Lineare Ungleichung:

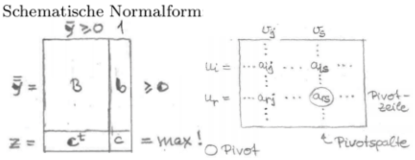
$$y := \mathbf{u}^t\mathbf{x} + u \geq 0$$

Ungleichungssystem mit l linearen Ungleichungen

$$y_i := \mathbf{a}_i^t\mathbf{x} + a_i \geq 0$$

bilden konvexes Polyeder S, *Simplex!*

Schematische Normalform



AUSTAUSCH:

Eingabe:
$A = [a_{ij}]_{i,j=1,1}^{m,n}$
r, s
Ausgabe:
$A' = [a'_{ij}]_{i,j=1,1}^{m,n}$

For i $\neq$ r, j $\neq$ s
$a'_{ij} \leftarrow a_{ij} - \frac{a_{is}a_{rj}}{a_{rs}}$
For i = r, j $\neq$ s (Pivotzeile)
$a'_{ij} \leftarrow -\frac{a_{ij}}{a_{rs}}$
For i $\neq$ r, j = s (Pivotspalte)
$a'_{ij} \leftarrow \frac{a_{ij}}{a_{rs}}$
For i = r, j = s (Pivot)
$a'_{ij} \leftarrow \frac{i}{a_{rs}}$

Algorithmus:

Eingabe:
Normalform B eines lin. Programms
Ausgabe:
Normalform geaendert, sodass $z(\mathbf{0})$=c=max

solange ein $c_s > 0$
falls alle $b_{is} \geq 0$
keine Loesung − Ende
sonst
bestimme r so, documentclass
$\frac{b_r}{b_{rs}} = \max_{b_{is}<0} \frac{b_i}{b_{is}}$
B ← AUSTAUSCH(B, r, s)

Aufwand im worst-case $\Omega\left(m^{\frac{n}{2}}\right)$. In Praxis meist in $O\left(m^2 n\right)$.

## Berechnung der Normalform

Gegeben ein lineares Programm

- finde zulässigen Punkt
- mache Punkt zum Ursprung
- führe n Tauschs $x_r$ mit $y_r$ durch

Damit Ursprung beim Tausch von $x_r$ mit $y_r$ gültig bleibt muss für alle $i > r$ gelten:

$$a_i - \frac{a_{ir}a_r}{a_{rr}} \geq 0$$

## Duale lineare Programme

Das lineare Programm

$$\mathbf{y} \geq 0$$
$$B^t\mathbf{y} + \mathbf{c} \leq 0$$
$$\mathbf{b}^t\mathbf{y} + c = min!$$

ist dual zu

$$\mathbf{x} \geq 0$$
$$B\mathbf{x} + \mathbf{b} \leq 0$$
$$\mathbf{c}^t\mathbf{y} + c = min!$$

Ersteres kann transformiert werden zu

$$\mathbf{y} \geq 0$$
$$-B^t\mathbf{y} - \mathbf{c} \geq 0$$
$$-\mathbf{b}^t\mathbf{y} - c = max!$$

## Ausgleichen mit Maximumsnorm

$A\mathbf{x} = \mathbf{a}$ überbestimmtes LGS. $\forall\mathbf{x}$ ist das Residuum

$$\mathbf{r} := (r_1...r_n)^t := A\mathbf{x} - \mathbf{a} \neq 0$$

Wir definieren $x_0 := \frac{1}{r}$ und $\bar{\mathbf{x}} := \mathbf{x}x_0$. Führt zu linearem Programm:

$$-A\bar{\mathbf{x}} + \mathbf{a}x_0 + \mathbf{e} \geq 0$$
$$A\bar{\mathbf{x}} - \mathbf{a}x_0 + \mathbf{e} \geq 0$$
$$x_0 = max!$$

## 1 Gradientenverfahren

Ausgehend vom Anfangspunkt $\mathbf{x_0} = [x_1, .., x_n]$, berechne Gradient der Kostenfunktion an dieser Stelle: $\bigtriangledown c(\mathbf{x_0}) = [\partial c/\partial x_1, ..., \partial c/\partial x_n]$. Der Nachfolger ist dann, $\mathbf{x_{i+1}} = \mathbf{x_i} + h * \bigtriangledown c(\mathbf{x_i})$, wobei $h > 0$, falls maximiert werden soll und $h < 0$, wenn minimiert wird.

## 3 stochastische verfahren

### 3.1 Allgemein

startzustand $\mathbf{q}$ -¿ einen nachbar $\mathbf{p}$ zufällig wählen -¿ schritt von ausgang zum neuen nachbar akzeptabel?, falls nein neuer nachbar, sonst von akzeptiertem zustand weiter, bis zurfieden

### 3.2 simuliertes tempern

Hier: Maximalzahl betrachteter nachbarn insgesamt (Bedingung fuer Terminieren), Folge sinkender temperaturen $t_1, t_2, ...$, Akzeptanzbedingung:
$c(\mathbf{p}) > c(\mathbf{q})\ \ OR\ \ exp(\frac{c(\mathbf{p}) - c(\mathbf{q})}{t_i}) > Zuffalszahl \in [0, 1]$, nächste Temperatur immer wenn akzeptiert wurde.

### 3.3 Schwellwert-Algorithmus

Akzeptanzbedingung: $c(\mathbf{p}) > c(\mathbf{q}) - \sigma$, wobei $\sigma$ immer weiter abgesenkt wird.

### 3.4 Sintflut-Maximierung

Akzeptanzbedingung: $c(\mathbf{p}) > F$, mit steigender unterer grenze F.

### 3.5 Rekordjagd-Algorithmus

Akzeptanzbedingung: $c(\mathbf{p}) \geq Rekord - \sigma$, bisher bester gesehener Wert, darf nicht um sinkende Toleranz unterschritten werden.

## 4 Evolutionäre Algorithmen

### 4.1 Allgemein

Population aus Individuen mit Merkmalsvektor $\mathbf{q}$ und Fitness c($\mathbf{q}$). Außerdem gibt es noch einen globalen Streuungsvektor $\sigma$, aus dem normal oder geometrisch verteilt mutiert wird. Individuen koennen sich fortpflanzen, bei mehreren Eltern Kreuzung, sonst Klon. Populationsgräße wird einigermaßen konstant gehalten, d.h. es werden immer wieder Loesungen verworfen.

### 4.2 Plus-Evolutionsstrategie

Erzeuge $\lambda$ Nachkommen, nur die $\mu$ fittesten Individuen aus den $\mu$ Eltern und $\lambda$ Nachkommen ueberleben.
Eltern werden pro Kind zufällig aus Population gewählt.

### 4.3 Komma-ES

Wie bei plus, allerdings sterben alle Eltern garantiert und nur $\mu$ fittesten Kinder ueberleben.

### 4.4 Klonen vs. Mehrere Eltern

Beim Klonen mutiere einfach mithilfe von $\sigma$ (s.o.).
Bei mehreren Eltern:

- Mischen: Es wird fuer jedes Merkmal ($q_i$) zufaellig bestimmt, von welchem Elternteil dieses Merkmal uebernommen wird.
- Mitteln: Es wird fuer jedes Merkmal der Mittelwert ueber die Werte der Eltern gebildet.

## 5 Genetische Algorithmen

Binäre Merkmalsvektoren, nur Nachkommen ueberleben.
Individuum klont sich mit Wahrscheinlichkeit $W(\mathbf{q}) = c(\mathbf{q})/\sum_{\mathbf{p}\in Pop} c(\mathbf{p})$. $\mu$ Eltern erzeugen immer $\mu$ Klon-Nachkommen.
Wähle unter den $\mu$ Klonen p% Individuen, die gekreuzt werden. Wähle daraus zufällige Paare. Aus a = (a1,...,an) und b = (b1,...,bn) entstehen c = (a1,...,aj,bj+1,...,bn) und d = (b1,...,bj,aj+1,...,an).
Dann kippe jedes Bit des Mermalsvektors mit sehr geringer Wahrscheinlichkeit.
Zusammengehörende Mermalsbits sollten möglichst nahe beisammen stehen, damit sie bei der Kreuzung nicht zerbrechen.

## 6 Partikel-Schwarm-Optimierung

Partikel jeweils mit Position $p_i(t)$, Fitness $f_i(p_i)$, Geschwindigkeit $v_i(t) = p_i(t) - p_i(t-1)$.
Algorithmus:
Initiale Position und Geschwindigkeit zufällig. Dann sei $q_i$ die bisher beste Position eines Partikels, q die global bisher beste Position.
Dann nächste Werte:
$v_i = v_i * \omega + (q_i - p_i) * c_1 * r_1 + (q - p_i) * c_2 * r_2$ r jeweils aus [0,1], c und $\omega$ konstant.
Die Konstanten können per Superschwarm optimiert werden.

## Schreibweise

$C$ ist De-Casteljau, $C^*$ Unterteilungsoperator mit Stufe $k$.

$$C\left(B_0^0, t\right) = B_0^1 B_1^1$$
$$C^*\left(B_0^0, k\right) := B_0^k ... B_{2^k-1}^k$$

Eingabe:
$\mathbf{b}_0^0...\mathbf{b}_n^0 \subset \mathbb{R}^d$
$t \in \mathbb{R}$
Ausgabe:
$\mathbf{b}_0^0...\mathbf{b}_0^n\ \mathbf{b}_0^n...\mathbf{b}_n^0 \subset \mathbb{R}^d$

For k = 1, ..., n
For i = 0, ..., n − k
$\mathbf{b}_i^k \leftarrow \mathbf{b}_i^{k-1}(1-t) + \mathbf{b}_{i+1}^{k-1}t$

## Konvergenz

Satz: Zu jedem Polygon $B = b_0...b_n$ gibt es ein Polygon $L_\infty(B)$ vom Grad $\leq n$, sodass

$$\sup_{[0,1]} |L_k(B) - L_\infty(B)| \in O\left(2^{-k}\right)$$

Folgerung:

$$\frac{d}{dt} L_\infty(B) = n L_\infty(\Delta B)$$

## Differenzenpolygon

$$\Delta B := \Delta\mathbf{b}_0...\mathbf{b}_n$$
$$:= (\Delta\mathbf{b}_0)...(\Delta\mathbf{b}_{n-1})$$
$$:= (\mathbf{b}_1 - \mathbf{b}_0)...(\mathbf{b}_n - \mathbf{b}_{n-1})$$

Zusammenhang von Differenzen mit normalem Polygon

$$B_0 B_1 := C^*(B, 1)$$
$$\Delta B_0 \Delta B_1 = \frac{1}{2} C^*(\Delta B, 1)$$
$$B_0...B_{2^k} := C^*(B, k)$$
$$\Delta B_0...\Delta B_{2^k} = 2^{-k} C^*(\Delta B, k)$$

Abhängigkeit von $\Delta B_0$ und $\Delta B_1$ bzgl. C: (Übung 9.2)

$$tL = \Delta B_0 \quad (1-t)R = \Delta B_1$$
$$\Rightarrow \frac{\Delta B_0}{t} \frac{\Delta B_1}{(1-t)} = C(\Delta B, t)$$

## Ableitung unterteilter Polygone

$$b(t) := L_k(B)$$
$$\beta_i^{k,j} := \frac{jn + i}{n2^k}$$
$$b_i^{k,j} := b^k\left(\beta_i^{k,j}\right)$$
$$\dot{b}^k(t) = \frac{\Delta b_i^{k,j}}{\Delta \beta_i^{k,j}} = n2^k \Delta b_i^{k,j}$$

## Kanonische Parametrisierung

Unterteilung eines Polygons in $P^k := \mathbf{P}_0^k...\mathbf{P}_{2^k-1}^k$ mit

$$P_j^k := \mathbf{p}_0^{k,j}...p_n^{k,j}$$
$$\mathbf{p}_i^{k,j} := \begin{bmatrix} \beta_i^{k,j} \\ \mathbf{b}_i^{k,j} \end{bmatrix}$$
$$\beta_i^{k,j} := \frac{j + \frac{i}{n}}{2^k}$$

Zu $P^k$ it die stückweise lineare Funktion $L_k(\mathbf{b}_0...\mathbf{b}_n)$