



Sommersemester 2006

Mehrkriterielle Optimierung mit Metaheuristiken (Vorlesung)

Prof. Dr. Günter Rudolph
 Fachbereich Informatik
 Lehrstuhl für Algorithm Engineering (LS XI)
 Fachgebiet *Computational Intelligence*



Kapitel 2: Grundlagen



Definition 2.1:

Binäre Relation $R \subseteq A \times A$, wobei A eine Menge.

Binäre Relation auf A heißt

- **reflexiv**, falls $\forall a \in A: (a,a) \in R$,
- **irreflexiv**, falls $\forall a \in A: (a,a) \notin R$,
- **symmetrisch**, falls $\forall a,b \in A: (a,b) \in R \Rightarrow (b,a) \in R$
- **asymmetrisch**, falls $\forall a,b \in A: (a,b) \in R \Rightarrow (b,a) \notin R$
- **antisymmetrisch**, falls $\forall a,b \in A: (a,b) \in R$ und $(b,a) \in R \Rightarrow a = b$
- **transitiv**, falls $\forall a,b,c \in A: (a,b) \in R$ und $(b,c) \in R \Rightarrow (a,c) \in R$
- **total** (oder **konnex**), falls $\forall a,b \in A: (a,b) \in R$ oder $(b,a) \in R$ ■

Kapitel 2: Grundlagen



Definition 2.2:

Binäre Relation auf A heißt

- **Quasiordnung** (oder **Präordnung**) \Leftrightarrow reflexiv, transitiv
- **Äquivalenzrelation** \Leftrightarrow reflexiv, transitiv, symmetrisch
- **partielle Ordnung** (oder **Halbordnung**) \Leftrightarrow reflexiv, transitiv, antisymmetrisch
- **lineare Ordnung** \Leftrightarrow totale Halbordnung
- **strenge partielle Ordnung** \Leftrightarrow irreflexiv, transitiv ■

Kapitel 2: Grundlagen



Definition 2.3:

Sei A eine Menge und R eine Relation über A . Das Paar (A, R) heißt

- **quasi geordnete Menge**, falls R Quasiordnung
- **halb geordnete Menge**, falls R Halbordnung
- **total** oder **linear geordnete Menge**, falls R lineare Ordnung ■

Satz 2.1:

Sei (A, \preceq) eine quasigeordnete Menge.

- (1) Falls $a < b \Leftrightarrow a \preceq b$ und nicht $b \preceq a$, dann $<$ strenge Halbordnung.
- (2) Falls $a \sim b \Leftrightarrow a \preceq b$ oder $b \preceq a$, dann \sim Äquivalenzrelation.

Beweis:

ad (2): \sim reflexiv (weil \preceq reflexiv) und symmetrisch (durch Definition).

Sei $a, b, c \in A$: $a \sim b$ und $b \sim c$. Aus Transitivität von \preceq folgt:

$a \preceq b \preceq c \Rightarrow a \preceq c$ und $c \preceq b \preceq a \Rightarrow c \preceq a$, so dass aus

$a \preceq c$ und $c \preceq a$ endlich $a \sim c$ folgt.

ad (1): $<$ irreflexiv (durch Definition). ■

Definition 2.4:

Sei (A, \preceq) eine halbgeordnete Menge und $<$ strenge Halbordnung gemäß Satz 1.

- Falls $a < b$ für gewisse $a, b \in A$, dann wird b von a **dominiert**.
- Verschiedene $a, b \in A$ sind **vergleichbar**, falls entweder $a < b$ oder $b < a$; sonst sind sie **unvergleichbar** (symbolisiert durch $a \parallel b$).
- (A, \preceq) heißt **Kette**, falls alle verschiedene $a, b \in A$ vergleichbar.
- (A, \preceq) heißt **Antikette**, falls alle verschiedene $a, b \in A$ unvergleichbar. ■

Bemerkungen:

- Kette = linear geordnete Menge
- (\mathbb{R}, \leq) ist Kette
- (\mathbb{R}^d, \leq) mit $x \leq y \Leftrightarrow \forall i: x_i \leq y_i$ und $x \neq y$ ist keine Kette für $d > 1$.

Bsp: $\begin{pmatrix} 3 \\ 6 \end{pmatrix} \preceq \begin{pmatrix} 4 \\ 7 \end{pmatrix}$ und $\begin{pmatrix} 3 \\ 6 \end{pmatrix} \preceq \begin{pmatrix} 3 \\ 7 \end{pmatrix}$ aber $\begin{pmatrix} 3 \\ 6 \end{pmatrix} \parallel \begin{pmatrix} 2 \\ 7 \end{pmatrix}$ ■

Definition 2.5:

Sei (A, \preceq) eine halbgeordnete Menge und $<$ strenge Halbordnung gemäß Satz 1.

$a^* \in A$ **minimales Element** falls kein $a \in A$ existiert mit $a < a^*$.

$M(A, \preceq)$ bezeichnet **Menge der minimalen Elemente**.

$M(A, \preceq)$ **vollständig**, falls $\forall a \in A$ existiert $a^* \in M(A, \preceq)$ mit $a^* \preceq a$. ■

Bemerkungen:

- minimale Elemente haben gewisse „Optimalitätseigenschaft“
- $M(A, \preceq)$ vollständig, falls A endlich (und für viele andere Fälle) ■

Wie finden wir Menge der minimalen Elemente $M(A, \preceq)$?

Annahme: A sei endlich mit $n = |A|$

Algorithmus 2.1:

```

A* = { } // am Ende: A* = M(A, <)
for (i = 1; i <= |A|; i++) {
    for (j = 1; j <= |A|; j++)
        if (a[j] < a[i]) break;
    if (j > |A|) then A* = A* ∪ { a[i] }
}
    
```

Laufzeit: $O(n^2)$

→ geht das schneller? **Nein!** In dieser Allgemeinheit nicht!

Kapitel 2: Grundlagen

Beschleunigung möglich, falls Halbordnung bestimmte Struktur hat!

Halbordnung im \mathbb{R}^d für $d > 1$:

$$a \preceq b \Leftrightarrow \forall i = 1, \dots, d: a_i \leq b_i \wedge a \neq b$$

Beispiel: Algorithmus 1

6	5	3	7	2	1	8	9	4	6	5	1	8	6	4	2	8	8
5	4	7	8	3	5	1	7	6	8	7	6	9	4	3	2	5	4

Laufzeit: $O(n^2)$ wie gehabt!

erste Verbesserung: falls $a[i] \preceq a[j]$ dann Element j aus Liste löschen

→ keine Verbesserung der worst case Schranke

Kapitel 2: Grundlagen

besser:

Beispiel: **Algorithmus 2.2**

6	5	3	7	2	1	8	9	4	6	5	1	8	6	4	2	8	8
5	4	7	8	3	5	1	7	6	8	7	6	9	4	3	2	5	4

sortieren bzgl. 1. Komponente (bei Gleichheit bzgl. 2. Komponente)

$O(n \log n)$

1	1	2	2	3	4	4	5	5	6	6	6	7	8	8	8	8	9
5	6	2	3	7	3	6	4	7	4	5	8	8	1	4	5	9	7

wähle 2. Komponente des 1. Elements

lösche alle Elemente rechts davon bis 2. Komponente kleiner dieses Element wird neues Auswahlelement

$\Theta(n)$

Kapitel 2: Grundlagen

Definition 2.6

Sei $x, y \in \mathbb{R}^d$. x heisst **lexikographisch kleiner** als y , wenn gilt:

$$x \preceq_{\text{lex}} y \Leftrightarrow \text{Entweder } x_i < y_i \text{ oder } x_i = y_i \wedge (x_{i+1}, \dots, x_d) \preceq_{\text{lex}} (y_{i+1}, \dots, y_d).$$

$x^{(1)}, x^{(2)}, \dots, x^{(n)} \in \mathbb{R}^d$ heisst **lexikographisch geordnet** \Leftrightarrow

$$x^{(1)} \preceq_{\text{lex}} x^{(2)} \preceq_{\text{lex}} \dots \preceq_{\text{lex}} x^{(n)}$$

Kapitel 2: Grundlagen

Lemma 2.1

Sei $x^{(1)}, x^{(2)}, \dots, x^{(n)}$ lexikographisch geordnet.

$$\forall i = 1, \dots, n-1: \exists j > i: x^{(j)} \preceq x^{(i)}.$$

Beweis:

Folgt unmittelbar aus lexikographischer Sortierung. ■

Lemma 2.2

Sei $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$ lexikographisch geordnet.

Dann ist $x^{(1)} \in M(X, \preceq)$.

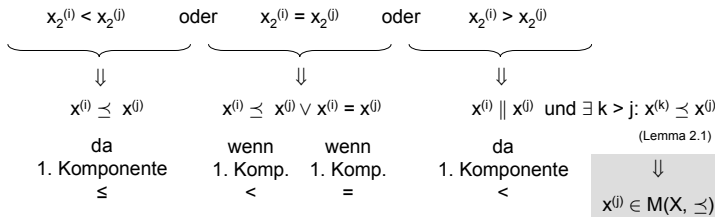
Beweis:

Folgt unmittelbar aus Lemma 2.1. ■

Korrektheit von Algorithmus 2.2 für d = 2

- Sei $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$ lexikographisch geordnet.
- Sei $i \in \{1, \dots, n\}$ derart, dass $x^{(i)} \in M(X, \preceq)$.
- Wg. Lemma 2.2 können wir mit $i = 1$ beginnen.

Es gilt $\forall j: j > i$:



Lemma 2.3:

Sei $C_d(n)$ max. Anzahl Vergleiche, um $M(V, \preceq)$ mit finitem $V \subset \mathbb{R}^d$ zu bestimmen.
 Es gilt: $C_{d-1}(n) \leq C_d(n)$ für $d \geq 2$ und $n = |V| < \infty$.

Beweis:

z.Z.: Berechnung für (d-1)-dim. Menge nicht schwerer als für d-dim. Menge!

- Sei $U \subset \mathbb{R}^{d-1}$ mit $|U| = n \Rightarrow C_{d-1}(n)$ Vergleiche nötig für $M(U, \preceq)$.
- Erweitere jedes $u \in U$ mit weiterer Komponente mit demselben Wert $\Rightarrow v =_{\text{def}}(u, w) = (u_1, u_2, \dots, u_{d-1}, w) \in V$.
- Offensichtlich: $u \in M(U, \preceq) \Leftrightarrow v = (u, w) \in M(V, \preceq)$, da $w = \text{const.}$ für alle u .
- Wir brauchen $C_d(n)$ Vergleiche für $M(V, \preceq)$ und bekommen $M(U, \preceq)$ durch Streichen der zusätzlichen Komponente $w \Rightarrow$ wir erhalten $M(U, \preceq)$ in höchstens $C_d(n)$ Vergleichen!
 $\Rightarrow C_{d-1}(n) \leq C_d(n)$ ■

$d > 2, n = 2^q, q \in \mathbb{N}$

Algorithmus 2.3

1. $R = \{x^{(1)}, \dots, x^{(n/2)}\}, S = \{x^{(n/2+1)}, \dots, x^{(n)}\}$ und jeweils lex. sortieren
2. $R^* = M(R, \preceq), S^* = M(S, \preceq)$
3. $T^* = \{s \in S^* \mid \nexists r \in R^*: r \preceq s\}$
4. Ausgabe: $R^* \cup T^*$

$$C_d(n) \leq \underbrace{2 C_d(n/2)}_{\text{Schritt 2}} + \underbrace{F_d(n/2, n/2)}_{\text{Schritt 3}}$$

$d > 2, n = 2^q, q \in \mathbb{N}$

$$F_d\left(\frac{n}{2}, \frac{n}{2}\right) \leq Q n (\log n)^{d-3}$$

$$\begin{aligned}
 C_d(n) &\leq 2 C_d\left(\frac{n}{2}\right) + F_d\left(\frac{n}{2}, \frac{n}{2}\right) \\
 &\leq 2^q \cdot C_d(1) + \sum_{i=1}^{q-1} 2^i F_d\left(\frac{n}{2^i}, \frac{n}{2^i}\right) \\
 &\leq n + Q \sum_{i=1}^{q-1} 2^i \frac{n}{2^i} \log\left(\frac{n}{2^i}\right)^{d-3} \\
 &= n + Q n \sum_{i=0}^{q-1} \log\left(\frac{n}{2^i}\right)^{d-3} \\
 &\leq n + Q n k (\log n)^{d-3} = O(n (\log n)^{d-2})
 \end{aligned}$$

Kung et al. (1975), JACM 22(4):469-476