Universität Dortmund

Wintersemester 2005/06

Fundamente der Computational Intelligence

(Vorlesung)

Prof. Dr. Günter Rudolph

Fachbereich Informatik

Lehrstuhl für Algorithm Engineering

Inhalt

- Grundlagen Optimierung
- Nachbarschaftssuche
- …

Nachbarschaftssuche

gegeben:

- Suchraum S
- Zielfunktion f: S $\rightarrow \mathbb{R}$ mit Zielsetzung Minimierung
- Nachbarschaftsstruktur N(x) für jedes x $\in$S → was ist N(x)?

```
wähle X₀ ∈ S; setze k = 0
repeat
  wähle Yₖ ∈ N(Xₖ)
  if f(Yₖ) < f(Xₖ) then Xₖ₊₁ = Yₖ else Xₖ₊₁ = Xₖ
  k = k + 1
until Terminierungbedingung = true
```

→ wie wählen?

Nachbarschaftssuche

Nachbarschaftsstruktur ist abhängig vom Suchraum:

- basiert meistens auf einer Metrik

Definition:

Sei M eine Menge. Die Funktion d: M $\times$ M $\rightarrow \mathbb{R}_{\geq 0}$ heißt <u>Metrik</u>, falls

1. $\forall$ x, y $\in$ M: d(x, y) $\geq$ 0, wobei d (x, y) = 0 $\Leftrightarrow$ x = y (Definitheit)
2. $\forall$ x, y $\in$ M: d(x, y) = d(y, x) (Symmetrie)
3. $\forall$ x, y, z $\in$ M: d(x, y) + d(y, z) $\geq$ d(x, z) (Dreiecksungleichung) ■

Nachbarschaftssuche

- $S = \mathbb{A}^n$ für $\mathbb{A} = \{ a_1, a_2, \ldots, a_K \}$

$$d(a,b) = \sum_{i=1}^{K} 1_{[a_i \neq b_i]}$$ Hamming-Abstand

- $S = \mathbb{B}^n$ für $\mathbb{B} = \{ 0, 1 \}$

$$\begin{aligned} d(a,b) &= \sum_{i=1}^{K} 1_{[a_i \neq b_i]} \\ &= \sum_{i=1}^{K} |\, a_i - b_i \,| = \sum_{i=1}^{K} a_i \otimes b_i \\ &= \sum_{i=1}^{K} [\, a_i\,(1-b_i) + (1-a_i)\,b_i \,] \end{aligned}$$

Nachbarschaftssuche

- Sei $S \neq \emptyset$ und $a, b \in S$:

$$d(a,b) = \begin{cases} 1, & \text{falls } a \neq b \\ 0, & \text{falls } a = b \end{cases}$$ diskrete Metrik

⇒ man kann jede nichtleere Menge mit einer Metrik ausstatten!
(allerdings ist die diskrete Metrik zur Nachbarschaftsspezifikation denkbar ungeeignet; warum?)

Satz:

Wenn d(a, b) eine Metrik auf S ist, dann ist auch $\dfrac{d(a,b)}{1 + d(a,b)}$ eine Metrik. ■

Nachbarschaftssuche

bisher: Metriken für abzählbare Mengen – überabzählbar?

Definition:

Sei M eine Menge. Die Funktion $|| \cdot ||: M \to \mathbb{R}_{\geq 0}$ heißt <u>Norm</u>, falls

1. $\forall x \in M: || x || = 0 \Rightarrow x = 0$ (Definitheit)
2. $\forall x \in M: \forall \alpha \in \mathbb{R}: || \alpha \cdot x || = | \alpha | \cdot || x ||$ (Homogenität)
3. $\forall x, y \in M: || x + y || \leq || x || + || y ||$ (Dreiecksungleichung) ■

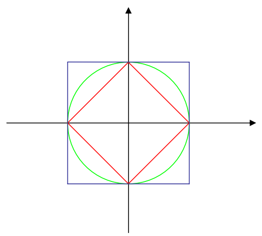
Satz:

Jede Norm induziert eine Metrik via $d(x, y) = || x – y ||$. ■

Nachbarschaftssuche

- $S = \mathbb{R}^n$ Hölder-Normen

$$d(a,b) = \left(\sum_{i=1}^{K} |\, a_i - b_i \,|^p \right)^{\frac{1}{p}} = \|a - b\|_p \qquad (p > 0)$$



p = 1 Betragssummennorm
p = 2 Euklidische Norm
p = ∞ Maximumnorm

Nachbarschaftssuche

Also:

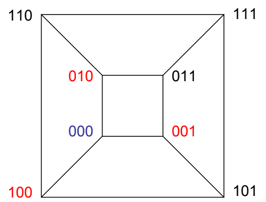
Nachbarschaft $N_r(x) = \{ y \in S : d(x, y) \leq r \}$ (enthält x)

punktierte Nachbarschaft $N_r(\dot{x}) = N_r(x) \setminus \{ x \}$ (häufig gemeint)

Sei $S = \mathbb{B}^n$.

$N_1(x) = \{ y \in \mathbb{B}^n : d(x, y) = 1 \}$

$N_1(000) = \{ 001, 010, 100 \}$



Nachbarschaftssuche

Elementauswahl:

● *Greedy method, steepest decent*

1. alle Elemente aus N(x) bewerten, d.h. Zielfunktionswert berechnen
2. wähle bestes Element aus N(x), bei gleich guten irgendeins davon

Probleme:
mitunter sehr viele Zielfunktionsberechnungen nötig
im $\mathbb{R}^n$ ist $|N(x)|$ überabzählbar $\Rightarrow$ so nur bei end. Nachbarschaften möglich

Nachbarschaftssuche

Elementauswahl:

● *First improvement*

Enumerationsschema für $N(x) = \{ e_1, e_2, \ldots, e_K \}$
aktuell bester Wert a* ist Parameter

1. Nachbarschaft aufzählen und Zielfunktionswert berechnen
2. Falls $f(e_i) < a^*$ dann wähle e_i

Probleme:
starke Abhängigkeit vom Enumerationsschema
funktioniert nur bei endlichen Nachbarschaften

Nachbarschaftssuche

Elementauswahl:

● *First improvement with random enumeration*

Enumerationsschema für $N(x) = \{ e_1, e_2, \ldots, e_K \}$ ist zufällige Permutation
aktuell bester Wert a* ist Parameter

1. Zufällige Permutation erzeugen
2. Nachbarschaft aufzählen und Zielfunktionswert berechnen
3. Falls $f(e_i) < a^*$ dann wähle e_i

Probleme:
funktioniert nur bei endlichen Nachbarschaften

Nachbarschaftssuche

Wie erzeugt man eine zufällige Permutation?

= „Gleichverteiltes Ziehen ohne Zurücklegen" (Urnenmodell)

W'keit für eine spezielle Permutation = $\frac{1}{K} \cdot \frac{1}{K-1} \cdot \frac{1}{K-2} \cdots \frac{1}{2} \cdot \frac{1}{1} = \frac{1}{K!}$

Realisierung: (lineare Laufzeit)

```
for i = 1 to K: A[i] = i
for i = 1 to K - 1:
  j = int_rand(i, K)       // Zufallszahl ∈ {i, i+1,…, K}
  swap(A[i], A[j])         // Vertauschen der Werte
endfor
```


Nachbarschaftssuche

Satz: (first improvement with random enumeration)
Wenn genau ein besseres Element in N(x) mit | N(x) | = N existiert, dann beträgt die Wahrscheinlichkeit, dass dieses Element genau beim k-ten Versuch getestet wird, den Wert 1 / N. Im Mittel werden (N+1) / 2 Versuche benötigt.

Beweis:
Erfolg genau im k-ten Versuch ⇒ vorher k-1 Misserfolge, dann Erfolg

$$\underbrace{\left[\prod_{i=0}^{k-2}\left(1-\frac{1}{N-i}\right)\right]}_{\text{Misserfolge}} \cdot \underbrace{\frac{1}{N-(k-1)}}_{\text{Erfolge}} = \frac{1}{N}$$

Erwartungswert = $\sum_{i=1}^{N} i \cdot \frac{1}{N} = \frac{1}{N} \cdot \frac{N(N+1)}{2} = \frac{N+1}{2}$ ■