

Vorlesung

# Effiziente Algorithmen und Komplexitätstheorie

Sommersemester 2008

Ingo Wegener

# Was bisher geschah...

- starke Zusammenhangskomponenten ✓
- Matchings ✓
- Flussproblem ✓
- Amortisierte Analyse ✓
- String Matching ✓
- Hidden-Markow-Modelle
  - Decodierungsproblem ✓
  - Auswertungsproblem ✓
  - Lernproblem

# Hidden Markov Modelle

## „Markov-Kette mit Ausgabe“

### Definition 7.1

Ein Hidden Markov Modell (HMM) ist definiert durch eine Markov-Kette mit endlicher Zustandsmenge

$Q = \{0, 1, \dots, |Q| - 1\}$  und Transitionswahrscheinlichkeiten  $a_{i,j}$  mit  $\sum_{j \in Q} a_{i,j} = 1$  für alle  $i \in Q$  und ein endliches Ausgabealphabet

$\Sigma$  und Ausgabewahrscheinlichkeiten  $e_i(b)$  mit  $\sum_{b \in \Sigma} e_i(b) = 1$  für alle  $i \in Q$ .

- Start **bei uns** immer in 0
- Zustände **verborgen**, nur Ausgabe beobachtbar

# Grundlegende Probleme

- **Decodierung**  
Finde eine wahrscheinlichste Zustandsfolge zu einer gegebenen Ausgabefolge.
- **Auswertung**  
Bestimme die Wahrscheinlichkeit einer gegebenen Ausgabefolge.
- **Lernen**  
Finde ein passendes HMM.

hier

- effizienter Algorithmus für Decodierung (**dynamische Programmierung**)
- effizienter Algorithmus für Auswertung (**dynamische Programmierung**)
- Heuristik für Lernen (**nur so zum Spaß**)

# Das Lernproblem

sehr schwierig

Wir lassen offen Wahl von  $|Q|$

Wie bestimmt man  $a_{i,j}$  und  $e_i(b)$  passend?

hier nur Heuristik

aber bekannt, verbreitet, empirisch oft gut

Ziel Finde  $W'$ keiten, die  $\text{Prob}(\sigma = x \mid M)$  maximieren

werden erreichen Finde zu HMM  $M$  und  $x \in \Sigma^n$  neues HMM  $M'$   
mit  $\text{Prob}(\sigma = x \mid M') \geq \text{Prob}(\sigma = x \mid M)$ ,  
hoffentlich oft  $\text{Prob}(\sigma = x \mid M') > \text{Prob}(\sigma = x \mid M)$

# Baum-Welch-Algorithmus

- 1 Starte mit W'keiten  $a_{i,j}$  und  $e_i(b)$ .
- 2 Schätze, wie oft Zustandswechsel  $i \rightsquigarrow j$  vorkommt.
- 3 Setze neue W'keiten proportional dazu.
- 4 Beurteile Unterschiede zwischen alten und neuen W'keiten.  
Setze ggf. bei 2 fort.

## Wie realisieren wir die Schätzung?

Definiere  $p_{i,j}(t) := \text{Prob}(q_t = i \wedge q_{t+1} = j \mid \sigma = x)$

### Lemma 7.5

$$p_{i,j}(t) = \frac{f_i(t) \cdot a_{i,j} \cdot e_j(x_{t+1}) \cdot b_j(t+1)}{\text{Prob}(\sigma=x)}$$

also  $p_{i,j}(t)$  mit bekannten Algorithmen effizient berechenbar

## Zum Beweis von Lemma 7.5

aus Definition  $p_{i,j}(t) = \text{Prob}(q_t = i \wedge q_{t+1} = j \mid \sigma = x)$

klar  $p_{i,j}(t) = \frac{\text{Prob}(q_t=i \wedge q_{t+1}=j \wedge \sigma=x)}{\text{Prob}(\sigma=x)}$

zu zeigen

$$\text{Prob}(q_t = i \wedge q_{t+1} = j \wedge \sigma = x) = f_i(t) \cdot a_{i,j} \cdot e_j(x_{t+1}) \cdot b_j(t+1)$$

Zeitpunkt  $t$  als Zäsur

$$\begin{aligned} & \text{Prob}(q_t = i \wedge q_{t+1} = j \wedge \sigma = x) \\ &= \text{Prob}(q_{t+1} = j \wedge (\sigma_{t+1}, \dots, \sigma_n) = (x_{t+1}, \dots, x_n) \wedge q_t = i \\ & \quad \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) \end{aligned}$$

$\text{Prob}(A \wedge B) = \text{Prob}(A \mid B) \cdot \text{Prob}(B)$  liefert

$$\begin{aligned} & \text{Prob}(q_{t+1} = j \wedge (\sigma_{t+1}, \dots, \sigma_n) = (x_{t+1}, \dots, x_n) \wedge q_t = i \\ & \quad \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) \\ &= \text{Prob}(q_{t+1} = j \wedge (\sigma_{t+1}, \dots, \sigma_n) = (x_{t+1}, \dots, x_n) \mid q_t = i \\ & \quad \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) \\ & \cdot \text{Prob}(q_t = i \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) \end{aligned}$$

# Schon wieder tapfer rechnen

## Wir haben

$$\begin{aligned} & \text{Prob}(q_{t+1} = j \wedge (\sigma_{t+1}, \dots, \sigma_n) = (x_{t+1}, \dots, x_n) \wedge q_t = i \\ & \quad \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) \\ &= \text{Prob}(q_{t+1} = j \wedge (\sigma_{t+1}, \dots, \sigma_n) = (x_{t+1}, \dots, x_n) \mid q_t = i \\ & \quad \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) \\ & \quad \cdot \text{Prob}(q_t = i \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) \end{aligned}$$

klar  $\text{Prob}(q_t = i \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) = f_i(t)$

## zu zeigen

$$\begin{aligned} & \text{Prob}(q_{t+1} = j \wedge (\sigma_{t+1}, \dots, \sigma_n) = (x_{t+1}, \dots, x_n) \mid q_t = i \\ & \quad \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) \\ &= a_{i,j} \cdot e_j(x_{t+1}) \cdot b_j(t+1) \end{aligned}$$

# Wieder passend abspalten

Abspalten von  $x_{t+1}$  und  $\text{Prob}(A \wedge B) = \dots$  liefern

$$\text{Prob}(q_{t+1} = j \wedge (\sigma_{t+1}, \dots, \sigma_n) = (x_{t+1}, \dots, x_n) \mid q_t = i$$

$$\wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t))$$

$$= \text{Prob}((\sigma_{t+2}, \dots, \sigma_n) = (x_{t+2}, \dots, x_n) \mid q_t = i$$

$$\wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t))$$

$$\wedge \sigma_{t+1} = x_{t+1} \wedge q_{t+1} = j)$$

$$\cdot \text{Prob}(\sigma_{t+1} = x_{t+1} \wedge q_{t+1} = j \mid q_t = i$$

$$\wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t))$$

scharfes Hinsehen liefert

$$\text{Prob}(q_{t+1} = j \wedge (\sigma_{t+1}, \dots, \sigma_n) = (x_{t+1}, \dots, x_n) \mid q_t = i$$

$$\wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t))$$

$$= \text{Prob}((\sigma_{t+2}, \dots, \sigma_n) = (x_{t+2}, \dots, x_n) \mid q_{t+1} = j)$$

$$\cdot \text{Prob}(\sigma_{t+1} = x_{t+1} \wedge q_{t+1} = j \mid q_t = i)$$

# Schrittweise Richtung Beweisende

Wir haben

$$\begin{aligned} & \text{Prob}(q_{t+1} = j \wedge (\sigma_{t+1}, \dots, \sigma_n) = (x_{t+1}, \dots, x_n) \mid q_t = i \\ & \quad \wedge (\sigma_1, \dots, \sigma_t) = (x_1, \dots, x_t)) \\ &= \text{Prob}((\sigma_{t+2}, \dots, \sigma_n) = (x_{t+2}, \dots, x_n) \mid q_{t+1} = j) \\ & \quad \cdot \text{Prob}(\sigma_{t+1} = x_{t+1} \wedge q_{t+1} = j \mid q_t = i) \end{aligned}$$

Beobachtung

$$\text{Prob}((\sigma_{t+2}, \dots, \sigma_n) = (x_{t+2}, \dots, x_n) \mid q_{t+1} = j) = b_j(t+1)$$

zu zeigen  $\text{Prob}(\sigma_{t+1} = x_{t+1} \wedge q_{t+1} = j \mid q_t = i) = a_{i,j} \cdot e_j(x_{t+1})$

$\text{Prob}(A \wedge B) = \text{Prob}(A \mid B) \cdot \text{Prob}(B)$  liefert

$$\begin{aligned} & \text{Prob}(\sigma_{t+1} = x_{t+1} \wedge q_{t+1} = j \mid q_t = i) \\ &= \text{Prob}(\sigma_{t+1} = x_{t+1} \mid q_{t+1} = j \wedge q_t = i) \cdot \text{Prob}(q_{t+1} = j \mid q_t = i) \end{aligned}$$

Beobachtung  $\text{Prob}(q_{t+1} = j \mid q_t = i) = a_{i,j}$

zu zeigen  $\text{Prob}(\sigma_{t+1} = x_{t+1} \mid q_{t+1} = j \wedge q_t = i) = e_j(x_{t+1})$

$p_{i,j}(t)$  berechnen

**zu zeigen**  $\text{Prob}(\sigma_{t+1} = x_{t+1} \mid q_{t+1} = j \wedge q_t = i) = e_j(x_{t+1})$

**klar**  $\text{Prob}(\sigma_{t+1} = x_{t+1} \mid q_{t+1} = j \wedge q_t = i)$   
 $= \text{Prob}(\sigma_{t+1} = x_{t+1} \mid q_{t+1} = j)$   
 $= e_j(x_{t+1})$



**also**  $p_{i,j}(t) = \text{Prob}(q_t = i \wedge q_{t+1} = j \mid \sigma = x)$   
 mit bekannten Algorithmen effizient berechenbar

## Zur Schätzung

Definiere  $T_{i,j} = \# \text{Zustandsübergänge } i \rightsquigarrow j$

Definiere  $T_{i,j}(t) = \begin{cases} 1 & \text{falls im } t\text{-ten Schritt } i \rightsquigarrow j \\ 0 & \text{sonst} \end{cases}$

klar  $T_{i,j} = \sum_{t=1}^{n-1} T_{i,j}(t)$

also 
$$\begin{aligned} \mathbf{E}(T_{i,j}) &= \mathbf{E}\left(\sum_{t=1}^{n-1} T_{i,j}(t)\right) \\ &= \sum_{t=1}^{n-1} \mathbf{E}(T_{i,j}(t)) \\ &= \sum_{t=1}^{n-1} \text{Prob}(T_{i,j}(t) = 1) \\ &= \sum_{t=1}^{n-1} p_{i,j}(t) \end{aligned}$$

# Wahl der Wahrscheinlichkeiten

**Definiere**  $T_i = \#$ erreiche Zustand  $i$   
 $\gamma_i(t) = \text{Prob}(q_t = i \mid \sigma = x)$

**klar**  $\gamma_i(t) = \sum_{j \in Q} p_{i,j}(t)$   
 $E(T_i) = \sum_{t=1}^{n-1} \gamma_i(t)$

**Setze**  $a'_{i,j} := \frac{E(T_{i,j})}{E(T_i)} = \frac{\sum_{t=1}^{n-1} p_{i,j}(t)}{\sum_{t=1}^{n-1} \gamma_i(t)}$   
 $e'_i(b) := \frac{\sum_{t=1}^{n-1} \gamma_i(t) \cdot \mathbb{1}_{x_t=b}}{E(T_i)} = \frac{\sum_{t=1}^{n-1} \gamma_i(t) \cdot \mathbb{1}_{x_t=b}}{\sum_{t=1}^{n-1} \gamma_i(t)}$

## Anmerkungen

- Man kann  $\text{Prob}(\sigma = x \mid M') \geq \text{Prob}(\sigma = x \mid M)$  zeigen.
- Man kann zeigen, dass das Verfahren konvergiert.

# Approximation

zunächst Begriffe

## Optimierungsproblem

- Menge von Instanzen  $I$
- Funktion  $S$ , die für alle  $w \in I$  Menge zulässiger Lösungen  $S(w)$  angibt
- Bewertungsfunktion  $v$ , die für alle  $w \in I$  und alle  $s \in S(w)$  Wert  $v(s)$  angibt
- Optimierungsziel: Maximierung oder Minimierung

zugehöriges

## Entscheidungsproblem

Eingabe  $w \in I$ ,  $k \in \mathbb{N}$

entscheide, ob  $\text{OPT}(w) \leq k$  (bei Minimierung)

bzw.  $\text{OPT}(w) \geq k$  (bei Maximierung)

# Klassifikation von Optimierungsproblemen

**Definition** Optimierungsproblem gehört zu  $\mathcal{NPO}$

- $I \in P$  (Instanz legal?)
- $\forall w \in I: S(w) \in P$  (Lösung zulässig?)
- $v$  polynomialzeitberechenbar
- zugehöriges Entscheidungsproblem  $\in NP$

**hier** nur Probleme aus  $\mathcal{NPO}$  betrachten

**Definition Güte**  $s \in S(w)$  zu  $w \in I$   
 $r := \max \left\{ \frac{v(s)}{\text{OPT}(w)}, \frac{\text{OPT}(w)}{v(s)} \right\}$   
 heißt **Güte** der Lösung  $s$

# Klassifikation von Approximationsalgorithmen

**Definition** Polynomialzeitalgorithmus  $A$ ,  
der immer Lösung mit Güte  $\leq r_A$  liefert,  
heißt  $r_A$ -Approximation

**Definition** Polynomialzeitalgorithmus  $A$ ,  
der für jede Güte  $r = 1 + \varepsilon > 0$   
immer Lösung mit Güte  $\leq r$  liefert,  
heißt **polynomielles Approximationsschema**

**Definition** polynomielles Approximationsschema  $A$ ,  
dessen Laufzeit polynomiell ist  
in Eingabelänge **und**  $\varepsilon^{-1}$ ,  
heißt **echt polynomielles Approximationsschema**

# Klassifikation von Optimierungsproblemen

- Definition** Optimierungsproblem  
mit  $r$ -Approximation für konstantes  $r \geq 1$   
heißt **approximierbar** und  
gehört zur Klasse  $\mathcal{APX}$
- Definition** Optimierungsproblem  
mit polynomielltem Approximationsschema  
gehört zur Klasse  $\mathcal{PTAS}$
- Definition** Optimierungsproblem  
mit echt polynomielltem Approximationsschema  
gehört zur Klasse  $\mathcal{FPTAS}$
- Definition** Optimierungsproblem  
mit zugehörigem Entscheidungsproblem in  $P$   
gehört zur Klasse  $\mathcal{PO}$

# Und jetzt?

**Beobachtung**  $\mathcal{PO} \subseteq \mathcal{FPTAS} \subseteq \mathcal{PTAS} \subseteq \mathcal{APX} \subseteq \mathcal{NPO}$

**Anmerkung**

$\mathcal{PO} \subsetneq \mathcal{FPTAS} \subsetneq \mathcal{PTAS} \subsetneq \mathcal{APX} \subsetneq \mathcal{NPO} \Leftrightarrow P \neq NP$

$\mathcal{PO} = \mathcal{FPTAS} = \mathcal{PTAS} = \mathcal{APX} = \mathcal{NPO} \Leftrightarrow P = NP$

**Wunsch** interessante Beispiele  
für verschiedene Approximationsalgorithmen

**erstes Beispiel** Rucksackproblem  
echt polynomielles Approximationsschema dazu

# Problemdefinition und Eigenschaften

**Erinnerung** **Rucksackproblem (KP)**  
**Eingabe** Gewichte  $g_1, g_2, \dots, g_n \in \mathbb{N}$   
 Nutzen  $v_1, v_2, \dots, v_n \in \mathbb{N}$   
 Gewichtsschranke  $G \in \mathbb{N}$   
**Ausgabe** Bepackung  $B \subseteq \{1, 2, \dots, n\}$   
 mit  $\sum_{i \in B} g_i \leq G$   
 und  $\sum_{i \in B} v_i$  maximal

**Erinnerung**  $KP \in \mathcal{NPO}$   
 zugehörige Entscheidungsvariante  $NP$ -vollständig

**harmlos**  $g_i \leq G$  für alle  $i$  voraussetzen ✓

**Erinnerung** pseudopolynomieller Algorithmus ( $O(n \cdot G)$ )

## Ein anderer pseudopolynomieller Algorithmus für KP

**Definiere**  $M_{i,V} := \min \left\{ \sum_{j \in B} g_j \mid B \subseteq \{1, 2, \dots, i\}, \sum_{j \in B} v_j = V \right\}$   
 minimales Gewicht,  
 mit dem Auswahl aus  $\{1, 2, \dots, i\}$  Nutzen  $V$  erzielt

„Randfälle“  $M_{0,0} = 0$   
 $M_{0,V} = \infty$  für  $V > 0$   
 $M_{i,V} = \infty$  für  $V < 0$

**Beobachtung**  $M_{i,V} = \min\{M_{i-1,V}, M_{i-1,V-v_i} + g_i\}$

**Beobachtung** dynamische Programmierung  
 $\rightsquigarrow$  Laufzeit  $O(n \cdot V_{\max})$

# KP mit kleinen Nutzenwerten schnell lösen

**Laufzeit**  $O(n \cdot V_{\max})$   
mit dynamischer Programmierung

**klar**  $V_{\max} = \sum_{i=1}^n v_i \leq n \cdot \max\{v_1, \dots, v_n\}$  **geeignet**

**klar** Bepackung selbst mit gleichem Aufwand bestimmbar

## Theorem 8.1

Es gibt einen Algorithmus für KP, der in Zeit  $O(n^2 \cdot \max\{v_1, \dots, v_n\})$  eine optimale Lösung berechnet.



# Ein winziges Beispiel

Eingabe  $G = 21$ , 

$v_i$	13	9	6	1	2
$g_i$	17	13	11	4	3

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	∞	∞	∞	∞	4	4
2	∞	∞	∞	∞	∞	3
3	∞	∞	∞	∞	∞	7
4	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	11	11	11
7	∞	∞	∞	∞	15	15
8	∞	∞	∞	∞	∞	14
9	∞	∞	13	13	13	13
10	∞	∞	∞	∞	17	17
11	∞	∞	∞	∞	∞	14
12	∞	∞	∞	∞	∞	∞
13	∞	17	17	17	17	17
14	∞	∞	∞	∞	21	21
15	∞	∞	∞	24	24	20

	0	1	2	3	4	5
16	∞	∞	∞	∞	28	24
17	∞	∞	∞	∞	∞	27
18	∞	∞	∞	∞	∞	31
19	∞	∞	∞	28	28	28
20	∞	∞	∞	∞	32	∞
21	∞	∞	∞	∞	∞	32
22	∞	∞	30	30	30	30
23	∞	∞	∞	∞	34	34
24	∞	∞	∞	∞	∞	33
25	∞	∞	∞	∞	∞	37
26	∞	∞	∞	∞	∞	∞
27	∞	∞	∞	∞	∞	∞
28	∞	∞	∞	41	41	41
29	∞	∞	∞	∞	45	45
30	∞	∞	∞	∞	∞	44
31	∞	∞	∞	∞	∞	48

Ausgabe 15 (mit  $B = \{1, 5\}$ )

# Ein echt polynomielles Approximationschema für KP

**Beobachtung** wenn alle  $v_i$  polynomiell klein  
Polynomialzeitalgorithmus verfügbar

**Idee** mache alle  $v_i$  polynomiell klein  

$$v_i \rightsquigarrow \tilde{v}_i := \lfloor \frac{v_i}{k} \rfloor$$
 $k$  gerade ausreichend groß

**zentrale Einsicht** Zulässigkeit davon unberührt

**Hoffnung** Rundungsfehler werden nicht zu groß

# Eine modifizierte, abgerundete Instanz

ursprüngliche Instanz Gewichte  $g_1, \dots, g_n \in \mathbb{N}$   
 Nutzen  $v_1, \dots, v_n \in \mathbb{N}$   
 Gewichtsschranke  $G \in \mathbb{N}$

modifizierte Instanz Gewichte  $g_1, \dots, g_n \in \mathbb{N}$   
 Nutzen  $\tilde{v}_1, \dots, \tilde{v}_n \in \mathbb{N}$   
 mit  $\tilde{v}_i := \lfloor \frac{v_i}{k} \rfloor$  mit  $k := \frac{\varepsilon \cdot \max\{v_1, \dots, v_n\}}{(1+\varepsilon) \cdot n}$   
 Gewichtsschranke  $G \in \mathbb{N}$

## Beobachtungen

- $k > 0$
- Bepackungen zulässig für beide Instanzen oder keine Instanz

# Laufzeit des pseudopolynomiellen Algorithmus

$$\begin{aligned}
 & O\left(n^2 \cdot \max\{\tilde{v}_1, \tilde{v}_2, \dots, \tilde{v}_n\}\right) \\
 = & O\left(n^2 \cdot \max\left\{\left\lfloor \frac{v_1}{k} \right\rfloor, \left\lfloor \frac{v_2}{k} \right\rfloor, \dots, \left\lfloor \frac{v_n}{k} \right\rfloor\right\}\right) \\
 = & O\left(n^2 \cdot \max_{1 \leq i \leq n} \left\{ \left\lfloor \frac{(1 + \varepsilon) \cdot n \cdot v_i}{\varepsilon \cdot \max\{v_1, v_2, \dots, v_n\}} \right\rfloor \right\}\right) \\
 = & O\left(n^2 \cdot \left\lfloor \frac{(1 + \varepsilon) \cdot n}{\varepsilon} \right\rfloor\right) = O\left(\frac{n^3}{\varepsilon} + n^3\right)
 \end{aligned}$$

also polynomiell in Eingabelänge und  $1/\varepsilon$

also  $k$  jedenfalls groß genug

# Abschätzung der Güte

**Betrachte** optimale Lösung  $B'$  für ursprüngliche Instanz  
 optimale Lösung  $B$  für modifizierte Instanz  
 berechnet durch pseudopolynomiellen Algorithmus

**zu zeigen** 
$$\frac{\sum_{i \in B'} v_i}{\sum_{i \in B} v_i} \leq 1 + \varepsilon$$

## Beobachtungen

- $\frac{v_i}{k} \geq \tilde{v}_i \geq \frac{v_i}{k} - 1$
- $\sum_{i \in B} \tilde{v}_i \geq \sum_{i \in B'} \tilde{v}_i$

weil  $B$  optimal für modifizierte Instanz

## Abschätzung zur Güte

$$\begin{aligned}
 \sum_{i \in B} v_i &\geq k \cdot \sum_{i \in B} \tilde{v}_i \geq k \cdot \sum_{i \in B'} \tilde{v}_i \geq k \cdot \sum_{i \in B'} \left( \frac{v_i}{k} - 1 \right) \\
 &= \left( \sum_{i \in B'} v_i \right) - k \cdot |B'| = \text{OPT} - k \cdot |B'| \\
 &\geq \text{OPT} - \frac{\varepsilon \cdot \max\{v_1, v_2, \dots, v_n\}}{(1 + \varepsilon) \cdot n} \cdot n \\
 &= \text{OPT} - \frac{\varepsilon}{1 + \varepsilon} \cdot \max\{v_1, v_2, \dots, v_n\} \\
 &\geq \text{OPT} - \frac{\varepsilon}{1 + \varepsilon} \cdot \text{OPT} \\
 &= \text{OPT} \cdot \left( 1 - \frac{\varepsilon}{1 + \varepsilon} \right) = \text{OPT} \cdot \frac{1}{1 + \varepsilon}
 \end{aligned}$$

# Zur Güte

Wir haben

$$\sum_{i \in B} v_i \geq \text{OPT} \cdot \frac{1}{1+\varepsilon}$$

$$= \left( \sum_{i \in B'} v_i \right) \cdot \frac{1}{1+\varepsilon}$$

also

$$\frac{\sum_{i \in B'} v_i}{\sum_{i \in B} v_i} \leq 1 + \varepsilon \checkmark$$

## Theorem 8.2

$\text{KP} \in \mathcal{FPTAS}$



# Das Traveling-Salesperson-Problem

Erinnerung **Traveling-Salesperson-Problem (TSP)**

**Eingabe** Distanzmatrix  $D = (d_{i,j})$  mit  $d_{i,j} \in \mathbb{N} \cup \{\infty\}$

**Ausgabe** „Rundreise“, „Tour“

Permutation  $\pi$  von  $\{1, 2, \dots, n\}$

mit  $\sum_{i=0}^{n-1} d_{\pi((i \bmod n)+1), \pi(((i+1) \bmod n)+1)}$  minimal

klar  $TSP \in \mathcal{NPO}$

Wie sieht es mit Approximationen aus?

**Behauptung** TSP **nicht approximierbar**  
 $TSP \notin \mathcal{APX}$  oder  $P = NP$

# Nichtapproximierbarkeit des TSP

**Betrachte** Eingabe  $G = (V, E)$  für Hamiltonkreis

**Definiere**  $|V| \times |V|$ -Matrix  $D$  mit

$$d_{i,j} = \begin{cases} 1 & \text{falls } \{i, j\} \in E \\ 2^n & \text{sonst} \end{cases}$$

**Beobachtung**  $\text{OPT} = \begin{cases} |V| & \text{falls } G \in \text{HC} \\ w > 2^n & \text{falls } G \notin \text{HC} \end{cases}$

**Annahme**  $\exists r$ -Approximation  $A$  für TSP

**Beobachtung**  $G \in \text{HC} \Rightarrow v(A(D)) \leq r \cdot n$   
 $G \notin \text{HC} \Rightarrow v(A(D)) > 2^n$

**also**  $(r < 2^n/n) \Rightarrow \text{HC} \in \text{P} \Rightarrow \text{P} = \text{NP}$

# Eingeschränkte TSP-Varianten

## metrisches TSP

- **zusätzlich symmetrisch**  $\forall i, j: d_{i,j} = d_{j,i}$
- **zusätzlich  $\Delta$ -Ungleichung**  $\forall i, j, k: d_{i,k} \leq d_{i,j} + d_{j,k}$

## euklidisches TSP

**Eingabe**  $n$  Punkte  $x_1, x_2, \dots, x_n \in \mathbb{R}^2$   
 Distanzen  $d_{i,j}$  implizit gegeben durch  

$$d_{i,j} := \sqrt{(x_i[1] - x_j[1])^2 + (x_i[2] - x_j[2])^2}$$

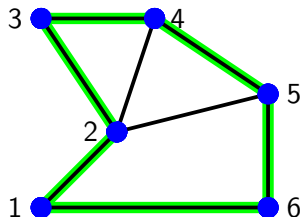
**Vorsicht** reelle Zahlen

**Anmerkung** im Kontext der Approximation nicht so kritisch

# Approximationen für das metrische TSP

Was hilft beim metrischen TSP zu besseren Approximationen?

Beobachtung „Mehrfachbesuche“ kostenlos reparierbar



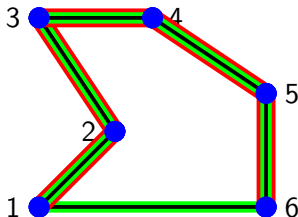
klar Knoten auslassen geht immer

aber nur beim metrischen TSP  
 $d_{4,5} \leq d_{4,2} + d_{2,5}$  wegen  $\Delta$ -Ungleichung  
also Gesamtkosten nicht größer

# Idee zur Approximation

Wie finden wir günstiges Tourgerüst?

Beobachtung Touren enthalten  
interessante bekannte Komponenten



Beobachtung enthält Spannbaum

darum  $\text{OPT} \geq v(\text{MST})$

also minimaler Spannbaum gutes „Gerüst“

# Von der Idee zum Algorithmus

## Definition 9.1

Ein ungerichteter *Multigraph*  $G = (V, E)$  ist definiert durch eine endliche Knotenmenge  $V$  und eine endliche Multimenge über  $(V \times V) \setminus \{\{v, v\} \mid v \in V\}$ .

Sei  $G = (V, E)$  ein ungerichteter Multigraph. Ein *Eulerkreis* ist ein Kreis in dem Graphen, der jede Kante genau einmal enthält.

## Lemma 9.2

Ein zusammenhängender Multigraph  $G = (V, E)$  enthält genau dann einen Eulerkreis, wenn alle Knoten geraden Grad haben.

**Anmerkung** Eulerkreise sogar in Linearzeit berechenbar

## Beweis von Lemma 9.2

„ $\Rightarrow$ “ Eulerkreis  $\rightsquigarrow$  alle Grade gerade

Berechne Knotengrade beim Durchlaufen des Eulerkreises  
je Betreten und je Verlassen +1

Beobachtung  $\forall$  Knoten:  $\#$ Betreten =  $\#$ Verlassen

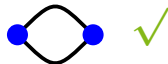
also alle Knotengrade gerade ✓

„ $\Leftarrow$ “ alle Grade gerade  $\rightsquigarrow$  Eulerkreis

vollständige Induktion über  $|E| = m$

Induktionsanfang leer für  $m \in \{0, 1\}$

für  $m = 2$



# Beweis von Lemma 9.2 „ $\Leftarrow$ “ $\forall$ Graph $G$ $\rightsquigarrow$ Eulerkreis

**Haben** zusammenhängenden Graph  $G = (V, E)$   
alle Knotengrade gerade,  $|m| > 2$

## Konstruiere Kreis

1. Starte in beliebigem Knoten  $v$ .
2. If  $\exists w$  unbesucht mit  $\{v, w\}$  unbenutzt
3. Then  $v := w$ ; Markiere  $v$  und  $\{v, w\}$ ; Weiter bei 2.
4. Else Wechsle zu benutztem Nachbarn  $w$ .

**Beobachtung** Endstück des Pfades ist Kreis  $K$

**Betrachte**  $G' = (V, E \setminus K)$

**Beobachtung** jeder Knoten in Bezug auf  $K$  Grad 0 oder 2

**also** jeder Knoten in  $G'$  mit geradem Grad

**darum** Induktionsannahme auf  $G'$  grundsätzlich anwendbar

## $G'$ und die Induktionsannahme

klar  $G' = (V, E \setminus K)$  enthält weniger Kanten

Ist  $G'$  auch zusammenhängend?

Einsicht muss nicht sein!

immerhin

Induktionsannahme  $\rightsquigarrow$  Zusammenhangskomponenten  $C_i$   
enthalten jeweils Eulerkreis

Beobachtung jedes  $C_i$  berührt  $K$

also EK aus  $C_i$  in  $K$  einfügen  $\rightsquigarrow$  Eulerkreis in  $G$



# Metrisches TSP mit MST und EK approximieren

## Algorithmus 9.3

1. Berechne MST  $T = (V, E_T)$  auf  $G$ .
2. Erzeuge Multigraphen  $G' = (V, E')$  mit  $E' = \{e, e \mid e \in E_T\}$ .
3. Berechne Eulerkreis  $K$  auf  $G'$ .
4. Berechne  $\pi$ , indem  $K$  durchlaufen und mehrfaches Vorkommen von Knoten entfernt wird.
5. Ausgabe  $\pi$

## Theorem 9.4

Algorithmus 9.3 hat Laufzeit  $O(n^2)$  und ist eine 2-Approximation für das metrische TSP.

# Beweis von Theorem 9.4

## Was ist zu zeigen?

- ① Laufzeit
- ② Korrektheit
- ③ Güte

### zur Laufzeit

- ① MST-Berechnung in Zeit  $O(n^2)$  (Algorithmus von Prim)
- ② Multigraphberechnung in Zeit  $O(n)$
- ③ Eulerkreisberechnung in Zeit  $O(n)$
- ④ Tourberechnung in Zeit  $O(n)$
- ⑤ Ausgabe in Zeit  $O(n)$

Gesamtlaufzeit  $O(n^2)$  ✓

### zur Korrektheit

nur **kritisch** ist  $G'$  zusammenhängend, alle Knotengrade gerade?

**Beobachtung** wegen Spannbaumverdopplung gesichert ✓

## Zur Güte

**Betrachte** Wert einer optimalen Lösung OPT

**Betrachte** Wert des Spannbaums  $v(T) = \sum_{\{i,j\} \in E_T} d_{i,j}$

**Erinnerung**  $v(T) \leq \text{OPT}$   
weil optimale Tour Spannäume enthält

**also**  $\sum_{\{i,j\} \in K} d_{i,j} = \sum_{\{i,j\} \in E'} d_{i,j} = 2v(T) \leq 2\text{OPT}$

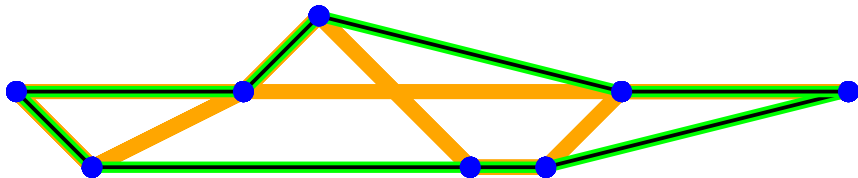
**also**  $v(\pi) = \sum_{i=0}^{n-1} d_{\pi((i \bmod n)+1), \pi(((i+1) \bmod n)+1)}$   
 $\leq 2v(T) \leq 2\text{OPT}$

**also**  $\frac{v(\pi)}{\text{OPT}} \leq 2$



# Beispiel

Graph  $G$ ,  $N = 8$ , Kantengewicht = Abstände



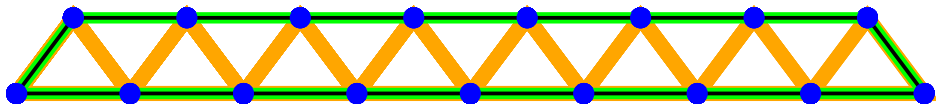
**MST**

**Tour aus Eulerkreis**    Länge  $\approx 24,3$

**optimale Tour**    Länge  $\approx 23,1$

## Noch ein Beispiel

Graph  $G$ ,  $N = 17$ , Kantengewicht = Abstände



**MST**

**Tour aus Eulerkreis** Länge  $16 \cdot 1,25 + 8 \cdot 1,5 = 32$

**optimale Tour** Länge  $8 \cdot 1,5 + 7 \cdot 1,5 + 2 \cdot 1,25 = 25$

# Bessere Approximationen

Wir haben 2-Approximation für das metrische TSP  
metrisches TSP  $\in \mathcal{APX}$

naheliegende Frage Wie geht es besser?

etwas präziser Wo verlieren wir viel?

klar Faktor 2 allein durch Spannbaumverdopplung

Beobachtung Abschätzung Gewinn durch Abkürzungen  
problematisch

Folgerung Können wir Spannbaumverdopplung einsparen?

# Einsparen der Spannbaumverdopplung

Warum verdoppeln wir den Spannbaum überhaupt?

**Erinnerung** damit alle Knoten geraden Grad haben  
für Eulerkreis-Konstruktion

**Beobachtung** zusätzliche Kanten nur nötig  
für Knoten mit ungeradem Grad

**Idee** Identifiziere Knoten mit ungeradem Grad in  $T$ ,  
verbinde paarweise durch möglichst günstige Kanten

**Erinnerung** „verbinde paarweise“  $\leftrightarrow$  perfektes Matching

**also** Suche kostenminimales perfektes Matching.

**Fakt** kostenminimale perfekte Matchings  
in Zeit  $O(n^3)$  berechenbar

# Algorithmus von Christofides

## Algorithmus 9.5

1. Berechne MST  $T = (V, E_T)$  auf  $G$ .
2. Berechne Menge  $V'$  der Knoten mit ungeradem Grad in  $T$ .
3. Berechne kostenmin. perfektes Matching  $M$  auf Knoten in  $V'$ .
4. Erzeuge Multigraph  $G' = (V, E')$  mit  $E' = E_T \cup M$ .
5. Berechne Eulerkreis  $K$  auf  $G'$ .
6. Berechne  $\pi$ , indem  $K$  durchlaufen und mehrfaches Vorkommen von Knoten entfernt wird.
7. Ausgabe  $\pi$

## Theorem 9.6

Der Algorithmus von Christofides hat Laufzeit  $O(n^3)$  und ist eine  $(3/2)$ -Approximation für das metrische TSP.

# Beweis von Theorem 9.6

## Was ist zu zeigen?

- ① Laufzeit
- ② Korrektheit
- ③ Güte

## zur Laufzeit

- ① MST-Berechnung in Zeit  $O(n^2)$  (Algorithmus von Prim)
- ② Berechnung von  $V'$  in Zeit  $O(n)$
- ③ Matchingberechnung in Zeit  $O(n^3)$
- ④ Multigraphberechnung in Zeit  $O(n)$
- ⑤ Eulerkreisberechnung in Zeit  $O(n)$
- ⑥ Tourberechnung in Zeit  $O(n)$
- ⑦ Ausgabe in Zeit  $O(n)$

Gesamtlaufzeit  $O(n^3)$  ✓

## Zur Korrektheit

Ist  $G'$  zusammenhängend und sind alle Knotengrade gerade?

Zusammenhang folgt aus Spannbaum ✓

gerade Knotengrade gerade Knoten in  $T$  unverändert in  $G'$  ✓

ungerade Knoten in  $T$  mit 1 neuen Kante in  $G'$  ✓

Ist  $|V'|$  gerade?

Beobachtung Summe aller Knotengrade in  $T$  gerade

Beobachtung Summe aller Knotengrade  
der Knoten mit geradem Grad in  $T$  gerade

also Summe aller Knotengrade  
der Knoten mit ungeradem Grad in  $T$  gerade

darum Anzahl der Knoten mit ungeradem Grad in  $T$  gerade ✓

# Zur Güte

wie vorhin  $v(\pi) \leq v(G') = v(T) + v(M) \leq \text{OPT} + v(M)$

zu zeigen  $v(M) \leq \text{OPT}/2$

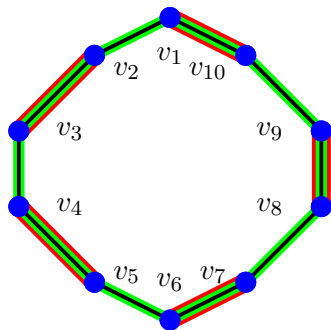
Betrachte Knoten aus  $V'$  in optimaler Tour  $\pi_{\text{OPT}}$   
Benenne so, dass in  $\pi_{\text{OPT}}$   
in Reihenfolge  $v_1, v_2, \dots, v_{|V'|}$  durchlaufen

Betrachte Kreis  $C = v_1, v_2, \dots, v_{|V'|}, v_1$

Beobachtung  $v(C) \leq v(\pi_{\text{OPT}}) = \text{OPT}$   
wegen  $\Delta$ -Ungleichung

# Abschätzung $v(M)$

Wir haben kostenminimales perfektes Matching auf Knoten in  $V'$   
 $V' = \{v_1, v_2, \dots, v_{|V'|}\}$   
 Kreis  $C = v_1, v_2, \dots, v_{|V'|}$  mit  $v(C) \leq v(\pi_{\text{OPT}}) = \text{OPT}$



Matching  $M_1$

Matching  $M_2$

Beobachtung  $v(M_1) + v(M_2) = v(C)$

Beobachtung  $\min\{v(M_1), v(M_2)\} \leq v(C)/2$

Beobachtung  $v(M) \leq \min\{v(M_1), v(M_2)\}$

also  $v(M) \leq v(C)/2 \leq \text{OPT}/2 \quad \square$

## Anmerkungen zum metrischen TSP

- Algorithmus von Christofides beste bekannte Approximation seit 1976
- obere Schranke scharf: entsprechende Worst-Case-Instanzen bekannt
- Existenz besserer Approximationen **offen**
- **bekannt**  $P \neq NP \Rightarrow$  metrisches TSP  $\notin PTAS$

also für polynomielles Approximationsschema  
weitere Einschränkung nötig

als nächstes euklidisches TSP