Approximationsalgorithmen

liefern in polynomieller Zeit Lösungen für Optimierungs-probleme, die vom Optimum nur um einen festen Faktor (die ***Güte*** des Appr. Algo) entfernt sind.

TSP: Falls in (G,w) die kürzeste Rundreise Länge k hat, muss ein Appr. Alg. mit Güte c eine Rundreise der Länge $\leq c \cdot k$ liefern.
[Minimierungsproblem, $c > 1$]

Knapsack: Falls $(\underline{g},\underline{w})$ eine Lösung mit Wert w erlaubt, muss ein Approx. Algo mit Güte c eine Lösung mit Wert $\leq c \cdot w$ liefern.
[Maximierungsproblem, c < 1]

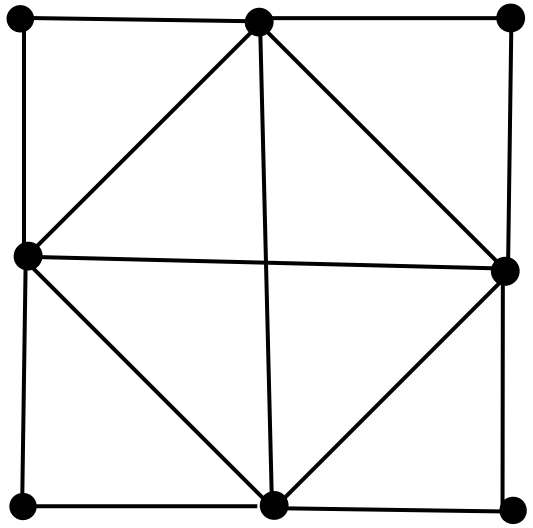
TSP := { $\langle G,w,k \rangle$ | (G,w) enth. einen Hamiltonkreis mit Gewicht $\leq k$ }

Approximationsalgorithmus für Knotenüberdeckung (VC)

Lemma: (i) Die Knoten eines nicht erweiterbaren Matchings *M* bilden eine Knotenüberdeckung.

(ii) Diese ist höchstens doppelt so groß wie eine minimale.

Beispiele für Matchings:
a) nicht erweiterbar (Gr. 2)
b) maximal (Größe 4)



Insbes. die Kanten des Matchings müssen überdeckt sein

Greedy Algorithmus für nicht erweiterbares Matching approximiert **VertexCover** in Laufzeit O(|*E*|) mit Güte 2.

Ein *Matching* in *G*=(*V*,*E*) ist eine Teilmenge *M* von *E*, in der keine zwei Kanten einen gemeinsamen Knoten enthalten.

Approximationsalgorithmus für das **metrische TSP**

TECHNISCHE UNIVERSITÄT DARMSTADT
Complexity Theory

TSP := { ⟨*G*,*w*,*k*⟩ | (*G*,*w*) enth. einen Hamiltonkreis mit Gewicht ≤*k* }

Eingabe: $w : V \times V \to \mathbb{N}$ symmetr. Kantengewichtsfunktion

Gesucht: Rundreise (Permut. π von V) mit min. Gewicht

MTSP Einschränkung: w erfülle Dreiecksungleichung:

$w(a,c) \leq w(a,b) + w(b,c)$ für alle $a,b,c \in V$.

Entscheidungsproblem **MTSP** bleibt **NP**-vollständig!

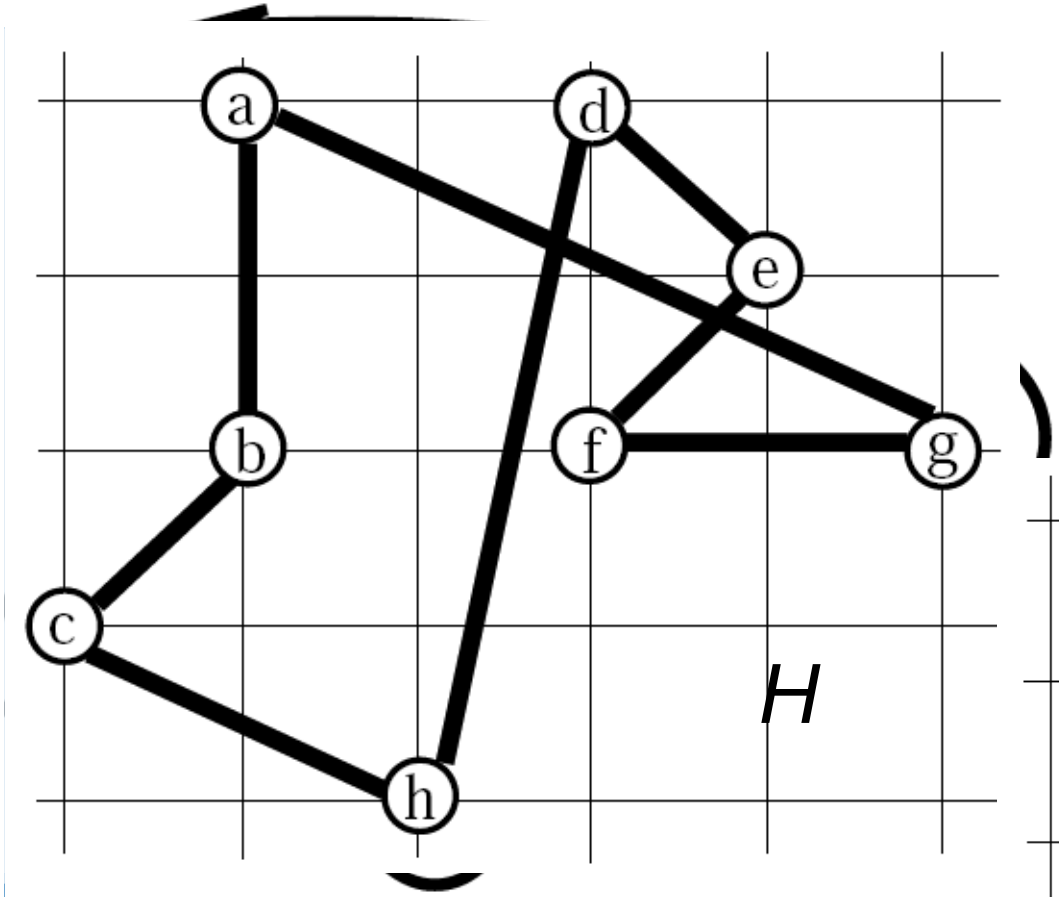
Polynom. Approximationsalgorithmus mit Güte 2:

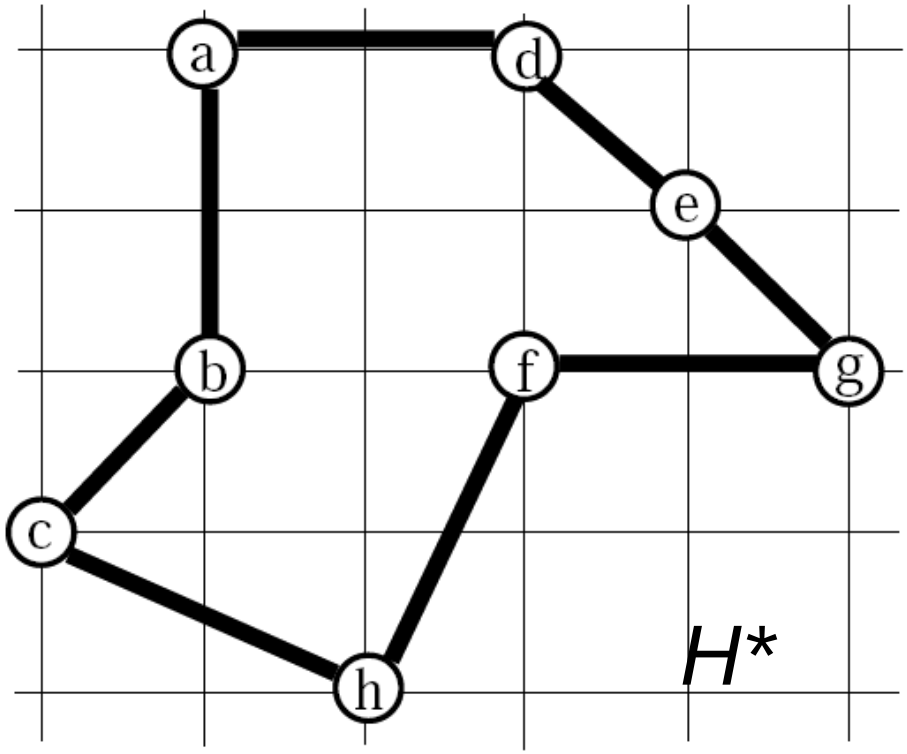
- Berechne Minimalen Spannbaum T in (G,w).
2. Starte bei bel. Knoten, durchlaufe T in Tiefensuche

Wurzel links rechts

ETSP zusätzl. Einschränkung: $V \subseteq \mathbb{R}^2$, $w(a,b) := \|a-b\|_2$ ist **NP**-schwer. Liegt in **NP**? → Dissertation Prof. Blömer

2. Zähle Knoten von *T* in Tiefensuche (W, L, R) auf





Beweis des Approximationsfaktors 2

TECHNISCHE UNIVERSITÄT DARMSTADT

w erfülle 3ecksungleichung, *T* sei Minimaler Spannbaum.
Algorithmus zählt die Knoten von *T* in Tiefensuche auf.

Sei *F* die Folge der in Tiefensuche durchlaufenen Kanten,
H die ausgegebene Tour, *H** eine optimale Tour.

Für Kanten $e_1,\ldots,e_k$ schreibe $L(e_1,\ldots,e_k) := w(e_1)+\ldots+w(e_k)$.

(i) $L(T) \le L(H^*)$, da wir aus *H** durch Entfernen irgendeiner Kante einen Spannbaum mit Kosten $\le L(H^*)$ erzeugen können.

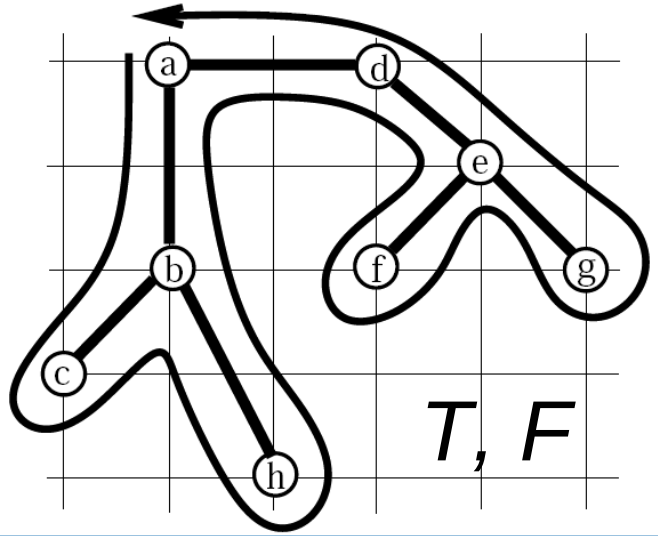
Weil in der Folge *F* jede Kante von *T* genau zwei Mal auftaucht:

(ii) $L(F) = 2 \cdot L(T)$

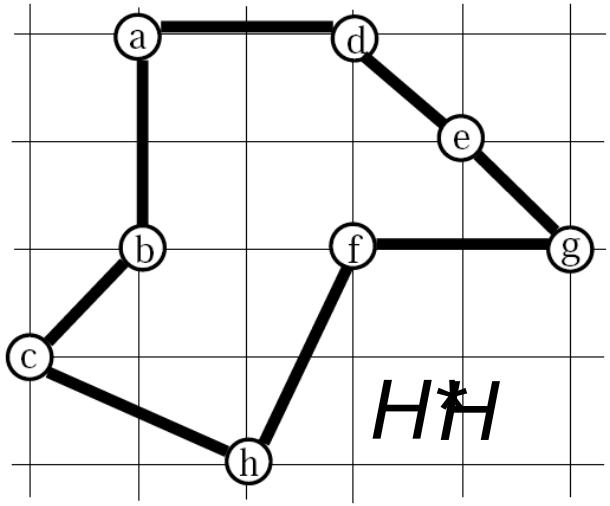
Wegen Dreiecksungl.:

(iii) $L(H) \le L(F)$

$\Rightarrow L(H) \le L(F) = 2 \cdot L(T) \le 2 \cdot L(H^*)$



T, F

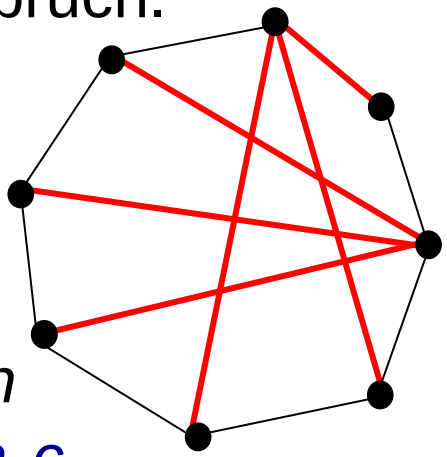


*H*H*

Grenzen der Approximierbarkeit

Satz: Falls $\mathcal{P} \neq \mathcal{NP}$ gilt,
gibt es kein polynom. Approx.Algo für **TSP** mit konstanter Güte.

Beweis: Wir nehmen an, es gäbe einen polynomiellen Approximationsalgorithmus A für **TSP** mit konstanter Güte $c \in \mathbb{N}$ und entwickeln daraus einen polynomiellen Algorithmus B für **HC**.
Da letzteres $\mathcal{NP}$-vollständig ist, folgt $\mathcal{P}=\mathcal{NP}$: Widerspruch.



Algorithmus B, Eingabe Graph $G=(V,E)$, $n:=|V|$.
Definiere $w(u,v) := 1$ falls $\{u,v\} \in E$;
$w(u,v) := n \cdot c$ falls $\{u,v\} \notin E$.

keine 3ecks-Ungleichung!

$\langle G \rangle \in \mathbf{HC} \Rightarrow$ w enthält Hamiltonkreis mit Gewicht n
$\Rightarrow$ Algo A findet einen mit Gewicht $\leq n \cdot c$
$\langle G \rangle \notin \mathbf{HC} \Rightarrow$ Jeder Hamiltonkreis von w hat Gewicht $\geq n \cdot c + n - 1 > n \cdot c$

HC := { $\langle G \rangle$ | G enthält einen Hamiltonkreis }
TSP := { $\langle G,w,k \rangle$ | (G,w) enth. einen Hamiltonkreis mit Gewicht $\leq k$ }

Knapsack *als Maximierungsproblem*

Gegeben: Werte und Gewichte $w_1,\ldots,w_n,g_1,\ldots,g_n \in \mathbb{N}$ sowie Gewichtsschranke g.

Gesucht: eine Teilmenge $S \subseteq \{1,\ldots,n\}$ mit $\sum_{p\in S} g_p \leq g$, die $\sum_{p\in S} w_p$ maximiert.

Algorithmische Idee: Dynamische Programmierung

Für $S \subseteq \{1,\ldots,n\}$ sei $\mathrm{gew}(S) := \sum_{j\in S} g_j$, $\mathrm{wert}(S) := \sum_{p\in S} w_p$.

Knapsack *als Minimierungsproblem*

Für $S \subseteq \{1,\ldots,n\}$: $\text{gew}(S) := \sum_{p \in S} g_p$, $\text{wert}(S) := \sum_{p \in S} w_p$

a) Suche S mit $\text{gew}(S) \leq g$, das $\text{wert}(S)$ maximiert: W

b) Suche S mit $\text{wert}(S) \geq w$, das $\text{gew}(S)$ minimiert: G

Sei $\boldsymbol{G_j(w)} := \min \{ \text{gew}(S) \mid S \subseteq \{1,\ldots,j\}, \text{wert}(S) \geq w \}$

kleinstmögliches Gewicht, mit dem unter den ersten j Objekten Wert mindestens w erzielt werden kann.

Falls kein solches S existiert, sei $F_j(w) := \infty$.

Lemma: i) $W = \max \{ w \mid G_n(w) \leq g \}$

ii) $G_j(w) = 0$ für $w \leq 0$

iii) $G_0(w) = \infty$ für $w > 0$

iv) $G_j(w) = \min \{ G_{j-1}(w) , g_j + G_{j-1}(w-w_j) \}$

$G_n(1) \leq G_n(2) \leq G_n(3) \ldots$
$\leq G_n(w) \leq g < G_n(w+1)$

Exakter Algo für Knapsack

Suche S mit $\text{gew}(S) \le g$, das $\text{wert}(S)$ maximiert

```
LET  w:=0.
WHILE  G_n(w)≤g  DO
        BEGIN
        w := w + 1
        FOR  j:=1…n  DO
                G_j(w) := min { G_{j-1}(w) , g_j + G_{j-1}(w-w_j) }
        END
PRINT w-1
```

Laufzeit $O(n \cdot \text{opt} \cdot n)$

Eingabegröße $\approx$ $\sum_j \log_2(w_j) + \log_2(g_j)$

opt bis zu $\sum_j w_j$

polynomiell nur für „viele kleine“ Pakete

Lemma iv) $G_j(w) = \min \{ G_{j-1}(w) , g_j + G_{j-1}(w-w_j) \}$

$G_n(1) \le G_n(2) \le G_n(3) \le \ldots \le G_n(w) \le g < G_n(w+1)$

Approximationsschema für Knapsack

Lemma a) Für $0 \le v_j$ gilt: $\text{MaxKnap}(\underline{g},\underline{w}) \le \text{MaxKnap}(\underline{g},\underline{w}+\underline{v})$

b) und für $v_j \le \ell$: $\text{MaxKnap}(\underline{g},\underline{w}+\underline{v}) \le \text{MaxKnap}(\underline{g},\underline{w}) + n \cdot \ell$

c) sowie $\text{MaxKnap}(\underline{g},k \cdot \underline{w}) = k \cdot \text{MaxKnap}(\underline{g},\underline{w})$

$$\lfloor \underline{w}/k \rfloor \cdot k \le w < \lfloor \underline{w}/k \rfloor \cdot k + k$$

Skalierungsmethode: Fixiere k und setze $w_j' := \lfloor w_j/k \rfloor$

Berechne $\text{opt}' := k \cdot \text{MaxKnap}(g,g_1,\ldots,g_n,w_1',\ldots,w_n')$ und J' mittels **`ExactKnapsack`** in Zeit $\text{poly}(n) \cdot W/k$. Dann

$$\text{opt} \ge \text{opt}' = \text{MaxKnap}(g, \lfloor \underline{w}/k \rfloor \cdot k) > \text{MaxKnap}(g,\underline{w}-k)$$

$$\ge \text{opt} - n \cdot k \ge \text{opt} \cdot (1 - n \cdot k/W)$$

Setze nun $k := \varepsilon \cdot W/n$.

oBdA $g_j \le g \Rightarrow$

$\max_j w_j =: W \le \text{opt} \le n \cdot W$

$$\text{MaxKnap}(\underline{g},\underline{w}) = \max\{ \textstyle\sum_{j \in J} w_j : J \subseteq \{1..n\}, \sum_{j \in J} g_j \le g \}$$

Nicht-/Approximierbarkeit

- **VC** kann in $\mathcal{P}$ mit Güte 2 approximiert werden,
- ebenso **MTSP**: mit Güte 2
- **Knapsack** $\in \mathcal{NP}$ kann in $\mathcal{P}$ mit Güte (1-ε) approximiert werden für beliebiges ε>0.
- Ebenso **SubsetSum** als Spezialfall mit $w_i=g_i$; Reduktion **3SAT** $\leqslant_p$ **SubsetSum** benutzte große Zahlen!
- **TSP** erlaubt in $\mathcal{P}$ keine Approx. mit konst. Güte
- Kann **CLIQUE** trivial mit Güte *n* approximieren;
- aber in $\mathcal{P} \neq \mathcal{NP}$ nicht mit Güte $O(n^{1-\varepsilon})$ für beliebiges ε>0 **(Johan Håstad 1996)**