- Mi, 4.2., 12–14 Vorbesprechung zum ADM-Seminar SS 09
- VL ADM III: Integer Linear Programming
  www.math.tu-berlin.de/coga/teaching/ss09/ILP/

LP als Werkzeug für Approximationsalgorithmen

Einfaches Runden

- Löse LP-Relaxation zu IP → polynomial viele Variablen ✓
  → exp. viele Variablen → effiziente Separierung?
- Runde LP(I) zu zul. Lösung A(I) → Zulässigkeit?
- Zeige $A(I) \leq \rho \cdot LP(I)$
- z.B. VC:
  - $x_v \geq 0.5 \iff x_v^* = 1$
  - $x_u + x_v = 1 \quad \forall (u,v) \in E \Rightarrow$ zulässig
  - worst case: $x_u = x_v = 0.5 \Rightarrow \rho = 2$.
- LP lösen erforderlich.

Randomisiertes Runden

z.B. MAX-k-SAT

- werfe Münze für jedes x_i

$\Rightarrow \mathbb{E}[I] \geq \left(1 - \left(\frac{1}{2}\right)^k\right) OPT(I)$

- gewichtete Wahrscheinlichkeit an LP

$y_i = 1 \iff x_i = \text{true}$

$z_j = 1 \iff C_j = \text{true}$

$T_j :=$ Menge der nicht-negierten Variablen in C_j

$F_j :=$ —— " —— negierten —— in ——

$$\max \sum_j z_j \qquad \text{(LP)}$$

$$\sum_{i \in T_j} y_i + \sum_{i \in F_j} (1 - y_i) \geq z_j$$

- Löse LP und mit Wkt. y_i, $x_i = \text{true}$.

$\Rightarrow \mathbb{E}[I] \geq 0.623\ OPT(I)$

Trick: Mittelwert zur Auswahl eines Algorithmus:

$\Rightarrow \frac{3}{4}$-Approximation

Duale Schranken

- Konstruiere zzg. Lösung $A(I)$ kombinatorisch
- Zeige $A(I) \leq \rho \cdot dual(I) \quad \left(\leq \rho \cdot LP(I)\right)$

z.B. VC:

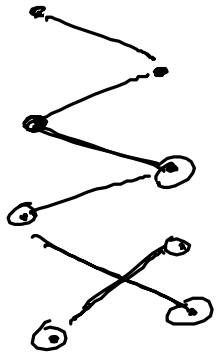
$$(P) \quad \min \sum_{v \in C} x_v \qquad x_u + x_v \geq 1 \quad \forall (u,v) \in E \qquad x_u \geq 0$$

$$(D) \quad \max_{e \in E} y_e \qquad \sum_{e \in \delta(v)} y_e \leq 1 \qquad y_e \geq 0$$

$\Rightarrow$ jedes Matching M ist dual zulässig ✓

↳ konstruiere inklusionsmaximales Matching M

- $x_u = x_v = 1 \quad \forall (u,v) \in M$
- Lösung ist VC, da M inklusionsmaximal



- $A(I) = 2 \cdot |M| \leq 2 \cdot \text{dual}(I)$

$\Rightarrow$ 2-Approximation.

Primal-Dual Approximationsalgorithmen

$$(P) \quad \min c^T x \qquad Ax \geq b \qquad x \geq 0$$

$$(D) \quad \max y^T b \qquad y^T A \leq c \qquad y \geq 0$$

HITTING SET:

Instanz: Grundmenge $E = \{e_1, \dots, e_n\}$, $c_e \geq 0 \quad \forall e \in E$

$T_1, \dots, T_p \subseteq E$.

Lösung: $A \subseteq E$ mit $T_i \cap A \neq \emptyset \; \forall i$, $\sum_{e \in A} c_e$ minimal

verallgemeinert: • Vertex Cover: $E = V$, $T_i = e_i$, e_i Kante.

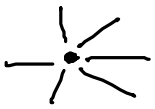
• kürzester s-t-Weg: $E \hat{=}$ Kanten

$T_i \hat{=}$ s-t-Schnitte in Graphen

• Minimum Spanning Tree: $E \hat{=}$ Kanten

$T_i \hat{=}$ Schnitte im Graph

• Perfektes Matching: $E \hat{=}$ Kanten
($c_e = 1$) $T_i \hat{=} \delta(v)$ $\forall v \in V$



Idee: Konstruiere gleichzeitig • ganzz. x

• duales y,

die relaxierte kompl. Schlupf erfüllen, d.h.

$$e \in A \Rightarrow \sum_{i: e \in T_i} y_i = c_e \qquad \forall e$$

$$y_i > 0 \Rightarrow |A \cap T_i| \leq \beta \qquad \forall i$$

Vorgehen: • eliminiere dualen Schlupf in dualer NB $e \in E$

• füge $e \in E$ A hinzu.

Algorithmus 1 (Grundversion)

• $y = 0$
• $A = \emptyset$
while (A unzulässig) {
• finde T_k mit $(T_k \cap A) = \emptyset$
• erhöhe y_k bis $\exists e \in E$ mit $\sum_{i: e \in T_i} y_i = c_e$
• $A := A \cup \{e\}$
}

Laufzeit: • $\leq |E|$ Iterationen $\Rightarrow$ höchstens $|E|$ positive y_i

• Finde von verletzter Teilung T_i? "Orakel"

$\Rightarrow$ "Orakel-polynomial" ($\exists$ polynomielles Orakel $\Rightarrow$ Algo polynomial).

Güte: rel. Bed. v. kompl. Schnitt.

$e \in A \Rightarrow \sum_{i: e \in T_i} y_i = c_e$ nach Konstruktion erfüllt

$\forall i: \quad y_i > 0 \Rightarrow |A \cap T_i| \leq \beta$

$\Rightarrow \beta$-Approximation für $\beta := \max_i |T_i|$.

z.B.: Vertex Cover: $|T_i| \equiv 2 \Rightarrow$ 2-Approximation

Allgemeine Ideen/Regeln für Verbesserung von β:

① Wähle T_i als inklusionsminimale verletzte Menge

z.B. s-t-kürzester Weg

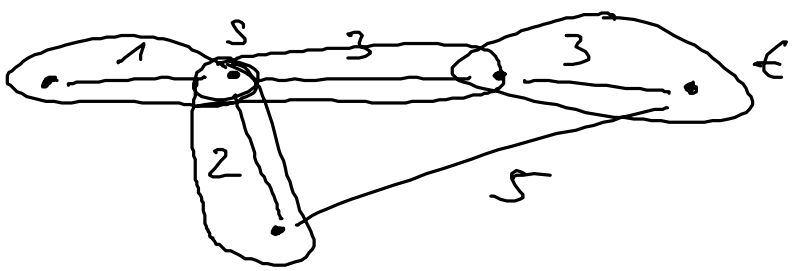
• Zu Beginn $A = \emptyset$, $T_i = \delta(S)$ mit $s \in S$, $t \notin S$

• inklusionsminimale verletzte Schnitt ist $\delta(\{s\})$.

$\Rightarrow$ billigste Kante in $\delta(s)$ wird in A aufgenommen

$\Rightarrow \{s, v\}$ ist im nächsten Schritt inklusionsminimale verletzte Menge

Bsp.:



$\Rightarrow$ A enthält am Ende des Algorithmus eine Ob-

menge eine zul. Lösung.

(2) Rückwärtslöschen

- Beim Hinzufügen von e zu A ist e notwendig für die Zulässigkeit von A
- Am Ende des Algorithmus vielleicht nicht mehr

Algorithmus 2:

- $y = 0$
- $A = \emptyset$
- $l = 0$

while $(\exists T_k : (A \cap T_k) = \emptyset)$ {

- $l := l + 1$
- erhöhe y_k, bis für ein e_l gilt $\sum_{i : e_l \in T_i} y_i = c_{e_l}$
- $A := A \cup \{e_l\}$

}

for $(j = l;\ j > 0,\ j--)$ {

if $(A \setminus \{e_l\}$ zul.$)$ {

$A := A \setminus \{e_l\}$

}

}

Analyse:

- Sei $T(A)$ die vollständige Menge, die zum Hinzufügen von e_j zu A geführt hat
- wenn e_j beim Rückwärtslöschen nicht aus A entfernt

$$\Rightarrow \{e_1, \dots, e_{j-1}\} \cap T_i = \emptyset$$

- Sei $B := A \cup \{e_1, \dots, e_{j-1}\}$

B hat <u>minimale Erweiterung</u> von $\{e_1, \dots, e_{j-1}\}$

$\Rightarrow |A \cap T(A)| \leq |B \cap T_i|$

Satz: Sei $T(A)$ die im Algorithmus gefundene verletzte Menge zu A. Für

$$\beta := \max_{\tilde{A} \subseteq E \text{ unzul.}} \; \max_{\substack{B \text{ min.} \\ \text{Erweiterung} \\ \text{von } \tilde{A}}} |B \cap T(\tilde{A})|$$

ist Algorithmus 2 eine β-Approximation.

Beweis: $\beta \geq \max_{\substack{B \text{ ist min. Erweit.} \\ \text{eines } A \subseteq E}} |B \cap T(A)| \geq |A \cap T(A)| \quad \forall T(A)$.

$\Rightarrow$ Diese β erfüllt rel. kompl. Schlupf $\square$

Für kürzeste s-t-Weg:

Sei A im Algo noch unzulässig, $T(A)$ ein inklusionsminimaler verletzter Schnitt (vgl. (7)). Jede minimale Erweiterung B von A enthält s-t-Weg, der durch das Entfernen einer bel. Kante aus $B \setminus A$ unterbrochen wird.

$\Rightarrow$ B hat nur eine Kante mit $T(A)$ gemeinsam.

$\Rightarrow \max_{A \subseteq E \text{ unzul.}} \; \max_{\substack{B \text{ min.} \\ \text{Erw. von } A}} |B \cap T(A)| = 1$

$\Rightarrow$ Algorithmus ist exakt.