- ! Mi, 4.2., 12–14 Vorbesprechung zum ADM-Seminar SS 09
- VL ADM III: Integer Linear Programming
  www.math.tu-berlin.de/coga/teaching/st09/ILP/

LP als Werkzeug für Approximationsalgorithmen

Einfaches Runden

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

Randomisiertes Runden

z.B. MAX-k-SAT

- werfe Münze für jedes x_i

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

- gewichtete Wahrscheinlichkeiten 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 —— " ——

$$\max \sum_j z_j \qquad\qquad (LP)$$

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

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

$$\Rightarrow \quad E[I] \geq 0.623 \; OPT(I)$$

Trick: Münzwurf zur Auswahl eines Algorithmus:

$$\Rightarrow \tfrac{3}{4}\text{-Approximation}$$

Duale Schranken

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

z.B. VC:

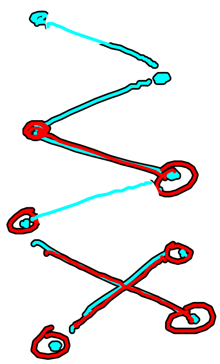
$$(P)\quad \min \sum_{v \in C} x_v \qquad x_u + x_v \geq 1 \quad \forall (u,v) \in E \qquad x_u \geq 0$$

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

$\Rightarrow$ 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



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

$\Rightarrow$ 2-Approximation.

Primal-Duale 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 $T_i \cap A \neq \emptyset \quad \forall i$, $\sum_{e \in A} c_e$ minimal verallgemeinert: • Vertex Cover: $E = V$, $T_i = e_i$, e_i Kante.

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

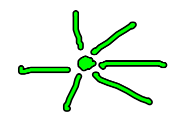
$T_i \mathrel{\hat=}$ s-t-Schnitte im Graphen

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

$T_i \mathrel{\hat=}$ Schnitte im Graphen

• Perfektes Matching ($c_e \equiv 1$): $E \mathrel{\hat=}$ Kanten

$T_i \mathrel{\hat=} \delta(v)$ $\forall v \in V$



<u>Idee:</u> Konstruiere gleichzeitig
• ganzz. x
• duales y,

die relaxierte kompl. Schlupf erfüllen, d.h.

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

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

<u>Vorgehen:</u> • eliminiere dualen Schlupf in dualer NB $e \in E$

• füge $e \in E$ A hinzu.

<u>Algorithmus 1 (Grundversion)</u>

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

<u>Laufzeit:</u> $\leq |E|$ Iterationen $\Rightarrow$ höchstens $|E|$ positive y_i

• Finden von verletzter Teilmenge T_k? "Orakel"

⇒ "Orakel-polynomial" (∃ polynomielles Orakel ⇒ Algo polynomial).

<u>Güte:</u> rel. Bed. v. kompl. Schlupf.

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

Z: $y_i > 0 \Rightarrow |A \cap T_i| \le \beta$

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

<u>z.B.:</u> Vertex Cover: $|T_i| = 2$ ⇒ 2-Approximation

<u>Allgemeine Ideen/Regeln für Verbesserung von β:</u>

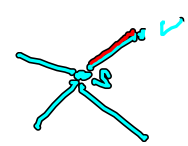
(1) Wähle T_k als inklusionsminimale verletzte Menge

z.B. s-t-kürzester Weg

• zu Beginn $A = \emptyset$, $T_1 = \delta(S)$ mit $s \in S$, $t \notin S$

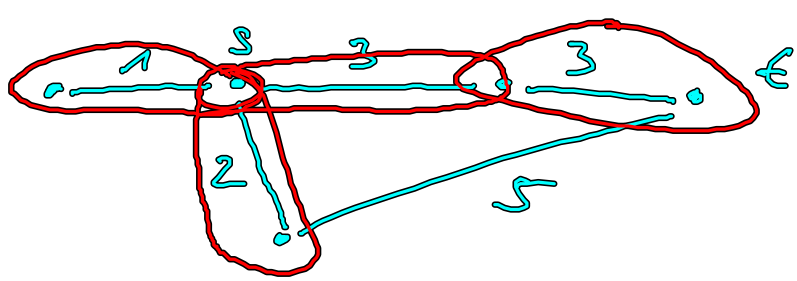
• inklusionsminimale verletzte Schnitt ist $\delta(\{s\})$.

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



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

Bsp.:



⇒ A enthält am Ende des Algorithmus eine Opt-menge ein 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 $\sum_{i : e \in T_i} y_i = c_{e_l}$
- $A := A \cup \{e_l\}$

}

for $(j = l;\ j > 0;\ j{-}{-})$ {
  if $(A \setminus \{e_l\}$ zul.$)$ {
    $A := A \setminus \{e_l\}$
  }
}

Analyse:
- Sei: $T^{(j)}$ die verletzte Menge, die zum Hinzufügen von e_j zu A geführt hat
- wurde 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 liefert minimale Erweiterung von $\{e_1, \dots, e_{j-1}\}$

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

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

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

ist Algorithmus 2 eine β-Approximation.

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

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

<u>Für kürzeste s-t-Weg:</u>

Sei A im Algo noch unzulässig, $T(A)$ eine inklusionsminimale verletzte Schnitt (vgl. 1). 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.

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

$\Rightarrow$ Algorithmus ist exakt.