Approximationsklassen

Ein Approximations-verfahren für ein Optimierungsproblem P ist ein Algorithmus A, der für jedes $x \in I_P$ eine Lösung $A(x) \in S_P(x)$ ausgibt.

Gesucht: Approximationsverfahren mit performance-Garantie!
(so wie der meistens betrachteten Beispiele!)

(1) Absolute Approximation

Def.: Für $x \in I$, $y \in S(x)$ ist der absolute Fehler
$D(x,y) := |m^*(x) - m(x,y)|$

Ein Approximations-Verfahren heisst absolut, wenn es eine Konstante k gibt mit
$$D(x, A(x)) \leq k \quad \text{f. alle } x \in I$$

Bsp: MINIMUM PLANAR GRAPH COLORING.

Algorithmus A, der in polynomieller Zeit:
- falls G 1, oder 2-Färbbar ist diesen optimal färbt
- andernfalls färbt er ihn mit 6 Farben.

Also ist f. alle G $\quad D(G, A(G)) \leq 3$.

Absolute Approximationsverfahren gibt es selten.

Satz: MAXIMUM KNAPSACK erlaubt kein absolutes Polynomialzeit Approximationsverfahren, falls $P \neq NP$

Beweis: Sei A ein solches mit absolutem Fehler $\leq k$
Für eine Instanz von MAXIMUM KNAPSACK, definiere eine neue mit $p_i' := p_i(k+1)$
Die einzige Lösung mit $D(x', A(x')) \leq k$ ist die optimale, also löst der Algorithmus das Problem exakt. □

(21) Relative Approximation

Def.: Performance Ratio, relativer Fehler (Folie) erklären

r-Approximation, r-approximierbar

Bsp: Lokale Suche ist ein 2-Approximationsalgorithmus für MAXIMUM CUT, PRIMAL DUAL 2-apx. für MLVC

:

MINIMUM SCHEDULING on IDENTICAL MACHINES ist $\frac{7}{6}$-approximierbar,
MINIMUM MULTICUT ist 2-approx.

ist für ein Minimierungsproblem mit Algorithmus A

$m(x, A(x)) \leq r \cdot m^*(x) + k$

so ist A kein r-Approximationsverfahren in diesem Sinn.

Analyse von FFD zeigt nur:

MINIMUM BIN PACKING ist 2,5-approximierbar,

da $m_{FFD}(x) \leq 1.5\, m^*(x) + 1 \leq 2.5\, m^*(x)$

Def. APX (Folie)

MAXIMUM SAT

Folgender Greedy-Algorithmus ist ein 2-Approximationsverfahren.

$C = D_1, \ldots, D_m$ Klauseln in x.

Finde Literal l mit den meisten Vorkommen.
Setze $l \to 1$ (ie. $x_i = 1$ if $l = x_i$, $x_i = 0$ if $l = \bar{x}_i$)
Lösche alle Klauseln D_i, in denen l vorkommt
Lösche $\bar{l}$ aus allen Klauseln, wo es vorkommt.
Lösche alle leeren Klauseln ⊛
Wiederhole bis C leer ist.

⊛ ist inkorrekt, diese Klausel bleiben unerfüllt.

Z.Z.: Gefundene Bewertung erfüllt mindestens $m/2$ Klauseln.
Durch Induktion nach n (Zahl der Variablen)
Induktionsanfang trivial ($n=1$)

Sei x_i die Variable in ℓ (höchstens vorkommendes Literal)

$m_1 :=$ Anzahl Klauseln, die x_i enthalten

m_2 ——— ——— $\overline{x_i}$ ———

oBdA $m_1 \geq m_2$. Algorithmus setzt $x_i \to 1$

Anzahl Klauseln, die noch übrigbleiben: $\geq m - m_1 - m_2$

$n-1$ Variablen

$\to$ I.H. Algorithmus findet Bewertung, die $\geq \frac{m - m_1 - m_2}{2}$ Klauseln erfüllt.

$\to$ Anzahl erfüllter Klauseln $\geq \frac{m - m_1 - m_2}{2} + m_1 \geq m/2$ □

Aber: TSP in APX $\to$ P=NP (Folie)

Angenommen TSP ∈ APX, also gibt es ein $r \geq 1$ so dass es ein r-Approximationsverfahren A für TSP gibt.

Wir zeigen, dass dann HC (Hamilton Kreis) in P ist.

Gegeben $G = (V, E)$. Definiere Instanz von TSP:

$$d(v_i, v_j) = \begin{cases} 1 & \{v_i, v_j\} \in E \\ 1 + nr & \text{sonst.} \end{cases}$$

Es gibt eine Tour mit Kosten n
gdw. G hat Hamilton-Kreis.

Jede andere (nicht-optimale) Tour hat Kosten
$\geq (n-1) + 1 + nr = n(1+r)$, also $R \geq 1+r$.

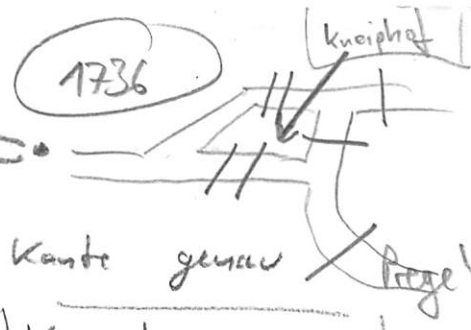
$\rightsquigarrow$ A findet die optimale Tour

Gibt es keinen Hamilton-Pfad in G, so hat jede Tour Kosten $\geq n(1+r)$.

Also:

G hat Hamilton Kreis gdw. A findet Tour der Länge n. □

Christofides Algorithmus



Königsberger Brückenproblem

Multigraph : kann parallele Kanten haben

Euler-Tour: geschlossener Pfad, der jede Kante genau einmal durchläuft.

G eulersch wenn es eine Euler Tour gibt

Satz: G eulersch g.d.w. jeder Knoten hat geraden Grad.

Es gibt einen polynomiellen Algorithmus zum Auffinden einer Euler Tour in Eulerschen Graphen.

Beobachtung: Gegeben eine Instanz von METRIC TSP.
Aus einem eulerschen Multigraphen E auf $\{c_1, \dots, c_n\}$, bei dem jede Kante e zwischen c_i, c_j das Gewicht $D(i,j) = d(e)$ hat, können wir eine Tour berechnen, die Kosten höchstens

$$\sum_{e \in E} d(e)$$

hat:

Jeder Knoten erscheint mindestens einmal in der Euler Tour, diese kann also aussehen

$$c_{\pi(1)} \xrightarrow{w_1} c_{\pi(2)} \xrightarrow{w_2} \cdots \xrightarrow{w_{n-1}} c_{\pi(n)} \xrightarrow{w_n} c_{\pi(1)}$$

w_i Weg von $c_{\pi(i)}$ nach $c_{\pi(i+1)}$. Wegen Δ ist

$D(\pi(i), \pi(i+1)) \leq$ Länge w_i. $\square$

Problem also: wie findet man so einen Eulerschen Graphen?
Idee: berechne minimalen Spannbaum T
erweitere diesen zu Eulerschen Graphen E

(1) verdoppele jede Kante in T

$\longrightarrow$ führt zu einem 2-Approximationsverfahren

Christofides:

nimm ein perfektes Matching auf den Knoten ungeraden Grades hinzu.

(kann in polynomialer Zeit gefunden werden: Primal Dual Algorithmus)

Für die Kosten m_{CH} der von Christofides gefundenen Lösung gilt:

$$m_{CH} \leq \frac{3}{2} m^*$$

Sei $d(T)$ die Summe der Gewichte im T

$d(M)$ M

Klar: $m_{CH} \leq d(T) + d(M)$. Wir zeigen

(1) $m^* \geq 2\, d(M)$

(2) $m^* \geq d(T)$

Damit $m_{CH} \leq m^* + \frac{1}{2} m^*$.

Zu (1):

Seien $c_1, \ldots, c_k$, $k = 2|M|$

die Knoten ungeraden Grades in T, in der Reihenfolge wie sie in der Tour auftreten.

Betrachte 2 Matchings:

$$M_1 : \{ (c_1, c_2) \ldots (c_{k-1}, c_k) \}$$

$$M_2 : \{ (c_2, c_3) \ldots (c_k, c_1) \}$$

Wegen Δ: $m^* \geq d(M_1) + d(M_2)$

D. M minimales Matching also $d(M_i) \geq d(M)$

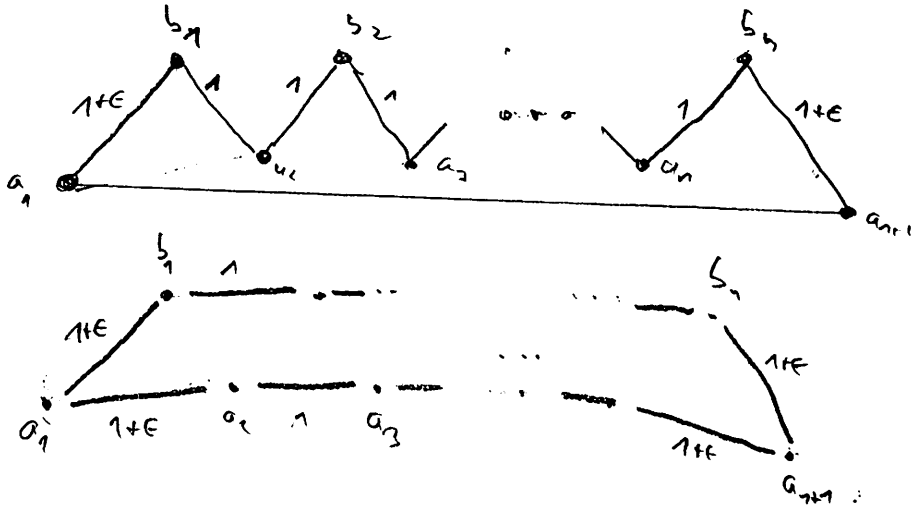
also $m^* \geq 2\, d(M)$.

Zu (2): Optimale Tour ist selbst ein Spannbaum + eine Kante. Da T minimaler Spannbaum:

$$m^* \geq d(T)$$

□

Die Schranke ist scharf, wie das folgende Bsp zeigt:



Tour, die Christofides findet

$3n + 2\epsilon$

Optimale Tour:

$2n+1+4\epsilon$

Performance Ratio $\longrightarrow \frac{3}{2}$ für $n \to \infty$

Approximationsschemata

Definition (Folie)

Laufzeit nur polynomial in $|x|$!

Klasse ATAS

zwischen PO und APX

Problem **Minimum Partition** (Folie) Trivial: jede Lösung $y \in S(x)$ hat $R(x,y) \leq 2$

Greedy-Algorithmus:

nach Gewicht sortieren.

jedes Teil jeweils in den Block mit geringerem Gesamtgewicht

→ liefert performance ratio $\frac{7}{6}$ Denn: Min PART = Min SCHED ON MACH w/ $p=2$, Greedy Alg = LPT

Verbesserung: Finde eine optimale Partition für die schwersten Objekte, wende für den Rest Greedy an.

→ Approximationsschema (Folie)

Analyse: Annahme: $r < 2$

$L = \frac{\sum X}{2}$ $\qquad m^* \geq L$

OBdA. $\sum Y_1 \geq \sum Y_2$ $\qquad \sum Y_1 \geq L \geq \sum Y_2$

X_h: das in Y_1 zuletzt eingefügte Element

$\sum Y_1 - a_h \leq \sum Y_2$

$\rightarrow \quad \sum Y_1 - L \leq \frac{a_h}{2}$

Fall 1: X_h in Phase 1 eingefügt

$\rightarrow$ Lösung optimal

Fall 2: X_h in Phase 2 eingefügt

$a_h \leq a_j$ für $j = 1..k$

$\rightarrow \quad 2L = \sum X \geq a_h (k+1)$

Insgesamt:

Performance ratio: $\frac{\sum Y_1}{m^*} \leq \frac{\sum Y_1}{L} \leq 1 + \frac{a_h}{2L} \leq 1 + \frac{1}{k+1}$

$\leq \frac{1}{\frac{2-r}{r-1} + 1} + 1 = r$

□

Laufzeit:

Sortieren:	$O(n \log n)$	} insgesamt polynomial in n
Optimum Bsp.:	$O(2^k)$	
Greedy-Teil	$O(n)$	

(aber exponentiell in k)

MAXIMUM INTEGER KNAPSACK:

wie MAX KNAPSACK, aber von jeder Sorte beliebig viel Vorrat.

Algorithmus: alle Teilmengen mit beschränkter Gewichtssumme $\leq \delta$, die potentielle Lösungen sind, auspobieren + greedy ergänzen.

Analyse

Sei c^* optimale Lösung. $c^*: X \to \mathbb{N}$.

Falls $\sum_{i=1}^{n} c^*(x_i) \leq \delta$:

Algorithmus findet optimale Lösung!

Sonst $\sum_{i=1}^{n} c^*(x_i) > \delta$. Definiere

$$C_\delta^*(x_i) = \begin{cases} c^*(x_i) & \sum_{j=1}^{i} c^*(x_j) \leq \delta \\ \delta - \sum_{j=1}^{i-1} c^*(x_j) & \sum_{j=1}^{i-1} c^*(x_j) \leq \delta \wedge \sum_{j=1}^{i} c^*(x_j) > \delta \\ 0 & \text{sonst} \end{cases}$$

C_δ^* gibt Sorte an, in viel wie möglich von optimaler, dann Rest

---

C_δ^* wird von Alg. berechnet.

$k := \max_i \; C_\delta^*(x_i) > 0$

$b_k := b - \sum_{i=1}^{k} a_i \, C_\delta^*(x_i)$

$m := \arg\max \left\{ \frac{p_k}{a_k}, \dots, \frac{p_n}{a_n} \right\}$

c Lösung, die Alg. liefert:

$$m(c) \geq m(C_\delta^*) + p_m \cdot \lfloor b_k / a_m \rfloor$$

und:

$$m^* \leq m(C_\delta^*) + b_k \cdot \frac{p_m}{a_m}$$

Also $R(c) := \frac{m^*}{m(c)} \leq \frac{m(c_\delta^*) + b_k \, p_m / a_m}{m(c_\delta^*) + p_m \lfloor b_k / a_m \rfloor}$

$$\leq \frac{m(c_\delta^*) + p_m \left(b_k/a_m - \lfloor b_k/a_m \rfloor \right)}{m(c_\delta^*)}$$

$$= 1 + \frac{p_m \left(b_k/a_m - \lfloor b_k/a_m \rfloor \right)}{m(c_\delta^*)}$$

$$\leq 1 + \frac{p_m}{m(c_\delta^*)} \leq 1 + \frac{p_k}{\delta p_k}$$

$$= 1 + \frac{1}{\delta} = r$$

□

Laufzeit:

im wesentlichen Anzahl der betrachteten F die ist $\binom{n + \lfloor \delta \rfloor + 1}{\lfloor \delta \rfloor} = O(n^\delta)$

$$= O\left(n^{1 + \frac{1}{1/\delta}}\right)$$

Unterschied zu PTAS für PARTITION

$O(2^k \cdot n^{O(1)})$ EPTAS

$P \neq NP \rightarrow$ MIN. BINPACKING $\notin$ PTAS

PARTITION

Instanz: $\{x_1, \dots, x_n\} = X$

Gewichte $w_1, \dots, w_n$

Frage: Gibt es $Y \subseteq X$ mit $\sum_{x_i \in Y} w_i = \sum_{x_i \in X \setminus Y} w_i$

Ist NP-vollständig!

Daraus: Instanz von MIN BIN PACKING:

$$B := \sum_{i=1}^{n} w_i \qquad \forall i \; w_i \leq B/2 .$$

obdA. sonst trivial nein!

Definiere $\langle\langle a_1 \dots a_n \rangle\rangle$ - X'

mit $a_i := \frac{2w_i}{B}$

Ist X positive Instanz

$\rightsquigarrow m^*(X') = 2$ ∎

sonst

$$m^*(X') \geq 3 .$$

also nach Lückentechnik:

kein r-Approximationsverfahren für

$$r < \frac{3}{2} .$$

D.h. MIN-BINPACKING $\notin$ APX:

$P \neq NP \rightsquigarrow$ PTAS $\subsetneq$ APX.

□

Lückentechnik:

Approximations-Algorithmus für P A
mit Performanz $r < 1+\varepsilon$ könnte Q entscheiden:

input $x \in \Sigma_Q$
$x' := f(x)$
$y := A(x')$
return $\left(m(x', y) < c(x) \cdot (1+\varepsilon)\right)$

dann:

$x \notin \Sigma_Q$:

$m^*(x') \geq c(x)(1+\varepsilon)$, denn

$m(x', y) \geq c(x)(1+\varepsilon)$

$x \in \Sigma_Q$:

$$\begin{aligned} m(x', y) &\leq r \cdot m^*(x') \\ &\leq r \cdot c(x) \\ &< (1+\varepsilon)\, c(x). \end{aligned}$$

□

$P \neq NP \longrightarrow$ kein beschränktes NP-schweres Opt.-problem in FPTAS.

Sei A ein FPTAS für Q; Maximierungsproblem (für Min. ähnlich).

Q beschränkt $\rightarrow$

$\forall$ alle $x \in I_Q$: $m^*(x) \leq p(|x|)$ Polynom $p(\cdot)$

Laufzeit von A: $q(|x|, \frac{1}{r-1})$ Polynom $q(\cdot,\cdot)$

Definiere $r := 1 + \frac{1}{p(|x|)}$

$A(x, r)$ liefert $y \in S(x)$ mit

$$\frac{m^*(x)}{m(x,y)} \leq r = \frac{p(|x|)+1}{p(|x|)}.$$

also $$m(x,y) \geq m^*(x) \frac{p(|x|)}{p(|x|)+1} = m^*(x) - \frac{m^*(x)}{p(|x|)+1}$$

$$\geq m^*(x) - 1$$

daher: $m(x,y) = m^*(x)$ □

Alg.: optimale Lösung in Zeit

$$q\left(|x|, \frac{1}{r-1}\right) = q(|x|, p(|x|))$$

polynomiell ! □

Q NP-schwer $\rightarrow$ $P = NP$