Übersicht

Absolute Approximation

Sei P ein Optimierungsproblem , $x \in I_P$ und $y \in S_P(x)$.

Definition: Absoluter Fehler

$$D(x, y) := |m^*(x) - m(x, y)|$$

Ein Approximationsalgorithmus A heißt absolut,
wenn $D(x, A(x) \leq k$ ist, für alle x und eine Konstante k.

Beispiel: Minimum Planar Graph Coloring

Satz

Falls **P** $\neq$ **NP**, *gibt es keinen absoluten Approximationsalgorithmus für* Maximum Knapsack.

Relative Approximation

Definition: Performanz:

$$R(x,y) := \max\left(\frac{m(x,y)}{m^*(x)}, \frac{m^*(x)}{m(x,y)}\right)$$

Relativer Fehler:

$$E(x,y) := \frac{|m^*(x) - m(x,y)|}{\max(m^*(x), m(x,y))} = 1 - \frac{1}{R(x,y)}$$

Für $r \geq 1$ ist ein r-Approximationsalgorithmus ein Algorithmus A, der für jedes $x \in I_P$ ein $A(x) \in S_P(x)$ mit $R(x, A(x)) \leq r$ liefert.

P heißt r-approximierbar, falls es einen r-Approximationsalgorithmus für P gibt.

Die Klasse **APX**

Approximations-algorithmen

Einführung
Grundlagen
Methoden zum Entwurf
Approximationsklassen
Absolute und relative Approximation
Die Klasse APX
Approximations-schemata

APX ist die Klasse aller Optimierungsprobleme $P \in$ **NPO**, die r-approximierbar für ein $r \geq 1$ sind.

$$\textbf{PO} \subseteq \textbf{APX} \subseteq \textbf{NPO}$$

APX enthält **NP**-schwere Probleme, z.B. Maximum Knapsack, also

$$\textbf{P} \neq \textbf{NP} \quad \Longrightarrow \quad \textbf{PO} \subsetneq \textbf{APX}$$

Maximum SAT

Instanz:	CNF $D_1 \wedge \ldots \wedge D_m$ in Variablen $x_1, \ldots, x_n$
Lösung:	Bewertung $\alpha : \{x_1, \ldots, x_n\} \to \{0, 1\}$
Maß:	Anzahl der erfüllten Klauseln $\#\{i; \alpha \models D_i\}$

Maximum SAT $\in$ **APX**.

Traveling Salespersons revisited

Satz

MINIMUM TRAVELING SALESPERSON $\notin$ **APX**,
außer wenn **P** $=$ **NP**.

Also gilt:

$$\mathbf{P} \neq \mathbf{NP} \quad \Longrightarrow \quad \mathbf{PO} \subsetneq \mathbf{APX} \subsetneq \mathbf{NPO}$$

Euler-Tour

Multigraph: kann parallele Kanten haben.

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

Satz (Euler)

Es gibt eine Euler Tour gdw. für alle Knoten v
$\deg v$ *gerade ist.*

Euler-Tour kann in polynomieller Zeit berechnet werden.

TSP-Tour aus Euler-Tour

Sei $V = \{c_1, \ldots, c_n\}$ mit $D \in \mathbb{R}^{n \times n}$ Instanz von Metric TSP

Sei $G = (V, E)$ Euler'scher Graph auf V.

Lemma

Aus Euler-Tour in G extrahiert man TSP-Tour π mit

$$m(\pi) \leq \sum_{\{c_i, c_j\} \in E} d(i, j)$$

Minimale Spannbäume

Minimum Spanning Tree

Instanz:	zusammenhängender Graph $G = (V, E)$, Kantengewichte $w : E \to \mathbb{N}$
Lösung:	Spannbaum $T \subseteq E$
Maß:	Gesamtgewicht $\sum_{e \in T} w(e)$

Satz

Minimum Spanning Tree *ist in* **PO**.

Approximations-algorithmen

Einführung
Grundlagen
Methoden zum Entwurf
Approximationsklassen
Absolute und relative Approximation
Die Klasse APX
Approximations-schemata

Ein 2-Approximationsalgorithmus

- Berechne minimalen Spannbaum T
- verdoppele jede Kante in T
- berechne eine Euler-Tour in diesem Graphen
- extrahiere daraus eine TSP-Tour

Satz

Für den Wert $m_A(x)$ der so gefundenen Tour gilt:

$$m_A(x) \leq 2m^*(x)$$

Korollar: MINIMUM METRIC TRAVELING SALESPERSON $\in$ **APX**

Matchings in gewichteten Graphen

Matching in $G = (V, E)$: Teilmenge $M \subseteq E$ mit

$$e_1 \cap e2 = \emptyset \quad \text{für alle} \quad e_1, e_2 \in E$$

Matching M ist perfekt, wenn $|M| = |V|/2$.

MINIMUM WEIGHT PERFECT MATCHING

Instanz: Graph $G = (V, E)$, der ein perfektes Matching hat
Kantengewichte $w : E \to \mathbb{N}$

Lösung: Perfektes Matching $M \subseteq E$

Maß: Gesamtgewicht $\sum_{e \in M} w(e)$

Satz

MINIMUM WEIGHT PERFECT MATCHING *ist in* **PO**.

Approximations-algorithmen

Einführung
Grundlagen
Methoden zum Entwurf
Approximationsklassen
Absolute und relative Approximation
Die Klasse APX
Approximations-schemata

Christofides' Algorithmus

- Berechne minimalen Spannbaum T
- sei U die Menge der Knoten mit ungeradem Grad in T
- berechne perfektes Matching M minimalen Gewichts auf U
- berechne eine Euler-Tour in $T \cup M$
- extrahiere daraus eine TSP-Tour

Satz

Für den Wert $m_C(x)$ der von Christofides' Algorithmus gefundenen Tour gilt:

$$m_A(x) \leq \frac{3}{2} m^*(x)$$

Approximationsschemata

Ein Approximationsschema für P ist ein Algorithmus A mit

- Eingabe: $x \in I_P$ und $1 < r \in \mathbb{Q}$
- Ausgabe: $A(x, r) \in S_P(x)$
- Performanz: $R(x, A(x, r)) \leq r$
- Laufzeit: $|x|^{O(1)}$

PTAS ist die Klasse aller Optimierungsprobleme in **NPO**, für die es ein Approximationsschema gibt.

$$\textbf{PO} \subseteq \textbf{PTAS} \subseteq \textbf{APX} \subseteq \textbf{NPO}$$

PTAS enthält **NP**-schwere Probleme (z.B. Maximum Knapsack):

$$\textbf{P} \neq \textbf{NP} \implies \textbf{PO} \subsetneq \textbf{PTAS}$$

Beispiel: ein Approximationsschema

Minimum Partition

Instanz:	Menge $X = \{x_1, \ldots, x_n\}$
	für $x_i \in X$ Gewicht $a_i \in \mathbb{N}$
Lösung:	Partition $X = Y_1 \uplus Y_2$
Maß:	$\max(\sum_{x_i \in Y_1} a_i \ , \ \sum_{x_i \in Y_2} a_i)$

Approximationsschema:

sortiere X so dass $a_1 \geq a_2 \geq \ldots \geq a_n$
$k := \lceil (2-r)/(r-1) \rceil$
finde eine optimale Partition $Y_1 \uplus Y_2 = \{x_1, \ldots, x_k\}$
for $j := k+1$ to n
  if $\sum Y_1 \leq \sum Y_2$
    then $Y_1 := Y_1 \cup \{x_j\}$
    else $Y_2 := Y_2 \cup \{x_j\}$

Ein Problem in **PTAS**

Maximum Integer Knapsack

Instanz:	Menge $X = \{x_1, \ldots, x_n\}$,
	für jedes $x_i \in X$: Gewicht $a_i \in \mathbb{N}$
	Wert $p_i \in \mathbb{N}$
	Kapazität $b \in \mathbb{N}$
Lösung:	Funktion $c : X \to \mathbb{N}$ mit $\sum_{x_i \in Y} c(x_i) a_i \leq b$
Maß:	Gesamtwert $\sum_{x_i \in Y} c(x_i) p_i$

Approximationsschema für Integer Knapsack

sortiere X so dass $p_1 \geq p_2 \geq \ldots \geq p_n$
$c := \vec{0}$
$\delta := \lceil \frac{1}{(r-1)} \rceil$
for all f with $\sum f(x_i) \leq \delta$ and $\sum f(x_i)a_i \leq b$
  $k := \max_i f(x_i) > 0$
  $b_k := b - \sum f(x_i)a_i$
  sei $x_m \in \{x_k, \ldots, x_n\}$ mit $\frac{p_m}{a_m}$ maximal
  $f(x_m) := f(x_m) + \lfloor \frac{b_k}{a_m} \rfloor$
  if $\sum f(x_i)p_i \geq \sum c(x_i)p_i$ then
    $c := f$

APX versus PTAS

Lückentechnik: Sei Q **NP**-vollständig und $P \in$ **NPO**. Wenn es Funktionen $f : I_Q \to I_P$ und $c : I_Q \to \mathbb{N}$ in **FP** gibt mit

$$m^*(f(x)) \leq c(x) \quad \text{für } x \in Y_Q$$
$$m^*(f(x)) \geq c(x)(1+G) \quad \text{für } x \in N_Q$$

für ein $G > 0$, dann ist P nicht r-approximierbar für $r < 1 + G$.

Satz

Falls **P** $\neq$ **NP**, *dann ist* Minimum Bin Packing *nicht in* **PTAS**.

$$\textbf{P} \neq \textbf{NP} \implies \textbf{PO} \subsetneq \textbf{PTAS} \subsetneq \textbf{APX} \subsetneq \textbf{NPO}$$

Approximations-algorithmen
Einführung
Grundlagen
Methoden zum Entwurf
Approximationsklassen
Absolute und relative Approximation
Die Klasse APX
Approximations-schemata

Volle Approximationsschemata

Definition: Ein volles Approximationsschema für P ist ein Algorithmus A mit

- Eingabe: $x \in I_P$ und $1 < r \in \mathbb{Q}$
- Ausgabe: $A(x, r) \in S_P(x)$
- Performanz: $R(x, A(x, r)) \leq r$
- Laufzeit polynomiell in x und $\frac{1}{r-1}$

FPTAS ist die Klasse aller Optimierungsprobleme in **NPO**, für die es ein volles Approximationsschema gibt.

$$\mathbf{PO} \subseteq \mathbf{FPTAS} \subseteq \mathbf{PTAS} \subseteq \mathbf{APX} \subseteq \mathbf{NPO}$$

FPTAS enthält **NP**-schwere Probleme (z.B. Maximum Knapsack):

$$\mathbf{P} \neq \mathbf{NP} \implies \mathbf{PO} \subsetneq \mathbf{FPTAS}$$

FPTAS vs. PTAS

Approximations-algorithmen

Einführung
Grundlagen
Methoden zum Entwurf
Approximationsklassen
Absolute und relative Approximation
Die Klasse APX
Approximations-schemata

Definition: Ein Problem $P \in \mathbf{NPO}$ ist beschränkt, falls es ein Polynom p gibt mit:

für alle $x \in I_P$ und $y \in S_P(x)$ ist $m(x, y) \leq p(|x|)$.

Satz

Falls $\mathbf{P} \neq \mathbf{NP}$*, ist kein beschränktes* **NP***-schweres Optimierungsproblem in* **FPTAS**.

Es gibt **NP**-schwere, beschränkte Probleme in **PTAS**, z.B. Maximum Planar Independent Set:

$$\mathbf{P} \neq \mathbf{NP} \implies \mathbf{PO} \subsetneq \mathbf{FPTAS} \subsetneq \mathbf{PTAS} \subsetneq \mathbf{APX} \subsetneq \mathbf{NPO}$$

Pseudopolynomielle Probleme

Sei $P \in \mathbf{NPO}$ mit Zahlen als Bestandteil der Instanzen,

für $x \in I_P$ sei $\max(x)$ das Maximum der Zahlen in x.

P heißt pseudopolynomiell, wenn es einen Algorithmus A gibt, der P_C in Zeit $p(|x|, \max(x))$ löst.

Beispiel: MAXIMUM KNAPSACK ist pseudopolynomiell.

Proposition

Sei $P \in \mathbf{FPTAS}$, so daß für alle $x \in I_P$ und $y \in S_P(x)$ gilt:
$m(x, y) \leq p(|x|, \max(x))$.
Dann ist P pseudopolynomiell.

Stark **NP**-schwere Probleme

Für $P \in \mathbf{NPO}$ und p Polynom ist Problem $P\lceil_p$ definiert durch:

- $I_{P\lceil_p} := \{x \in I_P \,,\, \max(x) \leq p(|x|)\}$
- $S_{P\lceil_p}(x) := S_P(x) \quad$ für $x \in I_{P\lceil_p}$
- $m_{P\lceil_p}(x, y) := m_P(x, y) \quad$ für $x \in I_{P\lceil_p}$ und $y \in S_{P\lceil_p}(x)$

P heißt stark **NP**-schwer, wenn $P\lceil_p$ **NP**-schwer ist.

Proposition

$\mathbf{P} \neq \mathbf{NP} \implies$
kein stark **NP**-schweres Problem ist pseudopolynomiell.