



Informatik III

WS 03/04

Prof. Dr. Dorothea Wagner

`dwagner@ira.uka.de`

Kapitel 4 : Komplexitätsklassen

Sprachen, Probleme, Zeitkomplexität

Beispiel: Traveling Salesman Problem (TSP)

Gegeben sei ein vollständiger Graph $G = (V, E)$, d.h.

$$V := \{1, \dots, n\}, \quad E := \{\{u, v\} \mid u, v \in V, u \neq v\},$$

sowie eine Längenfunktion $c: E \rightarrow \mathbb{Z}^+$.

- (1) Gesucht ist eine Tour (Rundreise), die alle Elemente aus V enthält und minimale Gesamtlänge unter allen solchen Touren hat. Gesucht ist also eine Permutation π auf V , so dass

$$\sum_{i=1}^{n-1} c(\pi(i), \pi(i+1)) + c(\pi(n), \pi(1)) \quad \text{minimal ist.}$$

Dies ist ein **Optimierungsproblem**.

Sprachen, Probleme, Zeitkomplexität

Eine „etwas schwächere“ Variante wäre:

(2) Gesucht ist die Länge einer minimalen Tour.

Diese Variante ist insofern „schwächer“, als mit (1) auch (2) lösbar ist. Noch „schwächer“ ist:

(3) Gegeben sei zusätzlich zu G und c auch ein Parameter $k \in \mathbb{Z}^+$. Die Frage ist nun: „Gibt es eine Tour, deren Länge höchstens k ist?“

Offensichtlich können wir mit (1) beziehungsweise (2) auch (3) lösen. (3) ist das zugehörige **Entscheidungsproblem** zu (1).

Sprachen, Probleme, Zeitkomplexität

4.1 Definition

*Ein **Kodierungsschema** s ordnet jedem Problembeispiel eines Problems eine Zeichenkette oder Kodierung über einem Alphabet Σ zu. Die **Inputlänge** eines Problembeispiels ist die Anzahl der Symbole seiner Kodierung. Dabei gibt es natürlich verschiedene Kodierungsschemata für ein bestimmtes Problem.*

4.2 Definition

Zwei Kodierungsschemata s_1, s_2 heißen **äquivalent** bezüglich eines Problems Π , falls es Polynome p_1, p_2 gibt, so dass gilt:

$$(|s_1(I)| = n \Rightarrow |s_2(I)| \leq p_2(n))$$

und

$$(|s_2(I)| = m \Rightarrow |s_1(I)| \leq p_1(m))$$

für alle Problembeispiele I von Π .

Sprachen, Probleme, Zeitkomplexität

*Ein Entscheidungsproblem Π können wir als Klasse von Problembeispielen D_{Π} auffassen. Eine Teilmenge dieser Klasse ist $J_{\Pi} \subseteq D_{\Pi}$, die Klasse der **Ja-Beispiele**, d.h. die Problembeispiele deren Antwort „Ja“ ist. Der Rest der Klasse $N_{\Pi} \subseteq D_{\Pi}$ ist die Klasse der **Nein-Beispiele**.*

Sprachen, Probleme, Zeitkomplexität

4.3 Definition

Die zu einem Problem Π und einem Kodierungsschema s zugehörige **Sprache** ist

$$L[\Pi, s] := \left\{ x \in \Sigma^* \mid \begin{array}{l} \Sigma \text{ ist das Alphabet zu } s \text{ und } x \text{ ist} \\ \text{Kodierung eines Ja-Beispiels } I \\ \text{von } \Pi \text{ unter } s, \text{ d.h. } I \in J_{\Pi} \end{array} \right\}$$

4.4 Definition

*Eine deterministische Turing-Maschine \mathcal{M} **löst** ein Entscheidungsproblem Π unter einem Kodierungsschema s , falls \mathcal{M} bei jeder Eingabe über dem Eingabe-Alphabet in einem Endzustand endet und $L_{\mathcal{M}} = L[\Pi, s]$ ist.*

Sprachen, Probleme, Zeitkomplexität

4.5 Definition

Für eine deterministische Turing-Maschine \mathcal{M} , die für alle Eingaben über dem Eingabe-Alphabet Σ hält, ist die **Zeitkomplexitätsfunktion** $T_{\mathcal{M}}: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ definiert durch

$$T_{\mathcal{M}}(n) = \max \left\{ m \mid \begin{array}{l} \text{es gibt eine Eingabe } x \in \Sigma^* \text{ mit} \\ |x| = n, \text{ so dass die Berechnung von} \\ \mathcal{M} \text{ bei Eingabe } x \text{ } m \text{ Berechnungs-} \\ \text{schritte (Übergänge) benötigt, bis} \\ \text{ein Endzustand erreicht wird} \end{array} \right.$$

4.6 Definition

Die Klasse \mathcal{P} ist die Menge aller Sprachen L (Probleme), für die eine deterministische Turing-Maschine existiert, deren Zeitkomplexitätsfunktion polynomial ist, d.h. es existiert ein Polynom p mit

$$T_{\mathcal{M}}(n) \leq p(n).$$

4.7 Satz

Falls es einen Algorithmus gibt, der das Entscheidungsproblem des TSP in polynomialer Zeit löst, so gibt es auch einen Algorithmus, der das Optimierungsproblem in polynomialer Zeit löst.

Nichtdeterministische TMs und die Klasse \mathcal{NP}

4.8 Definition

Die **Zeitkomplexitätsfunktion** $T_{\mathcal{M}}: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ einer nichtdeterministischen Turing-Maschine \mathcal{M} ist definiert durch

$$T_{\mathcal{M}}(n) := \max \left(\left\{ 1 \right\} \cup \left\{ m \mid \begin{array}{l} \text{es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n, \\ \text{so, dass die minimale Anzahl von} \\ \text{Schritten, die } \mathcal{M} \text{ benötigt, um} \\ x \text{ zu akzeptieren, } m \text{ ist} \end{array} \right. \right)$$

Nichtdeterministische TMs und die Klasse \mathcal{NP}

4.9 Definition

*Die Klasse \mathcal{NP} ist die Menge aller Sprachen L , für die es eine nichtdeterministische Turing-Maschine gibt, deren Zeitkomplexitätsfunktion polynomial beschränkt ist. (\mathcal{NP} steht für **nichtdeterministisch polynomial**.)*

\mathcal{NP} -vollständige Probleme

4.10 Definition

Eine **polynomiale Transformation** einer Sprache

$L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f: \Sigma_1^* \rightarrow \Sigma_2^*$ mit den Eigenschaften:

1. es existiert eine polynomiale deterministische Turing-Maschine, die f berechnet;
2. für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.

Wir schreiben dann $L_1 \propto L_2$ (L_1 ist polynomial transformierbar in L_2).

\mathcal{NP} -vollständige Probleme

4.11 Definition

Eine Sprache L heißt **\mathcal{NP} -vollständig**, falls gilt:

1. $L \in \mathcal{NP}$ und
2. für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$.

\mathcal{NP} -vollständige Probleme

4.12 Lemma

\propto ist transitiv, d.h. aus $L_1 \propto L_2$ und $L_2 \propto L_3$ folgt $L_1 \propto L_3$.

\mathcal{NP} -vollständige Probleme

4.13 Korollar

Falls $L_1, L_2 \in \mathcal{NP}$, $L_1 \propto L_2$ und L_1 \mathcal{NP} -vollständig, dann ist auch L_2 \mathcal{NP} -vollständig.

\mathcal{NP} -vollständige Probleme

Definition von SAT

Sei $U = \{u_1, \dots, u_m\}$ eine Menge von booleschen Variablen ($u_i, \overline{u_i}$ heißen Literale). Eine Wahrheitsbelegung für U ist eine Funktion $t: U \rightarrow \{\text{wahr}, \text{falsch}\}$. Eine **Klausel** ist ein Boole'scher Ausdruck der Form

$$y_1 \vee \dots \vee y_s$$

mit

$$y_i \in \underbrace{\{u_1, \dots, u_n\} \cup \{\overline{u_1}, \dots, \overline{u_m}\}}_{\text{Literalmenge}} \cup \{\text{wahr}, \text{falsch}\}$$

\mathcal{NP} -vollständige Probleme

Dann ist SAT wie folgt definiert:

Gegeben: Menge U von Variablen, Menge C von Klauseln über U

Frage: Existiert eine Wahrheitsbelegung von U , so dass C erfüllt wird, d.h. dass alle Klauseln aus C den Wahrheitswert `wahr` annehmen?

\mathcal{NP} -vollständige Probleme

4.14 Satz (von Steven Cook 1971)

SAT ist \mathcal{NP} -vollständig.

\mathcal{NP} -vollständige Probleme

Problem 3SAT

Gegeben: Menge U von Variablen, Menge C von Klauseln über U , wobei jede Klausel genau *drei* Literale enthält.

Frage: Existiert eine erfüllende Wahrheitsbelegung für C ?

\mathcal{NP} -vollständige Probleme

4.15 Satz

Das Problem 3SAT ist \mathcal{NP} -vollständig.

\mathcal{NP} -vollständige Probleme

4.16 Definition

Eine **Clique** ist eine Menge von Knoten, deren knoteninduzierter Subgraph vollständig ist.

Problem CLIQUE

Gegeben: Graph $G = (V, E)$ und ein Parameter $K \leq |V|$.

Frage: Gibt es in G eine Clique der Größe mindestens K ?

\mathcal{NP} -vollständige Probleme

4.17 Satz

Das CLIQUE-Problem ist \mathcal{NP} -vollständig.

\mathcal{NP} -vollständige Probleme

Problem 2SAT

Gegeben: Menge U von Variablen, Menge C von Klauseln über U , wobei jede Klausel genau zwei Literale enthält.

Frage: Existiert eine erfüllende Wahrheitsbelegung für C ?

\mathcal{NP} -vollständige Probleme

Problem Max2SAT

Gegeben: Menge U von Variablen, Menge C von Klauseln über U , wobei jede Klausel genau zwei Literale enthält und eine Zahl $K \in \mathbb{N}$.

Frage: Existiert eine Wahrheitsbelegung, die mindestens K Klauseln erfüllt?

\mathcal{NP} -vollständige Probleme

Problem Subgraphisomorphie

Gegeben: Graphen $G = (V, E)$ und $H = (V', E')$ mit
 $|V'| < |V|$

Frage: Gibt es eine Menge $U \subseteq V$ mit $|U| = |V'|$ und
eine bijektive Abbildung $\text{Iso}: V' \rightarrow U$, so dass
für alle $x, y \in V'$ gilt:

$$\{x, y\} \in E' \iff \{\text{Iso}(x), \text{Iso}(y)\} \in E$$

\mathcal{NP} -vollständige Probleme

Problem COLOR

Gegeben: ein Graph $G = (V, E)$ und ein Parameter $K \in \mathbb{N}$.

Frage: Gibt es eine Knotenfärbung von G mit höchstens K Farben, so dass je zwei adjazente Knoten verschiedene Farben besitzen?

3COLOR bezeichnet das Problem COLOR mit festem Parameter $K = 3$.

\mathcal{NP} -vollständige Probleme

4.18 Satz

3COLOR ist \mathcal{NP} -vollständig.

\mathcal{NP} -vollständige Probleme

Problem EXACT COVER

Gegeben: eine endliche Menge X und eine Familie \mathcal{S} von Teilmengen von X .

Frage: Existiert eine Menge $\mathcal{S}' \subseteq \mathcal{S}$, so dass jedes Element aus X in genau einer Menge aus \mathcal{S}' liegt?

\mathcal{NP} -vollständige Probleme

4.19 Satz

EXACT COVER ist \mathcal{NP} -vollständig.

\mathcal{NP} -vollständige Probleme

Problem SUBSET SUM

Gegeben: eine endliche Menge M , eine Gewichtsfunktion
 $w : M \rightarrow \mathbb{N}_0$ und $K \in \mathbb{N}_0$.

Frage: Existiert eine Teilmenge $M' \subseteq M$ mit

$$\sum_{a \in M'} w(a) = K ?$$

\mathcal{NP} -vollständige Probleme

4.20 Satz

SUBSET SUM ist \mathcal{NP} -vollständig.

\mathcal{NP} -vollständige Probleme

Problem PARTITION

Gegeben: eine endliche Menge M und eine Gewichtsfunktion $w : M \rightarrow \mathbb{N}_0$.

Frage: Existiert eine Teilmenge $M' \subseteq M$ mit

$$\sum_{a \in M'} w(a) = \sum_{a \in M \setminus M'} w(a) ?$$

\mathcal{NP} -vollständige Probleme

4.21 Korollar

PARTITION ist \mathcal{NP} -vollständig.

\mathcal{NP} -vollständige Probleme

Problem KNAPSACK

Gegeben: eine endliche Menge M , eine Gewichtsfunktion $w : M \rightarrow \mathbb{N}_0$, eine Kostenfunktion $c : M \rightarrow \mathbb{N}_0$ und $W, C \in \mathbb{N}_0$.

Frage: Existiert eine Teilmenge $M' \subseteq M$ mit

$$\sum_{a \in M'} w(a) \leq W \text{ und}$$

$$\sum_{a \in M'} c(a) \geq C ?$$

\mathcal{NP} -vollständige Probleme

4.22 Korollar

KNAPSACK ist \mathcal{NP} -vollständig.

4.23 Lemma

Ein beliebiges Beispiel m, w, c, W, C für KNAPSACK kann in $\mathcal{O}(|M| \cdot W)$ entschieden werden.

\mathcal{NP} -vollständige Probleme

4.24 Lemma

Sei L \mathcal{NP} -vollständig, dann gilt:

1. $L \in \mathcal{P} \implies \mathcal{P} = \mathcal{NP}$
2. $L \notin \mathcal{P}$, so gilt für alle \mathcal{NP} -vollständigen Sprachen L' , dass $L' \notin \mathcal{P}$ gilt.

Komplementsprachen

4.25 Definition

Die Klasse \mathcal{NPC} sei die Klasse der \mathcal{NP} -vollständigen Sprachen/Probleme. (\mathcal{NP} -complete). Die Klasse \mathcal{NPI} ist definiert durch $\mathcal{NPI} := \mathcal{NP} \setminus (\mathcal{P} \cup \mathcal{NPC})$. Die Klasse $\mathbf{co} - \mathcal{P}$ ist die Klasse aller Sprachen $\Sigma^* \setminus L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{P}$ (die Klasse der Komplementsprachen). Die Klasse $\mathbf{co} - \mathcal{NP}$ ist die Klasse aller Sprachen $\Sigma^* \setminus L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{NP}$.

Komplementsprachen

4.26 Satz (Ladner (1975))

Falls $\mathcal{P} \neq \mathcal{NP}$, so folgt $\mathcal{NPI} \neq \emptyset$.

Komplementsprachen

Problem co-TSP

Gegeben: Graph $G = (V, E)$, $c: E \rightarrow \mathbb{Z}^+$ und ein Parameter K .

Frage: Gibt es *keine* Tour der Länge $\leq K$?

Komplementsprachen

4.27 Lemma

Falls L \mathcal{NP} -vollständig ist und $L \in \text{co} - \mathcal{NP}$, so ist $\mathcal{NP} = \text{co} - \mathcal{NP}$.

Komplementsprachen

Problem Graphenisomorphie

Gegeben: Graphen $G = (V, E)$ und $H = (V', E')$ mit $|V| = |V'|$.

Frage: Sind G und H isomorph, d.h. existiert eine bijektive Abbildung

$$\text{Iso}: V' \rightarrow V$$

mit

$$\{x, y\} \in E' \iff \{\text{Iso}(x), \text{Iso}(y)\} \in E?$$

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

4.28 Definition

Ein **Suchproblem** Π wird beschrieben durch

1. die Menge der Problembeispiele D_Π und
2. für $I \in D_\Pi$ die Menge $S_\Pi(I)$ aller Lösungen von I .

Die „Lösung“ eines Suchproblems besteht in der Angabe einer Lösung aus $S_\Pi(I)$ für ein Problembeispiel $I \in D_\Pi$ mit $S_\Pi(I) \neq \emptyset$ und „Nein“ sonst.

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

Problem TSP–Suchproblem

Gegeben: Graph $G = (V, E)$ vollständig und gewichtet mit
 $c: E \rightarrow \mathbb{R}$.

Frage: Gib eine optimale Tour zu G bezüglich c an.

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

Problem Hamilton–Kreis (Suchproblem)

Gegeben: Ein ungerichteter, ungewichteter Graph

$$G = (V, E).$$

Frage: Gib einen Hamilton–Kreis in G an, falls einer existiert. Ein Hamilton–Kreis ist dabei eine Permutation π auf V , so dass $\{\pi(n), \pi(1)\} \in E$ und $\{\pi(i), \pi(i + 1)\} \in E$ für $1 \leq i \leq n - 1$ ist.

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

4.29 Definition

Ein **Aufzählungsproblem** Π ist gegeben durch

1. die Menge der Problembeispiele D_Π und
2. für $I \in D_\Pi$ die Menge $S_\Pi(I)$ aller Lösungen von I .

Die Lösung eines Aufzählungsproblem Π besteht in der Angabe der Kardinalität von $S_\Pi(I)$, d.h. $|S_\Pi(I)|$.

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

Problem Hamilton–Kreis (Aufzählungsproblem)

Gegeben: ein ungerichteter, ungewichteter Graph
 $G = (V, E)$.

Frage: Wieviele Hamilton–Kreise gibt es in G ?

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

4.30 Definition

Eine Funktion $f : \Sigma^* \rightarrow \Sigma^*$ **realisiert** eine Relation R , wenn für alle $x \in \Sigma^*$ gilt:

$$f(x) = \begin{cases} y & \text{falls es ein } y \in \Sigma^* \setminus \{\varepsilon\} \text{ gibt mit } (x, y) \in R \\ \varepsilon & \text{sonst} \end{cases}$$

Ein Algorithmus **löst** das durch R_{Π} beschriebene Suchproblem Π , wenn er eine Funktion berechnet, die R_{Π} realisiert.

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

4.31 Definition

Eine **Orakel-Turing-Maschine** zum Orakel $G : \Sigma^* \rightarrow \Sigma^*$ ist eine deterministische Turing-Maschine mit einem ausgezeichnetem **Orakelband** und zwei zusätzlichen Zuständen q_f und q_a . Dabei ist q_f der **Fragezustand** und q_a der **Antwortzustand** des Orakels. Die Arbeitsweise ist in allen Zuständen $q \neq q_f$ wie bei der normalen Turing-Maschine. Wenn der Zustand q_f angenommen wird, der Kopf sich auf Position i der Orakelbandes befindet und der Inhalt des Orakelbandes auf Position $1, \dots, i$ das Wort $y = y_1 \dots y_i$ ist, so ist der Übergang:

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

1. Fehlermeldung, falls $y \notin \Sigma^*$
2. in einem Schritt wird y auf dem Orakelband gelöscht und $g(y)$ auf die Positionen $1, \dots, |g(y)|$ des Orakelbandes geschrieben. Der Kopf des Orakelbandes springt auf Position 1 und der Folgezustand ist q_α .

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

4.32 Definition

Seien R, R' Relationen über Σ^* . Eine **Turing-Reduktion** \propto_T von R auf R' ($R \propto_T R'$), ist eine Orakel-Turing-Maschine \mathcal{M} , deren Orakel die Relation R' realisiert und die selber in polynomialer Zeit die Funktion f berechnet, die R realisiert.

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

4.33 Definition

Ein Suchproblem Π heißt \mathcal{NP} -schwer, falls es eine \mathcal{NP} -vollständige Sprache L gibt mit $L \propto_T \Pi$.

\mathcal{NP} -vollständige Probleme

Problem INTEGER PROGRAMMING

Gegeben: $a_{ij} \in \mathbb{N}_0$, $b_i, c_j \in \mathbb{N}_0$, $1 \leq i \leq m$, $1 \leq j \leq n$,
 $B \in \mathbb{N}_0$.

Frage: Existieren $x_1, \dots, x_n \in \mathbb{N}_0$, so dass

$$\sum_{j=1}^n c_j \cdot x_j = B \text{ und}$$

$$\underbrace{\sum_{j=1}^n a_{ij} \cdot x_j}_{A \cdot \bar{x} \leq \bar{b}} \leq b_i \text{ für } 1 \leq i \leq m ?$$

\mathcal{NP} -vollständige Probleme

4.34 Korollar

INTEGER PROGRAMMING ist \mathcal{NP} -schwer.

Weitere Komplexitätsklassen über \mathcal{NP} hinaus

4.35 Lemma

Falls L \mathcal{NP} -schwer ist, so ist auch L^c \mathcal{NP} -schwer. D.h. die Klasse der \mathcal{NP} -schweren Probleme ist bezüglich Komplementbildung abgeschlossen.

Approximationsalgorithmen für Optimierungsprobleme

4.36 Definition

Sei Π ein Optimierungsproblem. Ein polynomialer Algorithmus \mathcal{A} , der für jedes $I \in D_{\Pi}$ einen Wert $\mathcal{A}(I)$ liefert, mit

$$|\text{OPT}(I) - \mathcal{A}(I)| \leq K$$

und $K \in \mathbb{N}_0$ konstant, heißt **Approximationsalgorithmus mit Differenzengarantie** oder **absoluter Approximationsalgorithmus**.

Approximationsalgorithmen für Optimierungsprobleme

4.37 Satz

Falls $\mathcal{P} \neq \mathcal{NP}$, so gibt es keinen absoluten Approximationsalgorithmus \mathcal{A} für KNAPSACK.

Approximationsalgorithmen für Optimierungsprobleme

4.38 Satz

Falls $\mathcal{P} \neq \mathcal{NP}$, so gibt es keinen absoluten Approximationsalgorithmus \mathcal{A} für CLIQUE.

Approximationsalgorithmen für Optimierungsprobleme

4.39 Definition

Sei Π ein Optimierungsproblem. Ein polynomialer Algorithmus \mathcal{A} , der für jedes $I \in D_{\Pi}$ einen Wert $\mathcal{A}(I)$ liefert mit $R_{\mathcal{A}}(I) \leq K$, wobei $K \geq 1$ eine Konstante, und

$$\mathcal{R}_{\mathcal{A}}(I) := \begin{cases} \frac{\mathcal{A}(I)}{\text{OPT}(I)} & \text{falls } \Pi \text{ Minimierungsproblem} \\ \frac{\text{OPT}(I)}{\mathcal{A}(I)} & \text{falls } \Pi \text{ Maximierungsproblem} \end{cases}$$

heißt **Approximationsalgorithmus mit relativer Gütegarantie**. \mathcal{A} heißt **ε -approximativ**, falls $\mathcal{R}_{\mathcal{A}}(I) \leq 1 + \varepsilon$ für alle $I \in D_{\Pi}$.

4.40 Satz

Der Greedy-Algorithmus \mathcal{A} für KNAPSACK erfüllt $\mathcal{R}_{\mathcal{A}}(I) \leq 2$ für alle Instanzen I .

Approximationsalgorithmen für Optimierungsprobleme

4.41 Definition

Zu einem *polynomialen Approximationsalgorithmus* \mathcal{A} sei

$$\mathcal{R}_{\mathcal{A}}^{\infty} := \inf \left\{ r \geq 1 \mid \begin{array}{l} \text{es gibt ein } N_0 > 0, \text{ so dass} \\ \mathcal{R}_{\mathcal{A}}(I) \leq r \text{ für alle } I \text{ mit} \\ \text{OPT}(I) \geq N_0 \end{array} \right\}$$

Approximationsalgorithmen für Optimierungsprobleme

4.42 Satz

Falls $\mathcal{P} \neq \mathcal{NP}$, dann existiert kein relativer

Approximationsalgorithmus \mathcal{A} für COLOR mit $\mathcal{R}_{\mathcal{A}}^{\infty} \leq \frac{4}{3}$.

Approximationsalgorithmen für Optimierungsprobleme

4.43 Definition

Ein (polynomiales) **Approximationsschema (PAS)** für ein Optimierungsproblem Π ist eine Familie von Algorithmen $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$, so dass \mathcal{A}_ε ein ε -approximierender Algorithmus ist (d.h. $\mathcal{R}_{\mathcal{A}_\varepsilon} \leq 1 + \varepsilon$) für alle $\varepsilon > 0$. Dabei bedeutet polynomial wie üblich polynomial in der Größe des Inputs I .

Ein Approximationsschema $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$ heißt **vollpolynomial (FPAS)** falls seine Laufzeit zudem polynomial in $\frac{1}{\varepsilon}$ ist.

Approximationsalgorithmen für Optimierungsprobleme

4.44 Satz

Sei Π ein \mathcal{NP} -schweres Optimierungsproblem mit:

1. $\text{OPT}(I) \in \mathbb{N}$ für alle $I \in D_{\Pi}$, und
2. es existiert ein Polynom q mit $\text{OPT}(I) < q(\langle I \rangle)$ für alle $I \in D_{\Pi}$ wobei $\langle I \rangle$ die Inputlänge von I ist.

Falls $\mathcal{P} \neq \mathcal{NP}$, so gibt es kein FPAS $\{\mathcal{A}_{\varepsilon} \mid \varepsilon > 0\}$ für Π .

Approximationsalgorithmen für Optimierungsprobleme

Problem Spezielles KNAPSACK

Gegeben: Eine Menge von „Teilen“ $M = \{1, \dots, n\}$, Kosten $c_1, \dots, c_n \in \mathbb{N}_0$ und Gewichten $w_1, \dots, w_n \in \mathbb{N}$ sowie ein Gesamtgewicht $W \in \mathbb{N}$.

Frage: Gib eine Teilmenge M' von M an, so dass

$$\sum_{i \in M'} w_i \leq W \text{ und } \sum_{i \in M'} c_i \text{ maximal ist.}$$

Approximationsalgorithmen für Optimierungsprobleme

4.45 Satz

$\mathcal{R}_{\mathcal{A}_\varepsilon}(I) \leq 1 + \varepsilon$ für alle $I \in D_\Pi$ und die Laufzeit von \mathcal{A}_ε ist in $\mathcal{O}(n^3 \cdot \frac{1}{\varepsilon})$ für alle $\varepsilon > 0$, d.h. $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$ ist ein FPAS für das spezielle KNAPSACK.

Approximationsalgorithmen für Optimierungsprobleme

4.46 Satz

Sei Π ein Optimierungsproblem für das gilt:

- 1. $\text{OPT}(I) \in \mathbb{N}$ für alle $I \in D_{\Pi}$*
- 2. es existiert ein Polynom q mit
 $\text{OPT}(I) \leq q(\langle I \rangle + \max \#(I))$ ($\max \#(I)$ ist die
größte in I vorkommende Zahl)*

Falls Π ein FPAS hat, so hat es einen pseudopolynomialen optimalen Algorithmus.