

Beweis von Lemma B:

Benutze: $\log z \geq 1 - z^{-1}$ für $z \geq 1$ (*).

$$\left(\Leftrightarrow z \log z \geq z - 1 \Leftrightarrow z^z \geq \frac{1}{2} \cdot 2^z \Leftrightarrow \left(\frac{z}{2}\right)^z \geq \frac{1}{2}\right).$$

$$\begin{aligned} k &= \lceil \log_b r^* \rceil = \left\lceil \frac{\log r^*}{\log b} \right\rceil \leq \frac{\log r^*}{\log b} + 1 \stackrel{(*)}{\leq} \frac{\log r^*}{1 - 1/b} + 1 \\ &= \frac{b \cdot \log r^*}{b - 1} + \frac{b - 1}{b - 1} \stackrel{b < r^*}{<} \frac{r^* \log r^* + r^* - 1}{b - 1} \stackrel{\text{Def. } \beta}{=} \frac{1}{b - 1} \cdot \frac{\beta \cdot \varepsilon}{2 \cdot (1 + \varepsilon)}. \end{aligned}$$

Also folgt:

$$\frac{b - 1}{\beta} < \frac{\varepsilon}{2k(1 + \varepsilon)}$$

Auflösen von $b = 1 + \beta(r - 1)$ nach r , letzte Ungleichung:

$$r = \frac{b - 1}{\beta} + 1 < \frac{\varepsilon}{2k(1 + \varepsilon)} + 1 = \frac{\varepsilon + 2k(1 + \varepsilon)}{2k(1 + \varepsilon)}.$$

Hatten (letzte Folie):

$$r < \frac{\varepsilon + 2k(1 + \varepsilon)}{2k(1 + \varepsilon)}.$$

Damit folgt:

$$\begin{aligned} 1 - 2k \cdot (1 - 1/r) &> 1 - 2k \cdot \left(1 - \frac{2k(1 + \varepsilon)}{\varepsilon + 2k(1 + \varepsilon)}\right) \\ &= 1 - 2k \cdot \frac{\varepsilon}{\varepsilon + 2k(1 + \varepsilon)} = 1 - \frac{\varepsilon}{1 + \varepsilon + \varepsilon/(2k)} \\ &= \frac{1 + \varepsilon/(2k)}{1 + \varepsilon + \varepsilon/(2k)}. \end{aligned}$$

Also:

$$\frac{1}{1 - 2k \cdot (1 - 1/r)} < \frac{1 + \varepsilon + \varepsilon/(2k)}{1 + \varepsilon/(2k)} = 1 + \frac{\varepsilon}{1 + \varepsilon/(2k)} < 1 + \varepsilon. \quad \square$$

Neue Richtung:

Beweis von Lemma 12.4.2:

Ziel: beliebiges Problem A in MIN-APX $\xrightarrow{\text{PTAS}}$
geeignet konstruiertes Problem B in MAX-APX.

1. Idee: $v_A(x, s) \rightarrow v_B(x, s) := -v_A(x, s)$.

Problem: Lösungswerte müssen **positiv** sein.

2. Idee: b obere Schranke für $v_{A,\text{opt}}(x)$:
 $v_B(x, s) := b - v_A(x, s)$.

Problem: Falls b zu groß, alle neuen Lösungswerte $\approx b$.

Damit: Schlechte Lösungen für $A \rightarrow$ gute Lösungen für B .

Abhilfe: Obere Schranke adaptiv wählen, $b = b(x)$.

$A \in \text{MIN-APX}$: Es gibt Approximationsalgorithmus AL für A mit Worstcase-Approximationsgüte r^* .

O. B. d. A. (aufrunden): r^* ganzzahlig.

Sei $s^*(x)$ die von AL für Eingabe x berechnete Lösung.

Maximierungsproblem B :

- Eingaben für $B :=$ Eingaben für A .
- Für alle x : $S_B(x) := S_A(x)$ (Lösungsmengen gleich).
- **Neue Zielfunktion:** Für alle x und $s \in S_B(x)$:

$$v_B(x, s) := \max\{ 1, (r^* + 1) \cdot v_A(x, s^*(x)) - r^* \cdot v_A(x, s) \}.$$

$A \in \text{NPO}$: Lösungswerte ganzzahlig und positiv.

Damit auch $v_B(x, s)$ ganzzahlig und positiv.

Außerdem $v_B(x, s)$ in Polynomialzeit berechenbar.

Also zumindest $B \in \text{NPO}$.

$B \in \text{APX}$:

Benutze AL als Approximationsalgorithmus auch für B .

Beobachtungen:

(1) $v_B(x, s^*(x)) = v_A(x, s^*(x))$, denn:

$$\begin{aligned} v_B(x, s^*(x)) &= \max\{1, (r^* + 1) \cdot v_A(x, s^*(x)) - r^* \cdot v_A(x, s^*(x))\} \\ &= \max\{1, v_A(x, s^*(x))\} \stackrel{\text{Lsgs.-Werte in } \mathbb{N}}{=} v_A(x, s^*(x)). \end{aligned}$$

(2) $v_A(x, s^*(x)) = v_B(x, s^*(x)) \leq v_{B,\text{opt}}(x) \leq (r^* + 1) \cdot v_A(x, s^*(x))$.

Damit folgt insbesondere:

$$r_B(x, s^*(x)) = \frac{v_{B,\text{opt}}(x)}{v_B(x, s^*(x))} \leq r^* + 1.$$

\Rightarrow AL Approximationsalgo. für B mit Güte $\leq r^* + 1$.

$A \leq_{PTAS} B$:

- Für alle x : $f(x) := x$.
- $\alpha(\varepsilon) := \varepsilon/\beta$ mit $\beta := r^* + 1$, $\varepsilon > 0$ beliebig.

Definiere wieder $r := 1 + \alpha(\varepsilon)$. Dann:

$$1 + \varepsilon = 1 + \alpha(\varepsilon) \cdot \beta = 1 + \beta \cdot (r - 1).$$

Wollen g konstruieren, sodass für alle $s \in S_B(x)$:

$$r_B(x, s) \leq 1 + \alpha(\varepsilon) = r \Rightarrow$$

$$r_A(x, g(x, s, \varepsilon)) \leq 1 + \varepsilon = 1 + \beta(r - 1).$$

1. Fall: $v_B(x, s) = 1$.

In diesem Fall wähle $g(x, s, \varepsilon) := s^*(x)$.

Falls $v_A(x, s^*(x)) = 1$:

Lösungswerte aus $\mathbb{N} \Rightarrow s^*(x)$ optimale Lösung
und $r_A(x, g(x, s, \varepsilon)) = r_A(x, s^*(x)) = 1 \leq 1 + \varepsilon$.

Also sei $v_A(x, s^*(x)) > 1$, damit (wg. Ganzz.) $v_A(x, s^*(x)) \geq 2$.

$$r \geq r_B(x, s) = \frac{V_{B,\text{opt}}(X)}{V_B(X, s)} \stackrel{\text{Fall-Vor.}}{=} V_{B,\text{opt}}(X) \stackrel{\text{Beob. (2)}}{\geq} v_A(x, s^*(x)).$$

Es folgt:

$$\begin{aligned} r_A(x, g(x, s, \varepsilon)) - 1 &= r_A(x, s^*(x)) - 1 \leq r^* - 1 \\ &\leq (r^* + 1) \cdot (v_A(x, s^*(x)) - 1) \leq \beta \cdot (r - 1), \end{aligned}$$

Also: $r_A(x, g(x, s, \varepsilon)) \leq 1 + \beta \cdot (r - 1) = 1 + \varepsilon$.

2. Fall: $v_B(x, s) > 1$.

Wir wählen in diesem Fall $g(x, s, \varepsilon) := s$.

Vorüberlegung 1:

Wir setzen wie bisher voraus, dass

$$r_B(x, s) = \frac{V_{B,\text{opt}}(X)}{V_B(x, s)} \leq 1 + \alpha(\varepsilon) (= r), \text{ also}$$

$$V_B(x, s) \geq \frac{V_{B,\text{opt}}(X)}{1 + \alpha(\varepsilon)} \geq (1 - \alpha(\varepsilon)) \cdot V_{B,\text{opt}}(X). \quad (*)$$

Gemäß Definition von $v_B(x, s)$ für diesen Fall folgt:

$$V_B(x, s) = (r^* + 1) \cdot v_A(x, s^*(x)) - r^* \cdot v_A(x, s) \Rightarrow$$

$$v_A(x, s) = \frac{(r^* + 1) \cdot v_A(x, s^*(x)) - V_B(x, s)}{r^*}. \quad (**)$$

Vorüberlegung 2:

Sei $s_{A,\text{opt}}(x)$ eine Lösung von A mit dem minimalen Wert für Eingabe x , also $v_{A,\text{opt}}(x)$. Dann gilt:

$$v_{B,\text{opt}}(x) \geq v_B(x, s_{A,\text{opt}}(x))$$

Def. von v_B

$$\geq (r^* + 1) \cdot v_A(x, s^*(x)) - r^* \cdot v_A(x, s_{A,\text{opt}}(x))$$

$$= (r^* + 1) \cdot v_A(x, s^*(x)) - r^* \cdot v_{A,\text{opt}}(x).$$

Damit folgt:

$$(r^* + 1) \cdot v_A(x, s^*(x)) - v_{B,\text{opt}}(x) \leq r^* \cdot v_{A,\text{opt}}(x). \quad (***)$$

Jetzt Abschätzung von $v_A(x, s)$ durch $v_{A,\text{opt}}(x)$...

$$V_A(X, S) = \frac{(r^* + 1) \cdot v_A(X, s^*(X)) - v_B(X, S)}{r^*} \quad (**)$$

$$\begin{aligned} & \downarrow v_B(X, S) \geq (1 - \alpha(\varepsilon)) \cdot v_{B,\text{opt}}(X) \quad (*) \\ & \leq \frac{(r^* + 1) \cdot v_A(X, s^*(X)) - v_{B,\text{opt}}(X) + \alpha(\varepsilon) \cdot v_{B,\text{opt}}(X)}{r^*} \end{aligned}$$

$$\begin{aligned} & \downarrow (r^* + 1) \cdot v_A(X, s^*(X)) - v_{B,\text{opt}}(X) \leq r^* \cdot v_{A,\text{opt}}(X) \quad (***) \\ & \leq \frac{r^* \cdot v_{A,\text{opt}}(X) + \alpha(\varepsilon) \cdot v_{B,\text{opt}}(X)}{r^*} \end{aligned}$$

$$= v_{A,\text{opt}}(X) + \frac{\alpha(\varepsilon) \cdot v_{B,\text{opt}}(X)}{r^*}$$

$$\begin{aligned} & \downarrow v_{B,\text{opt}}(X) \leq (r^* + 1) \cdot v_A(X, s^*(X)), v_A(X, s^*(X)) \leq r^* \cdot v_{A,\text{opt}}(X) \\ & \leq v_{A,\text{opt}}(X) + \alpha(\varepsilon) \cdot (r^* + 1) \cdot v_{A,\text{opt}}(X) \end{aligned}$$

$$\begin{aligned} & \downarrow \alpha(\varepsilon) = \varepsilon/\beta = \varepsilon/(r^* + 1) \\ & = (1 + \varepsilon) \cdot v_{A,\text{opt}}(X). \end{aligned}$$

□

Fazit:

PCP-Theorie liefert die aktuell stärksten Werkzeuge, um Ergebnisse über die Grenzen von Approximationsalgorithmen zu beweisen:

- Nichtapproximierbarkeit mit konstanter oder sogar mit Eingabegröße wachsender Güte;
- Vollständigkeit für Klassen von Approximationsproblemen.

13. Platzkomplexität

Übersicht:

- Platz versus Zeit;
- Determinismus versus Nichtdeterminismus für Platzkomplexität;
- Platzkomplexität für randomisierte Algorithmen;
- Probleme, die mit logarithmischem Platz lösbar sind.

Teilweise in Kapitel 13 im Buch enthalten.

13.1 Grundlagen

Für deterministische TM M :

- **Speicherplatz für Eingabe x :**
Anzahl bei Rechnung auf x besuchter Speicherzellen.
- **(Worstcase-)Speicherplatz von M :**
Maximum des Speicherplatzes über alle Eingaben x .

Für nichtdeterministische / randomisierte TM M
auf nahe liegende Weise erweitern:

- **Speicherplatz für Eingabe x :**
Maximum über alle Rechenwege bzw. Zufallsbitstrings.
- **(Worstcase-)Speicherplatz von M :**
Maximum über alle Eingabe- und Zufallsbitstrings.

Deterministische Algorithmen benötigen für die meisten Probleme Zugriff auf die komplette Eingabe:
→ linearer Speicherplatz alleine dafür.

Konvention bei sublinearen Platzschranken:

- Eingabe auf separatem **Eingabeband**:
Band, das **nur lesbar ist**, Anfang und Ende der Eingabe markiert („\$ $x_1x_2 \dots x_n$ ϕ “)
- Ausgabe auf separatem **Ausgabeband**:
Einwegband (Kopf bewegt sich nach Schreiben eines Zeichens einen Schritt nach rechts), **nur beschreibbar**.

Dazu wie immer lesbares und beschreibbares Arbeitsband.
Nur der Speicher auf dem Arbeitsband wird gemessen.

Für Akzeptanz von Sprachen ohne Ausgabeband.

Motivation für sublineare Platzschranken:

- **Parallel Computation Thesis:**

Simulationen zwischen sequenziellen und parallelen Rechnermodellen liefern:

„Parallele Zeit entspricht sequenziellem Platz.“

Genauer: Polynomielle Verknüpfung der Maße.
Für konkrete Rechnermodelle bewiesen (später mehr).

- **Probleme für große Datenmengen:**

Z. B. Internet-Graph oder große Datenbank als Eingabe.
Dann linearer Speicherplatz nicht mehr okay.

Definition 13.1.1:

Für $s: \mathbb{N} \rightarrow \mathbb{R}$ enthält die Komplexitätsklasse $DSPACE(s(n))$ alle Entscheidungsprobleme, die von einer deterministischen TM mit Worstcase-Speicherplatz $\lceil s(n) \rceil$ für Eingaben der Länge n gelöst werden können. Analog $NSPACE(s(n))$ für nichtdeterministische TMs.

Wichtige Spezialfälle:

$$\begin{aligned} PSPACE &:= \bigcup_{k \in \mathbb{N}} DSPACE(n^k); \\ NPSPACE &:= \bigcup_{k \in \mathbb{N}} NSPACE(n^k); \\ LOGSPACE &:= L &:= DSPACE(\log n); \\ NLOGSPACE &:= NL &:= NSPACE(\log n). \end{aligned}$$

Randomisierte Platzkomplexitätsklassen später.

Proposition 13.1.2 (Bandkompression):

Für jede Funktion $s: \mathbb{N} \rightarrow \mathbb{R}$ und jedes $k \in \mathbb{N}$ gilt:

$$\text{DSPACE}(s(n)) = \text{DSPACE}(s(n)/k) \text{ und}$$

$$\text{NSPACE}(s(n)) = \text{NSPACE}(s(n)/k).$$

Konstante Faktoren bei Platzschranken also nicht wesentlich.

Beweisidee:

- Neues Arbeitsband mit k Spuren, jeweils k aufeinanderfolgende Zellen des ursprünglichen Bandes zusammenfassen.
- Aktive Spur (= alte Kopfposition innerhalb der neuen Zelle) im Zustand merken. □

Bezug zur Chomsky-Hierarchie (1/2)

Proposition 13.1.3: $DSPACE(0) = REG (= CH_3)$.

Beweisskizze:

„ \supseteq “: Trivial.

„ \subseteq “: Benutze dazu, dass **Zweiwege-DFAs**, d. h., DFAs, die ihren Eingabekopf in beide Richtungen bewegen dürfen, dieselbe Klasse von Sprachen erkennen wie gewöhnliche DFAs mit Einweg-Eingabeband.

Details: Siehe z. B. Hopcroft und Ullman. □

Bezug zur Chomsky-Hierarchie (2/2)

Satz 13.1.4: $\text{NSPACE}(n) = \text{CSL} (= \text{CH}_2)$.

CSL (*context sensitive languages*): Klasse der Sprachen, die von kontextsensitiven Grammatiken erzeugt werden:

Produktionen $u \rightarrow v$ außer $S \rightarrow \varepsilon$ erfüllen $|u| \leq |v|$.

Beweisskizze:

„ \supseteq “: Eingabe x . Stecke Bereich der Länge $|x|$ auf Arbeitsband ab. Rate Ableitung $S \rightarrow_* w$ auf diesem Bereich, d. h., Abbruch mit Verwerfen, falls erzeugtes Wort zu lang. Akzeptiere am Ende genau dann, wenn $w = x$.

„ \subseteq “: Simuliere Rechnung einer linear platzbeschränkten TM „rückwärts“. Produktionen bilden lokale Konfigurationsänderungen der TM „rückwärts“ nach. Details: Siehe wieder Hopcroft und Ullman. □

13.2 Platz versus Zeit

Proposition 13.2.1:

Jede $t(n)$ -zeitbeschränkte, deterministische oder nichtdeterministische TM benötigt höchstens Speicherplatz $t(n)$.

Beweis: In jedem Rechenschritt kann höchstens eine neue Speicherzelle aufgesucht werden. □

Folgerung: $P \subseteq PSPACE$, $NP \subseteq NPSPACE$.

(Okay, wir wissen schon viel mehr: Sogar $PH \subseteq PSPACE$.)

Was gilt in der umgekehrten Richtung?

Deterministische oder nichtdeterministische TM
mit folgender Einschränkung:

- TM für Sprachakzeptanz oder
- TM mit Ausgabe, für die jeder Rechenweg terminiert.

Beobachtung 1:

Für platzbeschränkte TM Schranke für Gesamtanzahl
verschiedener Konfigurationen bei fester Eingabelänge.

TM $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$,

Platzschranke $s(n)$, Eingabelänge n :

Konfiguration beschrieben durch Element aus

$$Q \times \Gamma^{s(n)} \times \{1, \dots, n\} \times \{1, \dots, s(n)\}$$

aktueller Zustand Inhalt des Arbeitsbandes Kopfposition für Eingabeband Kopfposition für Arbeitsband

Anzahl: $|Q| \cdot |\Gamma|^{s(n)} \cdot n \cdot s(n) \leq 2^{c(s(n)+\log n)}$, $c > 0$ Konstante.

Beachte: Über evtl. vorhandenes Ausgabeband wird in der Konfiguration nichts abgespeichert.

Beobachtung 2:

Falls Rechenweg mehr Schritte hat als Gesamtanzahl Konfigurationen: Konfiguration kommt mehrfach vor.

Länge von terminierenden Rechenwegen kann durch Gesamtanzahl Konfigurationen beschränkt werden, also höchstens $2^{c(s(n)+\log n)}$.

Modifikation der TM:

Zähle ausgeführte Rechenschritte mit, brich mit Verwerfen / Dummy-Ausgabe ab, sobald Zähler größer als $2^{c(s(n)+\log n)}$.

Bereich der Länge $c(s(n) + \log n)$ für Zähler muss abgesteckt werden können. Hinreichend: $s(n)$ platzkonstruierbar, d. h. $1^n \mapsto 1^{s(n)}$ in Platz $O(s(n))$ berechenbar.

Dann reichen Zeit $2^{O(s(n)+\log n)}$ und Platz $O(s(n) + \log n)$ insgesamt für modifizierte TM.

Satz 13.2.2:

Sei $s(n)$ platzkonstruierbar. Dann können deterministische, $s(n)$ -platzbeschränkte TMs, die Sprachen akzeptieren bzw. eine Ausgabe berechnen und immer anhalten, durch deterministische TMs simuliert werden, die mit Zeit $2^{O(s(n)+\log n)}$ und Platz $O(s(n) + \log n)$ auskommen. Analog für nichtdeterministische TMs.

Anwendungen:

- $L \subseteq P$.
- $PSPACE \subseteq EXP$.

$EXP := DTIME(2^{\text{poly}(n)})$, d. h., Entscheidungsprobleme, die von deterministischen TMs mit Rechenzeit $2^{\text{poly}(n)}$ lösbar sind.