Tutorium Berechenbarkeit und Komplexität

Tutorium 10 – Gruppe 1

Raimund Hensen

08.01.2019

NP und NP-schwer

Die vier Probleme A, B, C, D sind Entscheidungsprobleme, über die folgendes bekannt ist:

- $A \leq_p \mathrm{SAT} \leq_p B$
- $A \leq_p C$
- $B \leq_p \mathrm{SAT} \leq_p D$

Markieren Sie in der folgenden Tabelle die Aussagen, die wir mit Sicherheit wissen

(unabhängig davon, ob $P = NP$ oder $P \neq NP$ gilt):

	in NP	**NP-vollständig**	**NP-schwer**
A	×		
B	×	×	×
C			
D			×

Bin-Packing

Zeigen Sie für das BIN PACKING PROBLEM, dass, falls die Entscheidungsvariante in P ist, so kann auch die Optimierungsvariante in polynomialer Zeit gelöst werden.

Wir nehmen also an, dass die Entscheidungsvariante von BPP in P ist und konstruieren einen Polynomialzeit-Algorithmus $\mathcal{A}_{Opt}$, der eine optimale Lösung des BPP berechnet und dabei den Polynomialzeit-Algorithmus für die Entscheidungsvariante $\mathcal{A}_{Ent}$ als Unterprogramm benutzt.

Wir nutzen aus, dass man zwei Objekte x und y „zusammenkleben“ kann, indem man beide löscht und durch ein neues Objekt x' mit Gewicht $w(x') = w(x) + w(y)$ ersetzt. Damit wird erzwungen, dass x und y im gleichen Behälter landen. Falls sich dadurch der Wert der Lösung nicht ändert, so gibt es eine Verteilung, die bei gleichbleibender Qualität x und y dem gleichen Behälter zuordnet. Beachte, dass die Lösung durch das Zusammenkleben nicht besser werden kann.

Bin-Packing (Forts.)

Sei n die Anzahl an Elemente der BPP-Instanz. Durch eine binäre Suche können wir den Wert $b_{opt} \leq n$ der optimalen Lösung finden und damit b_{opt} in Polynomialzeit finden.

(*Hinweis: Lineare Suche ist bei diesem Problem auch möglich. Bei anderen Problemen mit in der Eingabe exponentiellen Suchraum (da Größe n in der Eingabe binär kodiert ist), geht dies nicht.*)

Der Algorithmus $\mathcal{A}_{Opt}$ zur Berechnung der optimalen Verteilung geht nun wie folgt vor:

- iteriere über alle Paare (x, y) von Objekten
- klebe das Paar (x, y) testweise zusammen und prüfe mit $\mathcal{A}_{Ent}$, ob es eine Verteilung der Objekte auf b_{opt} Behälter gibt.
  - Falls ja, bleiben x und y zusammengeklebt und der Algorithmus wird rekursiv weiter fortgesetzt, bis die Anzahl der Objekte b_{opt} ist.
  - Falls nein, wähle ein anderes Paar und „löse“ x und y wieder voneinander

Bin-Packing (Forts.)

Die auf diese Art erhaltene Verteilung ordnet jedem Behälter ein zusammengeklebtes Objekt zu. Zerlegt man diese Objekte wieder in seine Bestandteile, d.h. die einzelnen Objekte, aus denen es zusammengeklebt worden ist, erhält man die genaue Verteilung.

In jedem Rekursionsschritt wird die Eingabe um ein Objekt verringert; also werden, wenn n die Anzahl der ursprünglichen Objekte ist, $n - b_{opt}$ Rekursionen durchgeführt. Dabei benötigt jeder Rekursionsschritt maximal $n \cdot (n-1)$ viele Aufrufe von $\mathcal{A}_{Ent}$. Folglich ist die Laufzeit von $\mathcal{A}_{Opt}$ polynomiell in der Eingabelänge beschränkt.

NP-Vollständigkeit

Definition IndependentSet:

Eingabe: Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine unabhängige Knotenmenge $I \subseteq V$ mit $|I| \geq k$? (d.h. für alle $u, v \in I$ gilt $\{u, v\} \notin E$)

(a) Zeigen Sie IndependentSet $\in$ NP.

(b) Zeigen Sie Clique $\leq_p$ IndependentSet.

NP-Vollständigkeit

(a) Zeigen Sie $\textsc{IndependentSet} \in \mathrm{NP}$.

Sei $G = (V, E)$ und $k \in \mathbb{N}$ die Eingabe für $\textsc{IndependentSet}$.

Zertifikat: Kodiere eine Teilmenge $I \subseteq V$ als Binärstring der Länge $|V|$.

Verifizierer: Iteriere über alle Knotenpaare und prüfe ob für $u, v \in I$ immer $\{u, v\} \notin E$ folgt. Dies geschieht in polynomieller Zeit.

Der Verifizierer akzeptiert genau dann, wenn $(G, k) \in \textsc{IndependentSet}$. Somit gilt $\textsc{IndependentSet} \in \mathrm{NP}$.

NP-Vollständigkeit (Forts.)

(b) Zeigen Sie $\textsc{Clique} \leq_p \textsc{IndependentSet}$.

Konstruktion: Sei $G = (V, E)$ und $k \in \mathbb{N}$ die Eingabe für $\textsc{Clique}$. Konstruiere den Komplementgraphen

$$\overline{G} = (V, \overline{E}),$$

und verwende $(\overline{G}, k)$ als Eingabe für $\textsc{IndependentSet}$.

Die Konstruktion des Komplementgraphen ist in quadratischer Laufzeit möglich.

NP-Vollständigkeit (Forts.)

(b) Zeigen Sie $\textsc{Clique} \leq_p \textsc{IndependentSet}$.

Korrektheit:

„$\Rightarrow$“ Sei $(G, k) \in \textsc{Clique}$ und $C \subseteq V$, $|C| \geq k$ eine Clique in G.
Für $u, v \in C$, $u \neq v$ gilt $\{u, v\} \in E$, also $\{u, v\} \notin \overline{E}$.
Also ist C unabhängig in $\overline{G}$ und es gilt
$(\overline{G}, k) \in \textsc{IndependentSet}$.

„$\Leftarrow$“ Sei $(\overline{G}, k) \in \textsc{IndependentSet}$ und $I \subseteq V$, $|I| \geq k$ unabhängig in $\overline{G}$. Für $u, v \in I$, $u \neq v$ gilt $\{u, v\} \notin \overline{E}$, also $\{u, v\} \in E$.
Also ist I eine Clique in G und es gilt $(G, k) \in \textsc{Clique}$.

Somit gilt $\textsc{Clique} \leq_p \textsc{IndependentSet}$.

Zusammen mit $\mathrm{SAT} \leq_p \textsc{Clique}$ (VL) folgt
$\mathrm{SAT} \leq_p \textsc{IndependentSet}$, also ist $\textsc{IndependentSet}$ NP-schwer. Insgesamt ist $\textsc{IndependentSet}$ NP-vollständig.