Vorlesung 18
Ausblick: Algorithmen und Komplexität

Wdh.: NP-Vollständigkeit

Definition (NP-vollständig)

Ein Problem L heißt NP-vollständig (engl. NP-complete), falls gilt

1. $L \in \mathrm{NP}$, und
2. L ist NP-schwer.

Die Klasse der NP-vollständigen Probleme wird mit NPC bezeichnet.

Satz (Cook und Levin)

SAT ist NP-vollständig.

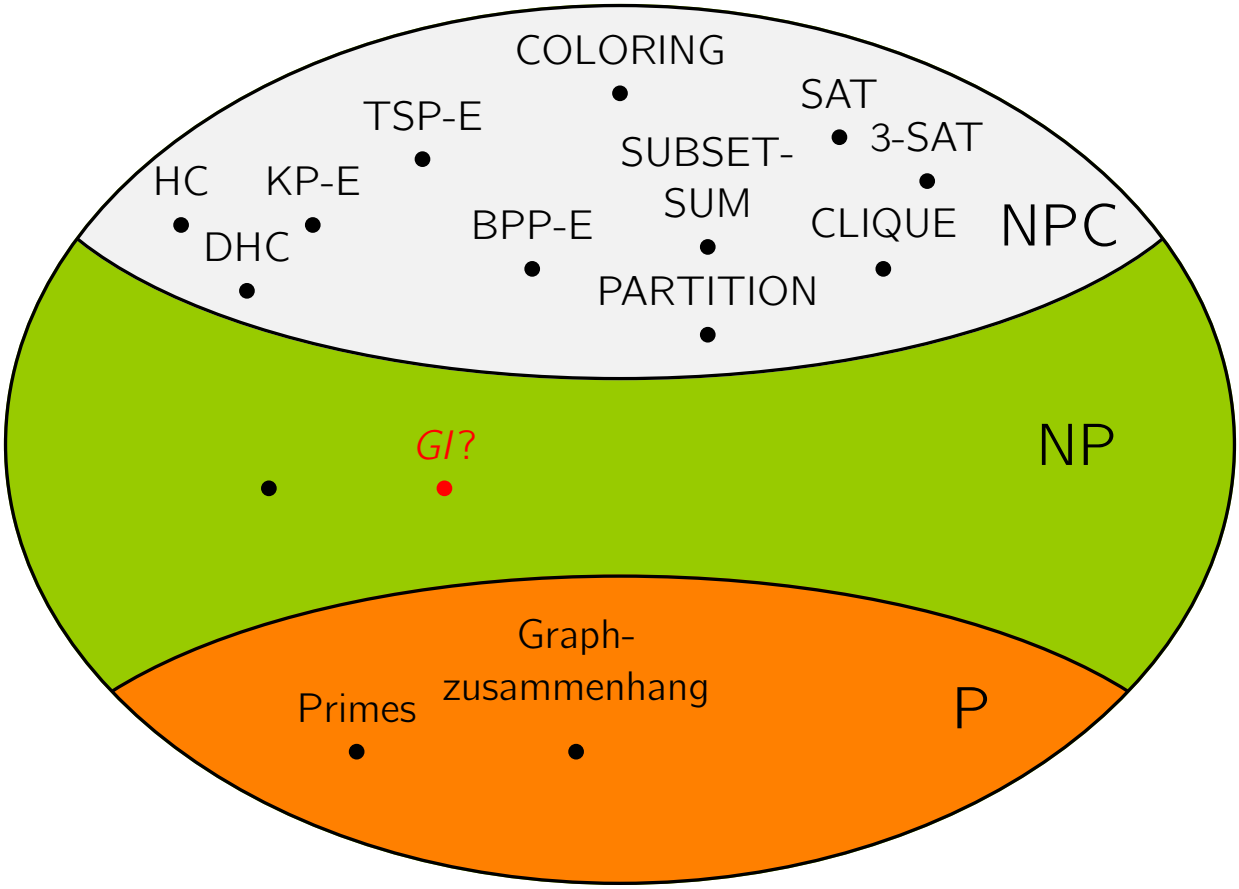
Lemma

3-SAT $\in$ NP und SAT $\leq_p$ 3-SAT.

Korollar

3-SAT ist NP-vollständig.

Wdh.: Die Komplexitätslandschaft



Warnung: Dieser Abbildung liegt die Annahme $P \neq NP$ zu Grunde.

Was tun mit NP-schweren Problemen?

Viele praxisrelevante Optmierungsprobleme sind NP-schwer, z.B. das Bin Packing Problem (BPP), das Rucksack Problem (KP) und das Traveling Salesperson Problem (TSP).

In der Praxis müssen die Probleme dennoch gelöst werden. Verschiedene Strategien können hier zum Erfolg führen; die wichtigsten sind:

- Approximationsalgorithmen
- Ausnutzen der Eingabestruktur durch spezielle Algorithmen
- Parametrisierte Algorithmen
- Randomisierte Algorithmen
- Heuristiken (ohne irgendwelche Garantien)

Ausnutzen der Eingabestruktur

Beispiel 1

- Straßennetze lassen sich durch Graphen modellieren, die (fast) planar sind.
- Es gibt effiziente Graphalgorithmen, die speziell auf diese planare Struktur ausnützen.

Beispiel 2

- Instanzen des TSP haben in der Praxis häufig eine metrische Struktur, d.h., die Distanzen sind symmetrisch und erfüllen die Dreiecksungleichung.
- Für das "metrische TSP" gibt es gute Approximationsalgorithmen.

Parametrische Analyse

In der Praxis wird es oft der Problemstellung nicht gerecht, die Effizienz eines Algorithmus allein in Abhängigkeit von der Eingabegröße zu messen. Eine verfeinerte Analyse, die andere Eingabeparameter berücksichtigt, kann zu besseren Resultaten führen.

Beispiel

Wir wollen eine Anfrage α in einer Datenbank $\mathcal{D}$ auswerten.

- Sei ℓ die Größe von α und m die Größe von $\mathcal{D}$.
  Die Eingabegröße ist dann $n = \ell + m$.
- Häufig ist m sehr groß und ℓ relativ klein. Ein Algorithmus mit einer Laufzeit von $2^\ell m$ kann deswegen durchaus gut sein, ein Algorithmen mit Laufzeiten wie m^ℓ oder gar $2^m \ell$ hingegen kaum.
- Analysieren wir die Algorithmen hingegen nur nach der Eingabegröße n, so wird dieser wichtige Unterschied nicht sichtbar.

Approximationsalgorithmen

Sei Π ein Optimierungsproblem. Für eine Instanz I von Π bezeichnen wir den optimalen Zielfunktionswert mit $opt(I)$.

- Ein α-Approximationsalgorithmus, $\alpha > 1$, für ein *Minimierungsproblem* Π berechnet für jede Instanz I von Π eine zulässige Lösung mit Zielfunktionswert höchstens $\alpha \cdot opt(I)$.
- Ein α-Approximationsalgorithmus, $\alpha < 1$, für ein *Maximierungsproblem* Π berechnet für jede Instanz I von Π eine zulässige Lösung mit Zielfunktionswert mindestens $\alpha \cdot opt(I)$.

Die Zahl α wird auch als Approximationsfaktor oder Approximationsgüte bezeichnet.

BPP – Approximationsalgorithmus

Zur Erinnerung:

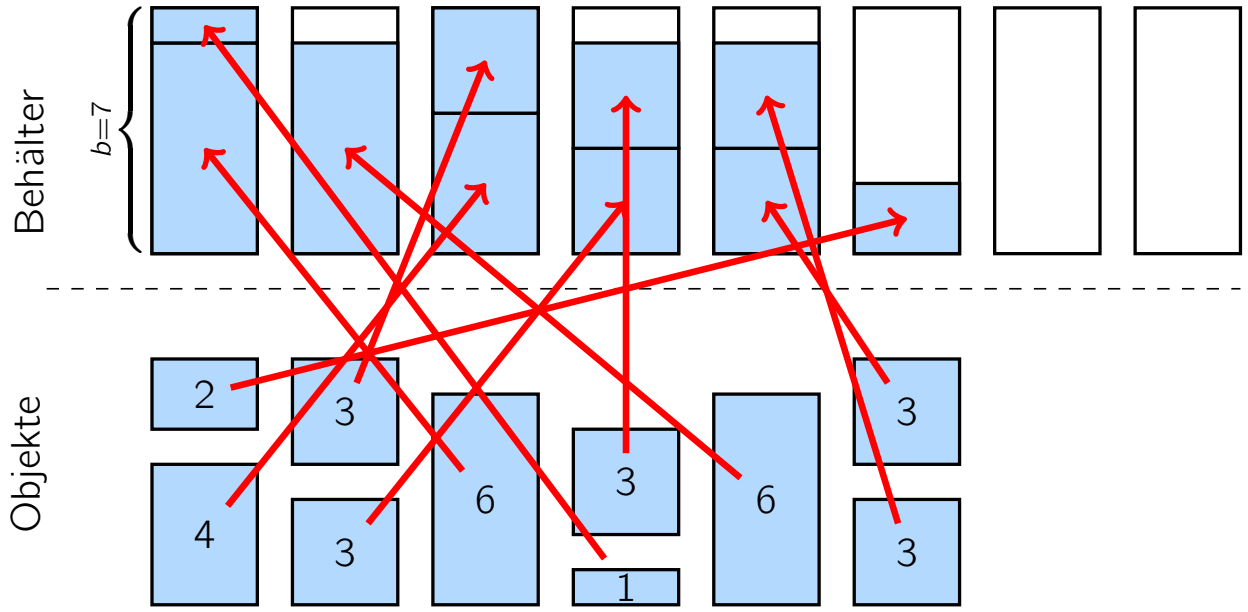
Beim Bin Packing Problem (BPP) suchen wir eine Verteilung von N Objekten mit Gewichten $w_1, \ldots, w_N$ auf eine möglichst kleine Anzahl von Behältern mit Gewichtskapazität jeweils b.

Satz

Für BPP gibt es einen effizienten 2-Approximationsalgorithmus.

Der Begriff effizienter Algorithmus wird als Synonym für Algorithmus mit polynomiell beschränkter Laufzeit verwendet.

Bin Packing Problem – Beispiel



Eine Lösung die $k = 6$ Behälter verwendet

BPP – Approximationsalgorithmus

Algorithmus FIRST FIT:

Initial seien alle Kisten *ungeöffnet*.

Betrachte die Objekte in beliebiger Reihenfolge:

- Wenn das gerade betrachtete Objekt in keine der geöffneten Kisten eingefügt werden kann, ohne die Kapazität zu überschreiten, dann *öffne* eine neue Kiste für dieses Objekt.
- Ansonsten füge das Objekt in die erste bereits geöffnete Kiste ein, in die es passt.

BPP – Approximationsalgorithmus

Analyse des Approximationsfaktors:

Wieviele geöffnete Kisten gibt es, die höchstens halb gefüllt sind?

- FIRST FIT öffnet nur dann eine neue Kiste und fügt ein Objekt mit Gewicht höchstens $b/2$ in diese Kiste ein, wenn keine geöffnete Kiste mit Gewicht höchstens $b/2$ existiert.
- Es folgt, dass es zu keinem Zeitpunkt zwei geöffnete Kisten gibt, die Gewicht höchsten $b/2$ enthalten.

Schlussfolgerung: Wenn am Ende k Kisten geöffnet sind, gibt es somit mindestens $k-1$ Kisten, die mehr als halb voll sind.

Sei $W = \sum_{i=1^N} w_i$. Es folgt $W > \frac{k-1}{2} b$.

Aus der trivialen unteren Schranke $opt \geq W/b$ folgt nun $\frac{k-1}{2} < opt$ und somit $k \leq 2opt$.

□

BPP – Untere Schranke für den Approximationsfaktor

Satz

Wenn $\mathrm{P} \neq \mathrm{NP}$*, dann gibt es für BPP keinen effizienten* α*-Approximationsalgorithmus mit* $\alpha < \frac{3}{2}$.

Beweis:

Wir zeigen, dass aus einem effizienten α-Approximationsalgorithmus mit $\alpha < \frac{3}{2}$ für BPP ein effizienter Algorithmus für PARTITION abgeleitet werden kann.

Das NP-vollständige PARTITON-Problem wäre dann in P und daraus würde $\mathrm{P} = \mathrm{NP}$ folgen.

BPP – Untere Schranke für den Approximationsfaktor

Analog zum Konzept der polynomiellen Reduktion, erzeugen wir aus einer Eingabe $a_1, \ldots, a_N$ für PARTITION eine Eingabe für BPP, indem wir $w_1 = a_1, \ldots, w_N = a_N$ und $b = (\sum_{i=1}^{N} a_i)/2$ setzen.

Es gilt:

- Falls es sich um eine JA-Instanz von Partition handelt, gibt es eine Lösung für die BPP-Instanz mit $k = 2$ Kisten.
- Im Falle einer JA-Instanz würde ein α-Approximationsalgorithmus für BPP somit eine Aufteilung auf zwei Kisten finden, weil eine Lösung mit drei oder mehr Kisten um mindestens den Faktor $\frac{3}{2} > \alpha$ vom Optimum abweichen würde.
- Falls es sich um eine NEIN-Instanz von Partition handelt, gibt es hingegen keine BPP-Lösung mit zwei Kisten.

Durch Aufruf des Approximationsalgorithmus für BPP kann man somit die JA- und NEIN-Instanzen von PARTITION unterscheiden. □

Nichtapproximierbarkeit vom allgemeinen TSP

Zur Erinnerung:

Beim Traveling Salesperson Problem (TSP) ist ein vollständiger Graph $G = (V, E)$ mit Kantenkosten $c(u, v) \in \mathbb{N}$ für $u, v \in V$ mit $c(u, v) = c(v, u)$ gegeben.

Gesucht ist eine Rundreise (ein Hamiltonkreis) mit kleinstmöglichen Kosten.

Satz

Sei $\alpha \geq 1$ beliebig gewählt. Wenn $\mathrm{P} \neq \mathrm{NP}$, dann gibt es für das TSP-Problem gibt es keinen effizienten α-Approximationsalgorithmus.

Nichtapproximierbarkeit vom allgemeinen TSP

Beweis:

Wir zeigen, dass aus einem effizienten α-Approximationsalgorithmus $\mathcal{A}$ für TSP ein effizienter Algorithmus $\mathcal{A}'$ für das Hamiltonkreisproblem (HC) abgeleitet werden könnte.

Algorithmus $\mathcal{A}'$:

- Sei $G' = (V, E')$ die Eingabe für HC.
- $\mathcal{A}'$ transformiert G' in einen gewichteten, vollständigen Graphen $G = (V, E)$, in dem nur die Kanten aus E' die Länge 1 haben, alle anderen Kanten in E erhalten die Länge αn.
- Dann ruft $\mathcal{A}'$ den Algorithmus $\mathcal{A}$ auf G auf.
- Falls $\mathcal{A}$ eine TSP-Tour der Länge höchstens αn findet, so gibt $\mathcal{A}'$ aus, dass G' einen Hamilton-Kreis enthält.
- Ansonsten meldet $\mathcal{A}'$, dass G' keinen Hamilton-Kreis enthält.

Nichtapproximierbarkeit vom allgemeinen TSP

Korrektheit von $\mathcal{A}'$:

- Zuerst nehmen wir an, G' enthält einen Hamilton-Kreis. Jeder Hamilton-Kreis in G' entspricht einer TSP-Tour der Länge n in G. $\mathcal{A}$ findet somit eine Tour der Länge höchstens αn.
- Jetzt nehmen wir an, G' enthält keinen Hamilton-Kreis. Dann haben alle TSP-Touren in G die Länge mindestens $\alpha n + n - 1 > \alpha n$.

Damit kann $\mathcal{A}'$ mit Hilfe von $\mathcal{A}$ die JA- und NEIN-Instanzen von HC unterscheiden.

Aus der Existenz von $\mathcal{A}$ würde somit $\mathrm{P} = \mathrm{NP}$ folgen. $\square$

Metrisches TSP – Definition

Beim metrischen TSP ist ein vollständiger Graph $G = (V, E)$ mit Kantenkosten $c(u, v) \in \mathbb{N}$ für $u, v \in V$ gegeben, die für beliebige Knoten u, v, w die folgenden Bedingungen erfüllen:

- $c(u, u) = 0$
- $c(u, v) = c(v, u)$ (Symmetrie)
- $c(u, w) \leq c(u, v) + c(v, w)$ (Dreiecksungleichung)

Gesucht ist eine Rundreise (ein *Hamiltonkreis*) mit kleinstmöglichen Kosten.

Metrisches TSP – NP-Schwere

Beobachtungen:

- Metrisches TSP ist ein Spezialfall des allgemeinen TSP.
- $\{1, 2\}$-TSP erfüllt die Dreiecksungleichung und ist somit ein Spezialfall des metrischen TSP.
- Aus der NP-Schwere von $\{1, 2\}$-TSP folgt somit die NP-Schwere vom metrischen TSP.

Metrisches TSP – Approximationsalgorithmus

Satz

Für das metrische TSP gibt es einen effizienten 2-Approximationsalgorithmus.

Beweis:

Algorithmus:

1. Finde einen minimalen Spannbaum T von G;
2. Verdopple die Kanten von T und erhalte dadurch den Euler-Graphen T';
3. Berechne eine Euler-Tour auf T';
4. Bereinige die Euler-Tour um wiederholt vorkommende Knoten.

Metrisches TSP – Approximationsalgorithmus

Analyse des Approximationsfaktors:

- Aus einer TSP-Tour können wir einen Spannbaum erzeugen, indem wir eine Kante löschen.
- Also ist ein minimaler Spannbaum nicht teurer als die Länge einer minimalen TSP-Tour.
- Die Kosten der berechneten Euler-Tour entsprechen den doppelten Kosten des minimalen Spannbaums, sind also höchstens zweimal so teuer wie die minimale TSP-Tour.
- Aufgrund der Dreiecksungleichung erhöht das Überspringen von mehrfach besuchten Knoten in Schritt 3 die Kosten nicht.

□

Metrisches TSP – Bemerkungen

- Der beste bekannte Approximationsalgorithmus für metrisches TSP war lange Zeit ein von Christofides im Jahr 1975 vorgestellter Algorithmus mit einem Approximationsfaktor von $\frac{3}{2}$.
- Erst 2021 wurde dieser Approximationsfaktor übertroffen: Karlin, Klein und Garan haben einen Approximationsalgorithmus für metrisches TSP mit einem Faktor $\frac{3}{2} - \epsilon$ für ein winziges

$$\epsilon > \frac{1}{1.000.000.000.000.000.000.000.000.000.000.000.000.000.000}$$

gefunden.
- Andererseits haben Karpinski, Lampis und Schmied 2015 bewisen, dass es unter der Annahme P $\neq$ NP keinen effizienten Approximationsalgorithmus mit einem Approximationsfaktor besser als $\frac{123}{122}$ gibt.

Approximationsschemata

Ein Approximationsschema ist ein Algorithmus, der es ermöglicht, für jedes vorgegebene $\epsilon > 0$ eine zulässige Lösung mit Approximationsfaktor $1 + \epsilon$ (bzw. $1 - \epsilon$) zu berechnen.

Satz

Für das Rucksackproblem (KP) gibt es ein polynomielles Approximationsschema.

Die Klasse PSPACE

- Sei PSPACE die Klasse derjenigen Probleme, die wir mit einem polynomiell beschränkten Band auf einer TM lösen können.
- Im Gegensatz zur Zeitkomplexität kann man nachweisen, dass diese Klasse sich nicht ändern würde, wenn wir sie bezüglich der Platzkomplexität auf NTM definieren würden.
  (*Satz von Savitch*)
- Wie verhält sich PSPACE zu NP? – Da sich der Kopf einer Turingmaschine pro Zeitschritt nur eine Position bewegen kann gilt

$$\text{NP} \subseteq \text{PSPACE} \ .$$

Die Klasse EXPTIME

- Die Klasse der Probleme mit einer Laufzeitschranke $2^{p(n)}$ auf einer TM für ein Polynom p bezeichnen wir als EXPTIME.
- Wie verhält sich EXPTIME zu NP? und wie zu PSPACE?
- Bei einer Speicherplatzbeschränkung der Größe $s(n)$ gibt es nur $2^{O(s(n))}$ viele verschiedenen Konfigurationen für eine Turingmaschine, so dass auch die Rechenzeit durch $2^{O(s(n))}$ beschränkt ist.
- Die Probleme in PSPACE können deshalb in Zeit $2^{p(n)}$ gelöst werden, so dass gilt

$$\text{PSPACE} \subseteq \text{EXPTIME} \ .$$

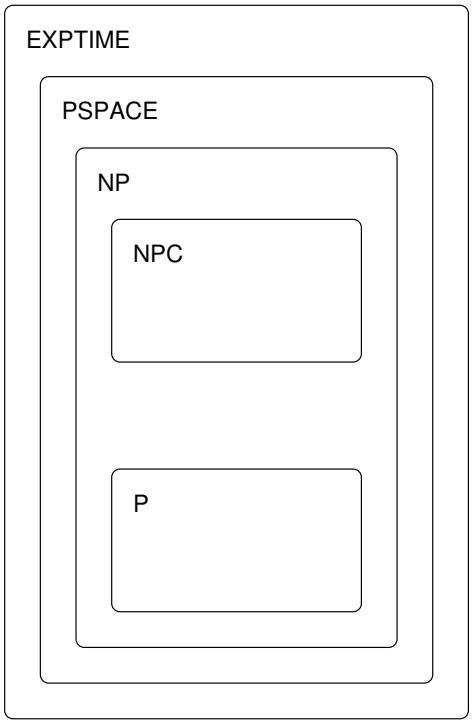
Zusammenfassung

- Wir haben gezeigt

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME.$$

- Es ist nicht bekannt, ob diese Inklusionen echt sind.
- Möglicherweise ist P = PSPACE oder NP = EXPTIME.
- Man weiß allerdings, dass $P \subsetneq EXPTIME$ (*Zeithierarchiesatz*).

Zusammenfassung



Mehr dazu in der Vorlesung Komplexitätstheorie.