NP-Vollständigkeit wichtiger Probleme

Sebastian Bernauer

22.03.2019

Inhalt I

Komplexitätsklassen

Der Operator $L_1 \ \leq_p L_2$ bedeutet, dass das L_1 polynomiell auf L_2 reduzierbar ist. Dies ist der Fall, wenn es eine polynomielle Transformation von L_1 nach L_2 gibt, so das heißt, wenn es eine von einer DTM in polynomieller Zeit berechenbare Funktion $f : \sum_1^* \rightarrow \sum_2^*$ gibt, so dass für alle $w \in \sum_1^*$ gilt:

$$w \in L_1 \leftrightarrow f(w) \in L_2$$

Komplexitätsklassen

Definition: $\leq_p$ "ordnet" Entscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

1. P - polynomiell lösbar

Komplexitätsklassen

Definition: $\leq_p$ "ordnet" Entscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

1. P - polynomiell lösbar
2. NP - nichtdeterministisch polynomiell lösbar

Komplexitätsklassen

Definition: $\leq_p$ "ordnet" Entscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

1. P - polynomiell lösbar
2. NP - nichtdeterministisch polynomiell lösbar
3. NP-vollständig
   $\rightarrow L \in NP$ und $\forall L' \in NP : L' \leq_p L$

Komplexitätsklassen

Definition: $\leq_p$ “ordnet” Entscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

1. P - polynomiell lösbar
2. NP - nichtdeterministisch polynomiell lösbar
3. NP-vollständig
   $\rightarrow L \in NP$ und $\forall L' \in NP : L' \leq_p L$
   $\rightarrow$Alle folgenden Probleme sind NP-vollständig

Komplexitätsklassen

Definition: $\leq_p$ "ordnet" Entscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

1. P - polynomiell lösbar
2. NP - nichtdeterministisch polynomiell lösbar
3. NP-vollständig
   $\rightarrow L \in NP$ und $\forall L' \in NP : L' \leq_p L$
   $\rightarrow$Alle folgenden Probleme sind NP-vollständig
4. NP-schwierig
   $\rightarrow \forall L' \in NP : L' \leq_p L$

Komplexitätsklassen

Definition: $\leq_p$ “ordnet” Entscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

1. P - polynomiell lösbar
2. NP - nichtdeterministisch polynomiell lösbar
3. NP-vollständig
   $\rightarrow L \in NP$ und $\forall L' \in NP : L' \leq_p L$
   $\rightarrow$Alle folgenden Probleme sind NP-vollständig
4. NP-schwierig
   $\rightarrow \forall L' \in NP : L' \leq_p L$
5. nicht rekursiv

Satisfiability Problem

Für natürliche Zahlen n und m seien m Klauseln über n Variablen gegeben. Eine Klausel ist die Disjunktion [Veroderung] von einigen Literalen x_i bzw. $\overline{x_i}$ mit $i, j \in \{1, ..., n\}$. Es soll entschieden werden, ob es eine Belegung $a = \{a_1, ..., a_n\} \in \{0, 1\}^n$ der Variablen $x_1, ..., x_n$ gibt, so dass alle Klauseln erfüllt sind.

Satisfiability Problem

Für natürliche Zahlen n und m seien m Klauseln über n Variablen gegeben. Eine Klausel ist die Disjunktion [Veroderung] von einigen Literalen x_i bzw. $\overline{x_i}$ mit $i, j \in \{1, ..., n\}$. Es soll entschieden werden, ob es eine Belegung $a = \{a_1, ..., a_n\} \in \{0, 1\}^n$ der Variablen $x_1, ..., x_n$ gibt, so dass alle Klauseln erfüllt sind.
Fragestellung: Existiert eine Wahrheitsbelegung der Variablen $x_1, ..., x_n$, so dass alle Klauseln erfüllt sind?

Satisfiability Problem

Für natürliche Zahlen n und m seien m Klauseln über n Variablen gegeben. Eine Klausel ist die Disjunktion [Veroderung] von einigen Literalen x_i bzw. $\overline{x_i}$ mit $i, j \in \{1, ..., n\}$. Es soll entschieden werden, ob es eine Belegung $a = \{a_1, ..., a_n\} \in \{0, 1\}^n$ der Variablen $x_1, ..., x_n$ gibt, so dass alle Klauseln erfüllt sind.

Fragestellung: Existiert eine Wahrheitsbelegung der Variablen $x_1, ..., x_n$, so dass alle Klauseln erfüllt sind?

$\rightarrow$Satz von Cook: SAT is NP-vollständig

3-SAT

- Spezialfall von SAT
- Jede Klausel enthält 3 Literale
- Zu Beweisen: SAT ist durch 3-SAT abbildbar und beide sind damit gleich komplex (NP-vollständig)

Beweis: 3-SAT $\leq_p$ SAT

Grundidee: Alle Klauseln des allgemeinen SAT in neue Klauseln der Länge 3 überführen.

- Klausel 1 Literal z
  $\rightarrow z \vee z \vee z$
- Klausel 2 Literale $z \vee y$
  $\rightarrow z \vee z \vee y$
- Klausel 3 Literale $z \vee y \vee z$
  $\rightarrow$Keine Änderung
- Klausel $\geq$ 4 Literale $z_1 \vee \ldots \vee z_k$
  $\rightarrow$siehe nächste Folie

Beweis: 3-SAT $\leq_p$ SAT

Beispiel: $k = 7$ Literale mit $z_1 \vee \ldots \vee z_k$:
(y_1, y_2, y_3, y_4 sind Hilfsvariablen)

- $z_1 \vee z_2 \vee y_1$
- $\overline{y_1} \vee z_3 \vee y_2$
- $\overline{y_2} \vee z_4 \vee y_3$
- $\overline{y_3} \vee z_5 \vee y_4$
- $\overline{y_4} \vee z_6 \vee z_7$

Beweis: 3-SAT $\leq_p$ SAT

Allgemein:

1. $z_1 \vee z_2 \vee y_{c,1}$
2. $\overline{y_{c,l}} \vee z_{l+2} \vee y_{c,l+1}$ für $1 \leq l \leq k-4$
3. $\overline{y_{c,k-3}} \vee z_{k-1} \vee z_k$

Index c für separate Hilfsvariablen für jede Klausel

Beweis: 3-SAT $\leq_p$ SAT

- SAT lässt sich durch 3-SAT abbilden
- 3-SAT $\leq_p$ SAT
- 3-SAT ist NP-vollständig

Clique Problem

In einem ungerichteten Graphen $G = (V, E)$ bildet die Knotenmenge $V‘ \subseteq V$ eine Clique, wenn für alle $v, v‘ \in V‘$ gilt $v, v‘ \in E$. [1]

Clique Problem

In einem ungerichteten Graphen $G = (V, E)$ bildet die Knotenmenge $V' \subseteq V$ eine Clique, wenn für alle $v, v' \in V'$ gilt $v, v' \in E$. [1]

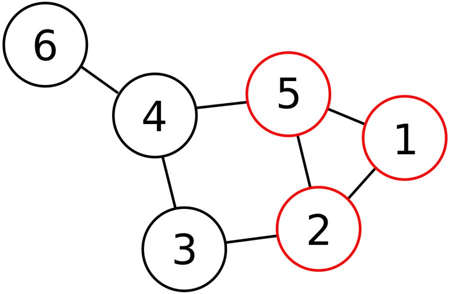

Abbildung: Ein Graph mit einer Clique der Größe 3.

Quelle: https://de.wikipedia.org/wiki/Clique_(Graphentheorie)#/media/File:6n-graf-clique.svg

Clique - Beispiel

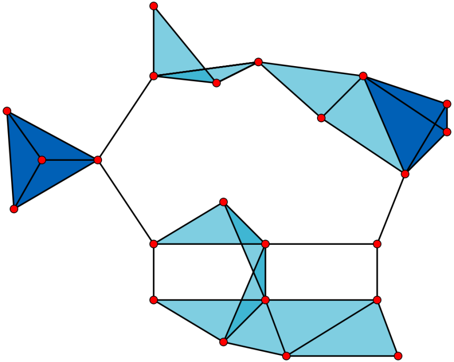

Abbildung: Ein Graph mit 2 Cliquen der Größe 4.

Quelle: https://en.wikipedia.org/wiki/Clique_(graph_theory)#/media/File:VR_complex.svg

Clique - Fragestellungen

1. Gibt es eine Clique der Größe k?
   $\rightarrow$Entscheidungsproblem

Clique - Fragestellungen

1. Gibt es eine Clique der Größe k?
   $\rightarrow$Entscheidungsproblem

2. Berechne das größte k, so dass eine Clique der Größe k vorhanden ist.
   $\rightarrow$Optimale Lösung

Clique - Fragestellungen

1. Gibt es eine Clique der Größe k?
   $\rightarrow$Entscheidungsproblem

2. Berechne das größte k, so dass eine Clique der Größe k vorhanden ist.
   $\rightarrow$Optimale Lösung

3. Berechne eine Clique mit dem größten k.
   $\rightarrow$Optimierungsproblem

Clique - Beweis

Beweis teilt sich in 2 Teile auf:

1. Clique ist in NP enthalten
2. SAT $\leq_p$ Clique

Clique - Beweis

Clique ist in NP enthalten.
Beweis:

1. NTM zählt Anzahl n der Knoten im Graphen

Clique - Beweis

Clique ist in NP enthalten.
Beweis:

1. NTM zählt Anzahl n der Knoten im Graphen
2. Rät Wort $w \in \{0,1\}^n$

Clique - Beweis

Clique ist in NP enthalten.
Beweis:

1. NTM zählt Anzahl n der Knoten im Graphen
2. Rät Wort $w \in \{0, 1\}^n$
3. Das Wort wird als Knotenauswahl interpretiert, V' enthält alle Knoten i mit $w_i = 1$

Clique - Beweis

Clique ist in NP enthalten.
Beweis:

1. NTM zählt Anzahl n der Knoten im Graphen
2. Rät Wort $w \in \{0,1\}^n$
3. Das Wort wird als Knotenauswahl interpretiert, V' enthält alle Knoten i mit $w_i = 1$
4. Es wird getestet, ob
   4.1 V' genau k Knoten beinhaltet.
   4.2 G eine Clique auf V' enthält

Clique - Beweis

Clique ist in NP enthalten.
Beweis:

1. NTM zählt Anzahl n der Knoten im Graphen
2. Rät Wort $w \in \{0,1\}^n$
3. Das Wort wird als Knotenauswahl interpretiert, V' enthält alle Knoten i mit $w_i = 1$
4. Es wird getestet, ob
   4.1 V' genau k Knoten beinhaltet.
   4.2 G eine Clique auf V' enthält

- Rechenaufwand ist polynomiell in der Knotenzahl n

Clique - Beweis

- Es wurde bereits bewiesen: NTM können durch DTM abgebildet werden
- Polynomielle Laufzeit + Nichtdeterminismus $\rightarrow$NP
- Daraus folgt: Clique ist in NP enthalten

Clique - Beweis

Beweis teilt sich in 2 Teile auf:

1. **Clique ist in NP enthalten** ✓
2. SAT $\leq_p$ Clique

Clique - Beweis

Es wurde bereits bewiesen, dass $\mathit{Clique} \in NP$ und SAT (und $3 - SAT$) NP-vollständig ist.
Nun ist zu beweisen, dass $SAT \leq_p \mathit{Clique}$.
Daraus folgt: *Clique* ist NP-vollständig.

Clique - Beweis

Konstruiere einen Graphen, der mittels *Clique* ein Problem löst, welches ein *SAT*-Problem ist.

1. Füge für jedes Literal in den Klauseln einen Knoten hinzu.

Clique - Beweis

Konstruiere einen Graphen, der mittels *Clique* ein Problem löst, welches ein *SAT*-Problem ist.

1. Füge für jedes Literal in den Klauseln einen Knoten hinzu.
2. Verbinde alle Literale außer folgende Kanten:
   - Klauselgruppen untereinander
   - Gegensätzliche Literale (z.B. x_1 und $\overline{x_1}$)

Clique - Beweis

Konstruiere einen Graphen, der mittels *Clique* ein Problem löst, welches ein *SAT*-Problem ist.

1. Füge für jedes Literal in den Klauseln einen Knoten hinzu.
2. Verbinde alle Literale außer folgende Kanten:
   - Klauselgruppen untereinander
   - Gegensätzliche Literale (z.B. x_1 und $\overline{x_1}$)
3. Suche eine Clique der Größe k, k ist die Anzahl der Klauseln. Da die Knoten einer Klauselgruppe nicht verbunden sind, muss aus jeder Klausel ein Literal "wahr" sein. Da die Literale in den Klauseln ODER-verknüpft sind, sind alle Klauseln erfüllt. [2]

Clique - Beweis

Umformung SAT-Problem zu Clique an der Tafel:

- $x_1 \vee \overline{x_2} \vee x_3$
- $\overline{x_1} \vee x_2$
- $x_3 \vee \overline{x_2}$

Clique - Beweis

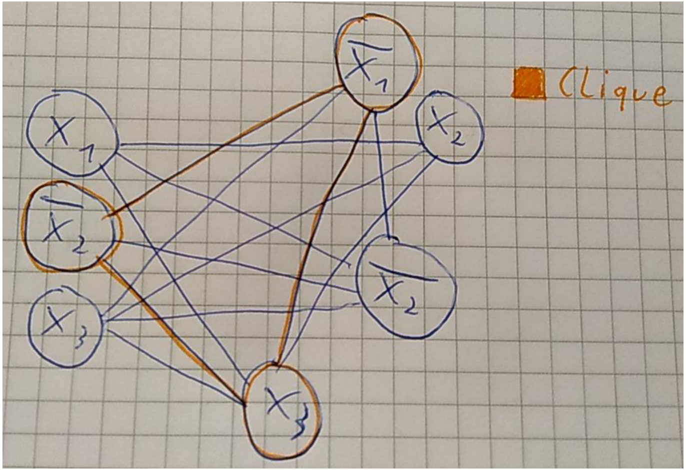

Abbildung: Graph nach Transformation von Clique- in SAT-Problem

Clique - Beweis

- *SAT*-Probleme können in ein *Clique*-Problem transferiert werden (mit polynomialen Zeitaufwand).
- $SAT \leq_p Clique$
- *Clique* ist NP-vollständig.

Knapsack Problem

Gegeben sind ein Rucksack und n Objekte mit Gewichten $g_1, ..., g_n \in \mathbb{N}$ sowie eine Gewichtsschranke G. Zusätzlich seien $a_1, ..., a_n \in \mathbb{N}$ die Nutzenwerte für die Objekte. [1]

Knapsack Problem

Gegeben sind ein Rucksack und n Objekte mit Gewichten $g_1, \ldots, g_n \in \mathbb{N}$ sowie eine Gewichtsschranke G. Zusätzlich seien $a_1, \ldots, a_n \in \mathbb{N}$ die Nutzenwerte für die Objekte. [1]

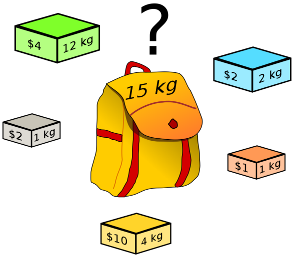

Abbildung: Ein zu befüllender Rucksack.

Quelle: https://de.wikipedia.org/wiki/Rucksackproblem#/media/File:Knapsack.svg

Knapsack - Fragestellungen

1. Gibt es - unter Beachtung des Limits - eine Beladung mit mindestens diesem Nutzwert?
   $\rightarrow$Entscheidungsproblem

Knapsack - Fragestellungen

1. Gibt es - unter Beachtung des Limits - eine Beladung mit mindestens diesem Nutzwert?
   →Entscheidungsproblem

2. Berechne den größtmöglichen Nutzwert.
   →Optimale Lösung

Knapsack - Fragestellungen

1. Gibt es - unter Beachtung des Limits - eine Beladung mit mindestens diesem Nutzwert?
   $\rightarrow$Entscheidungsproblem

2. Berechne den größtmöglichen Nutzwert.
   $\rightarrow$Optimale Lösung

3. Berechne die optimale Beladung.
   $\rightarrow$Optimierungsproblem

Knapsack - Beweis

Der Beweis sei an dieser Stelle vorausgesetzt. Es wird bewiesen, dass 3-SAT $\leq_p$ KP ist.
Für Interessierte ist er unter [1] im Kapitel 3.4.3 auf Seite 55 zu finden.

Partition Problem

Gegeben sind $b_1, ..., b_n \in \mathbb{N}$. Gibt es eine Teilmenge $I \subseteq \{1, ..., n\}$, so dass die Summe aller $b_i, i \in I$ gleich der Summe aller $b_i, i \notin I$ ist?
$\rightarrow$Teil eine Menge von Gewichten in 2 gleich schwere Haufen auf.

Partition - Beweis

Es wurde bereits bewiesen, dass ein (spezielles) Knapsack Problem $KP*$ NP-vollständig ist.
(Für $a_1, ..., a_n$ soll entschieden werden, ob es eine Auswahl gibt, so dass die Summe genau A beträgt).
Nun ist zu beweisen, dass $KP* \leq_p PARTITION$.
Daraus folgt: $PARTITION$ ist NP-vollständig.

Partition - Beweis

Sei $(a_1, ..., a_n, A)$ eine Eingabe für $KP*$.
Daraus konstruieren wir in polynomieller Zeit die Eingabe $(a_1, ..., a_n, S - A + 1, A + 1)$ für *PARTITION*, wobei S die Summe aller a_i ist.
Falls I eine Lösung für das KP ist, erhalten wir mit $I \cup \{n + 1\}$ eine Lösung für *PARTITION*, da

$$\sum_{i \in I} a_i + S - A + 1 = S + 1 = \sum_{1 \leq i \leq n} a_i + 1 = \sum_{i \notin I} a_i + A + 1$$

Sie Summe aller Zahlen in der Eingabe für *PARTITION* beträgt $2S + 2$.
Ein Lösung für *PARTITION* muss also so aussehen, dass jeder Teil sich zu $S + 1$ aufsummiert.

Partition - Beweis

Damit müssen die Zahlen $S - A + 1$ und $A + 1$ in verschiedenen Teilen sein. $(S - A + 1) + (A + 1) = (S + 2) > (S + 1)$ Die Zahlen, die $S - A + 1$ zu $S + 1$ ergänzen, haben die Summe A und bilden eine Lösung für die Eingabe von $KP*$.

Quellen

- Ingo Wegener. *Theoretische Informatik. Eine algorithmenorientierte Einführung*. Teubner, 2005.
- Weitz. *Clique ist NP-vollständig*. 2017. URL: https://www.youtube.com/watch?v=D3gkCTRMcKU (besucht am 25. 02. 2019).