NP-Vollständigkeit wichtiger Probleme

Sebastian Bernauer

25. Februar 2019

Inhalt I

Inhalt II

Beweis

HC

Problem

Beweis

TSP

Problem

Beweis

Komplexitätsklassen

Definition: $\leq_p$ ördnetËntscheidungsprobleme bezüglich ihrer Komplexität.

Aufsteigende Komplexitätsklassen:

1. P - polynomiell lösbar

Komplexitätsklassen

Definition: $\leq_p$ ördnetËntscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

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

Komplexitätsklassen

Definition: $\leq_p$ ördnetËntscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

1. P - polynomiell lösbar
2. NP - nichtdeterministisch polynomiell lösbar
3. NP-schwierig
   $\rightarrow \forall L‘ \in NP : L‘ \leq_p L$

Komplexitätsklassen

Definition: $\leq_p$ ördnetËntscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

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

Komplexitätsklassen

Definition: $\leq_p$ ördnetËntscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

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

Komplexitätsklassen

Definition: $\leq_p$ ördnetËntscheidungsprobleme bezüglich ihrer Komplexität.
Aufsteigende Komplexitätsklassen:

1. P - polynomiell lösbar
2. NP - nichtdeterministisch polynomiell lösbar
3. NP-schwierig
   $\rightarrow \forall L` \in NP : L` \leq_p L$
4. NP-vollständig
   $\rightarrow L \in NP$ und $\forall L` \in NP : L` \leq_p L$
   $\rightarrow$Alle folgenden Probleme sind NP-vollständig
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 $i, j \in \{1, \ldots, n\}$. Es soll entschieden werden, ob es eine Belegung $a = \{a_1, \ldots, a_n\} \in \{0, 1\}^n$ der Variablen $x_1, \ldots, x_n$ 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
- Es wurde bewiesen, dass SAT durch 3-SAT abbildbar und damit gleich komplex (NP-vollständig) ist

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]

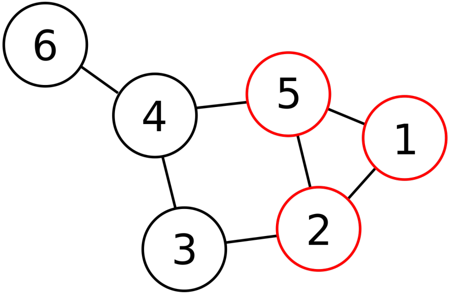

Abbildung: Ein Graph mit einer Clique der Größe 3.

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

Clique - Beispiel

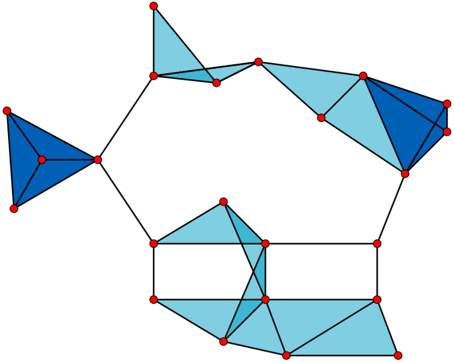

Abbildung: Ein Graph mit 2 Cliquen der Größe 4.

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

Es wurde bereits bewiesen, dass $CLIQUE \in NP$ und SAT NP-vollständig ist.
Nun ist zu beweisen, dass $SAT \leq_p CLIQUE$.
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 $\overline{x_1}$)
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.

Clique - Beweis

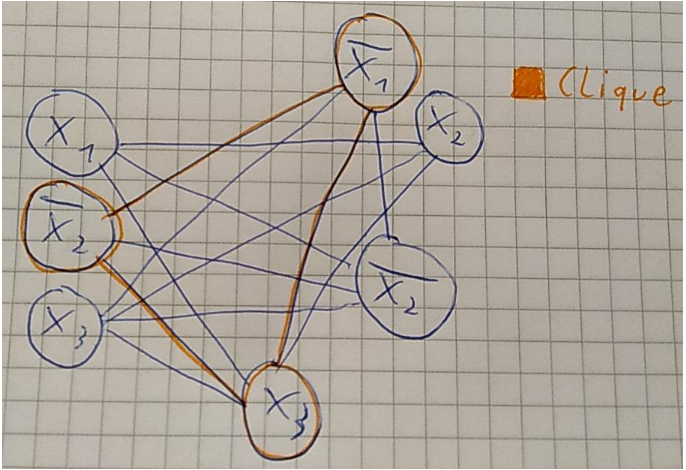

Abbildung: Graph nach Transformation von Clique- in SAT-Problem

Clique - Beweis

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

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]

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]

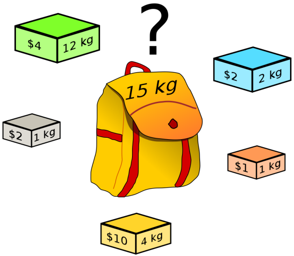

Abbildung: Ein zu befüllender Rucksack.

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?
   $\rightarrow$Entscheidungsproblem

2. Berechne den größtmöglichen Nutzwert.
   $\rightarrow$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

Beweis

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

Beweis

BP Problem

Problem

BP Beweis

Beweis

DHC Problem

Problem

DHC Beweis

Beweis

HC Problem

Problem

HC Beweis

Beweis

TSP Problem

Problem

TSP Beweis

Beweis

Quellen

Ingo Wegener. *Theoretische Informatik. Eine algorithmenorientierte Einführung*. Teubner, 2005.