Proseminar Theoretische Informatik
Die Komplexitätsklasse NP

Marvin Kleinert

25.11.2014

1 Wiederholung wichtiger Begriffe

1.1 Entscheidbarkeit

Sei $L \subseteq \Sigma^*$, dann gilt für die Entscheidbarkeit von L: L ist entscheidbar $\Leftrightarrow \exists$ TM M, die für jede Eingabe aus Σ^* anhält und alle $w \in L$ akzeptiert

1.2 Komplexitätsklasse P

P ist als Menge aller Sprachen definiert, die in polynomieller Zeit von einer deterministischen Einbandturingmaschine oder einem polynomiell äquivalenten Berechnungsmodell gelöst werden können. Daraus ergibt sich folgende Gleichung:

$$P = \bigcup_k TIME(n^k)$$

2 Definitionen

2.1 über Verifizierer

Die erste Möglichkeit die Komplexitätsklasse NP zu beschreiben ist mithilfe von Verifizierern. Ein Verifizierer V für eine Sprache A ist ein Algorithmus, der folgendermaßen funktioniert:

$$A = \left\{ w | \exists \text{ c: V akzeptiert (w,c) mit } |c| = |w|^k \right\}$$

Dabei ist der String c ein sog. Zertifikat, das eine richtige Lösung für das Kriterium der entsprechenden Sprache darstellt und als Referenz für die Verifizierung von w dient. Die Definition von NP sieht dann so aus:

$$NP = \{L | \text{L besitzt einen Polynomialzeit-Verifizierer, der alle } w \in L \text{ akzeptiert}\}$$

2.2 über nichtdeterministische TMs

Die zweite Möglichkeit sind nichtdeterministische Turingmaschinen. Eine solche ist analog zur deterministischen ein 7-Tupel $NTM = \{Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F\}$ mit einer veränderten Überführungsfunktion $\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{R, L\})$. Daraus kann sich eine nichtdeterministische Auswahl der nächsten Konfiguration ergeben. Der dabei entstehende Konfigurationsbaum benötigt einen Pfad von der Wurzel zu einer akzeptierenden Konfiguration in einem Blatt, damit sie ein Wort akzeptiert. Die Definition von NP über NTMs lautet dann:

$$\boxed{NP = \bigcup_k NTIME(n^k)}$$

mit:

$$NTIME(t(n)) = \{L | \text{L ist entscheidbar mit einer O(t(n)) nichtdeterministischen TM}\}$$

2.3 Äquivalenz beider Definitionen

Um zu zeigen, dass die beiden Definitionen äquivalent sind, nutzen wir folgende Idee: Wir simulieren den Verifizierer mit der NTM, indem sie das Zertifikat errät, und die NTM mit dem Verifizierer, indem er den akzeptierenden Pfad der NTM als Zertifiakt wählt.

1. Def. Verifizierer ⇒ Def. NTM
   Sei $A \in NP$ und V ein PNZ-Verifizierer für A. Es muss gezeigt werden, dass A entscheidbar ist mit einer PNZ-NTM N, die wie folgt konstruiert wird:
   - Eingabe: w mit $|w| = n$
   - Wähle nichtdeterministisch einen String c mit $|c| \leq n^k, k \in N$
   - Führe V auf Eingabe (w,c) aus; wenn V akzeptiert, akzeptiere; sonst verwerfe
2. Def. NTM ⇒ Def. Verifizierer
   Sei $A \in NP$ und N eine PNZ-NTM für A. Es muss gezeigt werden, dass A verifizierbar mit einem PNZ-Verifizierer V ist. Dazu konstruieren wir V folgendermaßen:
   - Eingabe: (w,c)
   - Simuliere N auf w und benutze c für die nichtdeterministische Auswahl
   - Wenn N akzeptiert, akzeptiere; sonst verwerfe

3 Einige Probleme in NP

3.1 HAMPATH

HAMPATH = $\{(G, s, t) |$G ist gerichteter Graph mit einem Hamilton-Pfad von s nach t$\}$

Ein Wort dieser Sprache ist z.B. dieser Graph mit dem schwarz markierten Hamiltonpfad von s nach t, der jeden Knoten in G genau einmal besucht:

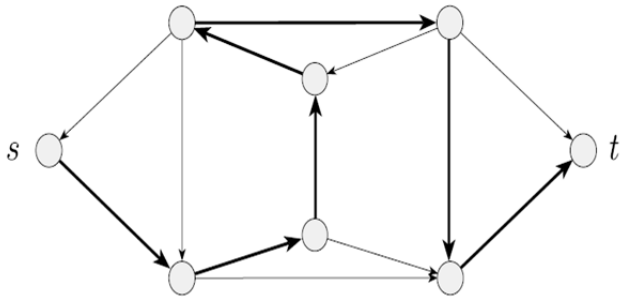

Abbildung 1: Graph mit Hamiltonpfad von s nach t

Für die Zugehörigkeit dieses Problems zu NP kann man entweder einen Verifizierer konstruieren, der HAMPATH in polynomieller Zeit verifiziert oder eine NTM, die HAMPATH in polynomieller Zeit entscheidet.

1. Verifizierer V: Eingabe $< (G, s, t), c >$
   Das Zertifikat ist in diesem Fall ein Hamiltonpfad von s nach t in G. Der Verifizierer gleicht nun c mit G ab und verifiziert das Wort (G,s,t), wenn jeder Knoten in c nur einmal besucht wird, alle Kanten in c auch in G vorkommen und s und t der Anfangs- bzw. Endknoten in c sind.
   Die Laufzeit in Abhängigkeit von $n = |(G, s, t)| = |V|^2 + c$ ist $O(n)$, da das Prüfen von doppelt besuchten Knoten $O(n)$, der Kantenabgleich zwischen c und G O(n) und die Überprüfung von Anfangs- und Endknoten $O(1)$ Zeit benötigen.

2. NTM: Eingabe (G,s,t)

   - Erzeuge eine Liste von m Zahlen $p_1, ..., p_m$ mit $m = |V|$, wobei jede Zahl nichtdeterministisch zwischen 1 und m ausgewählt wird
   - Prüfe, ob Zahlen mehrmals vorkommen; wenn ja, verwerfe
   - Prüfe, ob $s = p_1$ und $t = p_m$; wenn nicht, verwerfe
   - Prüfe für alle i zwischen 1 und (m-1), ob (p_i, p_{i+1}) eine Kante im Graphen G ist; wenn eine nicht zu G gehört, verwerfe; sonst akzeptiere

   Die Laufzeit ist polynomiell. Sie beträgt $O(n^2)$ mit $n = |(G, s, t)| = |V|^2 + c$, da die Auswahl der m Zahlen $O(n)$, das Prüfen von doppelten Knoten $O(n^2)$, das Prüfen des richtigen Anfangs- und Endknoten $O(n)$ und der Abgleich aller Kanten $O(n^2)$ benötigen.

3.2 CLIQUE

$$\text{CLIQUE} = \{(G, k) | \text{G ist ungerichteter Graph mit einer k-Clique}\}$$

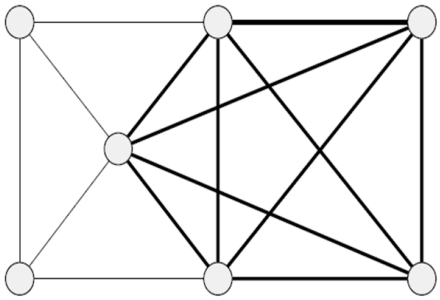

Abbildung 2: Graph mit 5-Clique

Auch CLIQUE liegt in NP. Für den Beweis lässt sich wieder ein PNZ-Verifizierer oder eine PNZ-NTM konstruieren:

1. Verifizierer V: Eingabe $< (G, k), c >$

   - Prüfe, ob c ein Untergraph mit k Knoten in G
   - Prüfe ob G alle Kanten enthält, die Knoten in c verbinden
   - Wenn beides gilt, akzeptiere; sonst verwerfe

2. NTM: Eingabe (G,k)

   - Wähle nichtdeterministisch eine Untermenge c mit k Knoten aus G
   - Prüfe, ob G alle Kanten enthält, die Knoten in c verbinden
   - Wenn ja, akzeptiere; wenn nein, verwerfe

Die Laufzeit ist für beide Konstruktionen polynomiell: Der Verifizierer braucht im schlimmsten Fall $O(n)$ mit $n = |(G, k)| = |V|^2 + c$, die NTM benötigt $O(n^2)$.

3.3 SUBSET-SUM

$$\text{SUBSET-SUM} = \left\{(S, t) | S = \{x_1, ..., x_k\}, \exists \{y_1, ..., y_l\} \subseteq S : \sum y_i = t\right\}$$

SUBSET-SUM ist in NP, was wieder mit einem PNZ-Verifizierer oder einer PNZ-NTM gezeigt werden kann:

1. Verifizierer V: Eingabe $< (S,t), c >$
   - Prüfe, ob c eine Menge von Zahlen ist, die addiert t ergeben
   - Prüfe, ob S alle Zahlen aus c enthält
   - Wenn beides zutrifft, akzeptiere; sonst verwerfe
2. NTM: Eingabe (S,t)
   - Wähle nichtdeterministisch eine Menge von Zahlen aus S
   - Prüfe, ob die Zahlen dieser Menge addiert t ergeben
   - Wenn ja, akzeptiere; wenn nein, verwerfe

Die Laufzeit ist in beiden Fällen polynomiell: Der Verifizierer muss im schlimmsten Fall $(n-1)$ Additionen und n Vergleiche machen mit $n = c_1 * |S| + c_2$, was zu einer Laufzeit von $O(n)$ führt. Die NTM wählt in $O(n)$ eine Menge aus und addiert in $O(n^2)$ die Zahlen der Menge, was zu einer Laufzeit von $O(n^2)$ führt.

4 coNP

Für die Komplemente der Sprachen in NP lässt sich im Allgemeinen nicht sagen, ob zu zu NP gehören oder nicht. Daher wird für diese Sprachen eine neue Klasse coNP definiert:

$$coNP = \{\overline{L} | L \in NP\}$$

Wir kennen zwar Komplemente, wie z.B. $\overline{PRIMES} = COMPOSITES$, die in coNP und NP liegen, es ist aber unklar, ob NP=coNP gilt.

5 P=NP?

Intuitiv scheint zu gelten, dass $P \subset NP$, da alle polynomiell entscheidbaren Probleme immer auch polynomiell verifizierbar sind und es darüber hinaus viele polynomiell verifizierbare Probleme gibt, für die wir bisher noch keinen Polynomialzeitalgotrithmus zum Entscheiden gefunden haben. Allerdings gibt es bis jetzt auch noch keinen Beweis dafür, dass eine Sprache in NP nicht in P liegt, sodass auch gelten könnte P=NP. Die Frage $P \stackrel{?}{=} NP$ gehört als eines der sieben Millenniums-Probleme zu den größten ungelösten Problemen der theoretischen Informatik.

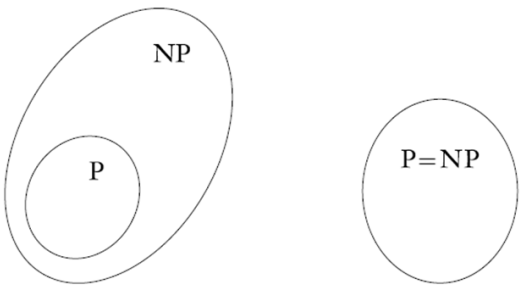

Abbildung 3: zwei Möglichkeiten: $P \subset NP$ oder P=NP

6 Literatur

M. Sipser Introduction to the Theory of Computation PWS Publ.Comp. 1997 Kap. 7.3