Prof. Dr. Hans-Jörg Kreowski 11. Januar 2006

Algorithmen auf Graphen (WS 2005/2006)

Ergänzung Kapitel 8

Die Klassen P und NP

Das P=NP-Problem, also die Frage, ob die Klasse der algorithmischen Probleme, die nichtdeterministisch mit polynomiellem Aufwand lösbar sind, und die Klasse der Probleme, die deterministisch mit polynomiellem Aufwand lösbar sind, übereinstimmen, ist eines der bedeutendsten ungelösten Probleme der Informatik (und der Mathematik). Um die Problematik detaillierter bearbeiten zu können, wird ein formaler Rahmen geschaffen, in dem sich die Klassen P und NP definieren und untersuchen lassen.

1. Die Betrachtung wird auf Entscheidungsprobleme der Form
$$D\colon IN \to \{\text{JA, NEIN}\}$$
beschränkt, wobei IN ein beliebiger Eingabebereich und D eine Abbildung ist.

2. In der Literatur werden Lösungsalgorithmen für D meist als Turing-Maschinen formalisiert. IN muss dann eine geeignete Menge von Wörtern sein, die als Eingaben auf das Band geschrieben und dann verarbeitet werden. Für alle $w \in IN$ muss dann gelten, dass genau dann ein Endzustand erreichbar ist, wenn $D(w) = \text{JA}$. Der Zeitaufwand wird als Zahl der Arbeitschritte der Turing-Maschine gemessen. Ist diese Zahl durch ein Polynom in Abhängigkeit von der Länge der Eingaben beschränkt, so liegt D in NP. Ist die Turing-Maschine, die das leistet, auch noch deterministisch, so ist D in P.

3. Der Vorteil dieser Definition ist, dass die Arbeitsschritte für alle Algorithmen einheitlich sind und alle ungefähr dieselbe konstante Zeit brauchen. Damit werden Aussagen über Aufwände gut vergleichbar (und sicher nicht Äpfel mit Birnen verglichen). Der Nachteil besteht darin, dass Algorithmen – wie insbesondere Algorithmen auf Graphen – praktisch nie als Turing-Maschinen modelliert werden.

4. Es wird deshalb hier ein Modellierungsrahmen geschaffen, der näher an übliche Beschreibungen von Algorithmen herankommt, aber dennoch P und NP zu definieren erlaubt. Wörter und Zustände werden durch Konfigurationen ersetzt, die frei wählbar sind und die Eingaben enthalten. Die Rolle der Endzustände übernimmt die spezielle Konfiguration JA. Statt der Wortlänge wird eine Konfigurationsgröße angenommen. Die Arbeitsschritte der Turing-Maschinen werden durch eine beliebige binäre Relation von Berechnungsschritten auf Konfigurationen ersetzt. Damit diese Schritte eine gewisse Homogenität aufweisen, wird vorausgesetzt, dass sie höchstens konstant lange dauern und höchstens konstant viel verändern.

5. Ein *algorithmisches Schema AS* für $D \colon IN \to \{\text{JA, NEIN}\}$ besteht aus
   - einer Menge K von *Konfigurationen* mit $IN \subseteq K$ und $\text{JA} \in K$,
   - einer Größenfunktion $size \colon K \to \mathbb{N}$ und
   - einer Menge von *(Berechnungs-)Schritten* $\to \subseteq K \times K$, so dass
     - jeder Schritt $con \to con'$ innerhalb einer Zeiteinheit c_{time} ausführbar ist und
     - $|\, size(con') - size(con) \,| \leq c_{size}$ für eine Konstante c_{size} gilt.

6. *AS* ist *korrekt bzgl. D,* falls für alle $con_0 \in IN$ gilt:

   $$D(con_0) = \text{JA g.d.w. eine Schrittfolge } con_0 \xrightarrow{*} \text{JA}$$

   existiert.

7. *AS* ist *polynomiell,* wenn ein Polynom p und eine Konstante c existieren, so dass für alle Schrittfolgen $con_0 \to con_1 \to \cdots \to con_k$ mit $con_0 \in IN$ gilt:

   $$k \leq c \cdot p(size(con_0)).$$

8. *AS* ist *nichtdeterministisch vom Grad* $b \in \mathbb{N}$, falls für alle $con \in K$ gilt:

   $$\#\{con' \mid con \to con'\} \leq b.$$

9. $D \in$ NP, falls ein nichtdeterministisches, polynomielles algorithmisches Schema *AS* existiert, das korrekt ist bzgl. D und einen konstanten Grad b besitzt. $D \in$ P, falls zusätzlich $b = 1$.

10. Offensichtlich gilt $P \subseteq NP$.

11. Es ist jedoch nicht bekannt, ob NP $\subseteq$ P. Die meisten Fachleute vermuten, dass das nicht gilt.

12. Sei $D \in$ NP. Sei *AS* ein polynomielles algorithmisches Schema mit Grad b, das korrekt ist bzgl. D. Sei $con_0 \in IN$ und Z die Zahl der maximalen Schrittfolgen $con_0 \xrightarrow{*} con$ (die sich nicht mehr verlängern lassen). Seien c eine Konstante und p ein Polynom, durch die *AS* polynomiell wird. Dann gilt:

    $$Z \leq b^{c \cdot p(size(con_0))}$$

13. Da mit den Voraussetzungen von Punkt 12 jede Schrittfolge, die bei einer Eingabe beginnt, Anfangsstück einer maximalen ist, lässt sich D dadurch berechnen, dass man alle Schrittfolgen aufbaut. Dieses Verfahren ist eine ad-hoc-Lösung, die oft ausschöpfende Suche genannt wird und sowohl durch Tiefen- als auch Breitensuche realisiert werden kann. Wenn $D(con_0) = \text{JA}$ gilt, reicht es, eine Schrittfolge $con_0 \xrightarrow{*} \text{JA}$ zu finden. Das könnte schnell gehen. Wenn aber $D(con_0) = \text{NEIN}$ gilt, hat man diese Antwort erst berechnet, wenn alle Schrittfolgen $con_0 \xrightarrow{*} con$ überprüft sind.

14. Insbesondere NP $\subseteq$ EXP (einfach exponentiell).