Kapitel 4: Komplexitätstheorie

Gliederung

↓ mit minimalen Änderungen durch B. Baumgarten

Zielstellung

- ... für diesen Abschnitt

  - Es geht uns darum, besser zu verstehen, warum es lösbare Entscheidungsprobleme gibt, die man höchstwahrscheinlich nicht effizient lösen kann.

  - Konkret geht es darum, besser zu verstehen, warum man vermutet, dass es effizient verifizierbare Entscheidungsprobleme gibt, die nicht effizient lösbar sind, d.h. unseren „Graubereich“ besser zu verstehen

... anders formuliert: man vermutet, dass $P \neq NP$ gilt (Da $P \subseteq NP$ gilt, bedeutet $P \neq NP$, dass es Entscheidungsprobleme gibt, die effizient verifizierbar aber nicht effizient lösbar sind.)

Anmerkungen

- Die Frage, ob die Vermutung, dass P ≠ NP gilt, richtig ist, oder vielleicht doch P = NP gilt, ist eines der zentralen Probleme in der Informatik.

Wer diese Frage beantworten kann, kann nicht nur (in Informatiker-Kreisen) berühmt sondern auch „reich“ werden:
Das P/NP-Problem gehört zu den sogenannten „Millennium-Problemen“, die mit je 1 Mio. $ dotiert sind.

Evtl. macht er/sie sich aber auch unbeliebt, denn konstruktive P=NP-Ergebnisse könnten existierende IT-Sicherheitsmechanismen gefährden.

Diskussion

- Warum ist das P/NP-Problem wichtig?

  - Es gibt einige tausend lösbare Entscheidungsprobleme, für die man bisher
    - nicht nachweisen konnte, dass sie effizient lösbar sind, sondern
    - nur nachweisen konnte, dass sie effizient verifizierbar sind.

Diese Entscheidungsprobleme sind meist Varianten von „praktisch relevanten“ algorithmischen Problemen, etwa von Konstruktions- bzw. Optimierungsproblemen.

Also gibt es auch einige tausend „praktisch relevante“ algorithmischen Probleme, für die man keine „praxis-tauglichen“ Lösungsalgorithmen kennt.

Diskussion

- ein „philosophischer“ Grund für die Vermutung, dass P ≠ NP gilt

  - Erfahrungsgemäß ist es – mit Blick auf eine interessierenden Aussage – in der Regel viele schwieriger, ...
    - einen Beweis zu finden, der belegt, dass diese Aussage korrekt ist,
    - als zu überprüfen, ob ein vorgeschlagener Beweis belegt, dass diese Aussage korrekt ist.
  - Deshalb vermutet man, dass P ≠ NP gilt.

Wenn ein Entscheidungsproblem (X,Y) effizient lösbar ist, so kann man herausfinden, ob eine Aussage der Form „$x \in Y$“ gilt.

Wenn ein Entscheidungsprobleme (X,Y) effizient verifizierbar ist, so kann man herausfinden, ob ein vorgeschlagener Beweis für eine Aussage der Form „$x \in Y$“ korrekt ist.

Diskussion

- ein „anderer“ Grund für die Vermutung, dass P ≠ NP gilt

  - In der Komplexitätsklasse NP gibt es „schwierigste“ Probleme, so genannte NP-vollständige Entscheidungsprobleme, die folgende Eigenschaften haben:

    - Wenn man nachweisen kann, dass ein einziges NP-vollständiges Entscheidungsproblem effizient lösbar ist, folgt sofort, dass jedes effizient verifizierbare Entscheidungsproblem auch effizient lösbar ist, d.h. P = NP wäre bewiesen.

Es ist noch nicht gelungen, einen effizienten Lösungsalgorithmus für irgendein NP-vollständiges Problem anzugeben.

Grundlagen

- Um den Begriff „NP-vollständiges Entscheidungsproblem“ zu präzisieren, müssen wir den in der Berechnungstheorie benutzen Reduktionsbegriff geeignet anpassen.
  - Seien (X_1,Y_1) und (X_2,Y_2) Entscheidungsprobleme.
  - Der komplexitätstheoretische Reduktionsbegriff soll es uns erlauben, die folgenden zwei Fragen miteinander in Beziehung zu setzen:
    - Ist (X_1,Y_1) ein effizient lösbares Entscheidungsproblem?
    - Ist (X_2,Y_2) ein effizient lösbares Entscheidungsproblem?

Zur Erinnerung

- Seien (X_1,Y_1) und (X_2,Y_2) Entscheidungsprobleme
- Sei t eine vollständige definierte Funktion von X_1 auf X_2, die (X_1,Y_1) auf (X_2,Y_2) reduziert, d.h. für alle $x \in X$ gilt:
  - wenn $x_1 \in Y_1$ gilt, so gilt $t(x_1) \in Y_2$
  - wenn $x_1 \notin Y_1$ gilt, so gilt $t(x_1) \notin Y_2$
- Dann gilt:

> (X_1,Y_1) ist Turing-reduzierbar auf (X_2,Y_2), wenn es ein Computerprogramm gibt, das die Funktion t berechnet, d.h. das für alle $x_1 \in X_1$ folgendes leistet:
>
> - Bei Eingabe $cod(x_1)$ wird die Ausgabe $cod(t(x_1))$ berechnet.

Mit der Funktion cod werden die Elemente aus X_1 und X_2 geeignet als Zeichenketten über einem Alphabet Σ kodiert.

Der angepasste Reduktionsbegriff

- Seien (X_1,Y_1) und (X_2,Y_2) Entscheidungsprobleme und cod eine geeignete Funktion, um die Elemente von X_1 und X_2 zu beschreiben;
- sei t eine vollständig definierte Funktion von X_1 auf X_2, die (X_1,Y_1) auf (X_2,Y_2) reduziert.
- Dann gilt:

> (X_1,Y_1) ist effizient auf (X_2,Y_2) reduzierbar, wenn es ein „praxistaugliches“ Computerprogramm gibt, das die Funktion t berechnet, d.h. das für alle $x_1 \in X_1$ folgendes leistet:
>
> - bei Eingabe $\text{cod}(x_1)$ wird die Ausgabe $\text{cod}(t(x_1))$ berechnet

Die Reduktionsfunktion t soll sich nun also sogar effizient berechnen lassen.

- Ein erstes (schon bekanntes) Beispiel
  - Im Folgenden seien A und B zwei endliche Listen natürlicher Zahlen.
  - Beim Entscheidungsproblem (X_1,Y_1) geht es um eine Antwort auf die Frage, ob die Liste B eine Permutation der Liste A ist
  - Beim Entscheidungsproblem (X_2,Y_2) geht es um eine Antwort auf die Frage, ob die beiden Listen A und B gleich sind

- *X_1 und X_2 umfassen alle Paare (A,B) von endlichen Listen natürlicher Zahlen.*
- *Y_1 enthält alle Listenpaare (A,B), bei denen B eine Permutation von A ist.*
- *Y_2 enthält alle Listenpaare (A,B), für die A = B gilt.*

Ein erstes (schon bekanntes) Beispiel (cont.)

- Sei x_1 = (A,B) gegeben.
- Sei $t(x_1)$ = (A‘,B‘), wobei A‘ die sortierte Version der Liste A und B‘ die sortierte Version der Liste B‘ ist.

- Wir wissen bereits, dass t eine geeignete Reduktionsfunktion ist.
- Da man zu einer gegebenen Liste effizient deren sortierte Version bestimmen kann, kann man die Reduktionsfunktion t effizient berechnen.

Ein zweites Beispiel

- Im Folgenden sei O eine endliche Menge von Objekten, w eine Funktion, die jedem Objekt aus O seinen Wert, d.h. eine natürliche Zahl, zuordnet, und b eine natürliche Zahl.
- Beim Partitionsproblem (X_P, Y_P) geht es um eine Antwort auf die Frage, ob die Menge O so in zwei Teilmengen O_1 und O_2 zerlegt werden kann, dass die Objekte in O_1 und O_2 denselben Gesamtwert haben.
- Beim Teilsummenproblem (X_{TS}, Y_{TS}) geht es um eine Antwort auf die Frage, ob es eine Teilmenge O‘ von O gibt, so dass die Objekte in O‘ den Gesamtwert b haben.

- *X_P enthält alle Paare (O,w); Y_P enthält alle Paare (O,w) mit ...*
- *X_{ST} enthält alle Tripel (O,w,b); Y_{ST} enthält alle Tripel (O,w,b) mit ...*

Ein zweites Beispiel (cont.)

- Sei $x_P = (O,w) \in X_P$ gegeben
- Sei $t(x_P) = (O,w',b) \in X_{ST}$ mit
  - $w'(o) := w(o)$, für alle Objekte $o \in O$
  - $b := \frac{1}{2} \sum_{o \in O} w'(o)$
- Offenbar ist t eine geeignete Reduktionsfunktion, denn …
  - genau dann wenn eine Teilmenge den halben Gesamtwert als Wertesumme hat, muss ihr Komplement dieselbe Wertesumme haben,
  - und die Funktion w' und die Zahl b (also auch t) kann man anhand der gegebenen Funktion w (sehr) effizient bestimmen.

Konsequenzen

- Seien (X_1,Y_1) und (X_2,Y_2) Entscheidungsprobleme
- Sei (X_1,Y_1) auf (X_2,Y_2) effizient reduzierbar

- Dann gilt die folgende Implikation:

> Wenn (X_2,Y_2) effizient lösbar ist, so ist auch (X_1,Y_1) effizient lösbar.

- Damit gilt auch deren Kontraposition:

> Wenn (X_1,Y_1) nicht effizient lösbar ist, so ist auch (X_2,Y_2) nicht effizient lösbar.

Dass (X_1,Y_1) auch effizient lösbar ist, liegt daran, dass die Komposition zweier effizient berechenbarer Funktionen eine effizient berechenbare Funktion „ergibt“.

- Ähnlich wie bei der Berechenbarkeit …

  - Seien (X_1,Y_1) und (X_2,Y_2) nicht-triviale Entscheidungsproblem (d.h. es gilt $\varnothing \neq Y_1 \neq X_1$ sowie $\varnothing \neq Y_2 \neq X_2$).

(X_1,Y_1)	(X_2,Y_2)	Wann ist (X_1,Y_1) effizient reduzierbar auf (X_2,Y_2)?
effizient lösbar	effizient lösbar	immer
effizient lösbar	nicht effizient lösbar	immer
nicht effizient lösbar	effizient lösbar	nie
nicht effizient lösbar	nicht effizient lösbar	vielleicht

Die zentrale Begriffsbildung

- Sei (X_{NPV}, Y_{NPV}) ein effizient verifizierbares Entscheidungsproblem, d.h. ein Entscheidungsproblem aus der Komplexitätsklasse NP.

- Dann definieren wir

> (X_{NPV}, Y_{NPV}) ist ein NP-vollständiges Entscheidungsproblem, wenn für alle effizient verifizierbaren Entscheidungsprobleme (X,Y), d.h. für alle Entscheidungsprobleme (X,Y) aus der Komplexitätsklasse NP, gilt:
>
> - (X,Y) ist auf (X_{NPV}, Y_{NPV}) effizient reduzierbar.

Konsequenzen

- Sei (X_{NPV}, Y_{NPV}) ein NP-vollständiges Entscheidungsproblem.
- Dann gilt offensichtlich die folgende Implikation:

> Wenn das Entscheidungsproblem (X_{NPV}, Y_{NPV}) effizient lösbar ist, so ist auch jedes andere effizient verifizierbare Entscheidungsproblem effizient lösbar, d.h. es gilt P = NP.

- Also gilt auch deren Kontraposition:

> Wenn P ≠ NP gilt, so ist das Entscheidungsproblem (X_{NPV}, Y_{NPV}) nicht effizient lösbar.

Weiteres Vorgehen

- Wir schauen uns jetzt einige NP-vollständige Entscheidungsprobleme an und diskutieren, weshalb sie NP-vollständig sind.
- Dabei hilft uns folgende Beobachtung:
  - Sei (X_{NPV}, Y_{NPV}) ein NP-vollständiges Problem;
  - sei (X,Y) ein effizient verifizierbares Entscheidungsproblem, d.h. (X,Y) gehört zur Komplexitätsklasse NP.
  - Dann gilt:

Wenn sich (X_{NPV}, Y_{NPV}) auf (X,Y) effizient reduzieren lässt, dann gilt:

- (X,Y) ist auch ein NP-vollständiges Entscheidungsproblem.

Das Erfüllbarkeitsproblem

- Bei diesem effizient verifizierbaren Entscheidungsproblem geht um folgendes algorithmisches Problem:

gegeben:	• eine aussagenlogische Formel F in konjunktiver Normalform
gesucht:	• die Antwort auf die Frage, ob F erfüllbar ist, d.h. ob es eine Belegung gibt, die F „wahr“ macht

Der zentrale Ergebnis

- Zu Beginn der 70iger Jahre der letzten Jahrhunderts wurde von Karp und – unabhängig davon – von Levin gezeigt, dass gilt:

> Das Erfüllbarkeitsproblem ist ein NP-vollständiges Problem.

Dieses Ergebnis wurde der Ausgangspunkt für alle weiteren solchen Ergebnisse.

Diskussion

- Anmerkungen
  - Nachzuweisen, dass es sich beim Erfüllbarkeitsproblem um ein NP-vollständiges Problem handelt, ist nicht ganz einfach, da man
    - zeigen muss, dass sich jedes andere effizient verifizierbare Entscheidungsproblem effizient auf das Erfüllbarkeitsproblem reduzieren lässt
    - aber nicht alle zu betrachtenden Entscheidungsproblem kennt.

Diese Schwierigkeiten bekommt man jedoch in den Griff, da man mit aussagenlogischen Formeln Turing-Maschinen beschreiben kann.

- Anmerkungen (cont.)
  - Nachzuweisen, dass weitere effizient verifizierbare Entscheidungsprobleme ebenfalls NP-vollständig sind, ist dann recht einfach, da man nur zeigen muss, dass
    - sich das Erfüllbarkeitsproblem auf das jeweils betrachtete Entscheidungsproblem effizient reduzieren lässt.

Mitunter wählt man statt des Erfüllbarkeitsproblems ein anderes NP-vollständiges Entscheidungsproblem als Ausgangspunkt.

Anmerkungen (cont.)

- Um die nachfolgenden Betrachtungen zu vereinfachen, wählen wir das 3-Erfüllbarkeitsproblem zum Ausgangspunkt.
- Bei diesen effizient verifizierbarem Entscheidungsproblem geht es darum, das folgende algorithmische Problem zu lösen.

gegeben:	• eine aussagenlogische Formel F in 3-konjunktiver Normalform (in jeder Disjunktion kommen genau drei paarweise verschiedene Literale vor)
gesucht:	• die Antwort auf die Frage, ob F erfüllbar ist, d.h. ob es eine Belegung gibt, die F „wahr“ macht

Das 3-Erfüllbarkeitsproblem

- Da man jeder aussagenlogischen Formel F in konjunktiver Normalform effizient eine erfüllbarkeitsäquivalente Formel F‘ in 3-konjunktiver Normalform zuordnen kann, gilt:

> Das 3-Erfüllbarkeitsproblem ist ein NP-vollständiges Problem.

Das k-Cliquenproblem

- Es sei daran erinnert, dass es bei diesem effizient verifizierbarem Entscheidungsproblem darum geht, folgendes algorithmisches Problem zu lösen:

gegeben:	• ein ungerichteter Graph G = (V,E) • eine Zahl k $\in$ N
gesucht:	• die Antwort auf die Frage, ob es im Graphen G eine Clique der Größe k gibt

Das k-Cliquenproblem (cont.)

- Indem man zeigt, dass sich das 3-Erfüllbarkeitsproblem auf das k-Cliquenproblem effizient reduzieren lässt, hat man nachgewiesen

> Das k-Cliquenproblem ist ein NP-vollständiges Problem.

Wir haben zu Beginn Vorlesung das 3-Erfüllbarkeitsproblem auf das k-Cliquenproblem reduziert.
Wenn man sich das Verfahren genau anschaut, sieht man, dass diese Reduktion effizient ist.

Anmerkungen

- Man kann auf ähnliche Art und Weise zeigen, dass sich das 3-Erfüllbarkeitsproblem auf das Teilsummenproblem effizient reduzieren lässt.
- Man kann zeigen, dass sich das Teilsummenproblem auf das Partitionsproblem effizient reduzieren lässt:
  - *Ist S der Gesamtwert von O, in dem man eine Teilmenge mit Wertesumme b sucht, so fügt man zu O noch zwei Elemente **y** mit Wert 2S–b und **z** mit Wert S+b hinzu.*
  - *Findet man im erweiterten O zwei „Hälften“ mit gleicher Wertesumme (jeweils 2S), so enthält eine von beiden y.*
  - *Deren Rest ohne y ist eine Teilmenge von O mit Wertesumme b !*
- Also gilt auch:

Das Teilsummenproblem ist ein NP-vollständiges Problem.

Das Partitionsproblem ist ein NP-vollständiges Problem.