10 Randomisiertes Approximieren

Im ersten Abschnitt dieses Kapitels führen wir Begriffe für verschiedene Varianten sogenannter polynomieller Approximationsschemata ein. Im zweiten Abschnitt stellen wir randomisierte polynomielle Approximationsschemata für das Problem vor, die Anzahl erfüllender Belegungen der Variablen einer Formel in DNF zu bestimmen.

10.1 Polynomielle Approximationsschemata

10.1 Im folgenden sei stets Π ein „Zählproblem“, das für jede Eingabe x eine gewisse Anzahl $\#_\Pi(x)$ (oder kurz $\#(x)$) von „Lösungen“ besitzt.

10.2 Jedem solchen Zählproblem Π entspricht ein Entscheidungsproblem E_Π, bei dem es um die Frage geht, ob es für eine Eingabe x eine Lösung gibt, d. h. ob $\#(x) > 0$ ist.

10.3 Definition Ein Problem Π gehört zur Klasse **#P**, wenn es eine nichtdeterministische Turingmaschine gibt, die in Polynomialzeit arbeitet und für jede Eingabe x genau $\#_\Pi(x)$ akzeptierende Berechnungen besitzt.

Ein Problem Π ist **#P**-*vollständig*, wenn jedes Problem $\Pi' \in$ **#P** von einer Turingmaschine in Polynomialzeit auf Π reduziert werden kann. ◇

Unter den **#P**-vollständigen Problemen finden sich unter anderem:

- Wieviele erfüllende Belegungen hat eine Formel, die in DNF gegeben ist?
- Wieviele erfüllende Belegungen hat eine 2sat-Formel?
- Wieviele perfekte Matchings hat ein gegebener bipartiter Graph?
- Was ist die Permanente einer gegebenen Booleschen Matrix?
- Wieviele topologische Sortierungen eines gegebenen DAG gibt es?

Dabei ist es insbesondere erstaunlich, dass die zugehörigen Entscheidungsvarianten der beiden ersten Probleme sehr einfach sind.

10.4 Lemma. Kann man ein **#P**-vollständiges Problem deterministisch in Polynomialzeit lösen, dann ist $\mathbf{P} = \mathbf{NP}$.

Es ist daher naheliegend, sich für Algorithmen zu interessieren, die Zählprobleme jedenfalls näherungsweise lösen:

10.5 Definition Ein *Approximationsschema (AS)* für Π ist ein deterministischer Algorithmus A, der für jede Eingabe x der Größe $n = |x|$ und für jedes $\varepsilon > 0$ eine Ausgabe $A(x)$ erzeugt, für die gilt:

$$(1-\varepsilon)\#(x) \leqslant A(x) \leqslant (1+\varepsilon)\#(x) \ .$$

$A(x)$ heißt dann eine ε-*Approximation* von $\#(x)$.

Ein *polynomielles Approximationsschema (PAS)* ist ein AS, dessen Laufzeit polynomiell in n ist. Ein *voll polynomielles Approximationsschema (FPAS)* ist ein AS, dessen Laufzeit polynomiell in n und $1/\varepsilon$ ist. ◇

Für **#P**-vollständige Probleme kennt man keine PAS oder gar FPAS. Die Situation wird besser, wenn man Approximationsschemata mit Zufallsentscheidungen erlaubt:

10.6 Definition Ein *randomisiertes Approximationsschema (RAS)* für Π ist ein randomisierter Algorithmus A, der für jede Eingabe x der Größe $n = |x|$ und für jedes $\varepsilon > 0$ eine Ausgabe $A(x)$ erzeugt, für die gilt:

$$\mathbf{P}((1-\varepsilon)\#(x) \leqslant A(x) \leqslant (1+\varepsilon)\#(x)) \geqslant \frac{3}{4}\,.$$

Ein *polynomielles randomisiertes Approximationsschema (PRAS)* ist ein RAS, dessen Laufzeit polynomiell in n ist. Ein *voll polynomielles randomisiertes Approximationsschema (FPRAS)* ist ein RAS, dessen Laufzeit polynomiell in n und $1/\varepsilon$ ist. ◇

Die Wahl der Wahrscheinlichkeit $3/4$ in Definition 10.6 ist bis zu einem gewissen Grad willkürlich. Man überlege sich als Übungsaufgabe, dass jede Konstante echt größer $1/2$ „genauso gut“ ist.

10.7 Definition Ein (ε,δ)-*FPRAS* ist ein FPRAS, das für jede Eingabe x mit einer Wahrscheinlichkeit größer oder gleich $1-\delta$ eine ε-Approximation berechnet und dessen Laufzeit polynomiell in n, $1/\varepsilon$ und $\log 1/\delta$ ist. ◇

10.8 Ein PRAS für ein Zählproblem Π kann man in einen randomisierten Algorithmus für das Entscheidungsproblem umwandeln, der zeigt, dass E_Π in **BPP** liegt.

Hätte man ein PRAS für ein **NP**-vollständiges Problem, so wäre **NP** $\subseteq$ **BPP**. Experten erwarten nicht, dass letzteres der Fall ist, und sie erwarten daher auch nicht, dass man ein PRAS für ein **NP**-vollständiges Problem findet.

10.2 Zählen von Lösungen von Formeln in DNF

10.9 Problem. Es sei $F(x_1,\ldots,x_n) = C_1 \vee \cdots \vee C_m$ mit $C_1 = l_{1,1} \wedge \cdots \wedge, l_{1,r_1}, \ldots, C_m = l_{m,1} \wedge \cdots \wedge, l_{m,r_m}$ eine Formel in DNF, so dass jedes Literal $l_{i,j}$ eine Variable x_k oder ihre Negation $\overline{x_k}$ ist und jede Variable in jeder Klausel C_i höchstens einmal (sei es normal oder negiert) vorkommt. Es sei $n \geqslant 1$ und $m \geqslant 1$.

Es bezeichne #F die Anzahl der F erfüllenden Variablenbelegungen $\mathbf{x} : \{x_1,\ldots,x_n\} \to \{0,1\}$. Die zu lösende Aufgabe besteht darin, zu gegebenem F die Zahl #F zu bestimmen.

10.10 Man weiß, dass dieses Problem **#P**-vollständig ist. Die Existenz eines deterministischen Algorithmus dafür, der in Polynomialzeit arbeitet, wird daher (analog wie bei **NP**) von vielen bezweifelt.

Wir werden im folgenden aber ein (ε,δ)-FPRAS vorstellen und analysieren.

Dazu gehen wir zunächst zu der folgenden allgemeineren Problemstellung über und untersuchen einen ersten einfachen Algorithmus.

10.11 Es sei U ein endliches „Universum“ und $G \subseteq U$ eine Teilmenge, deren Größe bestimmt werden soll. G sei gegeben mittels der charakteristische Funktion $g : U \to \{0,1\}$ von G; es sei also $g(u) = 1 \iff u \in G$. Gelegentlich benutzen wir die Abkürzung $\rho = |G|\,/\,|U|$.

Wir machen folgende Annahmen:

- g ist schnell zu berechnen, d. h. die Frage der Zugehörigkeit eines u zu G ist schnell zu entscheiden.

- Man kann schnell zufällig gleichverteilt ein u aus U auswählen.

Im Falle der Bestimmung von #F ist U die Menge aller 2^n möglichen Variablenbelegungen $\mathbf{x}$ und die Abbildung g ist gerade die durch die Auswertung $F(\mathbf{x})$ vermittelte.

10.12 Algorithmus.

$z \leftarrow 0$ ⟨*Variable zum Zählen der Lösungen*⟩
for $i \leftarrow 1$ **to** N **do**
  $u \leftarrow$ ⟨*zufällig gleichverteilt aus* U *ausgewähltes Element*⟩
  if $g(u) = 1$ **then**
    $z \leftarrow z + 1$
  fi
od
return $\frac{z}{N} \cdot |U|$

Wir gehen zunächst kurz darauf ein, dass dieser Algorithmus sinnvoll ist. Anschließend wird aber plausibel gemacht werden, dass er leider noch einen Nachteil hat.

10.13 Lemma. Es sei Y_i die Zufallsvariable mit

$$Y_i = \begin{cases} 1 & \text{falls im } i\text{-ten Schleifendurchlauf } g(u) = 1 \text{ ist} \\ 0 & \text{sonst} \end{cases}$$

und $Z = \frac{|U|}{N} \sum_{i=1}^{N} Y_i$. Dann ist $\mathbf{E}[Z] = |G|$.

10.14 Beweis. Offensichtlich ist $\mathbf{E}[Y_i] = |G| / |U|$. Wegen der Linearität des Erwartungswertes ist $\mathbf{E}[Z] = \frac{|U|}{N} \sum_{i=1}^{N} \mathbf{E}[Y_i] = \frac{|U|}{N} \cdot N \cdot \frac{|G|}{|U|} = |G|$. ∎

Durch dieses Lemma noch nicht beantwortet ist aber die Frage, wie groß man die Anzahl N der Schleifendurchläufe wählen sollte, damit die Wahrscheinlichkeit für ein „stark“ von $|G|$ abweichendes Ergebnis „klein“ ist.

Darauf gibt der folgende Satz einen Hinweis:

10.15 Satz. *Es seien δ und ε aus dem Intervall $(0; 1]$. Algorithmus 10.12 liefert mit einer Wahrscheinlichkeit größer oder gleich $1 - \delta$ eine ε-Approximation für $|G|$, falls gilt:*

$$N \geqslant \frac{5}{\varepsilon^2 \rho} \ln \frac{2}{\delta} .$$

10.16 Beweis. Es seien δ und ε beliebig aber fest. Es sei $Y = \sum_{i=1}^{N} Y_i$ und folglich $Z = |U| Y/N$.

Y ist binomialverteilt mit Erwartungswert $N\rho$ und auf Y sind die Ergebnisse über Chernoff-Schranken anwendbar, z. B. Korollar 4.21. Damit erhält man:

$$\begin{aligned} & \mathbf{P}((1-\varepsilon)|G| \leqslant Z \leqslant (1+\varepsilon)|G|) \\ = \; & \mathbf{P}((1-\varepsilon)N\rho \leqslant Y \leqslant (1+\varepsilon)N\rho) \\ = \; & 1 - \mathbf{P}((1-\varepsilon)N\rho > Y) - \mathbf{P}(Y > (1+\varepsilon)N\rho) \\ \geqslant \; & 1 - e^{-\varepsilon^2 N\rho/2} - e^{-\varepsilon^2 N\rho/5} \\ \geqslant \; & 1 - 2e^{-\varepsilon^2 N\rho/5} . \end{aligned}$$

Setzt man hier den oben angegebenen Wert für N ein, ergibt sich im Exponenten

$$-\frac{\varepsilon^2 N\rho}{5} = -\frac{\varepsilon^2 5\rho \ln \frac{2}{\delta}}{\varepsilon^2 \rho 5} = -\ln \frac{2}{\delta} = \ln \frac{\delta}{2}$$

und daher $\mathbf{P}((1-\varepsilon)|G| \leqslant Z \leqslant (1+\varepsilon)|G|) \geqslant 1 - \delta$. ∎

Der obige Satz besagt, dass man eine „gute“ Approximation erhält, wenn man unter anderem N umgekehrt proportional zu $\rho = |G| / |U|$ wählt. Man beachte, dass in der eingangs betrachteten Bestimmung von #F dieser Wert 2^{-n} sein *kann*. Dies bedeutete dann eine exponentielle Laufzeit des Algorithmus.

Und „bedauerlicherweise“ sind die Chernoff-Schranken recht gut. Das heißt, solch große Laufzeiten sind nicht nur hinreichend, sondern bei obigem einfachen Algorithmus auch notwendig.

Um zu einem *polynomiellen* RAS zu kommen, muss man also etwas anders vorgehen. Der problematische Fall oben liegt dann vor, wenn G sehr klein im Vergleich zu U ist. Wir werden im folgenden daher auf einen Algorithmus hinarbeiten, bei dem U von vorne herein kleiner ist. Für das DNF-Problem heißt das, dass man u. U. nicht mehr *alle* Variablenbelegungen in Betracht zieht, sondern nur manche.

10.17 Es sei V ein endliches Universum, von dem m Teilmengen $H_1, \ldots, H_m \subseteq V$ derart gegeben seien, dass gilt:

- Für alle i ist $|H_i|$ in Polynomialzeit berechenbar.
- Für alle i kann man aus H_i zufällig gleichverteilt ein Element auswählen.
- Für alle $v \in V$ und alle i kann man in Polynomialzeit feststellen, ob $v \in H_i$ ist oder nicht.

Die Aufgabe besteht darin, die Größe von $H = H_1 \cup \cdots \cup H_m$ zu bestimmen.

10.18 Im Fall unseres DNF-Problems werden später als H_i diejenigen Variablenbelegungen gewählt werden, die die Klausel C_i erfüllen. Da die Formel F in DNF vorliegt, ist dann $|H|$ gerade die Gesamtzahl von Variablenbelegungen, die F erfüllen.

10.19 Es sei $U = H_1 \dot\cup \cdots \dot\cup H_m$ die disjunkte Vereinigung aller H_i, also etwa $U = \{(v, i) \mid v \in H_i\}$. Offensichtlich ist $|U| \geqslant |H|$. (Da die H_i nicht paarweise disjunkt sein müssen, werden manche Elemente mehrfach gezählt.)

Es sei $\mathrm{cov}(v) = \{(v, i) \mid (v, i) \in U\}$. Offensichtlich gilt:

- Es gibt genau $|H|$ nichtleere Mengen $\mathrm{cov}(v)$.
- Die Mengen $\mathrm{cov}(v)$ sind paarweise disjunkt, partitionieren also U.
- $|U| = \sum |\mathrm{cov}(v)|$.
- Für alle $v \in V$ ist $|\mathrm{cov}(v)| \leqslant m$.

Es sei $f : U \to \{0, 1\}$ mit

$$f((v, i)) = \begin{cases} 1 & \text{falls } i = \min\{j \mid v \in H_j\} \\ 0 & \text{sonst} \end{cases}$$

und es sei $G = \{(v, i) \mid f((v, i)) = 1\}$.

Dann ist $|G| = |H|$, denn in G wird jedes $v \in H$ genau einmal gezählt, nämlich für das kleinste i mit $v \in H_i$. Zur Veranschaulichung ist die Situation in Abbildung 10.1 an einem Beispiel skizziert.

Um die Größe von $H \subseteq V$ zu bestimmen, kann man also auch die Größe von $G \subseteq U$ berechnen. Der entscheidende Punkt ist, dass im letzteren Fall die Größenverhältnisse „günstiger“ sind.

10.20 Lemma. $\rho = |G| / |U| \geqslant 1/m$.

Zum Beweis muss man nur beachten, dass $|U| = \sum_{v \in H} |\mathrm{cov}(v)| \leqslant \sum_{v \in H} m = m\,|H| = m\,|G|$ ist.

10.21 Algorithmus. Wir wenden nun das obige Schema an, um das DNF-Problem zu lösen. Als H_i wähle man die Menge aller Variablenbelegungen, die Klausel C_i erfüllen, und wende den Algorithmus wie in 10.12 beschrieben auf die Mengen G und U an, wie sie in Punkt 10.19 definiert wurden.

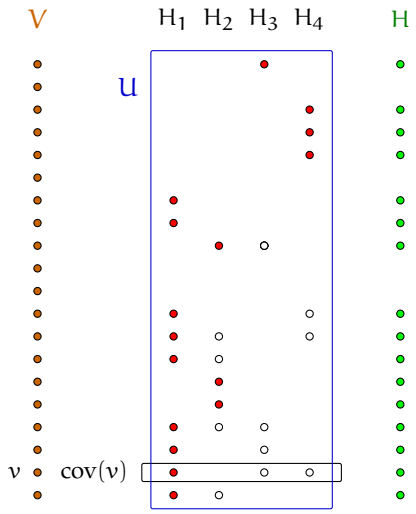

Abbildung 10.1: Ein Beispiel zu Punkt 10.19. Die Elemente von U, die zu G gehören, sind rot gekennzeichnet.

$\langle F = C_1 \vee \cdots \vee C_m$ *mit Klauseln* $C_i = l_{i,1} \wedge \cdots \wedge l_{i,r_i}$ *sei die vorgelegte Formel*$\rangle$
$\langle \delta$ *und* ε *seien die vorgegebenen Approximationsparameter*$\rangle$
$sizeU \leftarrow 0$
for $i \leftarrow 1$ **to** m **do**
  $sizeHi \leftarrow 2^{n-r_i}$
  $sizeU \leftarrow sizeU + sizeHi$
od
$N \leftarrow \frac{5m}{\varepsilon^2} \ln \frac{2}{\delta}$
$z \leftarrow 0$
for $j \leftarrow 1$ **to** N **do**
  $i \leftarrow \langle$*zufällig aus* $[1;m]$ *mit Wahrscheinlichkeit sizeHi/sizeU ausgewähltes Element*$\rangle$
  $x \leftarrow \langle$*Variablenbelegung; zufällig gleichverteilt ausgewählt aus denen, die Klausel* C_i *erfüllen*$\rangle$
  if $\neg\exists j < i : x \in H_j$ **then** $\langle x \in G\ ?\rangle$
    $z \leftarrow z + 1$
  fi
od
return $\frac{z}{N} \cdot sizeU$

10.22 Satz. *Algorithmus 10.21 ist ein* (ε, δ)*-FPRAS für das DNF-Problem.*

10.23 Beweis. Die Größe der Eingabe liegt in $\Omega(n+m)$ und in $O(nm)$. Nach Lemma 10.20 ist $\rho \geqslant 1/m$. Wir wollen nun Satz 10.15 anwenden, um folgern zu können, dass Algorithmus 10.21 tatsächlich ein (ε, δ)-FPRAS für das DNF-Problem ist.

Als H_i wird die Menge aller Variablenbelegungen benutzt, die Klausel C_i erfüllen. Es ist also $|H_i| = 2^{n-r_i}$. Damit muss noch bewiesen werden, dass in der zweiten Schleife durch die Zuweisungen an i und x stets zufällig ein Element aus U *gleichverteilt* ausgewählt wird. Die Wahrscheinlichkeit, ein bestimmtes (x,i) auszuwählen, ist aber gerade $|H_i| / \sum |H_i| \cdot 1/|H_i| = 1/\sum |H_i| = 1/|U|$.

Also ist Algorithmus 10.21 tatsächlich ein Spezialfall von Algorithmus 10.12.

Die Anzahl Schleifendurchläufe ist $N = \frac{5m}{\varepsilon^2} \ln \frac{2}{\delta}$, also polynomial in $1/\varepsilon$, $\log 1/\delta$ und in der Problemgröße. Wegen der Voraussetzungen aus Punkt 10.19 ist daher auch die Gesamtlaufzeit polynomial in den genannten Parametern. ■

