Approximations-Schemata, Estimator-Theorem, Expansion des Universums und Uniformes Sampling am Beispiel von # DNF

Sommerakademie 2007

Peter Zaspel

Inhaltsverzeichnis

1 Einleitung

2 Übersicht

Diese Ausarbeitung beschäftigt sich mit der approximativen Lösung von kombinatorischen Zählproblemen. Dazu wird im Folgenden erst der Begriff eines solchen Zählproblems näher erläutert und dessen Komplexität beschrieben.
In dem darauffolgenden Abschnitt wird die Methode der Expansion des Universums vorgestellt. Als allgemeine Definition folgt dann eine Übersicht über randomisierte Approximationsschemata.
Um die Expansion des Universums praktisch anwenden zu können, wird im nächsten Abschnitt die Monte-Carlo-Methode vorgestellt.
Fasst man alle beschriebenen Methoden zusammen, so erhält man ein Verfahren, mit dem ein kombinatorisches Zählproblem angenähert werden kann. Eine solche konkrete Anwendung wird im vorletzten Abschnitt am Beispiel des # DNF-Problems beschrieben.

2.1 Definitionen für kombinatorische Zählprobleme

Um sich auf eine gemeinsame Schreibweise zu verständigen, folgt eine allgemeine Definition eines kombinatorischen Optimierungsproblems:

Definition 1 (kombinatorisches Optimierungsproblem Π)
Gegeben durch:

- D*: Menge der (zulässigen) Eingaben*
- $\mathcal{S}(\mathcal{I})$ *mit* $I \in D$*: Menge der zulässigen Lösungen zu Eingabe* I
- $f : S(I) \rightarrow \mathbb{N}^{\neq 0}$ *(Bewertungsfunktion)*
- $ziel \in \{min, max\}$

Gesucht:
für $I \in D$ *eine Lösung* $\sigma_{opt} \in S(I)$ *so dass*

$$f(\sigma_{opt}) = ziel\{f(\sigma) \mid \sigma \in S(I)\}$$

$OPT(I) = f(\sigma_{opt})$ *ist der Wert einer optimalen Lösung.*

Ein kombinatorisches Zählproblem zielt im Gegensatz zu einem Optimierungsproblem nicht auf eine Maximierung bzw. Minimierung einer Bewertungsfunktion. Vielmehr geht es hierbei um die Anzahl der zulässigen Lösungen:

Definition 2 (kombinatorisches Zählproblem $\#\Pi$)
Gegeben:
kombinatorisches Optimierungsproblem Π

Gesucht:
$\#(I) = |S(I)|$ *(Anzahl zulässiger Lösungen zu Probleminstanz I)*

2.2 Beispiele

Am einfachsten macht man sich die Ziele von kombinatorischen Zählproblemen an ein paar Beispielen deutlich.

Beispiel 1 (Rucksack-Zählproblem $\#RUCKSACK$)
Gegeben: $RUCKSACK$ *(Standard-Knappsack-Problem)*
Gesucht: *Anzahl der Rucksackfüllungen, die maximales Füllgewicht nicht überschreiten*

Das Standard-Knappsack-Problem ist hierbei über eine gegebene Menge von Füll-Objekten mit Gewichten und ein maximales Füllgewicht für den Rucksack gegeben.

Beispiel 2 (Färbungen-Zählproblem $\#COL_k$)
Gegeben: COL_k *(Graph-Färbbarkeitsproblem mit k Farben)*
Gesucht: *Anzahl der korrekten Knotenfärbungen mit $\leq k$ Farben*

Das letzte Beispiel ist hier von besonderem Interesse, da im vorletzten Abschnitt ein Approximationsverfahren für dieses Problem vorgestellt wird:

Beispiel 3 (DNF-Zählproblem $\#DNF$)
Gegeben:
Probleminstanz Ψ ist Boolsche (n,m)-Formel in DNF über n Variablen $V = \{x_1, ..., x_n\}$
Also: $\Psi = C_1 \vee ... \vee C_m$ mit C_i Monome der Länge k_i bestehend aus Literalen x_j und $\bar{x}_k$. (Pro Monom kommt eine Variable maximal einmal vor.)
Gesucht:
Anzahl der zulässigen Belegungen, d.h. Belegung $b_\Psi : V \rightarrow \{TRUE, FALSE\}$ der Variablen so, dass Ψ erfüllt wird.

2.3 Komplexität der Aufgabe

Die im Zusammenhang von kombinatorischen Zählproblemen wichtige Komplexitätsklasse ist die Klasse $\#P$.
Umgangssprachlich könnte man diese Klasse dadurch beschreiben, dass in ihr alle Zählprobleme enthalten sind, deren zugehörige Entscheidungsprobleme in der Klasse NP liegen. Wie in NP lässt sich auch in $\#P$ der Begriff der $\#P$-Vollständigkeit angeben. Dieser überträgt sich analog. Also: Ein Problem p in $\#P$ ist $\#P$-vollständig, wenn sich alle Probleme aus $\#P$ mittels Polynomialzeitreduktion auf p reduzieren lassen.
Sinnvoll zum Verständnis der Komplexität von $\#P$-vollständigen Problemen ist das folgende Lemma:

Lemma 1
Wenn $P = NP$, dann lassen sich $\#P$-vollständige Probleme in Poly-Zeit lösen.

Diese Aussage lässt vermuten, dass im Falle von $P \neq NP$ ein solches Problem sehr schwer lösbar ist. So lange das exakte Verhältnis von P und NP ungeklärt ist, kann man also vermuten, dass sich $\#P$-vollständige Probleme unangenehm verhalten, also kein Polynomzeit-Algorithmus für die exakte Lösung existiert.

Das später approximierte # DNF ist gerade ein solches Problem:

Satz 1
$\#DNF$ *ist* $\#P$*-vollständig.*

3 Expansion des Universums

Kombinatorische Zählprobleme suchen nach der Kardinalität der Menge aller zulässigen Lösungen $S(I)$. Gut wäre es, wenn man eine obere Schranke für diese Kardinalität hätte. Diese ergibt sich durch die Angabe einer Obermenge, deren Größe bekannt ist. Zusammen mit einer Abschätzung für das Verhältnis von der tatsächlich gesuchten Größe und dieser oberen Schranke kann eine näherungsweise Aussage für die gesuchte Größe gemacht werden. Genau dies ist die Idee, die hinter der *Expansion des Universums* steckt:

Definition 3 (Expansion des Universums)
Gegeben:

- I *Instanz von* $\#\Pi$
- U_I *Menge mit* $S(I) \subseteq U_I$ *mit* $|U_I|$ *bekannt*
  U_I *ist das Universum von* $S(I)$

$$\xi = \frac{|U_I|}{\#(I)} = \frac{|U_I|}{|S(I)|}$$

  ξ *ist die Expansion des Universums*

Häufig wird im Zusammenhang mit Approximationsalgorithmen nach der relativen Güte einer genäherten Lösung gefragt:

Definition 4 (Relative Güte ρ_A)
Gegeben:

- $\#\Pi$ *kombinatorisches Zählproblem*
- $\#(I)$ *exakte Lösung*
- *Approximationsalgorithmus* A
- $A(I)$ *approximative Lösung des Problems*
  (mit $A(I) \leq \#(I)$ *und* $A(I) \geq \#(I)$ *erlaubt)*

relative Güte der Approximation:

$$\rho_A(I) = \max\left\{\frac{A(I)}{\#(I)}, \frac{\#(I)}{A(I)}\right\}$$

$\rho_A(n)$ *ist relative Güte bei Eingaben der Länge* $\leq n$.

Ist die relative Güte bekannt, so lässt sich die approximierte Lösung nach oben und unten abschätzen:

Lemma 2
Mit $|I| = n$ *gilt:*

$$\frac{1}{\rho_A(n)}\#(I) \leq A(I) \leq \rho_A(n)\#(I)$$

Zentraler Satz

Für die Näherung durch Expansion des Universums lässt sich eine Aussage zur relativen Güte machen.

Satz 2
Sei s *eine obere Schranke für* ξ*, also:* $\xi \leq s$.

Dann hat die Approximation

$$A(I) = \frac{|U_I|}{\sqrt{s}}$$

für den Wert $\#(I)$ *relative Güte von* $\sqrt{s}$.

4 Randomisierte Approximationsschemata

Bei der Näherung von kombinatorischen Zählproblemen $\#\Pi$ werden typischerweise Algorithmen A verwendet, die unter Eingabe einer Instanz I von $\#\Pi$ und von ϵ mit $0 < \epsilon < 1$ eine Näherung $A(I, \epsilon)$ berechnen. Für diese Algorithmen lassen sich Klasseneinteilungen angeben:

Definition 5 (polynomielles Zähl-Approximationsschema (PASC))
A *ist PASC für* $\#\Pi$*, falls* A *deterministisch, Laufzeit von* A $T(A) = O(poly(|I|))$ *und:*

$$|A(I, \epsilon) - \#(I)| \leq \epsilon\#(I)$$

Definition 6 (*streng* polynomielles Zähl-Approximationsschema (FPASC))
A *ist PASC und* $T(A) = O(poly(|I|, \frac{1}{\epsilon}))$

Neben diesen deterministischen Algorithmen, bei denen ein relativer Fehler von maximal ϵ garantiert wird, gibt es randomisierte Algorithmen, bei denen keine solchen Garantien mehr möglich sind. Bei den randomisierten Verfahren kann man nur noch Wahrscheinlichkeiten dafür angeben, dass eine Schranke für einen relativen Fehler auch eingehalten wird:

Definition 7 (polynomielles randomisiertes Zähl-Approximationsschema)
*(kurz: **PRASC**)*
A *hat Laufzeit* $T(A) = O(poly(|I|))$ *und es gilt:*

$$Pr[|A(I,\epsilon) - \#(I)| \leq \epsilon \cdot \#(I)] \geq \frac{3}{4}$$

Definition 8 (*streng* polynomielles randomisiertes Zähl-Approximationsschema)
*(kurz: **FPRASC**)*
A *ist PRASC und hat Laufzeit* $T(A) = O(poly(|I|, \frac{1}{\epsilon}))$

Definition 9 ((ϵ,δ)-FPRASC)
A *ist ein FPRASC mit zusätzlicher Eingabe* δ *mit* $0 < \delta < 1$*, hat Laufzeit* $T(A) = O(poly(|I|, \frac{1}{\epsilon}, \log(\frac{1}{\delta})))$ *und erzeugt eine Ausgabe* $A(I,\epsilon,\delta)$ *mit:*

$$Pr[|A(I,\epsilon,\delta) - \#(I)| \leq \epsilon \cdot \#(I)] \geq 1 - \delta$$

Das Besondere an den (ϵ,δ)-FPRASC ist, dass man durch Vorgabe eines δ die Wahrscheinlichkeit für eine Einhaltung der Fehlerschranken beliebig hoch treiben kann. So lassen sich tatsächlich Lösungen beliebiger Genauigkeit erzeugen. Es ist also stets wünschenswert, ein (ϵ,δ)-FPRASC zur Verfügung zu haben.
Unter Vorgabe eines streng polynomiellen randomisierten Zähl-Approximationsschemas, lässt sich genau ein solches (ϵ,δ)-Verfahren erzeugen. Dies geschieht durch den $\Theta(\log(\frac{1}{\delta}))$-maligen Aufruf vom FPRASC-Algorithmus und anschließende Mittelung:

Algorithmus 1 (AMPL(A;ϵ,δ,I))
`for` τ`:= 1 to` T_δ `do`
    N_τ`:=` $A(I,\epsilon)$`;`
`return` $\frac{1}{T_\delta} \cdot \sum_{\tau=1}^{T_\delta} N_\tau$

Der folgende Satz formalisiert dieses Ergebnis:

Satz 3 (Wahrscheinlichkeitsverstärkung)
Gegeben:

- $\#\Pi$ *kombinatorisches Zählproblem*
- A *ein FPRASC für* $\#\Pi$
- $\delta < 1$

Mit $T_\delta = 8\lceil \ln(\frac{1}{\delta}) \rceil$ *ist AMPL(A;*ϵ*,*δ*,I) ein (*ϵ*,*δ*)-FPRASC für* $\#\Pi$.

Es ist also tatsächlich möglich aus einem FPRASC ein (ϵ,δ)-FPRASC zu erzeugen.

5 Monte-Carlo-Methode

5.1 Algorithmus

Die Monte-Carlo-Methode ist ein randomisiertes Verfahren bei dem durch wiederholtes Ziehen von Stichproben aus einer Menge, anschließendes Bewerten dieser Stichproben und Mittelung über die Ergebnisse eine Näherung an einen gesuchten Wert bestimmt werden kann.
Für die hier zu behandelnde Anwendung der Methode auf kombinatorische Zählprobleme $\#\Pi$ sei im Folgenden ein Universum U_I zur jeder Instanz I gegeben. Die Kardinalität von U_I sei bekannt. Im Zusammenhang der Monte-Carlo-Methode wird U_I auch als Stichprobenraum bezeichnet. Zudem sei mit $\xi = \frac{|U_I|}{\#(I)}$ die Expansion des Universums gegeben. $\chi : U_I \rightarrow \{0,1\}$ sei die charakteristische Funktion von $S(I)$, die wie folgt definiert ist:

$$\chi(u) = \begin{cases} 1 & falls\ u \in S(I) \\ 0 & sonst. \end{cases}$$

Um den Monte-Carlo-Algorithmus anwenden zu können, benötigt man zwei Algorithmen. Der erste ist ein Algorithmus UG mit Laufzeit $O(Poly(|I|))$, der Elemente (*Stichproben*) u aus U_I ausgibt, so dass für alle $u \in U_I$

$$Pr[u\ wird\ von\ UG\ ausgegeben] = \frac{1}{|U_I|}$$

gilt. Diesen Algorithmus nennt man auch *uniformen Stichprobengenerator*. Der zweite Algorithmus (BEANTWORTER) ist eine effiziente (also mit Laufzeit $O(Poly(|I|))$) Implementation der charakteristischen Funktion χ.

Hat man beide Algorithmen gegeben, kann man damit den Monte-Carlo-Algorithmus zur Annäherung an das kombinatorische Zählproblem durchführen:

Algorithmus 2 (Monte-Carlo-Algorithmus MC(T))

for $i := 1$ *to* T *do*
    (1) ziehe eine Stichprobe $u \in U_I$ *mittels UG;*
    (2) $Y_i := \chi(u)$ *mittels BEANTWORTER*
done;
$R := \frac{1}{T} \cdot \sum_{i=1}^{T} Y_i$*;*
gib $Z := R \cdot |U_I|$ *aus.*

In diesem Algorithmus wird also T Mal eine Stichprobe aus dem Universum gezogen und bewertet ob diese Stichprobe in der gesuchten Menge $S(I)$ liegt. Der Mittelwert R über die Auswertungen nähert sich für große T immer stärker der Inversen der Expansion des Universums ξ^{-1} an. Durch Multiplikation mit der Kardinalität des Universums erhält man eine Näherung für die gesuchte Größe. Diese Aussagen fasst das folgende Lemma zusammen:

Lemma 3

1. $E[R] = \xi^{-1}$ *($E[Y_i] = \xi^{-1}$ folgt aus Uniformität $\Rightarrow$ Beh.)*

2. $E[MC(T)] = \#(I) \quad (E[MC(T)] = E[Z] = \xi^{-1} \cdot |U_I| = \#(I))$

Da das Lemma nur Aussagen über die erwarteten Ergebnisse macht, liegt es nahe zu fragen, wie groß die Abweichungen von diesen Erwartungswerten (die Varianzen der Zufallsvariablen) sein können. Das folgende Lemma gibt Aufschluss darüber, wie häufig man Stichproben ziehen muss, um die Varianz klein zu bekommen:

Lemma 4

1. $Var[R] = \frac{\xi^{-1} \cdot (1-\xi^{-1})}{T}$
2. $Var[MC(T)] = |U_I|^2 \cdot \frac{\xi^{-1} \cdot (1-\xi^{-1})}{T}$

Beweis 1

1. Y_i *sind 0-1-Zufallsvariablen*
   $\Rightarrow Var[Y_i] = E[Y_i] \cdot (1 - E[Y_i]) = \xi^{-1} \cdot (1 - \xi^{-1})$
   $\Rightarrow Var[R] = Var\left[\frac{1}{T} \cdot \sum_{i=1}^{T} Y_i\right] = \frac{1}{T^2} \cdot \sum_{i=1}^{T} Var[Y_i] = \frac{\xi^{-1} \cdot (1-\xi^{-1})}{T}$
2. $Var[MC(T)] = Var[|U_I| \cdot R] = |U_I|^2 \cdot Var[R] = |U_I|^2 \cdot \frac{\xi^{-1} \cdot (1-\xi^{-1})}{T}$

5.2 Estimator-Theorem

Das folgende Estimator-Theorem liefert eine Aussage darüber, dass die Wahrscheinlichkeit für einen kleinen relativen Fehler bei der Ausgabe des Monte-Carlo-Algorithmus groß ist.

Satz 4 (Estimator-Theorem der Monte-Carlo-Methode)
Gegeben: $\epsilon > 0$*, beliebig, aber fest*

Mit $T_\xi(\epsilon) = \lceil \frac{4}{\epsilon^2} \cdot (\xi - 1) \rceil$ *gilt:*

$$Pr[|MC(T_\xi(\epsilon)) - \#(I)| \leq \epsilon \cdot \#(I)] \geq \frac{3}{4}$$

Beweis 2

$$Pr[|MC(T_\xi(\epsilon)) - \#(I)| \geq \epsilon \cdot \#(I)]$$
$$\overset{Tschebyscheff}{\leq} \frac{1}{\epsilon^2} \cdot \frac{Var[MC(T_\xi(\epsilon)]}{E[MC(T_\xi(\epsilon)]^2} \overset{Lemmata}{=} \frac{1}{\epsilon^2} \cdot \frac{|U_I|^2 . \frac{\xi^{-1} \cdot (1-\xi^{-1})}{T_\xi(\epsilon)}}{\#(I)^2}$$
$$\overset{Def.\xi}{=} \frac{1}{\epsilon^2} \cdot \xi^2 \cdot \frac{\xi^{-1} \cdot (1-\xi^{-1})}{T_\xi(\epsilon)} = \frac{1}{\epsilon^2} \cdot (\xi - 1) \cdot \frac{1}{T_\xi(\epsilon)} \overset{Def.T_\xi(\epsilon)}{\leq} \frac{1}{4}$$

Auf den ersten Blick mag man denken, dass dies genau die Definition eines FPRASC ist. Dies ist aber nicht der Fall, da die Anzahl der Stichproben T im Algorithmus linear in der Größe ξ ist. Dadurch ergibt sich eine Abhängigkeit vom gesuchten Wert $\#(I)$, die bei einem FPRASC nicht vorkommen darf. Schafft man es aber, eine konstante obere Schranke für ξ zu finden wird der Wert T unabhängig von der gesuchten Größe und der Algorithmus wird genau zu einem FPRASC.
Eine solche Abschätzung ist aber nicht für alle Probleme identisch, so dass man konkret für jedes Problem, auf dass man den Algorithmus anwenden möchte, eine solche

Schranke finden muss. Dies wird unter anderem im folgenden Abschnitt anhand des Beispiels $\#DNF$ vorgeführt.

6 Anwendungsbeispiel: # DNF

6.1 Grundidee

Setzt man die Methoden und Algorithmen aus den vorhergehenden Abschnitten zusammen, so bekommt man eine Art *Rezept* für die Fertigstellung eines Approximationsalgorithmus für kombinatorische Zählprobleme. Benötigt wird ein „Vernünftiges Universum für $S(I)$, eine Abschätzung für ξ, ein BEANTWORTER-Algorithmus und ein uniformer Generator für das Universum. Damit kann man den Monte-Carlo-Algorithmus ausführen und erhält grob formuliert für $\#DNF$ folgendes Ergebnis:

Es gibt ein (ϵ,δ)-FPRASC für $\#DNF$ mit der Laufzeit $O(const \cdot n \cdot \frac{1}{\epsilon} \cdot \log(\frac{1}{\delta}))$

Sei im Folgenden das DNF-Zählproblem gegeben, wie es im ersten Abschnitt als Beispiel angegeben wurde:

Beispiel 4 (DNF-Zählproblem $\#DNF$)
Gegeben:
Probleminstanz Ψ ist Boolsche (n,m)-Formel in DNF über n Variablen $V = \{x_1, ..., x_n\}$
Also: $\Psi = C_1 \vee ... \vee C_m$ mit C_i Monome der Länge k_i bestehend aus Literalen x_j und $\bar{x}_k$. (Pro Monom kommt eine Variable maximal einmal vor.)
Gesucht:
Anzahl $\#(\Psi)$ der zulässigen Belegungen, d.h. Belegung $u : V \rightarrow \{TRUE, FALSE\}$ der Variablen so, dass Ψ erfüllt wird ($u(\Psi)$=TRUE).

6.2 Wahl des Universums für # DNF

Um ein „vernünftiges“ (also möglichst kleines) Universum für $\#DNF$ angeben zu können mache man zunächst folgende zwei Bemerkungen.

Bemerkung 1
Erfüllende Belegungen für Ψ sind einfach bestimmbar, da die Erfüllung einer Klausel ausreicht.

Bemerkung 2
Die Anzahl $\#(C_j)$ der erfüllenden Belegungen einer einzelnen Klausel C_j ist unmittelbar berechenbar.

Die erste Bemerkung wird klar, wenn man bedenkt, dass eine DNF eine „Veroderung“ von Monomen ist. Ist ein Monom (eine Klausel) erfüllt, wird die gesamte DNF erfüllt. Eine solche Bemerkung hilft beim Bestimmen einer erfüllenden Belegung für eine DNF. Dafür wählt man einfach die Variablen so, dass ein Monom erfüllt wird und „würfelt“ alle restlichen Variablen aus.
Für das genaue Verständnis der zweiten Bemerkung ist folgendes Lemma nützlich:

Lemma 5
Gegeben: *$C = l_1 \wedge ... \wedge l_k$ Klausel der Boolschen (n,m)-Formel Ψ in DNF aus k Literalen.*

Dann gilt: Es gibt genau 2^{n-k} Belegungen, die C erfüllen. $\Rightarrow \#(C) = 2^{n-k}$.

Der Beweis ist ein einfaches Abzählargument.

Nun kann man sich für das $\#DNF$-Problem die Menge, deren Kardinalität bestimmt werden soll, näher anschauen. Dabei erkennt man folgende Beziehung:

$$\begin{aligned} S(\Psi) &= \bigcup_{j=1}^{m} \{u | u\ erfüllt\ C_j\} \\ &= \bigcup_{j=1}^{m} \{u | u\ erfüllt\ C_j, aber\ kein\ C_k, k < j\} \end{aligned}$$

Zu dieser Darstellung der Menge lässt sich ein Menge angeben, bei der offensichtlich die Kardinalität identisch ist:

$$S'(\Psi) = \bigcup_{j=1}^{m} \{(u, j) | u\ erfüllt\ C_j, aber\ kein\ C_k, k < j\}$$

Es gilt also:

$$\#(\Psi) = |S(\Psi)| = |S'(\Psi)|$$

Aber zu der neu definierten Menge $S'(\Psi)$ kann man wiederum leicht eine Obermenge angeben, deren Größe bekannt ist. Das Universum für $S'(\Psi)$ ist also:

Definition 10 (Ein Universum für $S'(\Psi)$)

$$\begin{aligned} U_\Psi &= \{(u, j) | u\ erfüllt\ C_j\} \\ &= \dot{\bigcup_{j=1}^{m}} \{(u, j) | u\ erfüllt\ C_j\} \end{aligned}$$

Dass es sich bei der Menge U_Ψ um eine Obermenge handelt, sollte direkt ersichtlich sein. Die exakte Größe dieser Menge ergibt sich durch Anwendung des Lemmas 5 zu:

$$|U_\Psi| = \sum_{j=1}^{m} 2^{n-k_j}$$

Dies ist klar, da das Universum die disjunkte Vereinigung über erfüllende Belegungen für jeweils genau eine Klausel ist. Die Anzahl der erfüllenden Belegungen pro Klausel ergibt sich genau aus dem Lemma.

6.3 Abschätzung für ξ

Als nächster Schritt muss eine obere Schranke für die Expansion des Universums gefunden werden. Diese liefert uns das Expansionslemma:

Lemma 6 (Expansionslemma)

$$\xi = \frac{|U_\Psi|}{|S'(\Psi)|} \leq m$$

(m die Anzahl der Klauseln)

Beweis 3
Das Universum $U_\Psi = \bigcup_{j=1}^{m} \{(u,j)|u\ erfüllt\ C_j\}$ enthält Tupel die aus allen erfüllenden Belegungen u und Werten j mit $j = 1, ..., m$ bestehen. j kann offensichtlich m Werte annehmen. u nimmt genau so viele Werte an, wie es erfüllende Belegungen gibt, also $|S(\Psi)|$. Damit folgt: $|U_\Psi| \leq m \cdot |S(\Psi)|$, was zur Behauptung führt.

6.4 BEANTWORTER-Algorithmus

Als weiterer wichtiger Schritt zur Anwendung des Monte-Carlo-Algorithmus gilt das Auffinden eines deterministischen Polynom-Zeit-Algorithmus für die charakteristische Funktion χ von $S'(\Psi)$ mit

$$\chi((u,j)) = \begin{cases} 1 & falls\ j = \min\{k|u\ erfüllt\ C_k\} \\ 0 & sonst. \end{cases}$$

Ein solcher Algorithmus ist aber schon durch eine leicht modifizierte Auswertung der DNF gegeben, die ganz offensichtlich deterministisch in Zeit $O(m \cdot n)$ möglich ist.

6.5 Uniformer Generator für das Universum

Die letzte „Zutat“ die noch fehlt, ist der Uniforme Generator UG. Dieser lässt sich durch folgenden Algorithmus beschreiben:

Algorithmus 3 (UG)

1. würfele ein $j \in \{1, ..., m\}$ mit Wahrscheinlichkeit $\frac{2^{n-k_j}}{|U_\Psi|}$;
2. setze die in C_j auftauchenden Variablen so, dass C_j wahr wird;
3. würfele eine Belegung der restlichen Variablen;
4. gib j und die teils berechnete, teils gewürfelte Belegung u aus.

UG ist genau der gesuchte Algorithmus, wie folgendes Lemma zeigt:

Lemma 7
Algorithmus UG ist ein uniformer Generator für U_Ψ.

Beweis 4
Sei u, j beliebig aber fest. Dann gilt:
$Pr[(u,j) \in U_\Psi \; wird \; gewählt]$
$= Pr[j \in \{1,..,m\} \; wird \; in \; Zeile \; 1 \; ausgewürfelt$
$\wedge \; die \; C_j \; erfüllende \; Belegung \; u \; wird \; ausgegeben]$
j wird vom Algorithmus mit Wahrscheinlichkeit $\frac{2^{n-k_j}}{|U_\Psi|}$ ausgewürfelt. Dass eine erfüllende Belegung für C_j ausgegeben wird ist durch die Formulierung des Algorithmus klar. Es bleibt also als anzugebende Wahrscheinlickeit für das zweite Ereignis die Wahrscheinlichkeit, mit der genau u und nicht eine andere erfüllende Belegung ausgegeben wird. Diese Wahrscheinlichkeit ergibt sich aus der Anzahl der möglichen erfüllenden Belegungen für C_j zu $\frac{1}{2^{n-k_j}}$. Es folgt:

$$Pr[(u,j) \in U_\Psi \; wird \; gewählt] = \frac{2^{n-k_j}}{|U_\Psi|} \cdot \frac{1}{2^{n-k_j}} = \frac{\mathbf{1}}{|\mathbf{U_\Psi}|}$$

6.6 Monte-Carlo-Algorithmus

Mit dem Universum U_ψ, der oberen Schranke für die Expansion ξ, dem BEANTWORTER-Algorithmus und dem uniformen Stichprobengenerator UG sind alle nötigen Voraussetzungen gegeben, um den Monte-Carlo-Algorithmus (MC) ausführen zu können. Dafür gilt nun folgender Satz:

Satz 5
Mit $T(\epsilon,\delta) = \lceil m \cdot \frac{4}{\epsilon^2} \ln \frac{2}{\delta} \rceil$ ist Algorithmus $MC(T(\epsilon,\delta))$ ein (ϵ,δ)-FPRASC für $\#DNF$ der Laufzeit $O(m^2 \cdot n \cdot \frac{1}{\epsilon} \cdot \log \frac{1}{\delta})$.

Beweis 5
Der Beweis ergibt sich durch das Zusammenfügen aller bisher zusammengestellten Informationen:
Zunächst führt man den Monte-Carlo-Algorithmus für das Universum U_ψ aus. Das Estimator-Theorem liefert eine Abschätzung für Fehlerwahrscheinlichkeit:

$$Pr[|MC(T_\xi(\epsilon)) - \#(I)| \leq \epsilon \cdot \#(I)] \geq \frac{3}{4}$$

Diese macht den Monte-Carlo-Algorithmus aber noch nicht zu einem FPRASC. Durch die Hinzunahme der Abschätzung für die Expansion (Expansionslemma) erfüllt er genau die Anforderungen an ein FPRASC.
Aus dem FPRASC wird dann mit dem Satz 3 zur Wahrscheinlichkeitsverstärkung und dem dazugehörigen Algorithmus ein (ϵ,δ)-FPRASC. Rechnet man die ganzen Laufzeit-Werte durch, bekommt man genau das Ergebnis aus dem Satz.

Insgesamt lässt sich also ein (ϵ,δ)-FPRASC für die Approximation des kombinatorischen Zählproblems $\#DNF$ angeben.

7 Zusammenfassung

Ziel dieser Ausarbeitung war es, Verfahren vorzustellen, um kombinatorische Zählprobleme approximativ zu lösen. Dies wird nötig, da diese Problemklasse vermutlich schwer zu berechnen ist. Die in den ersten Abschnitten vorgestellten Verfahren liefern ein generelles Schema, um sich solchen Problemen zu nähern. Dieses Schema wurde beispielhaft auf $\#DNF$ angewandt und liefert als Ergebnis ein (ϵ, δ) - streng polynomielles randomisiertes Zähl-Approximationsschema.

Literatur

[1] Rolf Wanka: Approximationsalgorithmen - Eine Einführung.
B.G. Teubner, Stuttgart-Leipzig, 2006