**Ferienakademie im Sarntal — Kurs 1**
**Moderne Suchmethoden der Informatik: Trends und Potenzial**

# Counting the Number of Satisfying Assignments

Stefan Ploner
Friedrich-Alexander-Universität Erlangen-Nürnberg

18. Oktober 2014

## 1 Einführung

Diese Ausarbeitung beschreibt einen Approximationsalgorithmus zum Lösen des $\#k$-SAT Problems, also zur näherungsweisen Bestimmung der Anzahl an Lösungen einer $k$-SAT Formel. Dazu werden zunächst in Kapitel 1 grundlegende Begriffe erklärt. In Kapitel 2 wird beschrieben, wie sich die Monte-Carlo Methode auf $\#k$-SAT anwenden lässt. Anschließend wird in Kapitel 3 auf Eliminationsbäume und $\ell$-Schnitte eingegangen, die kombiniert mit dem Monte-Carlo Ansatz aus Kapitel 2 zu einem hybriden Algorithmus führen, wie er in [2] vorgestellt wurde. In Kapitel 4 folgt eine Reduzierung der Größe der verwendeten Eliminationsbäume und in Kapitel 5 werden unabhängige Klauseln und Rekursion kombiniert, um die Laufzeit des Algorithmus noch weiter zu reduzieren. Die Verfahren der letzten beiden Kapitel, sowie die Abbildungen 3 und 5, stammen aus [1]. Abschließend wird ein Ausblick auf weitere mögliche Verbesserungen gegeben.

Eine Formel des allgemeinen Erfüllbarkeitsproblems (SAT) ist eine boolsche Formel über $n$ Variablen, die in konjunktiver Normalform (CNF) vorliegt. Eine CNF besteht aus Literalen, also nichtnegierten und negierten Variablen, die zunächst zu Klauseln mit der logischen Oder-Operation verknüpft werden. Die logische Verundung aller $m$ Klauseln bildet dann die CNF (siehe Abbildung 1).

| | | |
|---|---|---|
| Literale: | $l = \{x_i, \overline{x}_i \mid i \in \{1, \dots, n\}\}$ | $\overline{x}_2$ |
| Klauseln: | $C_j = l_1 \vee \dots \vee l_{k_j}$ | $(x_1 \vee \overline{x}_2)$ |
| CNF: | $\Phi = C_1 \wedge \dots \wedge C_m$ | $(x_1 \vee \overline{x}_2) \wedge (\overline{x}_3 \vee x_2)$ |

Abbildung 1: Eine CNF über $n$ Variablen, bestehend aus $m$ Klauseln $C_j$ mit jeweils $k_j$ Literalen.

Enthält jede Klausel einer CNF maximal $k$ Literale, so wird die Formel als $k$-CNF bezeichnet, das Erfüllbarkeitsproblem entsprechend $k$-SAT. Tabelle 1 zeigt die benötigte Laufzeit später verwendeter Algorithmen zum Lösen von $k$-SAT Problemen.

Entgegen dem gewöhnlichen Erfüllbarkeitsproblem, bei dem lediglich entschieden werden muss, ob eine Lösung existiert oder nicht, muss beim $\#k$-SAT Problem die Anzahl der Lösungen berechnet werden. Das Problem #2-SAT ist mit Wahlström's Algorithmus deterministisch in $\mathcal{O}^*(1.2377^n)$

| | Laufzeit | $\beta_k$ |
|---|---|---|
| $k = 3$ | $\mathcal{O}^*(1.30704^n = 2^{0.3864n})$ | 0.3864 |
| $k = 4$ | $\mathcal{O}^*(1.46899^n = 2^{0.5548n})$ | 0.5548 |

Tabelle 1: Laufzeiten der $k$-SAT Löser aus [4].

lösbar [3]. Diese Ausarbeitung beschreibt die Spezialfälle $k = 3$ und $k = 4$ eines in [1] vorgestellten randomisierten Zählapproximationsschemas (RASC), das in seiner allgemeinen Form auch für beliebige $k \geq 5$ genutzt werden kann.

Definition: Erfüllt ein Algorithmus $A$ folgende Eigenschaft, so ist er ein RASC für $\#(I)$:

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

- $I$: Instanz des Zählproblems
- $\#(I)$: exakte Lösung
- $A(I, \varepsilon)$: berechnete Lösung
- $\varepsilon$: obere Schranke der Abweichung

Die Abweichung von der exakten Lösung ist also mit Wahrscheinlichkeit größer als $\frac{3}{4}$ kleiner als $\varepsilon \cdot \#(I)$. Erfüllt die Ausgabe eines Approximationsalgorithmus die RASC-Eigenschaft, so lässt sich mit Hilfe von polynomiell vielen Wiederholungen die Wahrscheinlichkeit, dass die Ausgabe des Algorithmus innerhalb eines bestimmten Fehlerintervals liegt, beliebig nahe an 1 annähern. Diese Methode, bei der der Median von Mittelwerten berechnet wird, wird in der Literatur als *median of means*-Methode bezeichnet.

## 2 Anwendung der Monte-Carlo Methode auf #k-SAT

Die Monte-Carlo Methode ist ein Verfahren zum Abschätzen eines Verhältnisses zwischen einem Stichprobenraum $\mathcal{U}_I$ bekannter Größe $|\mathcal{U}_I|$ und einer darin enthaltenen Teilmenge $\mathcal{S}_I$. Dazu werden gleichverteilt Elemente aus dem Stichprobenraum ausgewählt und geprüft, ob sie auch in der Teilmenge enthalten sind. Anschließend kann man aus der Größe des Stichprobenraumes und dem geschätzten Verhältnis $R$ zwischen beiden Mengen auf die Größe der Teilmenge $|\mathcal{S}_I|$ schließen. Codebeispiel 1 zeigt den zugehörigen Pseudocode. Aus der Literatur, siehe zum Beispiel [5], ist bekannt, dass mit einer ausreichenden Anzahl $T$ an Stichproben die resultierende Approximation die RASC-Eigenschaft erfüllt.

Satz (Estimator Theorem der Monte-Carlo Methode):
Sei $\varepsilon > 0$. Mit $T(\varepsilon) \geq \frac{4}{\varepsilon^2}(\frac{|\mathcal{U}_I|}{|\mathcal{S}_I|} - 1)$ ist die Monte-Carlo Methode ein RASC, es gilt also:

$$\Pr[|MC(T(\varepsilon)) - |\mathcal{S}_I|| \leq \varepsilon \cdot |\mathcal{S}_I|] \geq \frac{3}{4}$$

Werden die Iterationen des Monte-Carlo Algorithmus in polynomieller Zeit ausgeführt, ergibt sich insgesamt also eine asymptotische Laufzeit von $\mathcal{O}^*(\frac{1}{\varepsilon^2} \cdot \frac{|\mathcal{U}_I|}{|\mathcal{S}_I|})$.

Für das Problem $\#\Phi$ dient als Stichprobenraum $\mathcal{U}_\Phi$ zunächst die Menge der Variablenbelegungen aller $n$ Variablen von $\Phi$. Die Anzahl der Elemente in dieser Menge ist dementsprechend $|\{0,1\}^n| = 2^n$. $\mathcal{S}_\Phi$ ist die Menge der erfüllenden Variablenbelegungen, deren Größe $|\mathcal{S}_\Phi| = \#\Phi$

```
Wiederhole T mal:
    Rate unabhaengig und gleichverteilt ein Element x aus U_I
    Pruefe, ob x ∈ S_I (''Treffer'')
Sei L die Anzahl der Treffer
R = L/T ist eine Abschaetzung fuer den Anteil richtiger Loesungen
R · |U_I| ist eine Abschaetzung von |S_I|
```

Codebeispiel 1: Allgemeine Monte-Carlo Methode zum Abschätzen der Größe einer Menge.

gesucht ist. Um gleichverteilte Stichrpoben zu ziehen, werden die Variablen jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ mit `true` und `false` belegt. Anschließend wird geprüft, ob die gezogene Belegung in $\mathcal{S}_\Phi$ enthalten ist, indem die Belegung in $\Phi$ eingesetzt und ausgewertet wird. Nun erhält man das Verhältnis $R$, das den Anteil der Lösungsmenge näherungsweise beschreibt und $R \cdot |\mathcal{U}_\Phi|$ ist eine Approximation von $\#\Phi$. Der Pseudocode für das beschriebene Verfahren ist in Codebeispiel 2 dargestellt.

```
Wiederhole T mal:
    Fuer jede Variable x_i:
        Mit Wahrscheinlichkeit 1/2: x_i := 1, sonst: x_i := 0
    Werte Φ aus, X_t := Die Belegung erfuellt Φ
R := 1/T · Σ_{t=1}^{T} X_t
Gib R · |U_Φ| aus
```

Codebeispiel 2: Monte-Carlo Methode zum Abschätzen der Anzahl Lösungen einer $k$-CNF.

Die nötige Anzahl an Wiederholungen zum erfüllen der RASC-Eigenschaft ist laut Estimator-Theorem von der Größe der Menge $\mathcal{S}_\Phi$ abhängig. Da diese die Gesuchte ist, muss stattdessen eine noch zu bestimmende untere Schranke $\ell$ verwendet werden, also eine Mindestanzahl erfüllender Variablenbelegungen. Da einzelne Iterationen dieses Monte-Carlo Algorithmus in polynomieller Zeit durchgeführt werden, gilt für die Laufzeit $T_{\mathsf{MC}} \in \mathcal{O}^*(\frac{1}{\varepsilon^2} \cdot \frac{|\mathcal{U}_\Phi|}{\ell}) \subseteq \mathcal{O}^*(\frac{1}{\varepsilon^2} \cdot \frac{|\mathcal{U}_\Phi|}{|\mathcal{S}_\Phi|})$, wenn $\ell$ eine gültige untere Schranke für $|\mathcal{S}_\Phi|$ ist. Abbildung 2 zeigt die Anzahl notwendiger Wiederholungen in Abhängigkeit von $\ell$. Lediglich einen SAT-Solver zu verwenden, um eine Untergrenze von mindestens einer erfüllbaren Belegung ($\ell = 1$) zu erhalten, genügt nicht, um die triviale Laufzeit von $2^n$ zu verringern. Eine Methode, die eine signifikant große untere Grenze $\ell$ berechnet, wird im folgenden Kapitel erläutert.

## 3 $\ell$-Schnitte und ein erster hybrider Algorithmus

Die Belegungen beliebiger SAT-Instanzen können mit Hilfe von Eliminationsbäumen dargestellt werden, die die Form eines ausgeglichenen Binärbaumes haben. Dabei steht die ursprüngliche Formel in der Wurzel. Die beiden Kindknoten erhält man, indem man eine noch nicht belegte Variable der ursprünglichen Formel mit `wahr` beziehungsweise `falsch` belegt. Dieser Schritt wird in einer festen Reihenfolge zum Belegen der Variablen rekursiv für alle Kinder solange wiederholt, bis alle $n$ Variablen belegt sind. Dadurch bleibt an den $2^n$ Blättern des Baumes jeweils ein Wahrheitswert

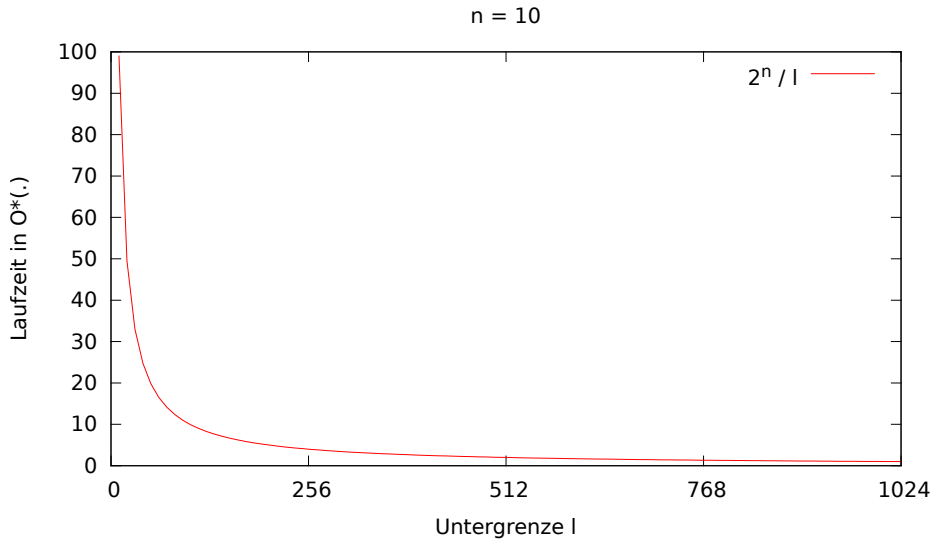


Abbildung 2: Laufzeit des Monte-Carlo Algorithmus in Abhängigkeit von $\ell$, also von der unteren Schranke für die Anzahl an Lösungen.

übrig. Die Anzahl der `wahr`-Blätter entspricht dann genau der Anzahl der erfüllenden Belegungen. Abbildung 3 (a) zeigt einen Eliminationsbaum einer 2-CNF über 3 Variablen.

Ein $\ell$-Schnitt ist ein Teilbaum eines Eliminationsbaumes, der die Wurzel enthält und genau $\ell$ Blätter hat. Weiterhin dürfen nur Knoten enthalten sein, deren zugeordnete Formel erfüllbar ist. Abbildung 3 (b) zeigt einen zum Eliminationsbaum (a) gehörigen 3-Schnitt. Im Allgemeinen kann ein $\ell$-Schnitt folgendermaßen berechnet werden: Beginnend an der Wurzel wird die zugehörige Formel eines Knotens mit Hilfe eines SAT-Solvers auf Erfüllbarkeit geprüft. Ist dies gegeben, so wird der Knoten dem $\ell$-Schnitt hinzugefügt und rekursiv mit den Kindknoten fortgeschritten, bis ein Baum mit $\ell$ Blättern gefunden wurde. Wird eine nicht erfüllbare Formel gefunden, so wird diese und deren ebenso nicht erfüllbare Kindknoten ignoriert. Kann die benötigte Anzahl an $\ell$ Blättern auf diese Weise nicht erreicht werden, so existiert kein Schnitt der Größe $\ell$. Die Anzahl erfüllender Belegungen der ursprünglichen Formel entspricht dann exakt der Anzahl an Blättern im konstruierten Baum. Wird ein $\ell$-Schnitt gefunden, so ist $\ell$ eine Untergrenze für die Anzahl erfüllender Belegungen.

Die asymptotische Laufzeit zur Erzeugung eines $\ell$-Schnittes lässt sich ermitteln, indem man den für den SAT-Solver benötigten Aufwand für alle Knoten des $\ell$-Schnittes aufsummiert. Da in jedem Level des Baumes eine Variable belegt wird, wird für die maximal $2^i$ Knoten in Level $i$ jeweils ein Aufwand von $2^{\beta_k \cdot (n-i)}$ aufgewendet. Da die Knotenanzahl eines $\ell$-Schnittes durch $n \cdot \ell$ beschränkt ist, ist die maximale Höhe des Schnitts $\lceil \log(n \cdot \ell) \rceil$. Es ergibt sich also die folgende Laufzeit, die nach dem Herausziehen der konstanten Faktoren mit Hilfe der Formel für die $n$-te Partialsumme der geometrischen Reihe vereinfacht werden kann:

$$T_{\text{cut}} \in \mathcal{O}^* \left( \sum_{i=0}^{\lceil \log(n \cdot \ell) \rceil} 2^i \cdot 2^{\beta_k \cdot (n-i)} \right) = \mathcal{O}^* \left( 2^{\beta_k \cdot n} \cdot \sum_{i=0}^{\lceil \log(n \cdot \ell) \rceil} 2^{(1-\beta_k) \cdot i} \right) = \mathcal{O}^* (2^{\beta_k \cdot n} \cdot \ell \underbrace{^{1-\beta_k}}_{\in [0;1)})$$

Wie schon beim Monte-Carlo Verfahren liegt auch hier die Worst-Case Laufzeit in $\mathcal{O}^*(2^n)$, die

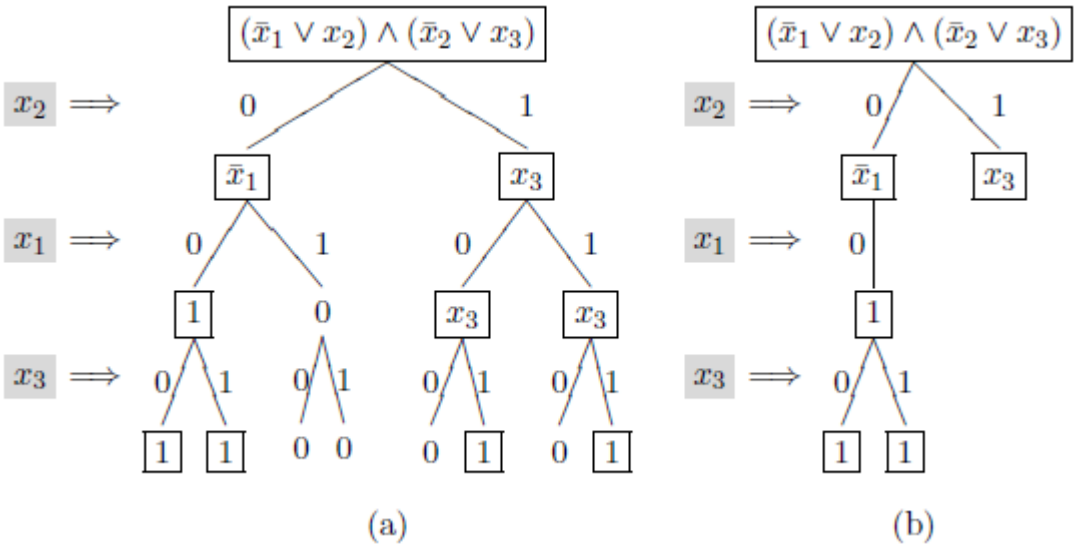


Abbildung 3: (a) Eliminationsbaum und (b) 3-Schnitt für die Formel $\Phi = (\overline{x}_1 \vee x_2) \wedge (\overline{x}_2 \vee x_3)$ und der Belegungsreihenfolge $(x_2, x_1, x_3)$. Erfüllbare Formeln sind eingerahmt. Die Anzahl der 1-Blätter des Eliminationsbaumes entspricht $\#\Phi$.

zur Berechnung eines $\ell$-Schnitts der Größe $2^n$ nötig ist. Im Gegensatz zur Monte-Carlo Methode tritt der Worst-Case jedoch für eine hohe Anzahl erfüllender Belegungen ein. Plottet man beide asymptotischen Laufzeiten in Abhängigkeit von $\#\Phi$ bzw. der maximalen Schnittgröße $\ell$, so ergibt sich, wie in Abbildung 4 dargestellt, ein Schnittpunkt der Laufzeiten. Dieser liegt auf Höhe des $\ell$-Wertes, bis zu dem die Berechnung eines $\ell$-Schnittes günstiger ist als die Verwendung der Monte-Carlo Methode mit selbiger Untergrenze. Führt die Berechnung des $\ell$-Schnitts nicht zu einer exakten Lösung, es wird also ein Schnitt der Größe $\ell$ gefunden, so kann $\ell$ als Untergrenze für die Monte-Carlo Methode genutzt werden.

Der optimale Punkt $\ell^*$ kann durch Ableiten und Null-Setzen der Summe beider Laufzeiten berechnet werden. Durch Einsetzen von $\ell^*$ ergibt sich schließlich die Gesamtlaufzeit des hybriden Algorithmus:

$$\begin{aligned} \ell^* &= \text{arg-min}_\ell \{\mathsf{T}_{\mathsf{Cut}} + \mathsf{T}_{\mathsf{MC}}\} \\ \frac{\partial}{\partial \ell}(\mathsf{T}_{\mathsf{Cut}} + \mathsf{T}_{\mathsf{MC}}) &\stackrel{!}{=} 0 \\ \ell^* &= 2^{\frac{1-\beta_k}{2-\beta_k} \cdot n} \\ T_{\text{hybrid}} &\in \mathcal{O}^*\left(\frac{1}{\varepsilon^2} \cdot 2^{\frac{1}{2-\beta_k} \cdot n}\right) \end{aligned}$$

Setzt man die anfangs genannten Werte für $\beta_k$ ein, so ergibt sich für $k = 3$ die Laufzeit $\mathcal{O}^*(\varepsilon^{-2} \cdot 1.5366^n)$ und für $k = 4$ die Laufzeit $\mathcal{O}^*(\varepsilon^{-2} \cdot 1.6155^n)$.

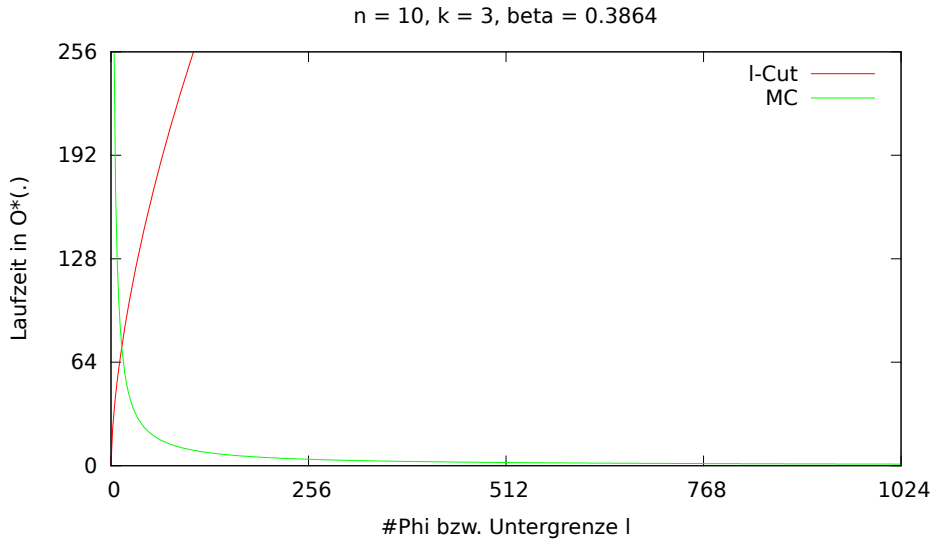


Abbildung 4: Schnittpunkt der Laufzeiten zur Berechnung von $\#\Phi$ mit Hilfe eines $\ell$-Schnittes in Abhängigkeit von $\#\Phi$ und einer Monte-Carlo Approximation bei gegebener unteren Schranke $\ell$.

# 4 Verkleinern des Eliminationsbaums

Belegt man bei der Erzeugung des Eliminationsbaumes die Variablen einer $k$-CNF nicht einzeln sondern klauselweise, also indem man alle Variablen einer in der Formel enthaltenen Klausel gleichzeitig belegt, so kann man direkt die Belegung weglassen, bei der keines der $k$ Literale der Klausel erfüllt wird. Für Klauseln mit $\kappa < k$ Literalen können $k - \kappa$ beliebige zusätzliche Variablen mit belegt werden. In diesem Fall können sogar $2^{k-\kappa} > 1$ Belegungen weggelassen werden. Es ergibt sich ein neuer Baum, in dem jeder Knoten maximal $2^k - 1$ Kinder hat. Die Höhe des Baumes reduziert sich um den Faktor $k$ auf $\lceil \frac{n}{k} \rceil$, die Anzahl enthaltener Knoten ist folglich durch $\lceil \frac{n}{k} \rceil \cdot \ell$ beschränkt. Ein Beispiel ist in Abbildung 5 dargestellt. Die Laufzeit berechnet sich nach dem selben Schema wie zuvor:

$$T_{\text{cut}} \in \mathcal{O}^* \left( \sum_{i=0}^{\lceil \log_{2^k-1}(\ell \cdot \frac{n}{k}) \rceil} (2^k - 1)^i \cdot 2^{\beta_k \cdot (n - k \cdot i)} \right) = \mathcal{O}^*(2^{\beta_k \cdot n} \cdot \ell^{1 - \beta_k \cdot \frac{k}{\log(2^k - 1)}})$$

Dementsprechend ändern sich auch der optimale Umschaltpunkt und die Laufzeit des hybriden Algorithmus wie folgt:

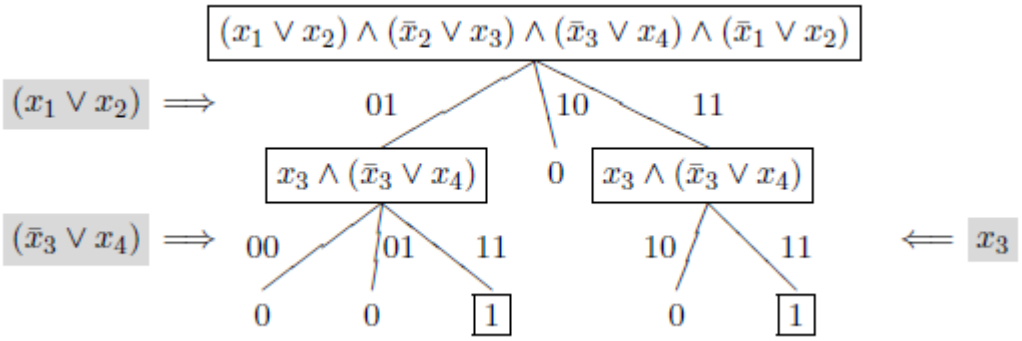


Abbildung 5: Verkleinerter Eliminationsbaum bei dem zunächst die Variablen der Klausel $(x_1 \vee x_2)$ in der Wurzel belegt wurden. Anschließend wurden die Variablen $x_3$ und $x_4$ belegt, wobei im linken Ast die Klausel $(\overline{x}_3 \vee x_4)$ und im rechten Teilbaum die Klausel $(x_3)$ zur Reduzierung der Kindknoten verwendet wurden.

$$\ell^* = \exp\left(\frac{1-\beta_k}{2-\beta_k \cdot \frac{k}{\log(2^k-1)}} \cdot n\right)$$

$$T_{\text{hybrid2}} \in \mathcal{O}^*\left(\frac{1}{\varepsilon^2} \cdot 2^{p_k \cdot n}\right)$$

$$\text{wobei } p_k = \frac{1-\beta_k \cdot \left(\frac{k}{\log(2^k-1)} - 1\right)}{2-\beta_k \cdot \frac{k}{\log(2^k-1)}}$$

Für $k = 3$ verbessert sich die Laufzeit auf $\mathcal{O}^*(\varepsilon^{-2} \cdot 1.5298^n)$, für $k = 4$ ergibt sich die neue Laufzeit $\mathcal{O}^*(\varepsilon^{-2} \cdot 1.6122^n)$.

# 5 Unabhängige Klauseln und Rekursion

Die bisherigen Laufzeitverbesserungen wurden durch eine möglichst effiziente Berechnung der für die Monte-Carlo Methode notwendigen unteren Schranke $\ell$ erzielt. In diesem Kapitel geht es nun darum, die zweite Laufzeit-bestimmende Größe des Monte-Carlo Algorithmus, die Größe des Universums $|\mathcal{U}_\Phi|$, zu verringern. Die hierfür verwendete Idee ist, nicht alle $2^n$ Variablenbelegungen für Stichproben zu verwenden, sondern nur diejenigen, die zumindest einen bestimmten Teil der Klauseln erfüllen. Dieser Teil wird so gewählt, dass erfüllende Belegungen einfach zu erzeugen sind und deren Anzahl direkt berechnet werden kann.

Hierfür soll zunächst der Begriff der Unabhängigkeit für Klauseln definiert werden, wozu wir die CNF $\Phi$ als Menge von Klauseln auffassen. Zwei Klauseln sind *unabhängig*, wenn sie keine gemeinsamen Variablen enthalten. Weiterhin ist eine Teilmenge an Klauseln $\psi \subseteq \Phi$ unabhängig, wenn die enthaltenen Klauseln paarweise unabhängig sind. Eine unabhängige Klauselmenge $\psi$ ist maximal, wenn jede weitere Klausel aus $\Phi$ beim Hinzufügen zu $\psi$ die Eigenschaft der Unabhängigkeit verletzen würde. Damit taucht mindestens eine Variable jeder Klausel aus $\Phi$ auch in $\psi$ auf. Im folgenden wird zunächst analysiert, welche Auswirkungen die Verwendung einer so beschaffene

Formel $\psi$ auf die Laufzeit der Monte-Carlo Methode hat. Die Konstruktion einer entsprechenden Formel folgt am Ende dieses Kapitels.

Da alle Klauseln von $\psi$ auch in $\Phi$ enthalten sind, erfüllt jede erfüllende Belegung von $\Phi$ auch $\psi$. Da $\Phi$ (in nicht-trivialen Fällen) aber weitere Klauseln enthält, kann die Menge erfüllender Belegungen von $\psi$, $\mathcal{S}_\psi$, als Stichprobenraum für $\mathcal{S}_\Phi$ verwendet werden. Da $\psi$ mindestens eine Klausel enthält ist $\mathcal{S}_\psi$ zusätzlich immer kleiner als $\mathcal{U}_\Phi$.

$$\mathcal{U}_\Phi = \{0,1\}^n \supsetneq \mathcal{S}_\psi \supseteq \mathcal{S}_\Phi$$

Um eine Stichprobe aus $\mathcal{S}_\psi$ zu ziehen, wird zunächst eine erfüllende Belegung von $\psi$ ausgewürfelt. Dazu wird für jede Klausel mit $\kappa$ Literalen uniform verteilt eine der $2^\kappa - 1$ erfüllenden Belegungen ausgewürfelt. Anschließend werden alle nicht in $\psi$ enthaltenen Variablen uniform verteilt mit `wahr` oder `falsch` belegt. Es ergibt sich der in Codebeispiel 3 dargestellte neue Pseudocode für den Monte-Carlo Algorithmus.

```
Wiederhole T mal:
    Fuer jede Klausel in ψ
        Wuerfle von 1 bis 2^k − 1, erfuelle d. Literale entsprechend d. Bits
    Fuer jede uebrige Variable x_i:
        Mit Wahrscheinlichkeit 1/2: x_i := 1, sonst: x_i := 0
    Werte Φ aus, X_t := Die Belegung erfuellt Φ
R := 1/T · Σ_{t=1}^{T} X_t
Gib R · |U_Φ| aus
```

Codebeispiel 3: Monte-Carlo Methode mit verkleinertem Stichprobenraum $\mathcal{S}_\psi$.

Die Größe von $\mathcal{S}_\psi$ hängt von der Anzahl an Klauseln in $\psi$ ab. Aus der Unabhängigkeit der Klauseln folgt, dass für die Variablen jeder Klausel $2^\kappa - 1 \leq 2^k - 1$ erfüllende Belegungen existieren. Für alle diese Belegungen können die Variablen, die nicht in $\psi$ auftauchen, beliebig belegt werden, was die Anzahl erfüllender Belegungen um den Faktor $2^{n-\sum_j^{|\psi|}\kappa_j} \geq 2^{n-k\cdot|\psi|}$ vergrößert. Da jede Klausel mit $\kappa < k$ Variablen die Größe von $\mathcal{S}_\psi$ um mehr als $2^{k-\kappa}$ reduziert, gilt folgende untere Grenze für die Größe des neuen Stichprobenraums:

$$|\mathcal{S}_\psi| \leq (2^k - 1)^{|\psi|} \cdot 2^{n-k\cdot|\psi|} = \underbrace{2^n}_{=|\mathcal{U}_\Phi|} \cdot \underbrace{(1-2^{-k})^{|\psi|}}_{<1}$$

In die allgemeine Formel für die Laufzeit der Monte-Carlo Methode eingesetzt ergibt sich folgende, mindestens um den Faktor $(1-2^{-k})^{|\psi|}$ verringerte Laufzeit:

$$\mathcal{O}^*\left(\frac{1}{\varepsilon^2}\cdot\frac{2^n}{\ell}\cdot(1-2^{-k})^{|\psi|}\right)$$

Aus dieser Formel wird ersichtlich, dass $|\psi|$ bezüglich $n$ mindestens in $\omega(1)$ liegen muss, damit der Faktor in $\mathcal{O}^*$-Notation nicht vernachlässigt wird. Betrachtet man die Fälle genauer, bei denen $|\psi|$ besonders klein ist, so erkennt man, dass durch die Belegung einer geringen Anzahl an

Variablen in jeder Klausel aus $\Phi$ mindestens eine Variable belegt wird. Das ergibt sich durch die maximale Unabhängigkeit von $\psi$. Belegt man also alle Variablen von $\psi$, so erhält man eine Menge von Instanzen des Problems $\#(k-1)$-SAT, die rekursiv gelöst werden können. Die Summe der Lösungen dieser Instanzen entspricht dann der Lösung des ursprünglichen Problems. Abbildung 6 zeigt den Extremfall einer 2-CNF, bei der alle Klauseln paarweise abhängig sind. Es gilt also $|\psi| = 1$, die Größe des Universums wäre um den Faktor $(1-2^{-k})^1 = \frac{3}{4}$ verkleinert worden.

$$\begin{aligned} \Phi &= (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\overline{x}_1 \vee x_4) \wedge (\overline{x}_1 \vee x_5) && \in \text{2-SAT} \\ \#\Phi &= \#\Phi|_{x_1=\text{true}} + \#\Phi|_{x_1=\text{false}} && \text{(vgl. Elim.-Baum)} \\ \Phi|_{x_1=\text{true}} &= (x_4) \wedge (x_5), \quad \Phi|_{x_1=\text{false}} = (x_2) \wedge (x_3) && \in \text{1-SAT} \end{aligned}$$

Abbildung 6: Da die Variable $x_1$ in jeder Klausel der 2-CNF $\Phi$ auftaucht, sind alle Klauseln voneinander abhängig. Belegt man die Variable $x_1$, so verschwindet entsprechend ein Literal aus jeder Klausel und die resultierenden Formeln sind 1-CNFs.

Durch die Reduzierung von $k$ sind die neu erzeugten Instanzen, solange sich ihre Anzahl in bestimmten Grenzen hält, schneller lösbar als die ursprüngliche. Die Anzahl kleinerer Instanzen entspricht der Anzahl erfüllender Belegungen von $\psi$, also $(2^k-1)^{|\psi|}$. Dies führt zu folgender rekursiven asymptotischen Laufzeit:

$$T_{\text{Rekursion}} \in \mathcal{O}^*\left((2^k-1)^{|\psi|} \cdot \mathsf{T}(\#(k-1)\text{-SAT})\right)$$

Der Worst-Case liegt in der Größenordnung $2^n$ und tritt ein, wenn die ursprüngliche Formel $\Phi$ bereits unabhängig ist. Da dies genau der Best-Case für die am Anfang dieses Kapitels erläuterte Reduzierung der Größe des Stichprobenraumes ist, liegt die Kombination beider Verfahren in einem weiteren hybriden Algorithmus auf der Hand. Die Konstruktion der Formel $\psi$ kann dabei einfach nach dem Greedy Prinzip erfolgen, indem man solange weitere Klauseln der Formel hinzufügt, bis diese maximal unabhängig ist. Je nach der Anzahl an Klauseln in $\psi$ wird mit Hilfe eines Umschaltpunktes $\hat{m}$ dann entweder der rekursive oder der zu Beginn des Kapitels um $\psi$ erweiterte bisherige Lösungsansatz ausgewählt. Der resultierende Algorithmus ist in Codebeispiel 4 als Pseudocode dargestellt.

Durch die Verwendung des Umschaltpunktes $\hat{m}$ wird die maximale Anzahl rekursiver Aufrufe durch $(2^k-1)^{\hat{m}-1}$ und die minimale Reduzierung der Größe des Universums durch $(1-2^{-k})^{\hat{m}}$ beschränkt. Der optimale Umschaltpunkt $\hat{m}^*$ kann, zusammen mit $\ell^*$, wieder durch Ableiten und Null-Setzen der partiellen Ableitungen von der Summe der Laufzeiten bestimmt werden. Es ergeben sich die optimalen Parameter $\hat{m}^* = 0.0587n$ und $\ell^* = 1.2372^n$. Eingesetzt erhält man für $k=3$ die Laufzeit $\mathcal{O}^*(\varepsilon^{-2} \cdot 1.5181^n)$ und für $k=4$ die Laufzeit $\mathcal{O}^*(\varepsilon^{-2} \cdot 1.6105^n)$.

# 6 Abschließende Bemerkungen

Die verwendeten Werte für $\beta_k$ stammen von randomisierten SAT-Solvern. Die Ergebnisse wurden, aus Gründen der Vereinfachung, in dieser Ausarbeitung jedoch als exakt angenommen. Um trotzdem für den gesamten Algorithmus die RASC-Eigenschaft zu erhalten, muss die Erfolgswahrscheinlichkeit der SAT-Solver mit Hilfe der bereits in der Einleitung angesprochenen *Median of*

```
Berechne ψ nach dem Greedy-Prinzip
Wenn |ψ| < m̂:
    Wenn k − 1 = 2: Verwende Wahlstroem's Algorithmus
    Sonst: Berechne rekursiv die Summe der Loesungen der #(k − 1)-SAT
           Instanzen aller ψ-erfuellenden Belegungen
Sonst:
    Berechne zuerst einen ℓ-Cut
        ℓ-Cut existiert nicht: #Φ entspricht der # Blaetter im Schnitt
        ℓ-Cut existiert: ℓ ist eine Untergrenze der Loesungen von Φ
    Starte MC mit der Untergrenze ℓ und dem Universum S_ψ
```

Codebeispiel 4: Hybrider Algorithmus mit Nutzung der unabhängigen Klauseln und Rekursion.

*Means*-Methode verstärkt werden. Da dies nur eine von der $\mathcal{O}^*$-Notation vernachlässigte polynomielle Anzahl zusätzlicher Wiederholungen erfordert, ändert sich die asymptotische Laufzeit dadurch nicht.

Für die Rekursion und die Verkleinerung des Universums im Monte-Carlo Algorithmus ist es nicht notwendig, ausschließlich Klauseln zu verwenden. Solange die Länge der verwendeten Teilformeln konstant ist, kann die Anzahl an Lösungen ohne zusätzlichen asymptotischen Aufwand berechnet werden. Dies führt, wie in [1] analysiert, zu einer weiteren Reduzierung der asymptotischen Laufzeit.

Weiterhin fällt beim Erstellen der $\ell$-Schnitte auf, dass sehr ähnliche Formeln vom SAT-Solver gelöst werden müssen. Ob sich diese Ähnlichkeiten durch Verwendung eines spezialisierten SAT-Solvers zugunsten einer besseren Laufzeit ausnutzen lassen, ist ein noch offenes Problem.

## Literatur


[1] Manuel Schmitt, Rolf Wanka. Exploiting Independent Subformulas: A Faster Approximation Scheme for #k -SAT. In *Information Processing Letters*, pages 337–344, 2013. doi:10.1109/FOCS.2011.22.

[2] Marc Thurley. An approximation algorithm for #k-SAT in: *Proc. 29th Int. Symp. on Theoretical Aspects of Computer Science (STACS)*, pp. 78-87. doi:10.4230/LIPIcs.STACS.2012.78

[3] Magnus Wahlström. A tighter bound for counting max-weight solutions to 2SAT instances. In *Proc. 3rd Int. Workshop on Parameterized and Exact Computation (IWPEC)*, pages 202–213, 2008. doi:10.1007/978-3-540-79723-4 19.

[4] Timon Hertli. 3-SAT faster and simpler – Unique-SAT bounds for PPSZ hold in general. In *Proc. 52nd IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 277–284, 2011. doi:10.1109/FOCS.2011.22.

[5] Rajeev Motwani, Prabhakar Raghavan. Randomized Algorithms. *Cambridge University Press*, page 311, 1995.