RANDOMISIERTES RUNDEN

SOMMERAKADEMIE GÖRLITZ 2007
NICOLAS WEBER

1. Einführung

Manche Probleme lassen sich durch "nicht-deterministische" Algorithmen schnell und im Mittel gut näherungsweise lösen. Das wird hier am Beispiel Max-SAT demonstriert[1].

Definition 1.1. Gegeben sei eine Menge von Variablen $V = \{x_1, \ldots, x_n\}$. Ein Literal l_i ist entweder eine Variable $x_i \in V$ oder deren Negation $\bar{x_i}$. Eine Klausel $C = l_1 \vee \ldots \vee l_k$ist eine Disjunktion von Literalen[2]. Eine Boolesche (n, m)-Formel $\Phi = C_1 \wedge \ldots \wedge C_m$ in konjugierter Normalform (KNF) ist eine Konjunktion von Klauseln. Wir nehmen an, dass alle Variablen in einer Klausel verschieden sind.

Das Entscheidbarkeitsproblem SAT ist die Frage, ob es eine Variablenbelegung $b : V \to \{\text{false}, \text{true}\}$ gibt, so dass $b(\Phi)$ für ein gegebenes Φ wahr wird, also alle Klauseln erfüllt werden können. In Max-SAT hingegen sollen nur möglichst viele Klauseln erfüllt werden:

Definition 1.2. Eine Probleminstanz Φ zu Max-SAT ist eine (n, m)-Formel in KNF. Eine zulässige Lösung ist eine Variablenbelegung $b : V \to \{\text{false}, \text{true}\}$. Die Zielfunktion ist $\text{wahr}(b, \Phi) := |\{j | C_j \in \Phi, b(C_j) = \text{true}\}|$, und sie soll maximiert werden.

Im Folgenden sollen einige "einfache" Lösungsalgorithmen für Max-SAT besprochen werden. Als ersten Versuch setzen probieren wir einfach die beiden Belegungen $b_f := (\text{false}, \ldots, \text{false})$ und $b_t := (\text{true}, \ldots, \text{true})$ aus und nehmen diejenige dieser beiden Belegungen, die mehr Klauseln erfüllt. Das gibt schon eine relative Güte von 2:

Theorem 1.3. Sei Φ eine (n, m)-Formel in KNF. Dann ist

$$\max\{\text{wahr}(b_f, \Phi), \text{wahr}(b_t, \Phi)\} \geq \frac{1}{2}m.$$

Beweis. Da alle Variablen auf den gleichen Wert gesetzt werden, kann man allen Variablen den gleichen Namen x geben und die Klauseln dadurch vereinfachen. Sei C_+ die Menge der Klauseln, die nur aus x bestehen, C_- die Menge der Klauseln, die nur aus $\bar{x}$ bestehen und C_0 die Menge der Klauseln, die wie $x \vee \bar{x}$ aussehen. Für b_f sind die Klauseln in C_- und C_0 erfüllt, für b_t entsprechend C_+ und C_0. Es gibt m Klauseln, also ist $|C_+| + |C_-| + |C_0| = m$. Nach der Definition von max

[1] Diese Ausarbeitung ist fast identisch mit dem Abschnitten 6.1, 6.2 und 6.5 aus dem Buch "Approximationsalgorithmen" von Rolf Wanka [**?**].

[2] Jede Klausel kann beliebige Literale enthalten, daher wäre $l_{s(n)}$ korrekter. Das lasse ich aber der Lesbarkeit halber mal bleiben.

gilt $\max\{a,b\} \geq a$ und $\max\{a,b\} \geq b$ und damit $\max\{a,b\} \geq \frac{a+b}{2}$. Damit (und mit $|C_0| \geq 0$) ergibt sich

$$\max\{\text{wahr}(b_f,\Phi), \text{wahr}(b_t,\Phi)\} \geq \frac{|C_-|+|C_0|+|C_+|+|C_0|}{2} = \frac{m+|C_0|}{2} \geq \frac{1}{2}m.$$

Es gibt offensichtlich Formeln, in denen mit diesem Verfahren nur $\frac{m}{2}$ Klauseln erfüllt werden, zum Beispiel wenn die Hälfte der Klauseln die Form $C_j = x$ haben und die andere Hälfte $C_j = \bar{x}$ – die Abschätzung ist also scharf. □

Wenn eine Klausel viele verschiedene Literale enthält, sagt einem die Intuition, dass man die Klausel erfüllen können sollte – schliesslich muss nur eines der Literale wahr werden, um die ganze Klausel zu erfüllen. Wir wollen nun einen sehr einfachen nicht-deterministischen Algorithmus betrachten, dessen erwartete relative Güte mit der Klauselläng wächst. Als "Nebenprodukt" erhalten wir einen Existenzsatz für eine "gute" Lösung für Klauseln mit "vielen" Variablen.

Algorithm 1.4. Setze jede der Variablen in Φ mit Wahrscheinlichkeit $\frac{1}{2}$ auf true und mit Wahrscheinlichkeit $\frac{1}{2}$ auf false.

Algorithmus 1.4 wird im Folgenden "Algorithmus A" genannt.

Diese Entscheidung wird für jede Variable unabhängig von den anderen Variablen getreffen, es gilt also $Pr[x_i = \alpha \wedge x_j = \beta] = Pr[x_i = \alpha] \cdot Pr[x_j = \beta]$.

Mit diesem sehr einfachen Verfahren ergibt sich:

Lemma 1.5. Sei k_j die Anzahl der Literale in C_j. Es gilt

$$Pr[A \text{ erfüllt } C_j] = 1 - \frac{1}{2^{k_j}}.$$

Beweis. Klausel C_j ist nur nicht erfüllt, wenn alle k_j in ihr enthaltene Literal falsch ist. Die Wahrscheinlichkeit dafür ist wegen der Unabhängigkeit einfach $\left(\frac{1}{2}\right)^{k_j}$. □

Damit lässt sich der Erwartungswert der von A erzeugten Belegung b_A bestimmen – also die Anzahl Klauseln, die von A "im Durchschnitt" bei "vielen" Versuchen auf einer Eingabe richtig geraten wird.

Theorem 1.6. Für jede (n,m)-Formel Φ in KNF, in der jede Klausel **mindestens** k Literale hat, gilt

$$E[A(\Phi)] := E[\text{wahr}(b_A,\Phi)] \geq \left(1 - \frac{1}{2^k}\right) \cdot m.$$

Beweis. Für $j \in \{1,\ldots,m\}$ sei Z_j eine Zufallsvariable mit

$$Z_j := \begin{cases} 1 & \text{falls } b_A(C_j) = \text{true} \\ 0 & \text{sonst.} \end{cases}$$

Z_j ist also genau dann 1, wenn C_j von der zufälligen Belegung erfüllt wird. Damit gilt $E[Z_j] = 0 \cdot Pr[b_A(C_j) = \text{false}] + 1 \cdot Pr[b_A(C_j) = \text{true}] = Pr[A \text{ erfüllt } C_j]$. Mit Lemma 1.5 folgt

$$E[\text{wahr}(b_A,\Phi)] = E\left[\sum_{j=1}^{m} Z_j\right] = \sum_{j=1}^{m} E[Z_j] = \sum_{j=1}^{m}\left(1 - \frac{1}{2^{k_j}}\right) \geq \left(1 - \frac{1}{2^k}\right) \cdot m.$$

Beachte, dass diese Zeile auch den exakten Erwartungswert von Algorithmus A enthält. Die Abschätzung wird durch die Annahme $\forall j.k_j \geq k$ möglich. □

Note 1.7. Aus diesem Satz folgt sofort, dass es unter den Voraussetzungen des vorigen Satzes immer eine Belegung mit mindestens $\left(1 - \frac{1}{2^k}\right) \cdot m$ wahren Klauseln gibt – wäre das nicht so, müsste der Erwartungswert kleiner sein. Dieses Argument nennt sich Probabilistische Methode.

Es lassen sich Beispiele finden, in denen es höchstens so viele erfüllbare Klauseln gibt (also einen Zeugen für diese Abschätzung). Ein mögliches Beispiel bei bei gegebenem k eine $\left(k, 2^k\right)$-Formel, in der in jeder Klausel jede Variable entweder selbst oder verneint vorkommt, und in der alle 2^k möglichen solchen Klauseln vorkommen. Für jede mögliche Belegung ist genau eine Klausel nicht erfüllt, es gibt also nur $2^k - 1 = \left(1 - \frac{1}{2^k}\right) \cdot m$ erfüllte Klauseln.

Corollary 1.8. Sei $2^k > m$. Dann ist jede (n, m)-Formel mit mindestens k Literalen erfüllbar.

Beweis. Wegen der vorigen Notiz und der Voraussetzung gilt

$$m \geq \text{wahr}\,(b, \Phi) \geq \left(1 - \frac{1}{2^k}\right) \cdot m > \left(1 - \frac{1}{m}\right) \cdot m = m - 1.$$

Da wahr$(\cdot, \cdot)$ ganzzahlig ist und echt größer als $m - 1$, muss es in diesem Fall den Wert m annehmen (einen größeren Wert kann es sicher nicht annehmen – es können höchstens nur so viele Klauseln erfüllt sein, wie es gibt). Das aber nur am Rande. □

Definition 1.9. Unsere bisherigen Qualitätsgrößen können nicht mit randomisierten Algorithmen umgehen. Daher definieren wir die erwartete relative Güte als

$$E\left[\varrho_A(I)\right] := \max\left\{\frac{OPT(I)}{E[A(I)]}, \frac{E[A(I)]}{OPT(I)}\right\}.$$

Wie zuvor nimmt das Maximum bei Maximierungsproblemen den ersten, bei Minimierungsproblemen den zweiten Wert an.

Aus Satz 1.6 ist der Erwartungswert von Algorithmus A bekannt, damit ergibt sich:

Theorem 1.10. Algorithmus A hat für jede (n, m)-Formel in KNF mit **mindestens** k Literalen eine erwartete relative Güte von

$$E\left[\varrho_A(\Phi)\right] = \frac{OPT(\Phi)}{E[A(\Phi)]} \leq \frac{m}{E[A(\Phi)]} \leq \frac{m}{\left(1 - \frac{1}{2^k}\right) m} = \frac{1}{1 - \frac{1}{2^k}}.$$

Der Algorithmus hat eine lineare Laufzeit. Wenn die kürzeste Klausel mindestens zwei Literale enthält, ist die erwartete relative Güte von A höchstens $\frac{4}{3}$.

Beweis. Die letzte Behauptung ergibt sich durch Einsetzen von $k = 2$ in die obige Formel, der Rest wurde schon gezeigt. □

Note 1.11. Durch eine aufwändigere Analyse lässt sich zeigen, dass A für alle Eingaben eine erwartete relative Güte von $\frac{3}{2}$ hat.

2. Randomisiertes Runden

Im vorigen Abschnitt wurde ein sehr einfacher Algorithmus vorgestellt, der sich gutartig verhält, wenn es viele lange Klauseln gibt. In diesem Abschnitt wird ein etwas komplizierterer Algorithmus vorgestellt, der gutartig ist, wenn es nur kurze Klauseln gibt. Der in diesem Abschnitt entwickelte Algorithmus B wird vom Prinzip so vorgehen wie Algorithmus A, nur werden die Wahrscheinlichkeiten zum Raten einer Belegung anders gewählt.

Als erster Schritt wird Max-SAT in ein ganzzahliges lineares Problem ("ILP", integer linear problem) umformuliert. Sei $\Phi = C_1 \wedge \ldots \wedge C_m$ eine (n, m)-Formel in KNF. Wir nennen die Menge der Variablen, die in C_j nicht negiert vorkommen S_j^+, und die Menge der negierten Variablen S_j^-. Da wir (oBdA) annehmen, dass jede Variable in jeder Klausel nur einmal vorkommt, sind diese beiden Mengen disjunkt. Nun führen wir für jede Variable x_i eine 0-1-Variable $\hat{x_i}$ ein und für jede Klausel eine 0-1-Variable $\hat{Z_j}$. Hierbei entspricht 0 dem Wahrheitswert false und 1 dem Wahrheitswert true. Mit diesen Variablen lässt sich Max-SAT äquivalent wie folgt ausdrücken:

$$\begin{aligned} \text{maximiere} \quad & \sum_{j=1}^{m} \hat{Z_j} \\ \text{gemäß} \quad & \sum_{x_i \in S_j^+} \hat{x_i} + \sum_{x_i \in S_j^-} (1 - \hat{x_i}) \geq \hat{Z_j} \quad j = 1, \ldots, m \\ & \hat{x_i}, \hat{Z_j} \in \{0, 1\} \quad \forall i, j \end{aligned}$$

Diese Umformung des Problems nennt sich Arithmetisierung. Durch die Umformung ist das Problem erstmal nicht einfacher geworden, aber wenn man die letzte Nebenbedingung zu $0 \leq \hat{x_i}, \hat{Z_j} \leq 1$ aufweicht. Dieser Schritt nennt sich Relaxierung. Dann bleibt ein normales lineares Problem B_{rel} zurück, dass in Polynomzeit gelöst werden kann – und da die Anzahl der Nebenbedingungen und Variablen auch nur polynomiell von der Ursprungsproblemgröße abhängt, ist das lineare Problem effizient lösbar. Allerdings ist die Lösung des relaxierten Problems natürlich im allgemeinen keine Lösung des ursprüunglichen Problems mehr, und es gilt

$$\sum_{j=1}^{m} \hat{Z_j} = \mathsf{OPT}(B_{\mathsf{rel}}) \geq \mathsf{OPT}(B) = \mathsf{OPT}(\Phi).$$

Ein Beispiel hierfür ist $\Phi = (x_1 \vee x_2) \wedge (\bar{x_1} \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_2})$. In dieser Formel können höchstens drei Klauseln erfüllt werden, aber $\hat{x_1} = 0.5, \hat{x_2} = 0.5$ und $\hat{Z_j} = 1, j = 1, 2, 3, 4$ sind eine zulässige Lösung des relaxierten Problems.

Um von der rationalen Lösung von B_{rel} zurück zu einer ganzzahligen Lösung von B zu kommen, benutzt man das sogenannte randomisierte Runden. Sei hierfür eine stochastische Funktion $\pi : [0, 1] \to [0, 1]$ gegeben.

Algorithm 2.1. RandomisiertesRunden:
Für $i := 1$ bis n tue

$$\begin{cases} \text{mit Warscheinlichkeit } \pi(\hat{x_i}) & x_i := \text{true} \\ \text{mit Warscheinlichkeit } 1 - \pi(\hat{x_i}) & x_i := \text{false} \end{cases}$$

Dabei werden die Entscheidungen immer noch für jede Variable unabhängig getroffen, aber durch das Verwenden der Lösung des relaxierten Problems sind

die Entscheidungen nicht komplett willkürlich. Zunächst verwenden wir $\pi(x) = x$. Die Hintereinanderausführung von Lösen des relaxierten Problems und randomisiertem Runden ist Algorithmus B. Um diesen zu Untersuchen brauchen wir noch folgende Aussagen:

Fact 2.2. Für $a_i \geq 0$ gilt $\prod_{i=1}^{k} a_i \leq \left(\frac{1}{k}\sum_{i=i}^{k} a_i\right)^k$.

Fact 2.3. Sei $f(x)$ auf dem Interval $[a, b]$ konkav und gelte $f(a) \geq ma + n$ und $f(b) \geq mb + n$. Dann gilt für alle $x \in [a, b]$: $f(x) \geq mx + n$.

Lemma 2.4. C_j habe k_j Literale. Dann gilt

$$Pr[B \text{ erfüllt } C_j] \geq \left(1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right) \cdot \hat{Z}_j.$$

Beweis. Der Beweis erfolgt relativ direkt durch einsetzen.

$$\begin{aligned}
& Pr[B \text{ erfüllt } C_j] \\
= \quad & 1 - \left[\prod_{x_i \in S_j^+} (1 - \hat{x}_i)\right] \cdot \left[\prod_{x_i \in S_j^-} \hat{x}_i\right] \overset{(1)}{\geq} 1 - \left(\frac{\sum_{x_i \in S_j^+}(1 - \hat{x}_i) + \sum_{x_i \in S_j^-} \hat{x}_i}{k_j}\right)^{k_j} \\
= \quad & 1 - \left(\frac{\left|S_j^+\right| - \sum_{x_i \in S_j^+} \hat{x}_i + \left|S_j^-\right| - \sum_{x_i \in S_j^-}(1 - \hat{x}_i)}{k_j}\right)^{k_j} \\
= \quad & 1 - \left(\frac{k_j - \left(\sum_{x_i \in S_j^+} \hat{x}_i + \sum_{x_i \in S_j^-}(1 - \hat{x}_i)\right)}{k_j}\right)^{k_j} \\
\overset{(2)}{\geq} \quad & 1 - \left(1 - \frac{\hat{Z}_j}{k_j}\right)^{k_j} \overset{(3)}{\geq} \left(1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right) \cdot \hat{Z}_j
\end{aligned}$$

An Stelle (1) wurde Fakt 2.2 benutzt, an (2) die Nebenbedingung des linearen Problems, das von $\hat{Z}_j$ gelöst wird. An Stelle (3) wird Fakt 2.3 mit der Funktion $f(z) = 1 - \left(1 - \frac{z}{k}\right)^k$, der Steigung $m = 1 - \left(1 - \frac{1}{k}\right)^k$ und einem y-Achsenabschnitt von $n = 0$ benutzt. Die Konkavität auf $[0, 1]$ sieht man daran, dass dort $f''(z) \leq 0$ gilt, und die Ungleichungen auf den Endpunkten dieses Intervalls sind offensichtlich erfüllt.

□

Wie schon bei Algorithmus A ergibt sich damit eine Aussage über die erwartete Anzahl der von Algorithmus B erfüllten Formeln.

Theorem 2.5. Für jede (n, m)-Formel Φ in KNF mit **höchstens** k Literalen gilt

$$E[B(\Phi)] \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot OPT(\Phi).$$

Beweis. Wieder folgt der Beweis durch stures Rechnen, ähnlich wie im Beweis von Satz 1.6:

$$\begin{aligned} E[B(\Phi)] &= \sum_{j=1}^{m} Pr[B \text{ erfüllt } C_j] = \sum_{j=1}^{m} \left(1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right) \cdot \hat{Z}_j \\ &\geq \left(1 - \left(1 - \frac{1}{k}\right)^{k}\right) \cdot OPT(B_{rel}) \geq \left(1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right) \cdot OPT(\Phi) \end{aligned}$$

Bei der ersten Abschätzung wurde benutzt, dass $1 - \left(1 - \frac{1}{k}\right)^k$ mit wachsendem k monoton f'ällt (und nach Voraussetzung ist $k_j \leq k$). In die zweite Abschätzung geht ein, dass die Lösung für das relaxierte Problem bessergleich der Lösung des ursprünglichen Problems ist. Wieder enthält dieser Beweis eine explizite exakte Formel für den Erwartungswert für eine gegebene Formel Φ. □

Mit diesem Satz lässt sich die relative Güte von Algorithmus B abschätzen:

Theorem 2.6. Algorithmus B hat für jede (n, m)-Formel in KNF mit **höchstens** k Literalen eine erwartete relative Güte von

$$E\left[\varrho_B(\Phi)\right] = \frac{OPT(\Phi)}{E[B(\Phi)]} \geq \frac{1}{\left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot 1} \leq \frac{1}{1 - \frac{1}{e}} \approx 1.582.$$

Der letzte Teil dieser Abschätzung ist sogar unabhängig von k.

Beweis. Es gilt $1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - \frac{1}{e}$ für alle $k \in \mathbb{N}$. □

Für andere Wahlen von π lässt sich auch eine relative Güte von $\frac{4}{3}$ erzielen. Betrachte dazu zum Beispiel eine Funktion π mit $1 - \frac{1}{4^x} \leq \pi(x) \leq 4^{x-1}$ auf $[0, 1]$. Mit dieser Funktion wird jetzt ganz genau so vorgegangen wie zuvor: Zunächst erhalten wir eine Aussage über die Wahrscheinlichkeit, dass Algorithmus B mit dieser Rundungsfunktion $\pi(x)$ (im Folgenden B_π genannt) eine gegebene Klausel erfüllt, dann gewinnen wir eine Aussage über den Erwartungswert und schließlich eine Abschätzung für die relative Güte.

Lemma 2.7. Es gilt

$$Pr[B_\pi \text{ erfüllt } C_j] \geq \frac{3}{4} \cdot \hat{Z}_j.$$

Beweis. Der Beweis erfolgt relativ direkt durch einsetzen.

$$\begin{aligned}
& Pr[B_\pi \text{ erfüllt } C_j] \\
= \quad & 1 - \left[\prod_{x_i \in S_j^+} (1 - \pi(\hat{x}_i))\right] \cdot \left[\prod_{x_i \in S_j^-} \pi(\hat{x}_i)\right] \\
\overset{(1)}{\geq} \quad & 1 - \left[\prod_{x_i \in S_j^+} (1 - 1 + \frac{1}{4^x})\right] \cdot \left[\prod_{x_i \in S_j^-} 4^{x-1}\right] \\
= \quad & 1 - 4^{-\left(\sum_{x_i \in S_j^+} \hat{x}_i + \sum_{x_i \in S_j^-} (1 - \hat{x}_i)\right)} \\
\overset{(2)}{\geq} \quad & \frac{3}{4} \cdot \min\left\{1, \left[\sum_{x_i \in S_j^+} \hat{x}_i\right] + \left[\sum_{x_i \in S_j^-} (1 - \hat{x}_i)\right]\right\} \\
\overset{(3)}{\geq} \quad & \frac{3}{4} \cdot \min\left\{1, \hat{Z}_j\right\} = \frac{3}{4} \cdot \hat{Z}_j
\end{aligned}$$

An Stelle (1) wurden die Schranken für π benutzt. An Schritt (2) wurde die Funktion $1 - 4^{-x}$ für $x \geq 0$ durch $\frac{3}{4} \min\{1, x\}$ nach unten abgeschätzt – im Intervall $[0, 1]$ gilt diese Abschätzung durch $\frac{3}{4}x$ wegen Fakt 2.3, und für $x > 1$ gilt offensichtlich $1 - 4^{-x} \geq \frac{3}{4}$. An Stelle (3) wird schließlich wieder die Nebenbedingung des linearen Problems benutzt. □

Theorem 2.8. Für den Algorithmus B_π gilt

$$E[B_\pi(\Phi)] \geq \frac{3}{4} \cdot OPT(\Phi)$$

und

$$E\left[\varrho_{B_\pi}(\Phi)\right] \leq \frac{4}{3}.$$

Beweis. Die Rechnung läuft wie zuvor:

$$E[B_\pi(\Phi)] = \sum_{j=1}^{m} \frac{3}{4} \hat{Z}_j = \frac{3}{4} \cdot OPT(B_{rel}) \geq \frac{3}{4} \cdot OPT(\Phi)$$

für den Erwartungswert und $E\left[\varrho_{B_\pi}(\Phi)\right] \leq \frac{OPT(\Phi)}{E[B_\pi(\Phi)]} = \frac{OPT(\Phi)}{\frac{3}{4} \cdot OPT(\Phi)} = \frac{4}{3}$. □

Beachte, dass diese Schranke unabhängig von der Länge der Klauseln ist, und damit im Allgemeinen echt besser als die Algorithmen A und B. In dem Artikel von Goemnans und Williamson [**?**] sind noch andere Funktionen vorgestellt, die auf die selbe erwartete Güte führen.

3. Derandomisierung

Einige nicht-deterministische Algorithmen lassen sich durch eine mechanische Transformation so in einen deterministische Varianten umwandeln, dass sie ihr Vorgehen und ihre Laufzeit mehr oder weniger behalten. Das wird in diesem Abschnitt am Beispiel von Algorithmus A vorgeführt, das Verfahren kann aber auch etwas allgemeiner angewandt werden.

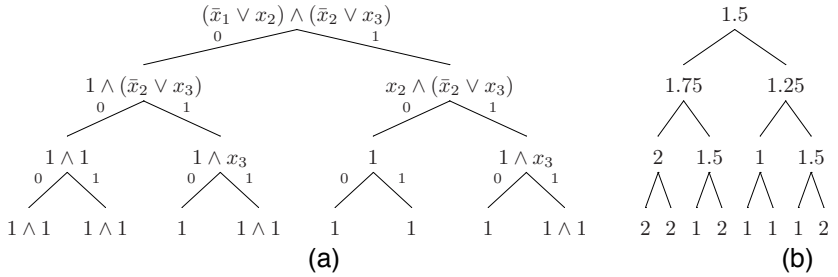

Abbildung 1. (a) Berechnungsbaum für ein Beispiel. (b) Die zugehörigen Erwartungswerte.

Zuerst definieren wir eine Substitutionsregel für Max-SAT: Sei $\Phi = C_1 \wedge \ldots \wedge C_m$ eine (n, m)-Formel in KNF und α sei true oder false. Dann ist $\Phi_{x_1=\alpha}$ die (n', m')-Formel in KNF mit $n' < n$ und $m' \leq m$, die man erhält, wenn man folgendes Verfahren auf alle Klauseln in Φ anwedet:

- Wenn x_1 in C_j nicht vorkommt, bleibt C_j unverändert.
- Wenn C_j wahr wird, wenn man α für x_1 einsetzt, wird C_j durch true, was wir jetzt als Klausel ansehen, ersetzt.
- Ansonsten streiche das Literal in C_j, das durch Einsetzen von α zu false ausgewertet wird. Wird C_j dadurch zur leeren Klausel, fällt sie ganz weg.

Diese Substitution lässt sich effizient berechnen. Mit ihrer Hilfe lässt sich ein "Ersetzungsbaum" konstruieren, indem man in alle Variablen der Reihe nach sowohl die Werte true und false einsetzt. Für ein Beispiel ist dies in Abb. 1a vorgeführt. Jedes substituierte Problem ist ein gültiges kleineres Problem.

Für Algorithmus A haben wir eine Formel hergeleitet, mit der wir für jede der Formeln in diesem Berechungsbaum die erwartete Anzahl der erfüllten Formeln effizient berechnen können (für Klauseln, die nur aus true bestehen, setzen wir in dieser Formel $k_j := \infty$). Der von dieser Formel berechnete Erwartungswert ist gleich dem bedingten Erwartungswert, den man erhält, wenn man für einen Knoten die Erwartungswerte der beiden Kinderknoten mit ihren jeweiligen Wahrscheinlichkeiten gewichtet (für Algorithmus A ist diese immer $\frac{1}{2}$) und addiert. Das kann in 1b am Beispiel überprüft werden, die Abbildung zeigt die Erwartungswerte für alle Knoten. Außerdem sind die Formeln in den Blättern deterministisch.

Es gilt

$$\begin{aligned} E[A(I)] &= Pr[x_1 = 0] \cdot E[A(I)|x_1 = 0] + Pr[x_1 = 1] \cdot E[A(I)|x_1 = 1] \\ &\leq \max\{E[A(I)|x_1 = 0], E[A(I)|x_1 = 1]\} \end{aligned}$$

Induktiv folgt, dass man die Erwartungswerte den Baum hinunter erhöhen kann, in jedem Knoten also die Erwartungswerte der beiden Kinder berechnen kann und die Belegung in Richtung des größeren Erwartungswert wählen. Als derandomisierten Algorithmus berechnen wir also in jedem Knoten die beiden Kind-Erwartungswerte. Ist der linke Wert größer, gehen wir den Baum links herab, ansonsten rechts. Damit ist dadurch die worst-case-Güte des Ergebnisses besser-gleich erwarteter Güte des ursprünglichen Algorithmusses A:

$$\varrho_{\text{DERAND\_A}}(I) \leq E[\varrho_A(I)]$$

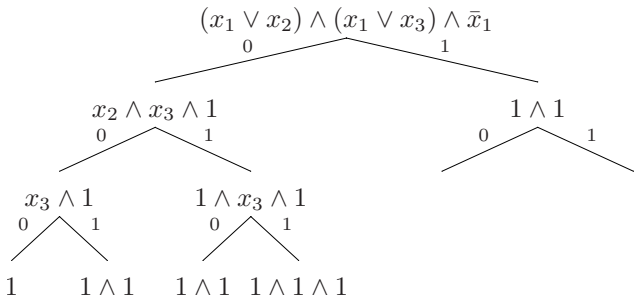

Abbildung 2. Teilweiser Berechnungsbaum für den Zeugen gegen die worst-case-Güten-Abschätzung.

Dieses Verfahren ist deterministisch, und immer noch in Polynomialzeit durchführbar. Es lässt sich auch für Algorithmus B anwenden. Die vorige Abschätzung für die Güte ist scharf, ein Zeuge dafür ist die $(3k, 3k)$-Formel

$$\Phi_k := \bigwedge_{i=0}^{k-1} \left((x_{3i+1} \vee x_{3i+2}) \wedge (x_{3i+2} \vee x_{3i+3}) \wedge \bar{x}_{3i+1} \right).$$

In Abb. 2 ist der (relevante) Berechnungsbaum für eine Instanz mit $k = 1$ angegeben. Die Erwartungswerte an den Kindern der Wurzel sind 2 ($\frac{1}{2}+\frac{1}{2}+1$) und 2 (1+1), damit geht der Algorithmus nach rechts und findet damit nur 2 erfüllte Klauseln – bei dem allgemeinen Beispiel entsprechend $2k$ viele, es können aber alle 3 (bzw. $3k$) erfüllt werden. Der derandomisierte Algorithmus hat also eine worst-case-Güte von $\frac{3}{2}$.

Die exakte Formel für die erwartete relative Güte von Algorithmus A ist

$$E[\varrho_A(I)] = \frac{OPT(I)}{\sum_{j=1}^{m} \left(1 - \frac{1}{2^{k_j}}\right)} = \frac{3k}{k \cdot (\frac{3}{4} + \frac{3}{4} + \frac{1}{2})} = \frac{3}{2}$$

da alle Klauseln erfüllt werden können ($OPT(I) = 3k$) und weil die Klauseln in Φ_k jeweils aus 2, 2 und 1 Literalen bestehen. Also wird für dieses Beispiel die Gleichheit in der Ungleichung angenommen.

Literatur

[1] Michel X Goemans and David P Williamson. A new $\frac{3}{4}$-approximation algorithm for max sat. Journal of the ACM, 1995.

[2] Rolf Wanka. Approximationsalgorithmen. Teubner, 2006.