Handout zur Sitzung am 16.01.06

Dominic de Carolis

30. Januar 2006

1 Kontext

In dem Papier *A Deterministic* $(2-\frac{2}{k+1})^n$ *Algorithm for k-SAT Based on Local Search* werden zwei, auf lokaler Suche basierende, Algorithmen zur Lösung von k-Sat-Problemen in Zeit $O^*((2-\frac{2}{k+1})^n)$ mit exponentiellem Platzbedarf, bzw. in Zeit $O^*((2-\frac{2}{k+1}+\varepsilon)^n)$ (für beliebig kleines ε) mit polynomiellen Platzbedarf vorgestellt. Später wird auf $k=3$ spezialisiert und die Laufzeit auf $1,481^n \cdot poly(n)$ verbessert.

In der vorigen Sitzung wurde bereits die lokale Suche innerhalb eines Balles mit Radius r vorgestellt und untersucht(Laufzeit k^r), sowie festgestellt, dass für jeden Hammingraum H_n ein Cover-Code der Länge n mit Radius $r \leq \rho n$ und Größe $\leq n\beta(n)2^{(1-h(\rho))n}$ existiert. Es wurde außerdem der Greedy-Algorithmus akzeptiert, der für H_n einen Cover-Code der Länge n und Größe $\leq n^2\beta(n)^2 2^{(1-(h(\rho))n}$ erzeugt.

In dieser Sitzung folgte nun die Analyse der Laufzeit dieses Algorithmus'(Lemma 5), sowie eine Verbesserung durch Unterteilung in Blöcke mit exponentiellem (Lemma 6) bzw. polynomiellem (Lemma 7) Platzbedarf. Dann folgte die Zusammenfassung der lokalen Suche mit der Cover-Code-Suche zum Hauptalgorithmus und seiner Analyse. Für den Sonderfall $k=3$ wurde der Algorithmus (noch nicht zuende)verbessert.

2 Konstruktion des Cover-Codes

2.1 Lemma 5

Seien $n \leq 1$, $0 < \rho < \frac{1}{2}$, und $\beta(n) = \sqrt{n\rho(1-\rho)}$. Dann kann ein Cover-Code der Länge n, Radius höchstens ρn, und Größe höchstens $n^2\beta(n)2^{(1-h(\rho))n}$ in Zeit $O^*(2^{3n})$ aufgebaut werden.

Da solch ein Cover-Code durch den Greedy-Algorithmus erstellt wird, ist die Zeitgrenze leicht zu erkennen: Jeder der möglichen $(\leq 2^n)$ Bälle wird überprüft ob er ein größter ist $(\leq 2^n)$. Dies wird höchstens 2^n mal iteriert.

Um eine bessere Laufzeit zu erlangen wird Lemma 6 eingeführt:

2.2 Lemma 6

Sei $d \leq 2$ ein Teiler von $n \geq 1$ und $0 < \rho < 1/2$. Dann existiert ein Polynom q_d, so dass ein Cover-Code der Länge n, Radius höchstens ρn, und Größe höchstens $q_d(n)2^{(1-h(\rho))n}$ in Zeit $O(q_d(n)(2^{\frac{3n}{d}} + 2^{(1-h(\rho))n}))$ erzeugt werden kann.

Um dies zu zeigen wird das Codewort in d Blöcke der Länge $\frac{n}{d}$ geteilt. Für jeden Block ist es laut Lemma 5 möglich, bei einem Radius $r \leq \frac{\rho n}{d}$ einen Cover-Code der

Größe $\leq (n)^2\beta(n)2^{(1-h(\rho))\frac{n}{d}}$ in Zeit $O^*(2^{\frac{3n}{d}})$ zu entwerfen, so dass die Größe des zusammengesetzten Cover-Codes höchstens $(n^2\beta(n)2^{(1-h(\rho))\frac{n}{d}})^d = n^{2d}\beta(n)^d 2^{(1-h(\rho))n}$ ist. Die einzelnen Cover-Codes für die $poly(n)2^{(1-h(\rho))n}$ Bälle lassen sich in Zeit $O^*(2^{3n/d})$ berechnen.

Lemma 7 lehrt, dass man nahezu die gleiche Laufzeit auch mit polynomiellem Platzbedarf erreichen kann.

2.3 Lemma 7

Seien $\delta > 0$ und $0 < \rho < 1/2$. Existiert eine Konstante $b = b(\delta,\rho)$, so dass für jedes $n = bl$, ein Cover-Code der Länge n, Radius höchstens ρn, und Größe höchstens $2^{(1-h(\rho)+\delta)n}$ mit polynomiellen Platzbedarf in Polynomzeit pro Codewort aufgebaut werden kann.

Dies ergibt sich direkt aus Lemma 6 und Korrolar 1.

3 Hauptalgorithmus und Analyse

Im Hauptalgorithmus wird also zuerst ein Cover-Code entworfen und der dann mittels lokaler Suche abgeklappert.

Zur Analyse des Hauptalgorithmus definiert man:

3.1 Definition

- $T_1(n,\rho,d)$ ist die Zeit zum Aufbau des Cover-Codes. Nach Lemma 6 ist also $T_1(n,\rho,d) = q_d(n)(2^{\frac{3n}{d}} + 2^{(1-h(\rho))n})$
- $B(n,\rho,d)$ ist die Größe des Cover-Codes, also die Anzahl der Bälle. Nach Lemma 6 ist also $B(n,\rho,d) = q_d 2^{(1-h(\rho))n}$
- $T_2(n,\rho)$ ist die Zeit pro Suche in einem Ball. Nach Lemma 3 ist also $T_2(n,\rho) = poly(n)k^{\rho n}$

3.2 Der Hauptalgorithmus

Also ist die Laufzeit im Hauptalgorithmus $T_1 + B * T_2$. Um den exponentiellen Teil zu minimieren wählt man $\rho = \frac{1}{k+1}$, und um T_1 kleiner zu bekommen als BT_2 und somit unwichtig werden zu lassen wählt man zb. $d = 6$.

1. Setze $\rho = \frac{1}{k+1}$.
2. Wende Lemma 6 mit $d = 6$ an, um einen Cover-Code C der Länge n und Radius höchstens ρn zu generieren.
3. Für jedes Codewort a in C durchlaufe Search(F, a, ρn). Liefer wahr zurück, wenn mindestens eine Prozedur wahr zurückliefert. Ansonsten liefer falsch zurück.

3.3 Theorem 1

Der Hauptalgorithmus löst k-SAT in Zeit $O^*((2 - \frac{2}{k+1})^n)$, wobei n die Zahl der Variablen in der Eingangsformel ist.

Dieses ist durch einfaches Umformen leicht zu ersehen...

3.4 Theorem 2

Für jedes $\varepsilon > 0$ und eine ganze Zahl k kann der Hauptalgorithmus so modifiziert werden, dass er in Zeit $O^*((2-\frac{2}{k+1}+\varepsilon)^n)$ ist und mit polynomiellem Platz arbeitet.

Wird im Hauptalgorithmus statt Lemma 6 Lemma 7 benutzt mit $\delta = log_2(\frac{k+1}{2k}\varepsilon + 1)$, so ist BT_2 durch Umformungen gleich $poly(n)(2-\frac{2}{k+1}+\varepsilon)^n$.

4 Verbesserung des Algorithmus bei 3-Sat

Durch besseres Auswählen der Klauseln, in denen Literale verändert werden, kann die Komplexität bei 3-Sat von $O^*(3^r)$ zu $O^*(2,848^r)$ verbessert werden.

4.1 Definition

- Eine (1,...,1,0,...,0)-Klausel mit m Einsen und p Nullen ist eine Klausel die unter der Belegung a genau m erfüllte und p unerfüllte Literale hat.
- Eine Formel F ist i-false unter der Belegung a wenn es innerhalb der Hammingdistanz i zu a keine erfüllende Belegung gibt.

4.2 Lemma 8

Sei F eine unter a unerfüllte Formel. Wenn $l_1 \vee l_2 \vee l_3$ eine (0,0,0)-Klausel ist und $\overline{l_i}$ eine (1)-Klausel ist, dann hat man folgende Äquivalenz: F ist geanu dann erfüllbar innerhalb einer Hamming Distanz $\leq r$ von a, wenn es ein $j \neq i$ gibt, so dass $F_{|l_j=1}$ erfüllbar innerhalb einer Hammingdistanz $\leq r-1$ von a ist.

Der Beweis ist analog zu dem von Lemma 1, nur dass ein Literal ausgeschlossen wird, weil es schon durch eine (1)-Klausel festgesetzt ist.

Um später eine Auswahl darüber zu treffen, welche Klauseln zum Verändern gewählt werden, definiert man:

4.3 Definition

Sei F eine nicht-triviale Formel in 3-CNF. Es gilt:

1. Sei F falsch unter a. F ist genau dann I-constrained bezüglich a, wenn
   - F eine (0)- oder (0,0)-Klausel hat, oder
   - F eine (0,0,0)-Klausel mit einem Literal l hat und $\bar{l}$ eine (1)-Klausel von F ist.
2. Sei F 1-false unter a. F ist genau dann II-constrained bezüglich a, wenn
   - F I-constrained bezüglich a ist, oder
   - F eine (0,0,0)-Klausel mit einem Literal l hat und $F_{|l=1}$ I-constrained ist.
3. Sei F 2-false unter a. F ist genau dann III-constrained bezüglich a, wenn
   - F I-constrained oder II-constrained bezüglich a ist, oder
   - F eine (0,0,0)-Klausel hat und für jedes Literal l dieser Klausel $F_{l=1}$ II-constrained ist.

4.4 Lemma 9

Sei F nicht-trivial und 3-false, l ein unter a unerfülltes Literal. Sei außerdem $F_{|l=1}$ nicht-trivial unter a (Es ist automatisch 2-false). Dann gilt: $F_{|l=1}$ ist III-constrained bezüglich a $\Rightarrow$ $F_{|l=1}$ ist II-constrained unter a oder F ist III-constrained unter a.

Hinweis zum Beweis: Wenn F nicht-trivial,1-false unter a und nicht I-constrained ist, dann ist $F_{|l=1}$ nicht-trivial für jedes Literal l von F, das unter a zu einer nicht erfüllten Klausel gehört. Innerhalb dieses Beweises wird I-, II-, III-constrained statt I-, II-, III-constrained bezüglich a benutzt.

Beweis: Wenn $F_{|l=1}$ III-constrained ist, dann ist $F_{|l=1}$ entweder I- oder II-contrained (Dann ist das Lemma erfüllt.), oder III-constrained aber nicht I- oder II-constrained. (Dieser Fall wird also behandelt.) Dann hat $F_{|l=1}$ eine (0,0,0)-Klausel $l_1 \vee l_2 \vee l_3$, und jedes $F_{|l=1,l_i=1}$ ist II-constrained. Wäre ein $F_{|l=1,l_i=1}$ I-constrained, dann wäre $F_{|l=1}$ II-constrained. Dieser Fall ist aber schon behandelt. Deswegen ist jedes $F_{|l=1,l_i=1}$ nicht I-constrained und hat eine (0,0,0)-Klausel $k_i \vee k_{i,1} \vee k_{i,2}$, und $G_i = F_{|l=1,l_i=1,k_i=1}$ ist I-constrained für i=1,2,3 und falsch unter a. Es gibt drei Fälle wenn G_i I-constrained ist:

1. (0,0)-Klausel. G_i hat eine (0,0)-Klausel $h_1 \vee h_2$. $h_1 \vee h_2$ wurde erzeugt bei $k_i = 1$. (Sonst wäre $F_{|l=1,l_i=1}$ schon I-constrained und damit $F_{|l=1}$ II-constrained.) Deshalb hat $F_{|l=1,l_i=1}$ eine (1,0,0)-Klausel $\overline{k_i} \vee h_1 \vee h_2$ und die (0,0,0)-Klausel $k_i \vee k_{i,1} \vee k_{i,2}$, wobei $h_1, h_2 \neq k_i$ ist. Daraus folgt, dass F schon die (1,0,0)-Klausel $\overline{k_i} \vee h_1 \vee h_2$ und die (0,0,0)-Klausel $k_i \vee k_{i,1} \vee k_{i,2}$, wobei $h_1, h_2 \neq k_i$ ist, hat, und damit $F_{|k_i=1}$ I-constrained ist. Deswegen ist F II-constrained.
2. (0)-Klausel. G_i hat eine (0)-Klausel h_i. h_i wurde erzeugt bei $k_i = 1$. (Sonst hätte $F_{|l=1,l_i=1}$ eine (0)-Klausel womit $F_{|l=1}$ II-constrained wär.) Deshalb hat $F_{l=1,l_i=1}$ eine (1,0)-Klausel $\overline{k_i} \vee h_i$ und die (0,0,0)-Klausel $k_i \vee k_{i,1} \vee k_{i,2}$, wobei $h_i \neq k_i$ ist. Dann gibt es zwei Möglichkeiten:
   - $F_{|l=1}$ hat $\overline{k_i} \vee h_i$, dann ist $F_{|l=1,k_i=1}$ I-constrained und $F_{|l=1}$ damit II-constrained.
   - $F_{|l=1}$ hat $\overline{l_i} \vee \overline{k_i} \vee h_i$, da F diese und die (0,0,0)-Klauseln $(l_1 \vee l_2 \vee l_3)$ und $(k_i \vee k_{i,1} \vee k_{i,2})$ auch hat, ist auch $F_{l_i=1}$II-constrained. Es bleibt nur zu zeigen, dass $F_{|l_j=1}$ für $j \neq i$ II-constrained ist, damit F III-constrained ist. Dadurch, dass die anderen $F_{|l=1,l_i=1}$ ja gleichzeitig überprüft werden, ist dieser Fall abgeschlossen, wenn alle Fälle abgeschlossen sind. (Inklusive dieser Art des Abschlusses)
3. (0,0,0)-Klausel und (1)-Klausel. G_i hat eine (0,0,0)-Klausel $h_i' \vee h_i'' \vee h_i'''$ und eine (1)-Klausel $\overline{h_i'}$. Dann gibt es 2 Möglichkeiten:
   - $F_{|l=1}$ hat die (1,1)-Klausel $\overline{k_i} \vee \overline{h_i'}$, dann ist $F_{|l=1,k_i=1}$ I-constrained und damit $F_{|l=1}$ II-constrained.
   - $F_{|l=1}$ hat die (1,1,1)-Klausel $\overline{l_i} \vee \overline{k_i} \vee \overline{h_i'}$. Da F diese und die (0,0,0)-Klausel $(l_1 \vee l_2 \vee l_3)$ auch hat, ist auch $F_{l_i=1}$II-constrained (hat die (1,1)-Klausel $\overline{k_i} \vee \overline{h_i'}$, fällt also auf den Möglichkeit hier drüber). Es bleibt nur zu zeigen, dass $F_{|l_j=1}$ für $j \neq i$ II-constrained ist damit F III-constrained ist. Dadurch, dass die anderen $F_{|l=1,l_i=1}$ ja gleichzeitig überprüft werden, ist dieser Fall abgeschlossen, wenn alle Fälle abgeschlossen sind (Inklusive dieser Art des Abschlusses).

Damit sind alle Fälle abgeschlossen, der Beweis bewiesen und die Sitzung gesessen.