On the Parameterized Complexity of Exact Satisfiability Problems

Thomas Knott

1 Überblick

In dem Papier *On the Parameterized Compelxity of Exact Satisfiability Problems* werden verschiedene Ausprägungen des Erfüllbarkeitsproblems mittels parametrisierter Komplexität klassifiziert. In dieser Sitzung wird gezeigt, dass MonoMaxExactSATund ResMaxExactSAT in FPT enthalten sind, und dass für q-MaxExactSATein linearer Problemkern vorliegt.

2 Definitionen

2.1 Fixed Parameter Tractable

Eine Sprache $\mathfrak{L}$ ist parametrisiert wenn $\mathfrak{L} \subseteq \Sigma^* \times \mathbb{N}$. Ist $(x, k) \in \mathfrak{L}$, dann ist das Wort x eine Instanz des Problems und k der Parameter der Instanz. Die Sprache $\mathfrak{L}$ ist nun *fixed parameter tractable* wenn ein Algorithmus existiert, welcher in $O(f(k)p(|x|))$ entscheidet, ob (x, k) in $\mathfrak{L}$ ist mit $f : \mathbb{N} \to \mathbb{N}$ und einem Polynom p. Daher ist FPT die Komplexitätsklasse aller Sprachen, die *fpt* sind.

2.2 MaxExactSAT und ResMaxExactSAT

Eine CNF Formel ist eine Multimenge von Klauseln, und eine Klausel ist eine Multimenge von Literalen.
Eine Formel wird monoton genannt, wenn jedes Literal nur in positiver oder negativer Form auftritt.
Eine Klausel wird von einer Belegung exakt erfüllt, wenn genau ein Literal

von dieser auf wahr gesetzt wird; sind es mehr, ist die Klausel übererfüllt.
Beim MaxExactSAT Problem geht es nun um die Frage, ob eine Belegung existiert welche k Klauseln in einer Formel exakt erfüllt. Dies wird beim ResMaxExactSAT Problem noch eingeschränkt dadurch, dass keine überbelegten Klauseln existieren dürfen.

3 Monotone Formeln

Theorem 2. MonoMaxExactSAT $\in$ FPT, mit Klauseln als Mengen.

Beweis. Tritt eine Variable in mehr als k Klauseln auf, setzt man diese auf wahr und alle anderen auf falsch um mindestens k Klauseln exakt zu erfüllen. Ist dies nicht der Fall und die Formel besteht aus weniger als $k^2 + k$ Klauseln, existieren auch nur 2^{k^2+k} unterscheidbare Variablen und das Problem kann mittels aufwändiger Suche gelöst werden.
In jedem anderen Fall findet der folgende Algorithmus eine Belegung welche mindestens k Klauseln exakt erfüllt.

1. Konstruktiere einen Hypergraphen $G(F)$ in welchem die Klauseln der Formel als Knoten und die Variablen als Hyperkanten dargestellt werden.

2. Konstruktiere ein maximales Matching M. Deckt es mehr als k Knoten ab, setze die entsprechenden Variablen auf wahr und sei fertig.

3. Ist dies nicht der Fall, bestehen zwei Möglichkeiten:

   - Im Hypergraphen existiert mindestens eine echte Hyperkante: Die vom Matching repräsentierten Variablen werden auf falsch gesetzt und die abgedeckten Knoten/Klauseln entfernt. Weiter mit 2.
   - G ist „nur noch“ ein Graph: Die nicht von M abgedeckten Knoten bilden ein independent set I. Schritt 3 wurde höchstens $k-3$ mal ausgeführt und jedesmal wurden höchstens $k-1$ Klauseln gelöscht, daher existieren noch mindestens $2k$ Klauseln. M hat weniger als k Knoten abgedeckt, somit gilt $|I| > k$. Da die Knoten von I disjunkt sind, ist es einfach eine Belegung zu finden, welche die Formel exakt erfüllt.

4 Klauseln mit fester Länge

Lemma 4. Sei F eine Formel in q-CNF. Wenn ein Literal x in mindestens $k \cdot 2^{q-1}$ Klauseln auftritt gibt es eine Belegung der Variablen welche mindestens k Klauseln exakt erfüllt.

Beweis. Sei X die Menge von Klauseln, welche x enthalten und X^- die Menge von Klauseln, die man durch weglassen von x aus X erhält. Wenn nun eine Belegung existiert, sodass in X^- k Klauseln unerfüllt bleiben, existiert für X eine Belegung, welche k Klauseln durch x exakt erfüllt.
Der Erwartungswert der Anzahl von erfüllten Klauseln in X^- durch eine zufällige Belegung ist:

$$E \leq \left(1 - \tfrac{1}{2^{q-1}}\right) \cdot |X^-|$$

Nach der first moment Methode gilt, dass eine Belegung existiert, welche höchstens E Klauseln erfüllt. Daher existiert eine Belegung, welche

$$|X^-| - E \geq \left(1 - \left(1 - \tfrac{1}{2^{q-1}}\right)\right) \cdot |X^-| \geq \tfrac{1}{2^{q-1}} \cdot k \cdot 2^{q-1} = k$$

Klauseln aus X^- nicht erfüllt. Und somit eine Belegung, welche k Klauseln aus X erfüllt.

Lemma 5. Sei F eine Formel in q-CNF mit $|F| \geq k^2 q 2^q$. Dann existiert eine Belegung, welche mindestens k Klauseln exakt erfüllt.

Beweis. Wenn ein Literal in mindestens $k \cdot 2^{q-1}$ Klauseln auftritt, gilt nach Lemma 4, dass mindestens k Klausel exakt erfüllt werden können. Trifft dies nicht zu, wählt man eine beliebige Klausel C und löscht alle anderen Klauseln welche eine Variable aus C enthalten. Wiederholt man dies k mal, entfernt man in jeder Iteration maximal $q \cdot k \cdot 2^q$ Klauseln und insgesamt weniger als $q \cdot k^2 \cdot 2^q$. Man erhält daher eine Belegung welche mindestens k Klauseln exakt erfüllt.

5 ResMaxExactSAT

In diesem Abschnitt wird gezeigt, dass ResMaxExactSAT in FPT liegt. Hierbei wird unterschieden zwischen monotonen und nicht monotonen For-

meln.

5.1 Nicht monotone Formeln

Bei nicht monotonen Formel verzweigen wir bei $x, \overline{x}$, dadurch wird mindestens eine Klausel exakt erfüllt. Nach k Iterationen werden k Klauseln exakt erfüllt und man sucht eine mögliche Lösung für den Rest der Formel.

5.2 Monotone Formeln

Zu Beginn wird eine andere Repräsentation der Formel gewählt. Für jede Variable gibt eine Menge an in welchen Klauseln sie auftritt.

$$F = \{\{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}, \{x_2, x_3\}, \{x_4, x_5\}\}$$

$$O_1 = \{C_1, C_2\}, O_2 = \{C_1, C_3\}, O_3 = \{C_1, C_2, C_3\}, O_4 = \{C_2, C_4\}, O_5 = \{C_4\}$$

Definition. $M = (M_1, \ldots, M_l)$ ist ein $(k_1, \ldots, k_l)$-Approximator für F, wenn paarweise disjunkte O_i existieren mit $|O_i| = k_i$ und $M_i \subseteq O_i$. Wenn $M_i = O_i, 1 \leq i \leq l$ gilt, sagt man der Approximator ist gesättigt.

Beispiel. Für

$$X_1 = \{C_1, C_2\}, X_2 = \{C_1, C_3\}, X_3 = \{C_1, C_2, C_3\},$$
$$X_4 = \{C_2, C_4\}, X_5 = \{C_4\}$$
und
$$O_1 = X_2 = \{C_1, C_3\}, O_2 = X_4 = \{C_2, C_4\}$$
wäre $M_1 = \{C_1\}, M_2 = \{C_4\}$ ein $(2, 2)$-Approximator.

Lemma 6. Sei F eine monotone Formel. Ist $(M_1, \ldots, M_l)$ ein ungesättigter $(k_1, \ldots, k_l)-$*Approximator* für F, dann fügt $Improve(F, M_1, \ldots, M_l, k_1, \ldots, k_l)$ den Mengen $M_1, \ldots, M_l$ in mindestens einem nicht-deterministischen Zweig Elemente hinzu, ohne die Approximator-Eigenschaft zu zerstören.

Beweis. *Improve* fügt an zwei Stellen den M_i Elemente hinzu. Passiert dies vor dem zweiten *return*-Befehl, so ist $(P_1 \cup \cdots \cup P_l)$ ein gesättigter Approximator. Passiert es nach der i-ten Iteration vor dem ersten *return*-Befehl, dann gilt:

```
1  Improve(F, M_1, ..., M_l, k_1, ..., k_l)
2  P_i := M_i, ∀i ∈ I = {1...l};
3  for  i = 1...l  do
4              if  ∃ variable x :
5                          −x ist nicht in P_j für alle j ≠ i
6                          −x ist in allen Klauseln von M_i
7                          −x tritt in genau k_i Klauseln auf
8              then  P_i := clauses(x) ;
9              else  rate Klausel C ∈ P_1 ∪ ··· ∪ P_l − (M_1 ∪ ··· ∪ M_l);
10                                   M_i := M_i ∪ {C} ;
11                                   return
12 fi  od ;
13 M_i := P_i, ∀i ∈ {1...l} ;
14 return ;
```

Abbildung 1: Improve(..)

Nach Voraussetzung existieren Variablen $v_1 \ldots, v_l$ und Mengen $O_1, \ldots, O_l$, so dass $M_i \subseteq O_i$. v_i widerspricht jedoch einer der **if** Bedingungen und da dies nicht die 2. und 3. Bedingung sein kann, existiert für $i \neq j$ eine Klausel C in O_i und P_i jedoch nicht in $M_i = P_i$.
Wird die richtige Klausel geraten wird diese zu M_i hinzugefügt.

Lemma 7. Der Algorithmus A gibt in allen nicht-deterministischen Zweigen *no* zurück, wenn keine Lösung existiert. Andernfalls gibt A in mindestens einem Zweig die korrekte Lösung und in allen anderen *no* zurück.

Beweis. Der Algorithmus A überprüft die Korrektheit der Lösung vor der Rückgabe, so dass keine falschen Antworten möglich sind. Da $(M_1, \ldots, M_l)$ ein $(k_1, \ldots, k_l)$-Approximator ist führt, nach Lemma 6, das richtige Raten zu einer Schleifen Invarianz.

Proposition 1. Sei F eine monotone Formel in welcher jede Variable höchsten k-mal auftritt.

```
1  Input:  monotone Formel F, Zahlen k_1, ..., k_l
2  Output:  {v_1, ..., v_l} zugehörig zu (k_1, ..., k_l)-Approximator
3
4  M_i := ∅ , ∀i ∈ I = {1, ..., k_l};
5  while |M_1| + ··· + |M_l| < k_1 + ··· + k_l  do
6            Improve(F, M_1, ..., M_l, k_1, ..., k_l);
7  od;
8  if  ∃v_1, ..., v_l, v_i in genau den Klauseln von M_i
9            then return {v_1, ..., v_l}
10 else return no fi;
```

Abbildung 2: Algorithmus A

Wenn eine Belegung existiert, welche mindestens k Klauseln exakt erfüllt, existiert auch eine Belegung welche $k - 2k$ Klauseln exakt erfüllt.

Theorem 4. Es existiert ein Algorithmus, welcher monotone ResMaxExactSAT in $O((2k)^{4k} \cdot poly(|F|))$ entscheidet.

Beweis.

```
1  Rate Größe aller Mengen O = (O_1, ..., O_l)
2  Initialisiere den Approximator M = (M_1, ..., M_l)
3  while M nicht vollständig
4            do Improve(...)
5  od
```

Anzahl der Zweige:
Für Schritt 1 gibt es höchstens $(2k)^{2k}$ verschiedene Möglichkeiten. Desweiteren gibt es $\binom{k}{l}$ Möglichkeiten den Approximator zu initialisieren und höchstens $2k$ neue Klauseln, jede aus $2k$ Möglichkeiten.