Unit-Conflict-Free Max-Sat Parameterized Above Expectation

June 2, 2010

Abstract

In the study of Fixed-Parameter Tractability (FPT) one looks at computationally hard problems which come with a natural parameter, k, and attempts to restrict the combinatorial explosion to a function of k only. Thus, for small k, the problem can be solved in a reasonable time. In the Max-Sat problem, the aim is to pick an assignment to satisfy the maximum number of clauses in a formula F in Conjunctive Normal Form (CNF). By an averaging argument, it is always possible to satisfy $m/2$ clauses (where F has m clauses). This only occurs when for a variable x, both clauses (x) and $(\bar{x})$ appear in F, which is easy to spot. We say a formula is unit-conflict free (UCF) if for every variable x at most one of the clauses (x) and $(\bar{x})$ appear in F. For a UCF CNF formula F, it is always possible to satisfy at least $p \cdot m$ clauses, where $p = (\sqrt{5} - 1)/2$. The parameterized form of the problem thus asks whether it is possible to satisfy at least $p \cdot m + k$ clauses of F. I'll describe the algorithm showing this problem is fixed-parameter tractable, obtained in joint work with G. Gutin, M. Jones, A. Yeo (arXiv:1004.0526v1).

Definition: Fixed-Parameter Tractability

- Interested in study of problems in NP, so under the assumption P $\neq$ NP, there is no polynomial-time algorithm to solve the problem
- Instead, separate out a parameter, and try to restrict the exponential growth to this factor only
- FPT iff the problem can be solved in time $f(k) \cdot n^{O(1)}$, where n is the size of the problem, $f(k)$ will be superpolynomial, but is likely to be small enough to make the problem practically solvable for small values of k
- Example: Vertex Cover, no known polynomial-time algorithm, but can be solved in time $2O^{(k)}n^{O(1)}$ where k is the size of the vertex cover we require by an exhaustive search.

FPT above a Lower Bound

- If a natural Parameter has a large lower bound (for example, in Max-Sat, one can always satisfy at least $n/2$ clauses), then this doesn't work well as a Parameter - the answer is YES unless k is large, in which case $f(k)$ will be impractical
- Instead, ask "is there an assignment satisfying at least $m/2 + k$ clauses?" (where m is the total number of clauses)
- Here we are paramterising above the known lower bound ($m/2$)

When does the tight lower bound for Max-Sat occur?

- In fact, the only way that the maximum number of clauses satisfied is $m/2$ is when for every clause (x), the clause $(\bar{x})$ also exists
- It is quite easy to spot when this happens!
- So why not consider a problem where at most one of (x) and $(\bar{x})$ exist.
- Call such a system *unit-conflict free* (UCF)

UCF-MAX-SAT-AE

UCF-MAX-SAT-AE
Instance: A UCF CNF formula $I = (V, C)$.
Parameter: The integer k.
Question: Is $\mathrm{Max}(I) \geq p \cdot m + k$?

Where,

- C is a multiset of clauses
- $\mathrm{Max}(I)$ is the maximum number of satisfied clauses
- $p = (\sqrt{5} - 1)/2$

This is a natural above-average question, since it has been shown that a value of $p \cdot m$ is always achievable. The nicest proof of this is by Yannakakis (1994), by a probabilistic argument.

Reduced Form

For an instance I of UCF-MAX-SAT-AE we firstly reduce it according to the following rule:

An instance $I = (V, C)$ of UCF-MAX-SAT-AE *reduced* if the following conditions hold:

1. All clauses in I have the form (x) or $(\bar{x} \vee \bar{y})$ for some $x, y \in V$.
2. For every variable $x \in V$, the clause (x) is in C.

Once UCF-MAX-SAT-AE is shown to be fixed-parameter traceable for this, the generalised version can be shown to also hold (once a couple of minor technicalities are handled)

The Algorithm

While $|V| > 0$, repeat the following:

1. For each $x \in V$, calculate $w_v(x)$ and $w_v(\bar{x})$.
2. Mark some variables TRUE or FALSE, according to the cases:

   A: *There exists $x \in V$ with $w_v(x) \geq w_v(\bar{x})$.* Pick one such x and assign it TRUE.

   B: *A is false, and there exists $x \in V$ with $w_v(x) \geq p \cdot w_v(\bar{x}) + (1-p)/2$.* Pick one such x and assign it TRUE.

   C: *A and B are false, and there exists a good clause.* Pick such a good clause $(\bar{x} \vee \bar{y})$, with (without loss of generality) $\epsilon(x) \geq \epsilon(y)$, and assign y FALSE and x TRUE.

   D: *A, B and C are false.* Pick any clause $(\bar{x} \vee \bar{y})$ and pick z such that both clauses $(\bar{x} \vee \bar{z})$ and $(\bar{y} \vee \bar{z})$ exist. Consider the six clauses $(x), (y), (z), (\bar{x} \vee \bar{y}), (\bar{x} \vee \bar{z}), (\bar{y} \vee \bar{z})$, and all 2^3 combinations of assignments to the variables x, y, z, picking the assignment satisfying the maximum possible weight from the six clauses.

The Algorithm (contd.)

3. Perform the following simplification: For any variable x assigned FALSE, remove any clause containing $\bar{x}$, remove any unit clause (x), and remove x from V. For any variable x assigned TRUE, remove any unit clause (x), and remove $\bar{x}$ from any clause containing $\bar{x}$.
4. For each y remaining, if there is a clause of the form $(\bar{y})$, do the following: If the weight of this clause is greater than $w_v(y)$, then replace all clauses containing the variable y (that is, literals y or $\bar{y}$) with one clause (y) of weight $w_v(\bar{y}) - w_v(y)$. Otherwise remove $(\bar{y})$ from C and change the weight of (y) to $w(y) - w(\bar{y})$.

Important Lemma

This algorithm gives us an assignment which achieves the claim in the following Lemma:

Lemma

If $I = (V, C)$ *is a reduced instance of* UCF-MAX-SAT-AE, *then there exists a solution to* I *with weight at least*

$$p \cdot w(C) + \frac{|V|(2 - 3p)}{2},$$

where p *is the positive root of* $p^2 + p - 1 = 0$, *and such a solution can be found in polynomial time.*

Proof: Outline

- Inductive
- Clearly algorithm is polynomial time
- If Cases A, B, C hold then calculations show we can mark suitably
- Otherwise, there is a small subset of clauses only containing 3 variables, and so brute force can be applied

Definitions

- We call a clause $(\bar{x} \vee \bar{y})$ *good* if for every literal $\bar{z}$, the set of clauses containing $\bar{z}$ is not equal to $\{(\bar{x} \vee \bar{z}), (\bar{y} \vee \bar{z})\}$.
- We define $w_v(x)$ to be the total weight of all clauses containing the literal x, $w_v(\bar{x})$ similarly (Note that $w_v(\bar{x})$ is different from $w(\bar{x})$)
- $\epsilon(x) = w_v(x) - p \cdot w_v(\bar{x})$
- $\gamma = (2 - 3p)/2 = (p - 1)^2/2$
- $\Delta(I) = \mathrm{Max}(I) - p \cdot w(C)$.

Two Lemmas

Lemma

For an instance I, if we assign a variable x TRUE, and run Steps 3 and 4 of the algorithm, the resulting instance I' satisfies the following:

$$\Delta(I) \geq \Delta(I') + (1 - p)\epsilon(x).$$

Furthermore, we have $|V'| = |V| - 1$, unless there exists $y \in V'$ such that (y) and $(\bar{x} \vee \bar{y})$ are the only clauses containing y and they have the same weight. In this case, y is removed from V'.

Lemma

For an instance I, if we assign a variable x FALSE, and run Steps 3 and 4 of the algorithm, the resulting instance I' has $|V'| = |V| - 1$ and satisfies the following:

$$\Delta(I) \geq \Delta(I') - p \cdot \epsilon(x).$$

Proof Outline (contd)

Using these two Lemmas, one works through each case, showing that after running one iteration, the new system will inductively satisfy the Main Lemma.

Generalisations

- UCF is also known as 2-satisfiability
- Can be generalised a system is *t-satisfiable* iff any subset of t clauses can be simultaneously satisfied
- Can ask - is the generalised version of this problem fixed parameter tractable
- Unfortunately, the lower bound is not known
- As t tends to infinity, it is known (Trevisan, 1997) that the lower bound tends to $3/4$

3-satisfiable UCF

The lower bound for 3-satisfiability is known, so the parametrised problem can be defined as follows:

Instance: A 3-satisfiable CNF formula $I = (V, C)$.
Parameter: The integer k.
Question: Is $\mathrm{Max}(I) \geq 2|C|/3 + k$*?*

This question can be extended to $t > 3$.
It is not known whether this is fpt. Naive attempts at generalising the method for 2-satisfiable instances are likely to be difficult - many cases will be generated.

End