15-859FF: Coping with Intractability **CMU, Fall 2019**
Lecture #9: Sparsification cont. and 2-CNF deletion October 2, 2019
Lecturer: Venkatesan Guruswami Scribe: Peter Manohar

1 Finishing sparsification

Recall the algorithm for sparsification from the previous lecture, due to [IPZ98]. We start with a 2SAT instance ϕ. On each step we find a "weak sunflower" of clauses $C_1, \ldots, C_t$ where $t \geq \theta_i$, all the clauses have size j, and the core $C = \cap_{\ell=1}^{t} C_\ell$ has size $j - i$. Then, we branch on the sunflower by either adding the core C or all of the petals $C_\ell \setminus C$.

We want to show that the number of clauses containing at most i literals added on any path in the tree is $\leq \beta_i n$ for some constant β_i. In the last lecture we showed that adding a j-clause (where $j < i$) in the path can eliminate at most $2\theta_{i-1}$ i-clauses. This implies that the number of clauses with $\leq i$ literals added on any path is at most

$$\underbrace{\beta_{i-1}n}_{\#\text{ clauses with } < i \text{ literals}} + \underbrace{\theta_{i-1}n}_{\#\ i\text{-clauses remaining at leaf}} + \underbrace{\beta_{i-1}n(2\theta_{i-1})}_{\#\ i\text{-clauses eliminated by a } j\text{-clause for } j<i} \leq 4\beta_{i-1}\theta_{i-1}n \ .$$

So, we can set $\beta_i = 4\beta_{i-1}\theta_{i-1}$ for $i \geq 2$. Note that $\beta_1 = 1$, as there are only n clauses containing one literal (and each clause can only be added once).

We now set $\theta_i = \alpha\beta_i$, where α is some constant that depends on k and ε. Then $\beta_i = 4\beta_{i-1}\theta_{i-1} = 4\alpha\beta_{i-1}^2$, which implies that $\beta_i = (4\alpha)^{2^{i-1}-1}$.

On any path, the number times we branch right on a weak sunflower with *petals* of size i is at most $\beta_i n/\theta_i = n/\alpha$, because we only add at most $\beta_i n$ i-clauses on a path, and each of these sunflowers has at least θ_i petals. So, by summing over all i we conclude that the total number of right branches is at most kn/α.

Since at most $\beta_{k-1}n$ clauses are added along any path, we get that

$$\#\text{ leaves} \leq \binom{\beta_{k-1}n}{kn/\alpha} \leq \left(\frac{\beta_{k-1}e\alpha}{k}\right)^{\frac{kn}{\alpha}} \leq \left((4\alpha)^{2^{k-1}}\right)^{\frac{kn}{\alpha}} \leq 2^{O(2^k k \frac{\log\alpha}{\alpha})} \leq 2^{\varepsilon n} \ ,$$

for α large enough. This completes the proof of the sparsification lemma.

Remark 9.1. We have shown that k-SAT $\leq_{\text{SERF}}$ k-SAT$(c_{k,\varepsilon})$, where c is doubly-exponential in k and $1/\varepsilon$. It would be nice to get $c_{k,\varepsilon}$ to be, say, poly$(k, 1/\varepsilon)$.

Remark 9.2. We have defined the Exponential Time Hypothesis (ETH) and the Strong Exponential Time Hypothesis (SETH). From our definitions, it is a priori not clear that SETH implies ETH. The sparsification lemma can be used to show that this is true.

Proof that SETH implies ETH. Pick a k-SAT instance for large k, so that δ_k is arbitrarily close to 1. Sparsify the formula ϕ using the sparisification lemma, and then apply the standard reduction from k-SAT to 3-SAT (which is now linear time) to conclude. □

2 Almost 2-SAT/2-CNF Deletion

We know that 2-SAT is in P. However, given an unsatisfiable 2-SAT instance ϕ, finding the maximum number of clauses that can be satisfied (MAX-2-SAT) is NP-hard (this is via a simple

gadget reduction from 3-SAT). This is equivalent to minimizing the number of unsatisfied clauses or deleting the fewest clauses to make the formula satisfiable (2-CNF deletion). For a 2-SAT formula ϕ with n variables and m clauses, we define

$$\text{min}-\text{unsat}(\phi) := \min_{x \in \{0,1\}^n} \#\text{ of unsat clauses in } \phi(x) \ .$$

Approximating min−unsat(ϕ) is an interesting problem.

- No $O(1)$-factor approximation algorithm is known.
- Under $\mathsf{P} \neq \mathsf{NP}$, min−unsat($\phi$) is hard to approximate within a factor of 2.8, but even 2.9-factor hardness is not known.
- The best algorithm is nontrivial and achieves $O(\sqrt{\log n})$-factor approximation, via SDPs. $O(\log n)$-factor approximation is known using LPs.
- Under the Unique Games Conjecture (UGC), min−unsat(ϕ) has no c-factor approximation algorithm for any constant c. (First such UGC conditioned result [Kho02].)

2.1 Aside: the Unique Games Conjecture

Theorem 9.3 ([Kho02])**.** *Under UGC, it is hard to distinguish between the following two cases:* min−unsat(ϕ) $\leq \varepsilon m$*, or* min−unsat(ϕ) $\geq \sqrt{\varepsilon} m$*, for all* $\varepsilon > 0$*.*

Definition 9.4 (Unique Games)**.** An instance of Unique Games consists of a prime p, n variables $x_1, \ldots, x_n$, and constraints $x_i - x_j = \alpha_{ij}$, where $\alpha_{ij} \in \mathbb{F}_p$. The goal is to assign each x_i some value in $\mathbb{F}_p$ to maximize the number of constraints satisfied.

Definition 9.5 (Unique Games Conjecture)**.** For every $\delta > 0$, there exists a prime p such that distinguishing between the case when $\geq 1 - \delta$ fraction of constraints are satisfiable and $\leq \delta$ fraction of constraints are satisfiable is hard.

Note: Unique Games has an FPT algorithm.

2.2 2-CNF deletion is in FPT

We show that 2-CNF deletion has an FPT algorithm, when parametrized by min−unsat(ϕ).

Theorem 9.6. *Determining if* min−unsat(ϕ) $\leq k$ *can be done in time* $4^k \text{poly}(n)$*.*

Proof outline. We will show the above theorem via a sequence of FPT reductions. We begin by defining the two problems in the reduction.

- Variable Deletion Almost 2-SAT. Given a 2-SAT formula ϕ, can we find $\leq k$ variables such that removing them makes ϕ satisfiable? Removing a variable removes all clauses containing the variable or its negation. Parametrized by k.
- Vertex Cover above Matching. Given (G, k'), determine if there is a vertex cover of size $\leq k'$ in G. Parametrized by $k = k' - m(G)$, where $m(G)$ is the size of a maximum matching in G.

We will show that 2-CNF Deletion $\leq_{\text{FPT}}$ Variable Deletion Almost 2-SAT $\leq_{\text{FPT}}$ Vertex Cover above Matching, and then we show that Vertex Cover above Matching is in FPT. This will prove Theorem 9.6.

The first reduction is an exercise.

Exercise 9.7. Show that 2-CNF Deletion $\leq_{\text{FPT}}$ Variable Deletion Almost 2-SAT.

We now show the second reduction.

Lemma 9.8. *Variable Deletion Almost 2-SAT $\leq_{FPT}$ Vertex Cover above Matching*

Proof. Let ϕ be a 2-SAT formula with variables $x_1, \ldots, x_n$ and clauses $(\ell_i \vee \ell_j)$, where the ℓ's are literals. Define G as follows. G has $2n$ vertices $x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$. For every clause $(\ell_i \vee \ell_j)$, add the edge (ℓ_i, ℓ_j) to E. Note that $m(G) = n$.

Suppose that ϕ can be made satisfiable by removing k variables. Without loss of generality assume that these variables are $x_1, \ldots, x_k$. Let ν be the assignment that satisfies ϕ once these variables are removed. Let S be the set of vertices in G consisting of $\{x_1, \ldots, x_k, \bar{x}_1, \ldots, \bar{x}_k\} \cup \{x_i : \nu(x_i) = \text{ true}\} \cup \{\bar{x}_i : \nu(x_i) = \text{ false}\}$. Observe that $|S| = 2k + n - k = n + k = k + m(G)$. S is also a vertex cover since for every edge $(\ell_i, \ell_j) \in E$, either $\ell_i \vee \ell_j$ is satisfied by ν, in which case one of ℓ_i or ℓ_j is in $\{x_i : \nu(x_i) = \text{ true}\} \cup \{\bar{x}_i : \nu(x_i) = \text{ false}\} \subseteq S$, or else the clause is deleted from ϕ, in which case one of ℓ_i or ℓ_j is in $\{x_1, \ldots, x_k, \bar{x}_1, \ldots, \bar{x}_k\}$, and hence in S.

Conversely, suppose that G has a vertex cover S of size $\leq k + m(G) = k + n$. Let $S' = \{x_i : x_i, \bar{x}_i \in S\}$, and let $S'' = S \setminus S'$. Let ν be the assignment to the literals in S'' that sets ℓ_i to true if $\ell_i \in S''$, and false otherwise. Then ν clearly satisfies ϕ after removing the variables in S' from ϕ. We also have that $|S'| + |S''| = n$, and that $n + k \geq |S| = 2|S'| + |S''| = n + |S'|$, so $|S'| \leq k$ and therefore we have deleted at most k variables from ϕ. $\square$

We show the following theorem, which will complete the proof of Theorem 9.6

Theorem 9.9. *Vertex Cover above Matching has an algorithm with runtime* $4^{k-m(G)}\text{poly}(n)$.

Proof. We actually prove that a "harder" problem, Vertex Cover above LP, has an FPT algorithm. Vertex Cover above LP is defined similarly to Vertex Cover above Matching: given (G, k'), determine if there is a vertex cover of size $\leq k'$ in G. The problem is parametrized by $k = k' - \text{vc}^*(G)$, where $\text{vc}^*(G)$ is the value of the optimal solution to the LP relaxation of vertex cover. We will show that Vertex Cover above LP has an algorithm with runtime $4^{k-\text{vc}^*(G)}\text{poly}(n)$, which will prove the above theorem since $m(G) \leq \text{vc}^*(G)$.

Recall that the vertex cover LP always has a $\frac{1}{2}$-integral optimal solution, and such a solution can be found efficiently. Let $\vec{x}$ be an optimal $\frac{1}{2}$-integral solution, and let $V_0 = \{v : x_v = 0\}$, $V_{1/2} = \{v : x_v = 1/2\}$, and $V_1 = \{v : x_v = 1\}$. We know that G has a vertex cover of size $\leq k$ if and only if $G' := G[V_{1/2}]$ has a vertex cover of size $\leq k' := k - |V_1|$, so our algorithm recurses in G' until $V = V_{1/2}$. Since each iteration decreases the number of vertices by at least 1, there are at most n iterations each taking $\text{poly}(n)$ time. We note that on each iteration, the following relation holds.

Exercise 9.10. $k' - \text{vc}^*(G') = k - \text{vc}^*(G)$

This implies that the parameter does not change via this process.

When $V = V_{1/2}$, we then do the following branching algorithm. Pick $v \in G$, and branch on $G - v$ and $G - N(v)$. The parameter decreases by at least $\frac{1}{2}$ on each step (because each $x_u = \frac{1}{2}$), so the tree has depth $\leq 2k$. This gives us a $4^k\text{poly}(n)$ algorithm. $\square$

References

[IPZ98] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 653–663, 1998. 1

[Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 767–775, 2002. 2, 9.3