Lecture 24: Hardness Assumptions

December 2, 2013

Lecturer: Ryan O'Donnell *Scribe: Jeremy Karp*

1 Overview

This lecture is about hardness and computational problems that seem hard. Almost all of the theory of hardness is based on assumptions. We make assumptions about some problems, then we do reductions from one problem to another. Then, we want to make the minimal number of assumptions necessary to show computational hardness. In fact, all work on computational complexity and hardness is essentially in designing efficient algorithms.

2 NP-Hardness

A traditional assumption to make is that $3SAT$ is not in P or equivalently $P \neq NP$. From this assumption, we also see that Maximum Independent Set, for example, is not in P through a reduction. Such a reduction takes as input a $3CNF$ formula φ. The reduction itself is an algorithm that runs in polynomial time which outputs a graph G. For the reduction to work as we want it to, we prove that if φ is satisfiable then G has an independent set of size at least some value k and if φ is not satisfiable then the maximum independent set of G has size less than k. This reduction algorithm will be deterministic. This is the basic idea behind proofs of NP-hardness.

There are a few downsides to this approach. This only give you a worst-case hardness of a problem. For cryptographic purposes, it would be much better to have average-case hardness. We can think of average-case hardness as having an efficiently sampleable distribution of instances that are hard for polynomial time algorithms. Then, we could efficiently create problem instances if $3SAT$, which could then be reduced to hard instances of other problems. Also, we still don't really know how hard $3SAT$ is. Note that it has been shown that $3SAT \in O(1.31^n)$

3 Learning Parity with Noise

We can also consider other reasonable assumptions. In the last lecture we discussed the Learning With Errors (LWE) problem which is very useful for constructing cryptographic primitives with the assumption the LWE is hard. Recall that this problem is roughly solving noisy n-variable linear equations mod q where q is polynomial in n. LWE seems to be hard in the average case. Also, there is an efficient quantum reduction from the Gap Shortest

Vector Problem with factor $\widetilde{O}(n^{1.\epsilon})$ to LWE, which says if you can solve Gap SVP in worst case then you can solve LWE in the average case.

A related problem is Learning Parity with Noise (LPN), which is basically LWE mod 2. This problem has two parameters, n and $0 < \epsilon < \frac{1}{2}$. To generate instances of the problem, pick a "secret" string $s \sim \mathbb{F}_2^n$. The solving algorithm can ask for a noisy equation about s. When it asks for this, it receives $a_1 s_1 + \ldots + a_n s_n = b$. The solving algorithm gets to see $a \sim \mathbb{F}_2^n$, another uniformly random string, and b which is $a \cdot s$ with probability $1 - \epsilon$. One should think of ϵ as a small constant like .1. The task for the learner is to output s by interacting with this oracle that returns noisy linear equations.

The assumption we make is that LPN is not in P (or another similar assumption like LPN is sub-exponential). In other words, for all constants $0 < \epsilon < \frac{1}{2}$, any polynomial-time algorithm fails except with exponentially small probability. This gives a one way function and very efficient secret key encryption, but it is unknown in general if you can get public key encryption. You can, however, get public key encryption if $\epsilon = \frac{1}{\Theta(\sqrt{n})}$ [Ale03].

The fastest known algorithm for LWE runs in $2^{O(\frac{n}{\log n})}$ time and samples [BKW03]. This algorithm performs "clever" Gaussian elimination. As a side note, if the algorithm can only get polynomially many samples, then the fastest known algorithm runs in $2^{O(\frac{n}{\log \log n})}$ time [Lyu05]. This motivates the assumption that this problem cannot be done in sub-exponential time.

This problem is a search problem, but we could also make a decision problem to determine if one is getting completely random equations or actually getting the noisy linear equations. This decision problem is polynomial-time equivalent to the search problem.

4 Sparse-LPN

Sparse-LPN is the same as LPN except all equations have only $k \geq 3$ nonzero elements of a. We need to be careful if we are to assume this problem is hard. For example, this problem is easy if the algorithm can get $\gg n^k$ samples. This is because we would see every possible equation many times and could essentially remove the noise from the equations and know the correct answer with overwhelming probability. This problem is still easy with $\gg n^{\frac{k}{2}}$ samples, but this requires a more sophisticated algorithm.

However, it's more reasonable to assume that for $k = 3$ there is no polynomial-time algorithm if limited to $O(n)$ samples. Another way to think about this is to find a string s' which agrees wiht s on $\geq \frac{1}{2} + \epsilon$ fraction of the coordinates. The algorithm gets a set of $O(n)$ linear equations:

$$\begin{gathered} s_1 + s_{10} + s_9 = 0 \\ s_{53} + s_{18} + s_5 = 1 \\ \vdots \end{gathered}$$

and with high probability there's an assignment that satisfies a little bit less than $1 - \epsilon$ fraction of the equations but it is seemingly hard to find an assignment that satisfies $\frac{1}{2} + \epsilon$

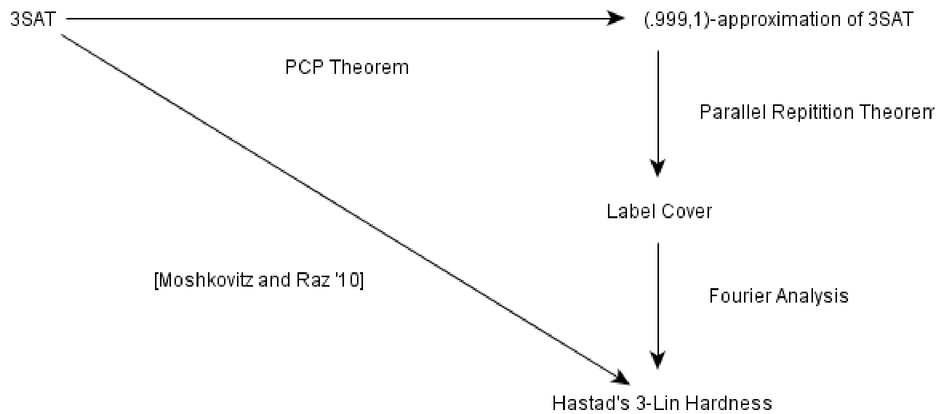

Figure 1: Outline of the proof of Håstad's theorem

fraction of the equations. Then this assumption is like saying that we can efficiently generate hard-seeming instances of this problem. We saw previously that this problem is NP-Hard, so this problem is hard in the worst case.

Theorem 4.1 ([HILL99]). *$(\frac{1}{2}+\epsilon, 1-\epsilon)$-approximating $MAX-3LIN$ (mod 2) is NP-Hard.*

Note, however, that if $k = 2$ then the Goemans-Williamson Max Cut approximation algorithm gives a $(1 - O(\sqrt{\epsilon}), 1 - \epsilon)$-approximation for $MAX - 2LIN$ (mod 2).

Håstad's theorem is frequently used in proving theorems of hardness of approximation. For example, this theorem is the basis for the best known NP-Hardness results of Max Cut, Max $2SAT$, TSP, robust Graph Isomorphism and many others. Then, assuming the underlying assumption, we can generate hard instances of all these problems.

5 Hard instances of $3SAT$

$3SAT$ is much more widely studied than solving linear equations, which gives support assumptions about the problem. It is NP-hard to distinguish a totally satisfiable instance and an unsatisfiable one, whereas it is easy to solve a satisfiable linear equation without noise. One way to make hard instances of $3SAT$ is to take a problem that's hard on average and encode it into a $3SAT$ instance. However, this is a bit inefficient and isn't really done in practice. Another possibility is to pick random clauses, but this fails because if the number of clauses is too small or large the instance will (with high probability) by satisfiable or unsatisfiable, respectively. A third possibility is to pick a secret solution s at random, then output random $3CNF$ clauses consistent with it. This, too, fails because the distribution of the clauses will not be uniform and can be used to the advantage of a SAT solver.

Let's now think about a decision problem, where the solver just has to decide if an instance is satisfiable. Suppose we pick m totally random clauses. If $c = \frac{m}{n}$, maybe there is a value of c where the probability that the instance is satisfiable is $\frac{1}{2}$. Empirically, it has been roughly shown that if $c_0 \approx 4.2$ then $\forall c > c_0,\ m > cn$ the instance is unsatisfiable with high probability and $\forall c < c_0,\ m < cn$ the instance is satisfiable with high probability. This statement has only been shown empirically, not proven. However, we do know that $\forall c > c_0,\ m > cn$ the instance is unsatisfiable with high probability if $c_0 \approx 4.5$ and $\forall c < c_0,\ m < cn$ the instance is satisfiable with high probability if $c_0 \approx 3.5$. If we want to be completely accurate, we say there exists a sequence $c_0(n)$ for which the previous claim holds [FB$^+$99]. This theorem allows for the possibility that this sequence could oscillate rather than being a constant.

A related hardness assumption is Feige's $R3SAT$ assumption: For all sufficiently large c, there does not exist a polynomial-time algorithm which refutes (proves unsatisfiable) a random $c \cdot n$-clause $3CNF$ instance with high probability. The proof of unsatisfiability needs to output "typical" or "not typical" where it outputs "typical" with high probability but can never output "typical" if the instance is unsatisfiable. Feige's $R3SAT$ assumption is nearly equivalent to the $k = 3$ version of the Sparse-LPN assumption.

6 Worst-case hardness of $CNF - SAT$

It is trivial to solve this problem in time $2^n\text{poly}(n)$ where $\text{poly}(n)$ is the number of clauses. A reasonable question is to ask if we can do better than this. In the case of $3SAT$ we can, $3SAT$ is solvable in $O(1.31^n)$ [Her11]. This result is from a succession of papers lowering the running time to solve $3SAT$. One notable algorithm from this series of papers is called WALK-SAT, for which we provide pseudo-code below [Sch99]. WALK-SAT is notable for being very simple yet successful at solving $3SAT$ with high probability in $\widetilde{O}(\frac{4}{3}^n)$.

Algorithm 1 WALK-SAT

```
For i=1 to Õ((4/3)^n)
   Pick a random assignment of variables
   For j=1 to O(n)
     Take any unsatisfactory clause and flip one of the variables
```

6.1 The Exponential Time Hypothesis

It's of course not known what the limit is to solving $3SAT$ quickly. However, some people are comfortable making the following assumption about the computational complexity of $3SAT$:

Definition 6.1. [IP99] The Exponential Time Hypothesis (ETH) states that there exists some $\delta > 0$ such that $3SAT \notin \text{Time}(2^{\delta n})$

It is important to note that there are NP-Complete problems with algorithms running in $O(2^{\sqrt{n}})$. This illustrates that ETH is a much stronger assumption than $P \neq NP$, although we think it is still reasonable.

Theorem 6.2. *[GJT76]The planar Hamiltonian path problem is NP-Complete.*

This theorem was proved with a reduction from $3SAT$. If we inspect the proof, we see that if the input size is n, then the size of the graph G outputted by the reduction is $O(n^2)$. Then, if we assume ETH, we see that planar Hamiltonian path is in $2^{\Omega(\sqrt{n})}$. It is also known that this problem can be solved in $2^{O(\sqrt{n})}$, which matches our observation nicely. Typically, basic NP-Completeness reductions have output size $O(n)$, so under ETH these problems all require $2^{\Omega(n)}$ time.

Fact 6.3. *There exists a time $1.3^k poly(n)$ algorithm for deciding if there is a vertex cover $\geq k$*

Algorithms such as the one from Fact 6.3 are called fixed parameter tractable algorithms. But, ETH implies that there does not exist a $2^{o(k)}\text{poly}(n)$ algorithm, suggesting that the algorithm from Fact 6.3 cannot be beat.

Fact 6.4. *There exists a time n^k algorithm for the k-clique problem.*

On the other hand, ETH implies that there does not exist a $n^{o(k)}$ algorithm for k-clique.

Fact 6.5. *There exists a time $2^{O(treewidth(G))}poly(n)$ algorithm for the Max Cut problem.*

The ETH implies that we cannot improve on this; there is no $2^{o(\text{treewidth}(G))}\text{poly}(n)$ algorithm.

Definition 6.6. The Strong Exponential Time Hypothesis states that there does not exist $\delta > 0$ such that $CNF - SAT$ is solvable in $2^{(1-\delta)n}\text{poly}(n)$.

Theorem 6.7. *[PW10] The Strong ETH implies that the k-Sum problem on n integers requires $n^{\Omega(k)}$.*

The k-Sum problem requires the solver to determine if some k of the n integers sum to 0.

7 Label Cover

The Label Cover problem is the baseline for many hardness of approximation results. An instance of Label Cover with domain size q is a special type of $2 - CSP$. We can think of an instance as a bipartite graph, where each edge (u, v) has a function $\pi_{uv} : [q] \to [q]$ where each edge is a constraint that is satisfied iff $\pi_{uv}(A(u)) = A(v)$ where $A(x)$ is the labeling of vertex x. Note that if the label functions are bijections then this is the Unique Games problem.

Theorem 7.1. *[Raz98] For all $\delta > 0$ there eqists $q = poly(\frac{1}{\delta})$ such that $(\delta, 1)$-approximating Label Cover with domain size q is NP-hard.*

Håstad also showed that for all ϵ, $(n^\epsilon, n^{1-\epsilon})$-approximating Maximum Independent Set is NP-Hard. One downside of Raz's result is that it maps $3SAT$ instances of size n to Label Cover instances of size $n^{O(\log(\frac{1}{\delta}))}$. Then ETH, along with Raz and Håstad's results show that $(.51, .99)$-approximating $3LIN$ requires 2^{n^ϵ} for some very small $\epsilon > 0$ which is not so great. An upside to this is that Moshkovitz and Raz [MR10] showed a reduction from a $3SAT$ instance of size n to Label Cover instances of size $n^{1+o(1)}$, even for subconstant δ. Then combining ETH with this result gets shows that $3LIN$ requires $2^{n^{1-o(1)}}$, which is nicer than the previous bound.

8 Unique Games

We mentioned earlier that if we require the functions in Label Cover instances to be bijections we have the Unique Games problem. This problem was identified by Khot [Kho02].

Conjecture 8.1 (Unique Games Conjecture)**.** *For all $\delta > 0$, there exists $q = q(\delta)$ such that $(\delta, 1-\delta)$-approximating Unique Games is NP-Hard.*

The Unique Games Conjecture implies many sharp inapproximability results. For example, Raghavendra [RS09] showed that the Unique Games Conjecture implies that Semidefinite Programming algorithms give optimal approximations for all CSPs. However, we aren't so sure how believable the Unique Games Conjecture is.

Theorem 8.2. *[ABS10] For all $\delta > 0$, there is a $(\delta, 1-\delta)$-approximation algorithm for Unique Games in time $2^{n^{O(\delta^{\frac{1}{3}})}}$.*

This doesn't mean the Unique Games is not NP-Hard, but it's a bit scary. Furthermore, we don't know how to generate hard instances of Unique Games and we are able to solve all known instances of Unique Games.

References

[ABS10] Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 563–572. IEEE, 2010.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 298–307. IEEE, 2003.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[FB+99] Ehud Friedgut, Jean Bourgain, et al. Sharp thresholds of graph properties, and the k-sat problem. *Journal of the American Mathematical Society*, 12(4):1017–1054, 1999.

[GJT76] Michael R Garey, David S. Johnson, and R Endre Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

[Her11] Timon Hertli. 3-sat faster and simpler-unique-sat bounds for ppsz hold in general. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 277–284. IEEE, 2011.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on*, pages 237–240. IEEE, 1999.

[Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775. ACM, 2002.

[Lyu05] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 378–389. Springer, 2005.

[MR10] Dana Moshkovitz and Ran Raz. Two-query pcp with subconstant error. *Journal of the ACM (JACM)*, 57(5):29, 2010.

[PW10] Mihai Pătraşcu and Ryan Williams. On the possibility of faster sat algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1065–1075. Society for Industrial and Applied Mathematics, 2010.

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

[RS09] Prasad Raghavendra and David Steurer. Integrality gaps for strong sdp relaxations of unique games. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 575–585. IEEE, 2009.

[Sch99] T Schoning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 410–414. IEEE, 1999.