CMPUT 675: Randomized Algorithms **Fall 2005**

Lecture 21: Nov 22

Lecturer: Mohammad R. Salavatipour *Scribe: Mohammad R. Salavatipour*

Remark: These notes are based on notes in an earlier course taught by me.

21.1 PCP, another characterization of NP

Many problems, such as Bin Packing and Knapsack, have PTAS's. A major open question for a long time was: "does MAX-3SAT have a PTAS?" A significant result on this line was the paper by Papadimitriou and M. Yannakakis ('92) where they defined the class Max-SNP. All problems in Max-SNP have constant approximation algorithms. They also defined the notion of completeness for this class and showed if a Max-SNP-complete problem has a PTAS then every Max-SNP problem has a PTAS.

Several well-known problems including Max-3SAT and TSP are Max-SNP-complete. The simple random truth assignment (and its derandomized version using the method of conditional probability) yields a a $\frac{7}{8}$-approximation for Max-3SAT. It turns out that even if there are not exactly 3 literals per clause this problem has a $\frac{7}{8}$-approximation algorithm. How about lower bounds? In other words, how hare is it to get a, say, $(\frac{7}{8}+0.01)$-approximation for Max-3SAT? To study these kinds of questions (hardness of approximation) we need to give another characterization of NP based on probabilistic verifiers. First let's take a look back at the standard definition of NP:

Definition 21.1 *A language $L \in$ NP if and only if there is a deterministic polynomial time verifier (i.e. algorithm) V that takes an input x and a proof y with $|y| = |x|^c$ for a constant $c > 0$ and it satisfies the following:*

- *Completeness: if $x \in L \Rightarrow \exists\, y$ such that $V(x, y) = 1$.*
- *Soundness: if $x \notin L \Rightarrow \forall y$, $V(x, y) = 0$.*

Now we define probabilistic verifiers that are provided with random bits (and make their decisions by using these random bits) and are also restricted to look at (i.e. query about) only a few bits of the alleged proof y and make their decision based on those few bits instead of reading the whole proof.

Definition 21.2 *A $(r(n), b(n))$-restricted verifier is a randomized verifier that uses at most $r(n)$ random bits. It runs in probabilistic polynomial time and reads/queries at most $b(n)$ bits of the proof.*

Now we define another proof system that uses probabilistic verifiers.

Definition 21.3 *A language $L \in \mathrm{PCP}(r(n), b(n))$ if and only if there is an $(r(n), b(n))$-restricted verifier V such that given an input x with length $|x| = n$ and a proposed proof π for x, it satisfies the following:*

- *Completeness: if $x \in L \Rightarrow$ there is a proof π such that $\Pr[V(x, \pi) = 1] = 1$.*
- *Soundness: if $x \notin L \Rightarrow$ for all "proofs" π, $\Pr[V(x, \pi) = 1] \leq \frac{1}{2}$.*

The probabilities in completeness and soundness given in definition above are 1 and $\frac{1}{2}$, respectively. A more general definition of PCP languages is by allowing these probabilities be some other constant values:

Definition 21.4 *For $0 \le s < c \le 1$, a language $L \in \mathrm{PCP}_{c,s}(r(n), b(n))$ if and only if there is a $(r(n), b(n))$-restricted verifier V such that given an input x with length $|x| = n$ and a proof π, it satisfies the following:*

- *Completeness: if $x \in L \Rightarrow \exists$ a proof π such that $\Pr[V(x,\pi) = 1] \ge C$.*
- *Soundness: if $x \notin L \Rightarrow \forall\pi$, $\Pr[V(x,\pi) = 1] \le S$.*

In the more general definition of PCP language, we need to have the following restrictions on parameters:

- c and s are constants and $0 \le S < C \le 1$. This is to make sure the verifier can give a correct answer with higher probability than a wrong answer.
- $r(n)$ and $b(n)$ are at most polynomial, this is to make sure the verifier runs in polynomial.
- Proofs are at most $2^{r(n)}$ bits long. The reason is that the verifier V uses at most $r(n)$ random bits, so it can access at most $2^{r(n)}$ positions, and it must be able access to any position of the proof.

From the definition it is obvious that $\mathrm{NP} \subseteq \mathrm{PCP}_{c,s}(0, \mathrm{poly}(n)) \subseteq \mathrm{PCP}_{c,s}(O(\log n), \mathrm{poly}(n))$.

Lemma 21.5 $\mathrm{PCP}_{c,s}(O(logn), \mathrm{poly}(n)) \subseteq \mathrm{NP}$

Proof: Let L be a language in $\mathrm{PCP}_{c,s}(O(\log n), \mathrm{poly}(n))$ with a verifier V. We construct a non-deterministic polytime Turing machine M for L. Starting with an input x, M guesses a proof π and simulates V on all $2^{O(\log n)} = \mathrm{poly}(n)$ possible random strings and M accepts if at least a fraction c of all these runs accept, rejects otherwise. Thus:

- if $x \in L \Rightarrow V(x,\pi)$ accepts with probability at least c; thus at least a fraction c of random strings cause the verifier V to accept which in turn causes M to accept.
- if $x \notin L$ then the verifier accepts with probability at most s which is smaller than c; thus for only a fraction of $< c$ of random strings verifier V accepts; M rejects.

Since there are $O(\mathrm{poly}(n))$ random strings of length $O(\log n)$ and each simulation takes polytime, the running time of M is polytime. ■

The above lemma and the observation before it implies that $\mathrm{PCP}_{c,s}(O(\log n), \mathrm{poly}(n)) = \mathrm{NP}$.

The remarkable PCP theorem, proved by Arora and Safra [1] and Arora, Lund, Motwani, Sudan, and Szegedy [2] states:

Theorem 21.6 (PCP Theorem) $\mathrm{NP} = \mathrm{PCP}_{1,\frac{1}{2}}(O(\log n, O(1))$

The original proof of PCP theorem was extremely difficult. There is a new a much simpler proof of PCP theorem using a different technique by Dinur [3] Basically, this miraculous theorem says that for every problem in NP there is a verifier that queries only a constant number of bits of the proof (regardless of the length of the proof) and with sufficiently high probability gives a correct answer whether the proof is correct or not. A direct consequence of PCP theorem is the following.

Theorem 21.7 *For some absolute constant $\epsilon > 0$, there is a gap-introducing reduction from SAT to MAX-3SAT such that it transforms a boolean formula ϕ for SAT to a boolean formula ψ with m clauses for MAX-3SAT such that:*

- *if ϕ is satisfiable, then $OPT(\psi) = m$.*
- *if ϕ is a NO-instance, then $OPT(\psi) \leq (1-\epsilon)m$.*

Corollary 21.8 *Approximating MAX-3SAT with a factor better then $(1-\epsilon)$ is NP-hard for some constant $\epsilon > 0$.*

In fact, it is not hard to prove that assuming Theorem 21.7 we can prove PCP theorem too. In other words, PCP theorem is equivalent to the statement of Theorem 21.7.

21.1.1 Improved PCP Theorems

As we saw, PCP theorem says: $\mathrm{NP} = \mathrm{PCP}_{1,\frac{1}{2}}(O(\log n), O(1))$. We are interested in reducing the number of query bits and also decrease the the probability of failure (i.e. better soundness). Our first theorem says that the number of query bits can be as small as 3 bits (and as you show in assignment 3 this is best possible).

Theorem 21.9 *For some $s < 1 : \mathrm{NP} = \mathrm{PCP}_{1,s}(O(\log n), 3)$*

Proof: Let $L \in \mathrm{NP}$ be an arbitrary language and y be an instance of L. By PCP theorem we can construct a 3CNF formula F such that:

- if $y \in L$ there is a truth assignment for F such that all clauses of F are satisfied.
- if $y \notin L$ then for any truth assignment for F at most $(1-\epsilon)$ fractions of clauses are satisfied.

We assume that the proof π given for y is the truth assignment to formula F above. The verifier V given y and proof π, first computes F in polytime. Then uses $O(\log n)$ bits to pick a random clause of F and query the truth assignment to its 3 variables (which of course requires only 3 bits of the proof). The verifier accepts if and only if the clause is satisfied. It is easy to see that:

- If $y \in L$ then there is a proof π s.t. V accepts with probability 1.
- If $y \notin L$ then for any proof π, V accepts with probability at most $1-\epsilon$.

■

The downside of this theorem is that, although the number of query bits is best possible, the soundness probability is much worse than $\frac{1}{2}$. The following theorem shows that we can actually keep the soundness probability arbitrary close to $\frac{1}{2}$ while reading only 3 bits of the proof.

Theorem 21.10 *[4] For all $\epsilon > 0$*

$$\mathrm{NP} = \mathrm{PCP}_{1,\frac{1}{2}+\epsilon}(O(\log n), 3).$$

Note that the 3 bits of the proof are selected by the verifier adaptively. That is, the position of the second bit checked may depend on the truth value of the first bit read. This theorem is tight by the following result:

Theorem 21.11 *[5]*

$$P = \mathrm{PCP}_{1,\frac{1}{2}}(O(\log n), 3).$$

21.2 Hardness of Clique, gap amplification using random walks

Definition 21.12 *(MAX-CLIQUE) Given a graph with n vertices, find a maximum size clique in it, i.e. a complete subgraph of maximum size.*

The best known algorithm for approximating clique has a factor of $O(\frac{n \cdot \log\log^2 n}{\log^3 n})$ which is quite high. Note that this is slightly better than the trivial algorithm which picks a single vertex and returns (which has approximation factor $O(n)$). It turns out that we cannot do much better than this. Håstad has proved that for any $\epsilon > 0$ there is no polytime approximation algorithm for clique with factor $n^{\frac{1}{2}-\epsilon}$ (if P$\neq$ NP) or $n^{1-\epsilon}$ (if ZPP$\neq$ NP). Our goal is to prove a polynomial hardness for clique.

To prove this hardness result, we start with a constant hardness result for Clique. Then show that it is hard to approximate Clique within *any* constant factor. Finally, we show how to improve this to a polynomial.

Consider a $\mathrm{PCP}_{1,\frac{1}{2}}(d\log n, q)$ verifier F for SAT, where d and q are constants. Such a verifier exists by PCP theorem. Let $r_1, r_2, ..., r_{n^d}$ be the set of all possible random strings to F. Given an instance ϕ of SAT, we construct a graph G from F and ϕ which will be an instance of Clique. G has one vertex $V_{r_i,\tau}$ for each pair (i, σ), where i is one of the the random strings r_i and σ is a truth assignment to q variables. The number of of all the vertices (i.e. the size of the graph) is $|V| = n^d 2^q$.

Equivalently, we define the vertices as follows. An accepting transcript for F on ϕ with random string r_i is q pairs $(p_1, a_1), \ldots, (p_q, a_q)$ s.t. for every truth assignment that has values $a_1, \ldots, a_q$ for variables $p_1, \ldots, p_q$, verifier F given r_i checks positions $p_1, \ldots, p_q$ in that order and accepts. Note that once we have ϕ, r_i, $a_1, \ldots, a_q$ it is easy to compute $p_1, \ldots, p_q$. For each transcript we have a vertex in G.

Two vertices (i, σ) and (i', σ') are adjacent iff σ, σ' don't assign different values to same variable, i.e. they are consistent.

If ϕ is a yes instance then there is a proof (truth assignment) π such that F accepts (given π) on all random strings. For each r_i, there is a corresponding σ (which has the same answers as in π) and is an accepting transcript. We have n^d random strings and therefore there are n^d vertices of G (corresponding to those). They form a clique because they come from the same truth assignment and therefore are consistent, i.e. adjacent; therefore G has a clique of size $\geq n^d$.

For the case ϕ is a no instance we want to show that every clique in G has size at most $\frac{n^d}{2}$. By way of contradiction suppose we have a clique C of size $c > \frac{n^d}{2}$. Assume that $(i_1, \sigma_1)...(i_c, \sigma_c)$ are the vertices in this clique. Therefore the transcripts $\sigma_1 ... \sigma_c$ (partial truth assignment) are all consistent. We can extend this truth assignment to a whole proof (truth assignment) such that on random strings $i_1, ..., i_c$, verifier F accepts. Therefore the verifier accepts for more than $\frac{n^d}{2}$ strings, which contradicts the assumption that ϕ is a no instance.

The gap created here is exactly the soundness probability of the verifier; so the the smaller S is, the larger gap we get.

By simulating a $\mathrm{PCP}_{1,\frac{1}{2}}(d\log n, q)$ verifier V for k times and accepting iff all of those simulations accept, we get a $\mathrm{PCP}_{1,\frac{1}{2^k}}(k \cdot d\log n, k \cdot q)$ verifier V'. Note that in this case the size of the construction G is $n^{kd}2^{kq}$, which is polynomial as long as k is some constant. This will show a hardness of 2^k, which is a constant.

Corollary 21.13 *For any constant S, it is NP-hard to approximation clique within a factor of S (say $S = 1/2^k$ in the above).*

To get an n^ε-gap, we need S to be polynomially small, and for that we need to repeat $k = \Omega(\log n)$ times. Therefore, $k \cdot q = O(\log n)$ which is Ok. But the length of random string becomes $k \log n = \Omega(\log^2 n)$, and the size of graph becomes super-polynomial. Therefore, to get a polynomial hardness result for Clique we need a PCP verifier with $O(\log n)$ random bits and polynomially small soundness probability, i.e. a $\mathrm{PCP}_{1,\frac{1}{n}}(O(\log n), O(\log n))$ verifier.

The trick here is to start with only $O(\log n)$ random bits and use random walks on expander graphs to generate $O(\log n)$ random strings, each of length about $\log n$.

Definition 21.14 *(Expander Graph) Every vertex has the same constant degree, say d, and for every non-empty set $S \subset V$, $|E(S, \overline{S})| \geq \min\{|S|, |\overline{S}|\}$.*

There are explicit construction of expander graphs. Let H be an expander graph with n^d nodes. For each node we assign a label which is a binary string of length $d \log n$. We can generate a random walk in H using only $O(\log n)$ random bits: we need $d \log n$ bits to choose the first vertex, and we only need constant number of random bits to choose one neighbor at every step. Therefore, to have a random walk of length $O(\log n)$ we need only $O(\log n)$ bits.

Theorem 21.15 *For any set S of vertices of H with $< \frac{n^d}{2}$ vertices, there is a constant k such that the probability that a random walk of length $k \log n$ lies entirely in S is $< \frac{1}{n}$.*

Proof sketch: Here is the outline of the proof of this theorem. By definition of the expander graphs, if you have a set S of vertices, we expect a constant fraction of edges out of the vertices of S be going into $\overline{S}$. Therefore, if you start a random walk from a vertex in S, at every step, there is a constant probability that this walk jumps into $\overline{S}$. So the probability that a random walk of length $\Omega(\log n)$ stays entirely within S is polynomially small. ∎

Using this theorem we can prove the following result.

Theorem 21.16 $\mathrm{PCP}_{1,\frac{1}{2}}(d \log n, q) \subseteq \mathrm{PCP}_{1,\frac{1}{n}}(O(\log n), O(\log n))$

Proof: Let $L \in \mathrm{PCP}_{1,\frac{1}{2}}(d \log n, q)$ and $\mathcal{F}$ be a verifier for L. We give a $\mathrm{PCP}_{1,\frac{1}{n}}(O(\log n), O(\log n))$ verifier $\mathcal{F}'$ for L.

$\mathcal{F}'$ builds graph the expander graph H of Theorem 21.15, then creates a random walk of length $k \log n$ using only $O(\log n)$ bits for some constant k (because choosing a neighbor, it takes only constant bits). This random walk yields $k \log n$ "random" string of length $d \log n$ each, which are the labels of the vertices of the walk. Then $\mathcal{F}'$ simulates $\mathcal{F}$ on each of these strings and accepts if and only if all these simulations accept.

Now, if $y \in L$ is a "yes" instance, then there is a proof π such that $\mathcal{F}$ accepts with probability 1 given π (i.e. on all random strings) which means $\mathcal{F}'$ accepts.

Suppose, $y \in L$ is a "no" instance. Thus $\mathcal{F}$ accepts on at most $\frac{n^d}{2}$ of the random strings. Let S be the set of vertices of H with those labels (i.e. these random strings on which $\mathcal{F}$ gives the wrong answer). Note that $|S| \leq \frac{n^d}{2}$. This means that $\mathcal{F}'$ accepts (wrongly) y only if the entire random walk is inside S. Now, based on Theorem 21.15 the probability that a random walk remains entirely in S is at most $\frac{1}{n}$ (if k is sufficiently large constant). Therefore, the probability that $\mathcal{F}'$ accepts y is at most $\frac{1}{n}$. ∎

Theorem 21.17 *For some $\delta > 0$, it is NP-hard to approximate clique within a factor of $\Omega(n^\delta)$.*

Proof: Given a SAT formula ϕ, let F be a $\mathrm{PCP}_{1,\frac{1}{n}}(d \log n, q \log n)$ verifier for it for some constants d and q. Construct the graph G from F in the same manner as we did earlier for the constant factor hardness. So the size of G is $n^d 2^{q \log n} = n^{d+q}$ and the gap created is equal to soundness probability, i.e. $\frac{1}{n}$. More specifically:

- If ϕ is a yes instance then G has a clique of size n^d.
- If ϕ is a no instance then every clique of G has size at most n^{d-1}.

This creates a gap of n^δ with $\delta = \frac{1}{d+q}$, which is a constant. ■

References

[1] S. Arora and S. Safra, *Probabilistic checking of proofs: a new characterization of NP*, J. ACM, 45(3):501-555, 1998. Earlier version in Proc. of IEEE FOCS 1992, pp 2-12.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, *Proof verification and intractability of approximation problems*, J. ACM, 45(1):70-122, 1998. Earlier version in Proc. IEEE FOCS 1992, pp 13-22.

[3] I. Dinur, *The PCP theorem by gap amplification*, Electronic Colloquium on Computational Complexity Report TR05-046, 2005.

[4] V. Guruswami, D. Lewin, M. Sudan, and L. Trevisan, *A tight characterization of NP with 3-query PCPs*, Proc. of 39th FOCS, 1998.

[5] H. Karloff and U. Zwick, *A $\frac{7}{8}$-approximation for Max-3SAT?*, in Proc of 38th FOCS, 1997.