18.405J/6.841J: Advanced Complexity Theory Spring 2016

Lecture 14: Intro to PCP

Prof. Dana Moshkovitz

Scribe: Dana Moshkovitz
Scribe Date: Spring 2015

1 Checking of Proofs and Hardness of Problems

The works of Cook, Levin and Karp is the years 1971-2 brought two important revelations to theoretical computer science:

- *Checkability:* Whether a problem can be verified efficiently is an interesting/important/crucial property. It is the very foundation of mathematics, and also a classification that theoretical computer science can contribute to its understanding. The class of efficiently verifiable problems is NP.
- *Hardness:* A large number of basic combinatorial problems are *as hard as any* of the problems in NP. Since for some NP problems, like 3SAT, even sub-exponential algorithms could not be found, this is evidence that the *NP-hard* problems are actually hard computationally.

The notion of reductions we will use in the course will be *Karp reductions*, meaning that an input to the initial problem is efficiently transformed to an input to the new problem. This is as opposed to the more general notion of *Cook reductions*, where the initial problem is solved using access to a black-box that solves the new problem. In fact, the reductions we will show will typically (but not always) be *Levin reductions*, meaning that a witness/proof for the new problem can be efficiently transformed to a witness/proof for the initial problem.

Examples: Some NP-complete problems that will be important for the course are:

1. "Does [your favorite math conjecture here] have a proof of length n?"
2. MAX-3SAT: Given clauses $C_1, \ldots, C_m$ over variables $x_1, \ldots, x_n$, where each clause is of the form $(x \vee y \vee z)$ where x, y, z are either variables or their negations, find an assignment to the variables that satisfies as many clauses as possible.
3. Constraint Satisfaction Problems (CSP): Given constraints $C_1, \ldots, C_m$ over variables $x_1, \ldots, x_n$, where each variable is over alphabet Σ, and each constraint is a predicate on q variables, find an assignment to the variables that satisfies as many constraints as possible.
4. LABEL-COVER: Given a bipartite graph $G = (A, B, E)$, alphabets Σ_A, Σ_B, and functions $\{P_e : \Sigma_A \to \Sigma_B\}_{e \in E}$, find assignments $f_A : A \to \Sigma_A$, $f_B : B \to \Sigma_B$ that *satisfy* as many of the edges as possible. We say that an edge $e = (a, b) \in E$ is *satisfied* if $P_e(f_A(a)) = f_B(b)$ ("projection test").

Note that both MAX-3SAT and LABEL-COVER can be seen as special cases of the constraint satisfaction problem.

Since 1972 there was a flurry of works showing that NP-hard problems come up almost everywhere: in planning networks, tours or circuits, in scheduling, in sequencing, and in optimizing mathematical programs. The areas where NP-hard problems arise are also diverse: they range from biology and chemistry to medicine and technology.

So all these NP-hard problems were not going anywhere – in real life people had to solve them. It became clear that researchers must find ways to cope with NP-hardness. Such ways included identifying easy special cases of NP-hard problems, designing approximation algorithms, devising faster exponential-time algorithms: time $(1.1)^n$ is much better than 2^n and may even be practical for small enough n's, or considering heuristics that do reasonably well in practice.

2 Approximation Algorithms

We will focus on approximations:

Definition 1 (Approximation algorithm)**.** *We say that an algorithm A is an α-approximation algorithm for a maximization problem Π, $0 < \alpha \leq 1$, if for* every *input x,*

$$\alpha \cdot \Pi(x) \leq A(x) \leq \Pi(x).$$

For minimization problems, A is a c-approximation, $c \geq 1$, if

$$\Pi(x) \leq A(x) \leq c \cdot \Pi(x)$$

For counting/search problems, A is a (α, c)-approximation, $0 < \alpha \leq 1 \leq c$, if

$$\alpha \cdot \Pi(x) \leq A(x) \leq c \cdot \Pi(x).$$

We say that a problem has a "polynomial-time approximation scheme" (PTAS) if it can be approximated to an arbitrary good precision, $(1 \pm \epsilon)$ for all $\epsilon > 0$. The running time depends on ϵ. If the running time depends polynomially on $1/\epsilon$, we say that it is an FPTAS: "a fully polynomial-time approximation scheme".

The approach of designing efficient approximation algorithms has been extremely successful. It resulted in the development of many new algorithmic methods, such as the Markov chain Monte-Carlo method (MCMC), linear programming (LP), semi-definite programming (SDP), spectral techniques, and more. More traditional algorithmic methods, such as greedy algorithms and divide-and-conquer, are also used.

A few prominent examples of approximation algorithms appear in Table 1.

2.1 Example: Approximation Algorithm for Max-3Sat

Next we show the $\frac{7}{8}$-approximation algorithm of MAX-3SAT. This algorithm demonstrates an important technique: the random assignment method.

INPUT: Clauses $C_1, \ldots, C_m$ over Boolean variables $x_1, \ldots, x_n$.

Problem	Approximation	Technique	Cite
Max-3Sat	$\frac{7}{8}$	random assignment	folklore
Vertex-Cover	2	combinatorial/LP	folklore
Set-Cover	$\ln n$	greedy/LP	folklore
Planar euclidean TSP	PTAS		[Mit99, Aro98]
Counting perfect matchings	FPTAS	MCMC	[JSV04]
Max-Cut	0.878...	SDP	[GW95]

Table 1: Notable approximation algorithms

1. Repeat:
2. Pick an assignment to $x_1, \ldots, x_n$ uniformly at random.
3. Until picked an assignment that satisfies at least $\frac{7}{8}$ fraction of clauses $C_1, \ldots, C_m$.

Lemma 2.1. *The expected fraction of clauses a random assignment satisfies is at least* $\frac{7}{8}$.

Proof. For $1 \leq i \leq m$, the probability that C_i is satisfied is exactly $\frac{7}{8}$. The lemma follows from the linearity of expectation. □

Note that Lemma 2.1 immediately implies that there exists an assignment that satisfies at least $\frac{7}{8}$ fraction of the clauses. This can be seen as an instance of the "Probabilistic Method" [AS92], where the existence of an object with a certain property is proved by arguing something about the probability that a random object has the property. We proceed to showing that, not only there exists an assignment satisfying $\frac{7}{8}$ fraction of the clauses, but such can also be found efficiently.

Lemma 2.2. *The fraction of assignments that satisfy at least* $\frac{7}{8}$ *fraction of the clauses is at least* $\frac{1}{m+1}$.

Proof. Denote the fraction of assignments that satisfy strictly less than $\frac{7}{8}$ fraction of the clauses by p. Since these assignments satisfy a natural number of clauses, this number is at most $(\frac{7}{8}m - \frac{1}{8})$ clauses. By Lemma 2.1,

$$p \cdot \left(\frac{7}{8} - \frac{1}{8m}\right) + (1 - p) \geq \frac{7}{8}.$$

So, $p \leq \frac{m}{m+1}$. □

Lemma 2.3. *The algorithm will stop after* $3(m+1)$ *iterations with probability at least* $\frac{2}{3}$.

Proof. By Lemma 2.2, the expected running time of the algorithm is $m+1$. By Markov's inequality, the probability the actual number is more than $3(m+1)$, is at most $\frac{1}{3}$. □

3 Hardness of Approximation

3.1 Gap Problems

First let us remove a technical obstacle out of the way: optimization problems are search problems and not decision problem, while complexity theory is typically better at handling decision problems.

To address this issue, we show that to prove hardness of approximation it is enough to prove hardness of a certain decision problem:

Lemma 3.1 (From approximation to decision)**.** *If it is NP-hard to distinguish whether* $\Pi(x) \geq c$ *or* $\Pi(x) \leq s$*, then it is NP-hard to approximate* Π *to within* c/s *for a minimization problem, or* s/c *for a maximization problem.*

The problem of distinguishing whether $\Pi(x) \geq c$ or $\Pi(x) \leq s$ is called a *gap problem*, and we denote it by $\Pi_{c,s}$. This is a kind of a *promise problem*, i.e., a problem where not all possible inputs are possible, only inputs that satisfy the promise (here: that $\Pi(x) \geq c$ or $\Pi(x) \leq s$).

3.2 Probabilistic Checking of Proofs

Perhaps surprisingly, in 1991 it was discovered that the problem of proving hardness of approximation is intimately related to a deep question about checking of proofs [FGL$^+$96]:

> "Can any mathematical proof be written in a form that can be checked probabilistically by making only a constant number of queries to the proof?"

This question is quite counter-intuitive: the proofs we are used to are sequential by definition; each proposition follows from the previous propositions. There is no way to verify correctness without checking every logical transition in the proof: it is possible that only one transition is false for the theorem to be false. It is possible for just one clause to be false for the formula not to be satisfied.

The question raises a revolutionary idea: proofs that are not sequential! That is, proofs that consist of many *local* tests, without apparent order between them, and *most* of the tests would be false if the theorem is false, not just one!

The very thought that such proofs can exist came from the extensive research of the notion of proofs that was conducted in theoretical computer science starting the 80's. This research centered around the ideas of *interactive proofs* (IP), proof systems where the randomized verifier makes polynomially-many rounds of interaction with the prover, and *multi-prover interactive proofs* (MIP), proof systems where the randomized verifier interacts with more than one prover, and the provers cannot communicate after seeing the verifier's questions.

The new notion was called *probabilistic checking of proofs* (PCP) [AS98]. It coincides with the notion of multi-prover interactive proofs if there is only one round of interaction between the verifier and the provers. This is because querying a proof can be simulated by sending each query to a different prover. Since the provers cannot communicate, the answer to each query depends only on the relevant query, and not on the other queries.

A better name for the new notion might have been *local checking of proofs*: proofs that can be checked *locally*, by reading only a constant number of their symbols. Such proofs are necessarily also *probabilistic*, since deterministically locally checkable proofs are simply constant-sized proofs.

Let us make the appropriate definitions:

Definition 2 (Probabilistic verifier)**.** *A probabilistic verifier is a probabilistic polynomial time Turing machine* V *that is given input* x*,* $|x| = n$*, and oracle access to a proof* π *over alphabet* Σ*.*

1. *We denote by $r(n)$ the maximal number of random bits that V uses on inputs of size n.*
2. *We denote by $q(n)$ the maximal number of queries that V uses on inputs of size n.*

We denote by $V^\pi(x, w)$ the output of V on input x, oracle access to π, and randomness w.

We denote by $\mathcal{V}[r, q]_\Sigma$ that set of all probabilistic verifiers with randomness r and number of queries q to proofs over alphabet Σ. We omit Σ if Σ is the binary alphabet.

Notice that a probabilistic verifier with randomness r and running time t can be simulated by a deterministic verifier than runs in time $2^r \cdot t$. The randomness allows lower running time and lower query complexity.

Note that a verifier with randomness r and q queries effectively accesses at most $2^r \cdot q$ locations in the proof. Thus, we can use $2^r \cdot q$ as a bound on the length of the proof, and $2^r \cdot q \cdot \log |\Sigma|$ as a bound on the length of the proof in bits.

Definition 3 (Probabilistic verification)**.** *We say that a language L has a probabilistic verifier with completeness c and soundness error s, if there is a probabilistic verifier V that satisfies:*

- Completeness: *For all $x \in L$, there exists a proof π, such that*

$$\Pr_{w \in \{0,1\}^r} [V^\pi(x, w) = 1] \geq c.$$

- Soundness: *For all $x \notin L$, for any proof π,*

$$\Pr_{w \in \{0,1\}^r} [V^\pi(x, w) = 1] \leq s.$$

If $V \in \mathcal{V}[r, q]_\Sigma$, we say that $L \in PCP_{c,s}[r, q]_\Sigma$ ("PCP" is "Probabilistically Checkable Proofs").

If $c = 1$ we say that the verifier has *perfect completeness*. Mostly, we will refer to verifiers with perfect completeness, or *almost-perfect completeness*, $1 - \delta$ for small $\delta > 0$, since these are more natural: if the proof is correct we usually want the verifier to always, or almost always, accept.

We typically want the soundness error to be as small as possible. Soundness error 0, or error smaller than $1/2^r$, corresponds to a deterministic verifier, and such cannot be local. However, arbitrarily small soundness error ε, as well error that goes to 0 with n ("sub-constant error"), is achievable. The soundness error for problems that are not in P is at least $\frac{1}{|\Sigma|^q}$. Thus the soundness error can get small, provided that the alphabet or the number of queries increase. Larger alphabet can be converted to a large number of queries and binary alphabet in a straightforward way, but this is not true vice-versa: given a verifier with many queries over a small alphabet, it is not immediate to get a verifier with larger alphabet, but fewer queries.

A verifier for an NP language that is not in P must perform at least two queries. In fact, any verifier can be efficiently transformed to a verifier that makes only two queries. The new verifier may need a non-binary alphabet, and, more importantly, its soundness error may be much larger than the soundness error of the original verifier.

3.3 The Connection Between Hardness of Approximation and Probabilistic Checking of Proofs

Lemma 3.2 (Probabilistic checking is equivalent to hardness of approximation)**.** MAX-3SAT$_{1,s}$ *is NP-hard for some* $s < 1$ *if and only if* 3SAT $\in PCP_{1,s'}[O(\log n), O(1)]$ *for some* $s' < 1$.

Proof. (i) Suppose MAX-3SAT$_{1,s}$ is NP-hard for some $s < 1$. Let us construct a probabilistic verifier V for 3SAT with completeness 1 and soundness error s.

Input: a formula φ.

1. Apply the reduction from 3SAT to MAX-3SAT$_{1,s}$, to transform the formula φ to clauses $C_1, \ldots, C_m$ over variables $x_1, \ldots, x_n$.
2. The proof is interpreted as an assignment to $x_1, \ldots, x_n$.
3. Verification: Pick a random $i \in m$, and check that C_i is satisfied.

Note that V uses $O(\log n)$ random bits to make three queries.

If φ is satisfiable, then there exists an assignment to $x_1, \ldots, x_n$ that satisfies all $C_1, \ldots, C_m$, and makes the verifier accept with probability 1.

If φ is not satisfiable, then any assignment to $x_1, \ldots, x_n$ satisfies at most s fraction of $C_1, \ldots, C_m$. Thus, V accepts with probability at most s.

(ii) Suppose that 3SAT $\in PCP_{1,s}[O(\log n), O(1)]$ for some $s < 1$. Let us show a reduction from 3SAT to MAX-3SAT$_{1,s'}$ for some $s' < 1$:

1. Let V be the probabilistic verifier for 3SAT. Note that V can be converted to a new verifier V' whose tests are of the form $(x \vee y \vee z)$ where x, y, z are either variables or their negations. The completeness of V' will remain 1 and the soundness error might increase to some other constant s'. The randomness of V' will remain $O(\log n)$ (although may be slightly larger than the randomness of V).
2. The reduction will produce all the $2^{O(\log n)} = n^{O(1)}$ clauses/tests that V' produces.

Note that the reduction is efficient. □

Note that the verifier we construct from a MAX-3SAT$_{1,s}$ NP-hardness uses its randomness only to decide which test to make out of $n^{O(1)}$ possible tests. It does not use the randomness for generating the tests (because the reduction to MAX-3SAT$_{1,s}$ is deterministic) or for performing the tests. Without loss of generality, the randomness is never used by the verifier for performing the tests, because whatever computational power r random bits buys one, it can be simulated in 2^r time. In our case the randomness is $r = O(\log n)$, and any randomized computation can be simulated in polynomial time.

Lemma 3.2 is phrased for MAX-3SAT$_{1,s}$ but it could be made more general. The completeness could have been replaced by any $c < 1$ (as long as the completeness of the verifier is changed

accordingly). The problem MAX-3SAT could have been replaced by any constraint satisfaction problem in which the constraints are rich enough to express any possible Boolean function, e.g., general CSP, quadratic equations over the finite field $GF(2)$, etc.

The connection between probabilistic checking and hardness of approximation drew attention to the question of whether NP has PCPs, and by 1992, the question was already answered positively:

Theorem 4 (The PCP Theorem [AS98, ALM+98])**.** $NP \subseteq PCP_{1,\frac{1}{2}}[O(\log n), O(1)]$.

Under plausible complexity assumptions, the randomness must be $\Omega(\log n)$. An interesting question is what is the constant in the Ω. This corresponds to the exponent of n in the number of possible constraints. The answer is that the constant can be 1, and the best randomness known is $\log n + O(\log \log n)$.

Corollary 3.3. *The following follow from Theorem 4:*

1. Two queries: *There are a fixed $s < 1$ and a fixed alphabet Σ, such that $NP \subseteq PCP_{1,s}[O(\log n), 2]_\Sigma$.*
2. Low error: *For any fixed $\epsilon > 0$, there is $q = q(\epsilon)$, such that $NP \subseteq PCP_{1,\epsilon}[O(\log n), q]$.*

It turns out that two queries and low error can be achieved simultaneously, but this is harder to prove. We will eventually prove:

Theorem 5 (Strong PCP Theorem [Raz98])**.** *For any fixed $\epsilon > 0$, there is an alphabet Σ (whose size depends on ϵ), such that $NP \subseteq PCP_{1,\epsilon}[O(\log n), 2]_\Sigma$.*

This theorem[1] is the basis of (almost?) all optimal hardness of approximation results.

References

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[Aro98] S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

[AS92] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley, 1992.

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[FGL+96] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

[GW95] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[1] in fact, in its stronger form, enforcing the tests to be projection tests: for any $\epsilon > 0$, for sufficiently large alphabets, LABEL-COVER$_{1,\epsilon}$ is NP-hard.

[JSV04] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, 2004.

[Mit99] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[Raz98] R. Raz. A parallel repetition theorem. In *SIAM Journal on Computing*, volume 27, pages 763–803, 1998.

MIT OpenCourseWare
https://ocw.mit.edu

18.405J / 6.841J Advanced Complexity Theory

Spring 2016

For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms.