The PCP theorem and hardness of approximation

Uriel Feige

5 + 12 March, 2003

1 Introduction

The content of this course is essentially a proof of one theorem, the PCP theorem, to be stated shortly. In fact, it is anticipated that parts of the proof will not be covered in the course. On the other hand, we will touch upon related subjects that are not needed for the proof of the PCP theorem. In the first two lectures we introduce the version of the PCP theorem that we wish to prove, and discuss it from various viewpoints.

Basic knowledge in computational complexity is assumed (such as the notions of NP-completeness), and basic undergraduate knowledge in mathematics.

1.1 3SAT

Recall the well known NP-complete problem of 3SAT. The input is a 3CNF formula over n variables. Namely, there is a collection of m clauses. Each clause contains 3 literals, where a literal is a variable or its negation. For example,

$$(x_1 \vee \bar{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \bar{x_4})$$

is a 3CNF formula with $n = 4$ and $m = 2$. A 3CNF formula is proper if the three literals in a clause are distinct. A 0/1 assignment to the variables satisfies a clause if at least one literal is assigned 1. An assignment is *satisfying* if it satisfies all clauses. A formula is *satisfiable* if it has a satisfying assignment. 3SAT is the problem of determining whether a 3CNF formula is satisfiable. As is well known, 3SAT is NP-complete. Namely, it is in NP (with a satisfying assignment being a witness for "yes instances"), and it is NP-hard, meaning that every problem in NP is reducible to 3SAT in polynomial time. Being NP-complete, there is no polynomial time algorithm for satisfiability unless P=NP.

1.2 Max-3SAT

3SAT is a decision problem. We may associate with it the following optimization problem, known as **max-3SAT**.
Input: a 3CNF formula. We use n to denote the number of variables and m to denote the number of clauses.
Output: an assignment to the variables.
Value: the number of clauses satisfied by the assignment.
Objective: find a solution of maximum value.

Clearly, max-3SAT is NP-hard, as otherwise we could distinguish between for mulas that are satisfiable and formulas that are not satisfiable. Technically, max-3SAT is not necessarily NP-complete, because it is not known to be in NP. Let m^* be the maximum number of clauses that are satisfied by the optimum solution. An assignment that satisfies m^* clauses is not necessarily a witness for max-3SAT, as it only certifies that m^* clauses are satisfiable, but does not certify that $m^* + 1$ clauses are not satisfiable. This last statement is a co-NP statement, and in fact co-NP-complete, and is not in NP unless $NP = coNP$ (which is unknown at the moment). Note however that if $P = NP$ (which implies the collapse of the whole polynomial hierarchy to P), then max-3SAT is in P.

1.3 Approximation algorithms

A common way for handling NP-hard optimization problems is by approximation algorithms. Namely, we design an a polynomial time algorithm (or sometimes, random polynomial time) that finds a suboptimal solution, but still of value not too far from the value of the optimal solution. In the context of max-3SAT, a natural measure for the value of an assignment is the number of clauses that it satisfies. The approximation ratio of an algorithm for max-3SAT on a particular instance is the ratio of the number of clauses that it satisfies to the number of clauses satisfied by the optimal solution. An algorithm is said to have approximation ρ if on every instance it has approximation ratio of at least ρ. Analogous definitions can be given for other optimization problems. Namely, the approximation ratio of an algorithm for a maximization problem is

$$\rho = \min_{\text{input } x} \frac{\text{alg}(x)}{\text{opt}(x)} \leq 1$$

and for a minimization problem it is

$$\rho = \max_{\text{input } x} \frac{\text{alg}(x)}{\text{opt}(x)} \geq 1.$$

Note that there are infinitely many input instances, so rather than min and max we should have taken inf and sup. More importantly, the approximation ratio sometimes deteriorates as a function of the size of x (the length of its representation in binary), and then the approximation ratio is expressed as a function of this size.

1.4 Estimation algorithms

In some cases, we are satisfied in estimating the value of the optimum solution, without necessarily exhibiting a solution of this value. An estimation algorithm outputs a range of values $[m_l, m_h]$ such that $m_l \leq m^* \leq m_h$. The estimation ratio of the algorithm is $\rho = m_l/m_h$ over the worst case instance. Every approximation algorithm with approximation ratio ρ is also an estimation algorithm with estimation ratio ρ. The converse need not hold. In fact, there are some optimization problems for which the known estimation ratios are better than the known approximation ratios. (One such example is the *domatic number* problem on d-regular graphs, for which the best estimation ratio is roughly $\ln d$, whereas the known best approximation ratio is inferior by a constant multiplicative factor. See details in [5].)

1.5 A simple example

Let us consider max-3SAT on proper 3CNF formulas. At best $m^* = m$.

Proposition 1 *For every proper max-3SAT instance, $m^* \geq 7m/8$.*

Proof: The probabilistic method provides a simple proof. Assign every variable the value 1 independently with probability 1/2. Every clause is satisfied with probability $1 - (1/2)^3 = 7/8$. (Here we used the assumption that the 3CNF formula is proper.) By linearity of expectation, the expected number of satisfied clause is $7m/8$. There must be at least one assignment achieving the expectation. $\square$

Hence for proper max-3SAT, the trivial estimation algorithm $7m/8 \leq m^* \leq m$ is correct and has estimation ratio 7/8.

What about approximation algorithms? We can use the method of conditional expectations to derandomize the probabilistic argument and give a greedy approximation algorithm with approximation ratio 7/8. (In fact, this algorithm predated the above probabilistic argument, and is due to Johnson [9].) Namely, we assign values to the variables one by one. When a literal is assigned 1, all clauses that contain it are satisfied and removed, and its complement is removed from the clauses that contain it. Note that the remaining formula need not be proper any more. We call a clause with j literals a j clause. For a variable x_i, let $n_j(i)$ for $j = 1, 2, 3$ be the number of j-clauses containing x_i, and similarly $\bar{n}_j(i)$ for $\bar{x}_i$. At the time that it is x_i's turn to be assigned a 0/1 value, define the weight of x_i to be

$$w(i) = n_3 + 2n_2 + 4n_1 - \bar{n}_3 - 2\bar{n}_2 - 4\bar{n}_1$$

and assign $x_i = 1$ if $w_i \geq 0$, and $x_i = 0$ if $w(i) < 0$.

Proposition 2 *The above greedy algorithm satisfies at least $7m/8$ clauses on proper instances of max-3SAT.*

Proof: Let X be the number of clauses satisfied. Let $E_i[X]$ denote the expected number of clauses satisfied when the first i variables are assigned by the greedy algorithm, and the rest are assigned at random. Hence we are interested in $E_n[X]$. From proposition 1 we know that $E_0[X] = 7m/8$. To complete the proof we show that for every $i \geq 1$, $E_i[X] \geq E_{i-1}[X]$.

Let $E_i^+[X]$ denote the expected number of satisfied clauses when the first $i-1$ variables are assigned by the greedy algorithm, $x_i = 1$, and the rest of the variables are assigned at random. Let $E_i^-[X]$ denote the expected number of satisfied clauses when the first $i-1$ variables are assigned by the greedy algorithm, $x_i = 0$, and the rest of the variables are assigned at random. Simple calculation show that $E_i^+[X] = E_{i-1}[X] + w_i/8$, and $E_i^+[X] = E_{i-1}[X] - w_i/8$. Our rule for choosing the assignment for x_i then ensures that

$$E_i[X] = \max[E_i^+[X], E_i^-[X]] \geq E_{i-1}[X].$$

□

For nonproper max-3SAT instances, achieving an approximation ratio of 7/8 is much more difficult. A computer assisted proof shows that a certain algorithm based on semidefinite programming has approximation ratio 7/8 [10].

1.6 Hardness of approximation/estimation

We have seen a trivial algorithm that estimates proper max-3SAT within a ratio of 7/8. But can we do better? How much better? What are the limitations for approximation/estimation algorithms?

In some cases, we can argue that there are no algorithms with better approximation ratios than some threshold. This needs to be under some assumption, because if $P = NP$ then max-3SAT can be approximated within a ratio of 1. The minimal assumption that we need to make is thus $P \neq NP$. This leads to NP-hardness of approximation results.

Let us see some examples. The chromatic number is NP-hard to estimate within ratios better than 4/3 because it is NP-hard to distinguish between 3-colorable and 4-colorable graphs. (We use an estimation ratio larger than 1 here because coloring is a minimization problem.) This is to some extent not a very informative example, because in may only reflect a fact that the chromatic number cannot be approximated with an additive factor of 1. However, as shown in [6], it is in fact NP-hard to distinguish between $3k$-colorable and $4k$-colorable graphs for every k, justifying a 4/3 hardness of approximation ratio.

A more striking example regarding hardness of approximation can be illustrated for the **k-center problem**.
Input: a parameter k, and n points in a metric space. Namely, between every two points there is a positive distance, and the distances satisfy the triangle inequality.
Output: a set of k of the points, the we shall call *centers*.
Value: the largest distance of a point from its closest center.
Objective: select k points so as to minimize the value.

Consider the following greedy algorithm. Select an arbitrary point as the first center. Then at every step, select the new center as the point furthest away from all existing centers (breaking ties arbitrarily).

Proposition 3 *The greedy algorithms has approximation ratio 2.*

Proof: Let r^* be the optimal value for the k center problem, namely, the maximum distance of a point to its closest center. Assume for contradiction that the value for greegy is $r > 2r^*$. Then in particular, all k centers chosen by the greedy algorithm and one more point are of mutual distance at least r from each other. To cover each of these $k+1$ points within a distance of r^*, the greedy algorithm must choose $k+1$ centers, as no center is at distance r^* from two such points (by the triangle inequality). As the optimal solution uses only k centers, this is a contradiction. $\square$

Can we achieve an approximation ratio better than 2?

Proposition 4 *It is NP-hard to estimate the optimum r^* for the k-center problem within a ratio strictly better than 2.*

Proof: By reduction from the question of whether a graph has a dominating set with k vertices, namely, a set of k vertices such that every other vertex is adjacent to at least one vertex in the set. The reduction simply views the graph as an instance of the k center problem, with vertices as points, and the metric induced by shortest path distance. On yes instances we have a k center problem with $r^* = 1$, and on no instances $r^* \geq 2$. Estimating the k-center problem within a ratio better than 2 would solve the decision problem for dominating set. $\square$

In general, we have the following pattern for proving NP-hardness of approximation results for problem max-Π. We reduce from an NP-hard problem Π'. Yes instances of Π' reduce to instances of max-Π with value at least v_h. No instances of Π' reduce to instances of max-Π with value at most v_l, where $v_l < v_h$. Using these so called *gap reductions*, we establish hardness of approximating max-Π within ratios better than $\rho = v_l/v_h$. (A similar approach works for proving NP-hardness of minimization problems.)

Note two aspects of these NP-hardness of approximation results.

1. They are also NP-hardness of estimation results.
2. They are algorithmic in the sense that they give a polynomial time algorithm that reduces any NP problem to gap instances of max-Π. Hence even though they convey a negative message (max-Π is hard to approximate), they contain positive content (a new polynomial time algorithm).

1.7 An approximation view of the PCP theorem

Recall that we have seen that proper max-3SAT can be trivially estimated within a ratio of 7/8, and asked whether we can do better. The PCP theorem has more than one form, and here is the form that we would like to prove in the course.

Theorem 5 *For every $\rho > 7/8$, it is NP-hard to distinguish between proper 3CNF formulas with $m^* = m$ and $m^* \leq \rho m$. In particular, it is NP-hard to approximate max-3SAT within a ratio strictly better than 7/8.*

Observe that both Proposition 4 and Theorem 5 deal with fairly natural optimization problems whose decision version (is there a solution with value at least as good as ...) is NP-complete, and establish a tight NP-hardness of approximation (estimation) for their respective problem. So why is it that the proof of Proposition 4 takes a few minutes, whereas the proof of Theorem 5 needs (and deserves) a full course? Where is the difference? Could it be that Theorem 5 also has a simple proof that we have overlooked?

Research Question. Find a simpler proof of the PCP theorem.

But let us try to explain why the PCP theorem appears to be deeper than the NP-hardness of approximation results that predated it (such as Proposition 4).

- In the development of the (current) proof of the PCP theorem, many new concepts and techniques were introduced, and these can be reused in other contexts.

- The set of techniques that were introduced in the proof of the PCP theorem are applicable for proving hardness of approximation results for a variety of other NP-hard combinatorial optimization problems. Often, the hardness of approximating Max-3SAT can be used in a "black-box" manner in order to prove hardness of approximation results for additional problems. The PCP theorem is a major revolution in the area of approximation algorithms, that led to answering many of the major open questions that predated it. For many of the most popular NP-hard combinatorial optimization problems we now pretty well understand the limits of approximation algorithms (in terms of best possible approximation ratios). Prior to the proof of the PCP theorem (which took several years), this was far from true, and for most problems people had no idea (or incorrect conjectures) regarding the best possible approximation ratios for them.

- The PCP theorem has additional formulations, not explicitly dealing with approximation algorithms, that are mathematically interesting. This does not appear to be true for Proposition 4. We shall elaborate on this shortly.

1.8 An algorithmic view of the PCP theorem

As we said in Section 1.6, NP-hardness of approximation results are in fact algorithmic results. Likewise, the PCP theorem is also an algorithm. Specifically, we have the following version of the PCP theorem.

The PCP theorem is a polynomial time algorithm that receives as input a 3CNF formula ϕ and a parameter $7/8 \leq \rho < 1$. It outputs a 3CNF formula ψ with the following properties.

- If ϕ is satisfiable, then ψ is satisfiable.

- If ϕ is not satisfiable, then any assignment satisfies at most a ρ fraction of the clauses of ψ.

Moreover, the properties above are also algorithmic in the following sense. There are two polynomial time algorithms C (for completeness) and S (for soundness) that satisfy:

- If w is a satisfying assignment for ϕ, then $C(\phi, w)$ is a satisfying assignment for ψ.

- If w is an assignment that satisfies more than a ρ fraction of the clauses in ψ, then $S(\phi, \psi, w)$ is a satisiying assignment for ϕ.

1.9 Probabilistically Checkable Proofs

PCP stands for *Probabilistically Checkable Proofs*, a term coined in [2], though appearing (sometimes implicitly, sometimes under different names) in earlier works.

Let us recall a classic view of NP, as deterministically checkable proofs.

A language L is in NP if there is a polynomial time verification procedure V with the following properties:

- **Completeness**. For every $x \in L$ there is a witness w such that $V(x, w)$ accepts.
- **Soundness**. For every $x \notin L$ and every w, $V(x, w)$ rejects.

The running time of V is assumed to be polynomial in $|x|$ (the length of the binary representation of x), implying that $|w|$ as well needs to be bounded by a polynomial in $|x|$.

Now let verifier V use randomness, in particular allowing V to use sampling of small parts of w, rather than read all of w. We still require $|w|$ to be polynomial in $|x|$. We use the notation V^w to indicate that V does not receive w as explicit input, but rather is allowed to query the value of individual bits in w.

Definition 1 *Language L is said to have a probabilistically checkable proof with parameters (r, q) if there is a randomized polynomial time algorithm V with the following properties:*

- **Completeness**. *For every $x \in L$ there is a witness w such that $V^w(x)$ accepts (with probability 1). (This is known as* perfect completeness.*)*
- **Soundness**. *For every $x \notin L$ and every w, $Pr[V^w(x) \text{ accepts}] \leq 1/2$. (We note that the choice of $1/2$ is somewhat arbitrary, and it can be replaced by any other constant.)*

$V^w(x)$ is nonadaptive *and works in three phases. First $V(x)$ uses up to r random bits in order to list at most q bit locations. Then it queries w at these q locations and gets back the bit values. Finally, based on the values received, it decides whether to accept or reject.*

Let $PCP(r, q)$ denote the class of languages that have probabilistically checkable proofs with parameters $c \cdot r$ and $c \cdot q$. We use the convention that for input x, $n = |x|$ denotes the length of the binary representation of x. We allow the parameters r and q to depend on n, but note that they must be bounded by the running time of V which is a polynomial in n.

Let us first note the following proposition.

Proposition 6 *In $PCP(r, q)$ with nonadaptive verifier, we may assume that $|w| \leq 2^{cr} \cdot cq$.*

Proof: V queries cq bit locations for some constant c. As V uses cr random bits, there are at most 2^{cr} possible sequences of queries. All other bit locations of w are irrelevant. $\square$

We note that if V is adaptive (in the sense that the location of a query may depend on the values of all previous queries, then $|w| \leq 2^{c(r+q)} \cdot cq$.

Corollary 7 *For every q, $PCP(\log n, q) \in NP$.*

Proof: By the bound of $r = O(\log n)$, $|w|$ is at most a polynomial in n. Hence w can be guessed. $\square$

A different formulation of the PCP theorem (in fact, the original one) is:

Theorem 8 *$NP \subset PCP(\log n, 1)$.*

This shows that for appropriately encoded NP-witnesses, sampling can drastically reduce the number of bit locations that one needs to read from the witness in order to verify membership in L. A constant number of bits suffice. To achieve this saving, soundness holds in a probabilistic sense, rather than unconditionally. Still, by repeated sampling the probability of error can be made arbitrarily small (at an exponential rate). Another hidden overhead is that the PCP witness w may be much longer than the original witness w, and the verification procedure V may be more complicated. We note that the proof of the PCP theorem is constructive in that it gives a polynomial time procedure for transforming a classical witness w to a PCP witness.

As both Theorems 5 and 8 are true, they are equivalent. But more interestingly, they are equivalent in the sense that one can easily be proved from the other, if we replace "for every $\rho > 7/8$" in the statement of Theorem 5 by "for some $\rho < 1$".

Proposition 9 *Theorem 5 (with "for some $\rho < 1$") implies Theorem 8.*

Proof: Consider an arbitrary $I \in L$ for some NP language L. By Theorem 5, there is a polynomial time reduction that reduces I to a 3CNF formula ϕ, and the reduction has the following properties:

1. If $I \in L$, then ϕ is satisfiable.

2. If $I \notin L$, then every assignment satisfies at most a ρ-fraction of the clauses.

The PCP witness for I is a satisfying assignment w for ϕ. Let m be the number of clauses in ϕ, and let k be the minimum integer such that $\rho^k \leq 1/2$. The verification procedure uses $k \log m$ random bits to select independently at random k clauses in ϕ, and then queries w for the assignments to the $3k$ variables in these clauses. If all k clauses are satisfied, V accepts.

As the reduction from I to ϕ is polynomial, and because $\rho < 1$ is some fixed constant, it follows that the number of random bits used is $O(\log |I|)$, and the number of bits queried is $O(1)$.

The PCP has the perfect completeness property, because V always accepts when w is a satisfying assignment for ϕ.

The soundness property follows from the fact that when $I \notin L$, then ϕ is at most ρ-satisfiable. Then for every candidate PCP witness w, at most a ρ-fraction of the clauses of ϕ are satisfied by it. The probability that k random clauses are among the satisfied clauses is thus at most $\rho^k \leq 1/2$. If at least one of the k clauses is not satisfied by w, V rejects. $\Box$

Proposition 10 *Theorem 8 implies Theorem 5 (for some $\rho < 1$).*

Proof: Let $L \in PCP(\log n, 1)$. We show a polynomial time reduction from L to max-3SAT.

Given input x for which we need to check if $x \in L$, consider verifier V from the $PCP(\log n, 1)$ system for L. Run V in polynomial time on each of the 2^r possible random strings, recording the q bit locations that it queries. Think of the bits of w as variables. Then we have a collection of 2^r tests, where each test depends on q Boolean variables. For each test, figure out its truth table by exhaustively running V on all possible sequences of q answer bits. This truth-table can be represented in q-CNF form (with $O(2^q)$ clauses), and thereafter, with the addition of auxiliary variables, can be transformed to 3-CNF form. It is important to note that the 3-CNF formula has the property that for every assignment to the original variables that satisfies the original test, there is an assignment to the auxiliary variables that satisfies the 3-CNF formula, and for any assignment that satisfies the 3-CNF formula, its projection on the original variables satisfies the original test. Considering all tests, we have a 3-CNF formula with at most $2^r 2^q q$ clauses (which also bounds the number of variables) that is satisfiable if $x \in L$, and at most ρ-satisfiable if $x \notin L$, for

$$\rho = 1 - \frac{1}{2q2^q}.$$

For every constant q, ρ is also a constant. $\Box$

To actually prove Theorem 5 ("for every $\rho > 7/8$") from its weak version ("for some $\rho < 1$") requires much more work, that will be discussed later in the course.

1.10 The PCP theorem as a game

It is sometimes beneficial to view the PCP theorem as a game. A player P claims to hold a satisfying assignment w to a formula ϕ. A verifier V may ask only yes/no questions about this w, and needs to either find w, or conclude that P was cheating in his answers. When ϕ is not satisfiable, then necessarily P is cheating, because no w exists. If V is computationally unlimited, he can reject without asking even a single question. However, V is computationally limited, and hence the need to ask question even when ϕ is not satisfiable.

A cleaner view of this game is to view ϕ as satisfiable, but P as computationally limited in the sense that P is unable to actually find a satisfying assignment for ϕ. Still, P claims to hold such an assignment w, and the role of V is to expose this bluff with as few yes/no questions as possible.

This view of the PCP theorem resembles playing the game of 20 questions (e.g., finding out a number in the range $[1, 1000000]$). However, one form of cheating that is common in the game of 20 questions is not available in the PCP game. Namely, P is nonadaptive: he needs to write answers to all possible questions in advance, and cannot base the answer to a question on the history of previous questions.

The design of efficient strategies for this game is the essence of constructing efficient PCPs. Informally, the number of queries q corresponds to the number of yes/no questions, and the number of random bits r limits the size of the pool of questions from which V can choose (and hence the number of answers that P has to prepare in advance).

The interpretation of the PCP theorem as a game becomes informative at the last stages of the proof, when one wants to get the smallest possible value for q. We shall return to this interpretation later in the course.

1.11 Historical prespective

The development of the PCP theorem as stated in Theorem 5 took several years, and many researches made important contributions towards the proof. It is easy to list 20 papers that made important technical or conceptual contributions with major impact on the development of the proof. For brevity, I list here only 5 such papers.

LFKN [11]. This paper by Lund, Fortnow, Karloff and Nisan showed the dramatic impact of combining randomness with proof checking. In particular, it showed that if one allows probabilistic verification, than not only

NP-statements can be verified, but also co-NP statements. This was a big surprise at the time, and even more so as there was "evidence" that this was impossible (via oracle results). The paper also introduced algebraic techniques to the area of proof checking (though such techniques were used earlier in other contexts). We shall see the LFKN protocol in the next lecture.

BFL [3]. This paper by Babai, Fortnow and Lund gave the first PCP theorem (and perhaps the most important one, if one cares only about PCPs and ignores approximation issues), showing that $NEXP = PCP(poly, poly)$. That is, randomness provably gives an exponential speedup in verification of proofs, as it is known that $NEXP \neq NP$. From a technical point of view, it introduced the multilinearity test. We shall not explicitly prove BFL's result in the course. One may think that there is really no need to separately prove BFL's result, once we establish the PCP theorem. Intuitively, scaling up $NP = PCP(\log n, 1)$ by an exponent would give $NEXP \subset PCP(poly, 1)$. However, the running time of the verifier might also scale up by an exponent, from polynomial to exponential. There are ways of avoiding this exponential blowup in running time, but we shall probably not discuss them in the course.

FGLSS [4]. Feige, Goldwasser, Lovasz, Safra and Szegedy made the connection between PCPs and hardness of approximation, and showed that $NEXP \subset PCP(poly, poly)$ implies that max-clique is hard to approximate unless EXPTIME=NEXPTIME. To get hardness of approximation results under assumptions closer to $P \neq NP$, they scaled down the BFL result, explicitly optimizing (or trying to optimize) the parameters r and q, and showed that $NP \in PCP(\log n \log\log n, \log n \log\log n)$. This established that max-clique is hard to approximate within constant factors unless $NP \subset TIME(n^{O(\log\log n)})$.

Let us digress from our historical overview and sketch how PCPs are related to max-clique. Recall that max-clique is the problem of finding a clique of maximum cardinality in a graph, where a clique is a set of vertices every two of which are connected by an edge. Given a PCP with r random bits and q query bits for 3SAT (or any other NP-complete language), and an input 3CNF formula ϕ, construct the following graph. There are 2^r clusters of vertices, one for every sequence of coin tosses of V. Within each cluster, there are 2^q vertices, one for each sequence of answers to the q queries of V. We remove vertices that correspond to sequences of answers on which V rejects. Connect two vertices by an edge if they are *consistent*. Namely, there is some string w (supposedly a PCP witness) on which the two sequences of answers are indeed the replies to the respective sequences of queries. In particular, vertices within a cluster form an independent set. The maximum size of a clique in this graph G is 2^r if ϕ is satisfiable, and at most $2^r/2$ if ϕ is not satisfiable. Hence if max-clique can be approximated within a ratio better than 2, then $NP \subset TIME(2^{O(r+q)})$. More details about this reduction should be supplied by students in the homework assignment about hardness of approximating min vertex cover.

ALMSS. Arora, Lund, Motwani, Sudan and Szegedy proved the PCP theorem (Theorem 8). They showed the importance of q being constant, by proving the NP-hardness of approximating max-3SAT. By known reductions from max-3SAT, it then followed that a multitude of other problems are NP-hard to approximate (including hardness of approximating max-clique within a ratio of n^ϵ).

Hastad [7, 8]. Here in fact I am referring to two papers, one on max-clique [7] and one on many other problems, including max-3SAT [8]. These works developed techniques such as the use of list decoding and of Fourier analysis that led to essentially tight hardness of approximation results for several problems, including Theorem 5.

There are many other important papers that contributed essential ingredients to the PCP theorem, and we shall mention some of them as we get to use their particular contributions.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy. "Proof verification and hardness of approximation problems". *JACM 45(3): 501–555 (1998).*

[2] S. Arora, S. Safra. "Probabilistic checking of proofs: a new characterization of NP". *JACM 45(1): 70–122 (1998).*

[3] L. Babai, L. Fortnow, C. Lund, "Non-deterministic exponential time has two-prover interactive protocols", *Computational Complexity, 1:3–40, 1991.*

[4] U. Feige, S. Goldwasser, L. Lovasz, S. Safra and M. Szegedy. "Interactive Proofs and the Hardness of Approximating Cliques". *Journal of the ACM, Vol. 43, No. 2, March 1996, pp. 268–292.*

[5] U. Feige, M. Halldorsson, G. Kortsarz and A. Srinivasan. "Approximating the domatic number". To appear in *SIAM Journal on Computing.* Preliminary version in *Proc. of 32nd STOC, 2000*, 134–143.

[6] M. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[7] J. Hastad. "Clique is hard to approximate within $n^{1-\epsilon}$". *Acta Mathematica* Vol. 182, 1999, 105–142.

[8] J. Hastad. “Some optimal inapproximability results”. *JACM* 48(4): 798–859 (2001).

[9] D. Johnson. “Approximation algorithms for combinatorial problems.” *Journal of Computer and System Sciences*, 9:256–278, 1974.

[10] H. Karloff and U. Zwick. “A 7/8-approximation algorithm for max-3SAT?” *Proceedings of the 38th Annual Aymposium on Foundations of Computer Science*, 406–415, 1997.

[11] C. Lund, L. Fortnow, H. Karloff, N. Nisan. “Algebraic Methods for Interactive Proof Systems.” *J. ACM*, **39** (1992), 859–868.