Limits of Approximation Algorithms: PCPs and Unique Games (DIMACS Tutorial Lecture notes)[1]

Organisers: Prahladh Harsha & Moses Charikar

[1]Jointly sponsored by the DIMACS Special Focus on Hardness of Approximation, the DIMACS Special Focus on Algorithmic Foundations of the Internet, and the Center for Computational Intractability with support from the National Security Agency and the National Science Foundation.

Preface

These are the lecture notes for the DIMACS Tutorial *Limits of Approximation Algorithms: PCPs and Unique Games* held at the DIMACS Center, CoRE Building, Rutgers University on 20-21 July, 2009. This tutorial was jointly sponsored by the DIMACS Special Focus on Hardness of Approximation, the DIMACS Special Focus on Algorithmic Foundations of the Internet, and the Center for Computational Intractability with support from the National Security Agency and the National Science Foundation.

The speakers at the tutorial were Matthew Andrews, Sanjeev Arora, Moses Charikar, Prahladh Harsha, Subhash Khot, Dana Moshkovitz and Lisa Zhang. We thank the scribes – Ashkan Aazami, Dev Desai, Igor Gorodezky, Geetha Jagannathan, Alexander S. Kulikov, Darakhshan J. Mir, Alantha Newman, Aleksandar Nikolov, David Pritchard and Gwen Spencer for their thorough and meticulous work.

Special thanks to Rebecca Wright and Tami Carpenter at DIMACS but for whose organizational support and help, this workshop would have been impossible. We thank Alantha Newman, a phone conversation with whom sparked the idea of this workshop. We thank the Imdadullah Khan and Aleksandar Nikolov for video recording the lectures. The video recordings of the lecture will be posted at the DIMACS tutorial webpage

`http://dimacs.rutgers.edu/Workshops/Limits/`

Any comments on these notes are always appreciated.

Prahladh Harsha
Moses Charikar
30 Nov, 2009.

Tutorial Announcement

DIMACS Tutorial
Limits of Approximation Algorithms: PCPs and Unique Games
DIMACS Center, CoRE Building, Rutgers University, July 20 - 21, 2009

Organizers:
* Prahladh Harsha, University of Texas, Austin
* Moses Charikar, Princeton University

This tutorial is jointly sponsored by the DIMACS Special Focus on Hardness of Approximation, the DIMACS Special Focus on Algorithmic Foundations of the Internet, and the Center for Computational Intractability with support from the National Security Agency and the National Science Foundation.

The theory of NP-completeness is one of the cornerstones of complexity theory in theoretical computer science. Approximation algorithms offer an important strategy for attacking computationally intractable problems, and approximation algorithms with performance guarantees have been designed for a host of important problems such as balanced cut, network design, Euclidean TSP, facility location, and machine scheduling. Many simple and broadly-applicable approximation techniques have emerged for some provably hard problems, while in other cases, inapproximability results demonstrate that achieving a suitably good approximate solution is no easier than finding an optimal one. The celebrated PCP theorem established that several fundamental optimization problems are not only hard to solve exactly but also hard to approximate. This work shows that a broad class of problems is very unlikely to have constant factor approximations, and in effect, establishes a threshold for such problems such that approximation beyond this threshold would imply P= NP. More recently, the unique games conjecture of Khot has emerged as a powerful hypothesis that has served as the basis for a variety of optimal inapproximability results.

This tutorial targets graduate students and others who are new to the field. It will aim to give participants a general overview of approximability, introduce them to important results in inapproximability, such as the PCP theorem and the unique games conjecture, and illustrate connections with mathematical programming techniques.

List of speakers: Matthew Andrews (Alcatel-Lucent Bell Laboratories), Sanjeev Arora (Princeton University), Moses Charikar (Princeton University), Prahladh Harsha (University of Texas, Austin), Subhash Khot (New York University), Dana Moshkovitz (Princeton University) and Lisa Zhang (Alcatel-Lucent Bell Laboratories)

Contents

Lecture 1

An Introduction to Approximation Algorithms

Sanjeev Arora

Scribe: Darakhshan J. Mir

20 July, 2009

In this lecture, we will introduce the notion of approximation algorithms and see examples of approximation algorithms for a variety of NP-hard optimization problems.

1.1 Introduction

Let Q be an optimization problem[1]. An optimal solution for an instance of this optimization problem is a feasible solution that achieves the best value for the objective function. Let $OPT(I)$ denote the value of the objective function for an optimal solution to an instance I.

Definition 1.1.1 (Approximation ratio)**.** *An algorithm for Q has an approximation ratio α if for instances I, the algorithm produces a solution of cost $\leq \alpha \cdot OPT(I)$ ($\alpha \geq 1$), if Q is a minimization problem and of cost $\geq \alpha \cdot OPT(I)$ if Q is a maximization problem.*

We are interested in polynomial-time approximation algorithms for NP-hard problems. How does a polynomial-time approximation algorithm know what the cost of the optimal solution is, which is NP-hard to compute? How does one guarantee that the output of the

[1]Formally, a (maximization) optimization problem is specified by two domains $\mathcal{X}, \mathcal{Y}$, a feasibility function $\text{feas} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ and an evaluation function $\text{value} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. An input instance to the problem is an element $x \in \mathcal{X}$. For each such x, the optimization problem is as follows:

$$OPT(x) = \max \{\text{value}(x, y) | y \in \mathcal{Y}, \text{feas}(x, y) = 1\}.$$

$OPT(x)$ is also called the optimal value.

algorithm is within α of the optimal solution when it is NP-hard to compute the optimal solution. In various examples below, we see techniques of handling this dilemma.

1.1.1 Examples

1. **2-approximation for metric Travelling Salesman Problem (metric-TSP):** Consider a complete graph G formed by n points in a metric space. Let d_{ij} be the distance between point i and j. The metric TSP problem is to find a minimum cost cycle that visits every point exactly once.

   The following observation relating the cost of the minimum spanning tree (MST) to the optimal TSP will be crucial in bounding the approximation ratio.

   Observation 1.1.2. *The cost of the Minimum spanning Tree (MST) is at most the optimal cost of TSP.*

   Algorithm A**:**

   (a) Find the MST

   (b) Double each edge

   (c) Do an "Eulerian transversal" and output its cost

   Observe that $TSP \leq cost(A) \leq 2 \cdot MST \leq 2 \cdot TSP$.

2. **A 1.5-approximation to metric-TSP:** The approximation ratio can be improved to 1.5 by modifying the above using an idea due to Christofides [Chr76]. Instead of doubling each edge of the MST as in the above algorithm, a minimum cost matching is added among all odd degree nodes. Observe that cost of matching $\leq \frac{1}{2}TSP$. So,

   $$Cost(Appx\text{-}algo) \leq MST + \frac{1}{2}TSP \leq 1.5 \cdot TSP$$

   It is to be noted that since 1976, there has been no further improvement on this approximation ratio.

The above examples are examples of approximation algorithms that attain a constant approximation ratio. In the next section, we will see how to get arbitrarily close to the optimal solution when designing an approximation algorithm, ie., approximation ratios arbitrarily close to 1.

1.2 Polynomial-time Approximation Scheme (PTAS)

A PTAS is a family of polynomial-time algorithms, such that for every $\varepsilon > 0$, there is an algorithm in this family that is an $(1 + \varepsilon)$ approximation to the NP-hard problem Q, if it is a minimization problem and an $(1 - \varepsilon)$-approximation if Q is a maximization problem.

The above definition allows the running time to arbitrarily depend on ε but for each ε it should be polynomial in the input size e.g. $n^{\frac{1}{\varepsilon}}$ or $n^{2^{\frac{1}{\varepsilon}}}$.

1.2.1 Type-1 PTAS

Various type of number problems typically have type-1 PTAS. The usual strategy is to try to round down the numbers involved , so the choice of numbers is small and then use Dynamic Programming. The classic example of such an approach is the Knapsack problem.

Knapsack problem Given a set of n items, of sizes $s_1, s_2 \dots s_n$ such that $s_i \leq 1 \ \forall i$, and profits $c_1, c_2, \dots, c_n$, associated with these items, and a knapsack of capacity 1, find a subset I of items whose total size is bounded by 1 such that the total profit is maximized.

The knapsack problem is NP-hard in general, however if the profits fall in a small-sized set, then there exists an efficient polynomial time algorithm.

Observation 1.2.1. *If the values $c_1, c_2, \dots c_n$ are in $[1, \dots, w]$, then the problem can be solved in $poly(n, w)$-time using dynamic programming.*

This naturally leads to the following approximation algorithm for knapsack.

$(1+\varepsilon)$-Approximation Algorithm

1. Let $c = \max_i c_i$.
2. Round down each c_i to the nearest multiple of $\frac{\varepsilon c}{n}$. Let this quantity be $r_i \cdot \left(\frac{\varepsilon c}{n}\right)$, i.e., $r_i = \lfloor c_i / \frac{\varepsilon c}{n} \rfloor$.
3. With these new quantities (r_i) as profits of items, use the standard Dynamic Programming algorithm, to find the most profitable set I'.

The number of r_i's is at most n/ε. Thus, the running time of this algorithm is at most $poly(n, n/\varepsilon) = poly(n, 1/\varepsilon)$. We now show that the above algorithm obtains a $(1-\varepsilon)$-approximation ratio

Claim 1.2.2. *$\sum_{i \in I'} c_i$ is an $(1-\varepsilon)$-approximation to OPT.*

Proof. Let O be the optimal set. For each item, rounding down of c_i causes a loss in profit of at most $\frac{\varepsilon c}{n}$. Hence the total loss due to rounding down is at most n times $\frac{\varepsilon c}{n}$. In other words,

$$\sum_{i \in O} c_i - \frac{\varepsilon c}{n} \cdot \sum_{i \in O} r_i \leq n \frac{\varepsilon c}{n} = c\varepsilon$$

Hence, $\frac{\varepsilon c}{n} \sum_{i \in O} r_i \geq OPT - c\varepsilon$. Now,

$$\sum_{i \in I'} c_i \geq \frac{\varepsilon c}{n} . \sum_{i \in I'} r_i \geq \frac{\varepsilon c}{n} \sum_{i \in O} r_i \geq OPT - \varepsilon c \geq (1-\varepsilon) OPT$$

The first inequality follows from the definition of r_i, the second from the fact that I' is an optimal solution with costs r_i's, the third from the above observation and the last from the fact that $OPT \geq c$. □

1.2.2 Type-2 PTAS

In these kinds of problems we define a set of "simple" solutions and find the minimum cost simple solution in polynomial time. Next, we show that an arbitrary solution may be modified to a simple solution without greatly affecting the cost.

Euclidean TSP A Euclidean TSP is a TSP instance where the points are in $\mathbb{R}^2$ and the distances are the corresponding Euclidean distances.

A trivial solution can be found in $n!$. Dynamic Programming finds a solution in $n^2 2^n$.

We now give a high-level description of a $n^{1/\varepsilon}$-time algorithm that achieves a $(1+\varepsilon)$-approximation ratio. Consider the smallest square that contains all n points. Use quad-tree partitioning to recursively partition each square into four subsquares until unit squares are obtained. We consider the number of times the tour path crosses a cell in the quad-tree. We construct the "simple solution" to the problem by restricting the tour to cross each dividing line $\leq \frac{6}{\varepsilon}$ times. We can then discretize the lines at these crossing points. Each square has $\leq \frac{24}{\varepsilon}$ number of crossing points. A tour may use each of these crossing points either 0, 1 or 2 times. So for the entire quadtree there are $\leq 3^{\frac{24}{\varepsilon}} = \exp(\frac{24}{\varepsilon})$ number of possibilities. For details see Arora's 2003 survey [Aro03].

In the next section, we will see examples of approximation algorithms which use linear programming and semi-definite programming.

1.3 Approximation Algorithms for MAXCUT

The MAX-CUT problem is as follows: Given a graph $G(V, E)$ with $|V| = n$, find $\max_{S \subset V} |E(S, \overline{S})|$.

The notation $E(S, \overline{S})$ refers to the set of all edges (i, j) such that vertex $i \in S$ and vertex $j \in \overline{S}$.

1.3.1 Integer Program Version

Define variable x_i, such that $x_i = 0$, if vertex $i \in S$ and $x_i = 1$, if $i \in \overline{S}$. We have the following integer program:

$$\begin{aligned} \text{Maximize} \sum_{(ij) \in E} & e_{ij} \\ \text{subject to} & \\ e_{ij} &\leq \min\{x_i + x_j, 2 - (x_i + x_j)\}, \ \forall (i,j) \in E \\ x_1, \ldots, x_n &\in \{0, 1\} \end{aligned}$$

Notice that $e_{ij} \neq 0 \iff x_i \neq x_j$.

1.3.2 Linear Program Relaxation and Randomized Rounding

This can be converted to a **Linear Program** as follows:

$$\begin{aligned} \text{Maximize} \sum_{(ij)\in E} e_{ij} & \\ \text{subject to} & \\ e_{ij} &\leq \min\{x_i + x_j, 2 - (x_i + x_j)\},\ \forall (i,j) \in E \\ x_1, \ldots, x_n &\in [0,1] \end{aligned}$$

Every solution to the Integer Program is also a solution to the Linear Program. So the objective function will only rise. If OPT_{LP} is the optimal solution to the LP, then:

$$\text{OPT}_{\text{LP}} \geq \text{MAX-CUT}$$

Randomized Rounding

We now round the LP-solution to obtain an integral solution as follows: form a set S by putting i in S with probability x_i. The expected number of edges in such a cut, $\mathbb{E}[|E(S,\bar{S})|]$ can be then calculated as follows:

$$\begin{aligned} \mathbb{E}[|E(S,\bar{S})|] &= \sum_{(i,j)\in E} Pr[(i,j) \text{ is in the cut}] \\ &= \sum_{(i,j)\in E} x_i(1-x_j) + x_j(1-x_i) \end{aligned}$$

The above calculates only an expected value of the cut, however if we repeat the above algorithm several times, it can be seen by Markov's inequality that we can get we can get very close to this value. We now show that this expected value is at least half the LP-optimal, which in turns means that it is at least half the MAX-CUT

Claim 1.3.1.

$$\mathbb{E}[|E(S,\bar{S})|] = \sum_{(ij)\in E} x_i(1-x_j) + x_j(1-x_i) \geq \frac{1}{2}\text{OPT}_{\text{LP}} \geq \frac{1}{2}\text{MAX-CUT}$$

Proof. We have

$$\text{OPT}_{\text{LP}} = \sum_{(ij)\in E} e_{ij} = \sum_{(i,j)\in E} \min\{(x_i + x_j), 2 - (x_i + x_j)\}$$

It can easily be checked that for any $x_i, x_j \in [0,1]$, we have

$$x_i(1-x_j) + x_j(1-x_i) \geq \frac{1}{2} \cdot \min\{(x_i + x_j), 2 - (x_i + x_j)\}.$$

Thus, a term by term comparison of the LHS of the inequality with OPT_{LP} reveals that $\mathbb{E}[|E(S,\bar{S})|] \geq \frac{1}{2}OPT_{LP} \geq \frac{1}{2}$MAX-CUT. □

We thus, have a 1/2-approximation algorithm for MAX-CUT using randomized rounding of the LP-relaxation of the problem. Actually, it is to be noted that the LP-relaxation is pretty stupid, the optimal to the LP is the trivial solution $x_i = 1/2$ for all i, which in turn leads to $OPT_{LP} = |E|$. But we do mention this example as it naturally leads to the following more powerful SDP relaxation.

1.3.3 Semi Definite Programming (SDP) Based Method

We will now sketch a 0.878-approximation to MAX-CUT due to Goemans and Williamson [GW95]. The main idea is to relax the integer problem defined above using vector valued variables. THE SDP relaxation is as follows:

$$\begin{aligned} \text{Maximize} \quad & \sum_{(i,j)\in E} \frac{(1 - \langle \mathbf{v}_i, \mathbf{v}_j \rangle)}{2} \\ \text{subject to} \quad & \langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1, \ \forall i \end{aligned}$$

Denote the optimal to the above SDP by OPT_{SDP}. We first observe that the SDP is in fact a relaxation of the integral problem. Let $\mathbf{v}_0$ be any vector of unit length, i.e., $\langle \mathbf{v}_0, \mathbf{v}_0 \rangle = 1$. Consider the optimal cut S that achieves MAX-CUT. Now define,

$$\mathbf{v}_i = \begin{cases} \mathbf{v}_0 & \text{if} \quad i \in S \\ -\mathbf{v}_0 & \text{if} \quad i \notin S \end{cases}, \forall i.$$

Consider the quantity $\frac{(1-\langle \mathbf{v}_i, \mathbf{v}_j \rangle)}{2}$. This is 0 if the vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ lie on the same side, and equals 1 if they lie on opposite sides. Thus, $OPT_{SDP} \geq$ MAX-CUT.

How do we round the SDP solution to obtain an integral solution. The novel rounding due to Goemans and Williamson is as follows: The SDP solution produces n vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Now pick a random hyperplane passing through the origin of the sphere and partition vectors according to which side tof the hyperplane they lie. Let $(S, \bar{S})$ be the cut obtained by the above rounding scheme. It is easy to see that

$$\begin{aligned} \mathbb{E}[|E(S,\bar{S})|] &= \sum_{(i,j)\in E} \Pr[(i,j) \in \text{cut}] \\ &= \sum_{(i,j)\in E} \Pr[\mathbf{v}_i, \mathbf{v}_j \text{ lie on opposite sides of the hyperplane}] \end{aligned}$$

Let θ_{ij} be the angle between vectors $\mathbf{v}_i$ and $\mathbf{v}_j$. Then the probability that they are cut is proportional to θ_{ij}, in fact exactly θ_{ij}/π. Thus,

$$\mathbb{E}[|E(S,\bar{S})|] = \sum_{(ij)\in E} \frac{\theta_{ij}}{\pi}$$

Let us know express OPT_{SDP} in terms of the θ_{ij}'s. Since $\theta_{ij} = cos^{-1}(\langle \mathbf{v}_i, \mathbf{v}_j \rangle)$, we have

$$OPT_{SDP} = \sum_{(i,j)\in E} \frac{(1 - cos\theta_{ij})}{2}$$

By a "miracle of nature"(Mathematica?) Goemans and Williamson observed that

$$\frac{\theta}{\pi} \geq (0.878\ldots) \times \frac{1 - cos\theta}{2},\ \forall \theta \in [0, \pi]$$

Hence,

$$\frac{\mathbb{E}[|E(S, \bar{S})|]}{\mathrm{OPT}_{\mathrm{SDP}}} \geq 0.8788.$$

Thus, we have a 0.878-approximation algorithm for MAX-CUT.

Lecture 2

The PCP Theorem: An Introduction

Dana Moshkovitz

Scribe: Alexander S. Kulikov

20 Jul, 2009

Complementing the first introduction lecture on approximation algorithms, this lecture will be an introduction to the limits of approximation algorithms. This will in turn naturally lead to the PCP Theorem, a ground-breaking discovery from the early 90's.

2.1 Optimization Problems and Gap Problems

The topic of this lecture is the hardness of approximation. But to talk about hardness of approximation, we first need to talk about optimization problems. Recall the definition of optimization problems from the earlier lecture. Let us begin by giving an example of a canonical optimization problem.

Definition 2.1.1 (MAX-3SAT)**.** *The* maximum 3-satisfiability problem *(MAX-3SAT) is: Given a* 3*-CNF formula* φ *(each clause contains* exactly *three literals) with* m *clauses, what is the maximum fraction of the clauses that can be satisfied simultaneously by any assignment to the variables?*

We first prove the following important claim.

Claim 2.1.2. *There exists an assignment that satisfies at least* $7/8$ *fraction of clauses.*

Proof. The proof is a classical example of the probabilistic method. Take a random assignment (each variable of a given formula is assigned either 0 or 1 randomly and independently). Let Y_i be a random variable indicating whether the i-th clause is satisfied. For any

$1 \leq i \leq m$ (where m is the number of clauses),

$$\mathbb{E}Y_i = 0 \cdot \frac{1}{8} + 1 \cdot \frac{7}{8},$$

as exactly one of eight possible assignments of Boolean constants to the variables of the i-th clause falsifies this clause. Here we use the fact that each clause contains exactly three literals.

Now, let Y be a random variable equal to the number of satisfied clauses: $Y = \sum_{i=1}^{m} Y_i$. Then, by linearity of expectation,

$$\mathbb{E}Y = \mathbb{E}\sum_{i=1}^{m} Y_i = \sum_{i=1}^{m} \mathbb{E}Y_i = \frac{7m}{8}.$$

Since a random assignment satisfies a fraction 7/8 of all clauses, there must exist an assignment satisfying at least as many clauses. □

The natural question to ask is if we can do better? Can we find an assignment that satisfies more clauses. Let us phrase this question more formally. For this, we first recall the definition of approximation algorithms from the previous lecture.

Definition 2.1.3. *An algorithm C for a maximization optimization problem is called α-approximation (where $0 \leq \alpha \leq 1$), if for every input x, the algorithm C outputs a value which is at least α times the optimal value, i.e.,*

$$\alpha \cdot \mathrm{OPT}(x) \leq C(x) \leq \mathrm{OPT}(x).$$

Claim 2.1.2 implies immediately that there exists an efficient (i.e., polynomial time) 7/8-approximation algorithm for MAX-3SAT. The natural question is whether there exists an approximation algorithm that attains a better approximation ratio. The answer is that such an algorithm is not known. The question that we are going to consider in this lecture is whether we can prove that such an algorithm does not exist. Of course, if we want to prove this, we have to assume that P≠NP, because otherwise there is an efficient 1-approximation algorithm.

There is some technical barrier here. We are talking about optimization problems, i.e., problems where our goal is to compute something. It is however much more convenient to consider decision problems (or languages), where we have only two possible answers: yes or no. So, we are going to transform an optimization problem to a decision problem. Namely, we show that hardness of a certain decision problem implies hardness of approximation of the corresponding optimization problem.

Definition 2.1.4. *For a maximization problem I and $A < B \in \mathbb{R}^+$, the corresponding $[A, B]$-gap problem is the following promise decision problem*[1]*:*

$$\begin{aligned} \mathrm{YES} &= \{x | \mathrm{OPT}(x) \geq B\} \\ \mathrm{NO} &= \{x | \mathrm{OPT}(x) < A\} \end{aligned}$$

[1] A promise problem Π is specified by a pair $(\mathrm{YES}, \mathrm{NO})$ where $\mathrm{YES}, \mathrm{NO} \in \{0,1\}^*$ and YES and NO are disjoint sets. Note there is no requirement that $\mathrm{YES} \cup \mathrm{NO} = \{0,1\}^*$. This is the only difference between promise problems and languages.

We now relate the hardness of the maximization problem to the hardness of the gap problem.

Theorem 2.1.5. *If the $[A, B]$-gap version of a maximization problem is NP-hard, then it is NP-hard to approximate the maximization problem to within a factor A/B.*

Proof. Assume, for the sake of contradiction, that there is a polynomial time A/B-approximation algorithm C for a maximization problem under consideration. We are going to show that this algorithm can be used in order to solve the gap problem in polynomial time.

The algorithm for the gap problem is: for a given input x, if $C(x) \geq A$, return "yes", otherwise return "no".

Indeed, if x is a yes-instance for the gap problem, i.e., $\mathrm{OPT}(x) \geq B$, then

$$C(x) \geq A/B \cdot \mathrm{OPT}(x) \geq A/B \cdot B = A$$

and we answer "yes" correctly. If, on the other hand, $\mathrm{OPT}(x) < A$, then

$$C(x) \leq \mathrm{OPT}(x) < A$$

and we give the correct "no" answer. □

Thus, to show hardness of approximation to within a particular factor, it suffices to show hardness of the corresponding gap problem. Hence from now onwards, we focus on gap problems.

2.2 Probabilistic Checking of Proofs

We will now see a surprising alternate description of the hardness of gap problems. The alternate description is in terms of probabilistically checkable proofs, called PCPs for short.

2.2.1 Checking of Proofs

Let us first recall the classical notion of proof checking. NP is the class of languages that have a deterministic polynomial-time verifier. For every input x in the language, there exists a proof that convinces the verifier that x is in the language. For every input x not in the language, there is no proof that convinces the verifier that x is in the language.

For example, when the language is 3SAT, the input is a 3CNF formula φ. A proof for the satisfiability of φ is an assignment to the variables that satisfies φ.

A verifier that checks such a proof may need to go over the entire proof before it can know whether φ is satisfiable: the assignment can satisfy all the clauses in φ, but the last one to be checked.

2.2.2 Local Checking of Proofs

Can we find some other proof for the satisfiability of φ that can be checked *locally*, by querying only a *constant* number of symbols from the proof?

For this to be possible, we allow the queries to be chosen in a randomized manner (otherwise, effectively the proof is only of constant size, and a language that can be decided using a

polynomial-time verifier with access to such a proof can be decided using a polynomial-time algorithm). The queries should be chosen using at most a logarithmic number of random bits. The logarithmic bound ensures that the verifier can be, in particular, transformed into a standard, deterministic polynomial time, verifier. The deterministic verifier would just perform *all* possible checks, instead of one chosen at random. Since the number of random bits is logarithmic, the total number of possible checks is polynomial. The number of queries the deterministic verifier makes to the proof is polynomial as well.

To summarize, we want a verifier that given the input and a proof, tosses a logarithmic number of random coins and uses them to make a constant number of queries to the proof. If the input is in the language, there should exist a proof that the verifier accepts with probability at least B. If the input is not in the language, for any proof, the verifier should accept with probability at most A. The *error* probability is the probability that the verifier does not decide correctly, i.e., $1 - B + A$. If $B = 1$, we say that the verifier has *perfect completeness*, i.e., it never errs on inputs in the language.

2.2.3 The Connection to The Hardness of Gap Problems

The NP-hardness of approximation of 3SAT is in fact *equivalent* to the existence of local verifiers for NP:

Hardness ⇒ Local Verifier

For $[A, 1]$-gap-MAX-3SAT, there is a verifier that makes only 3 queries to the proof, has perfect completeness, and errs with probability at most A!

On input formula φ, the proof is a satisfying assignment for φ. The verifier chooses a random clause of φ, reads the assignment to the three variables of the clause, and checks if the clause is satisfied. The verifier uses $\log m$ random bits, where m is the number of clauses. If φ is satisfiable, the verifier accepts the proof with probability 1. If not, at most A fraction of all clauses of φ can be satisfied simultaneously, so the verifier accepts with probability at most A.

Moreover, the NP-hardness of $[A, 1]$-gap-MAX-3SAT yields local verifiers for all NP languages! More precisely,

Claim 2.2.1. *If $[A, 1]$-gap-MAX-3SAT is NP-hard, then every NP language L has a* probabilistically checkable proof *(PCP). That is, there is an efficient randomized verifier that uses only logarithmic number of coin tosses and queries 3 proof symbols, such that*

- *if $x \in L$, then there exists a proof that is always accepted;*
- *if $x \notin L$, then for any proof the probability to err and accept is at most A.*

Note that the probability of error can be reduced from A to ε by repeating the action of the verifier $k = O(\frac{\log 1/\varepsilon}{\log 1/A})$ times, thus making $O(k)$ queries.

Local Verifier ⇒ Hardness

What about the other direction? Do local verifiers for NP imply the NP-hardness of gap-MAX-3SAT, which would in turn imply inapproximability of MAX-3SAT?

For starters, let us assume that every language in NP has a verifier that makes three bit queries and whose acceptance predicate is the OR of the three variables (or their negations). Assume that for inputs in the language the verfier always accepts a vaild proof, while for inputs not in the language, for any proof, the verifier accepts with probability at most A. From the above correspondence between local verifiers and gap problems, we get that $[A, 1]$-gap-MAX-3SAT is NP-hard.

What if the verifier instead reads a constant number of bits (not 3) and its acceptance predicate is some other Boolean function on these constant number of bits? We will now use the fact that every Boolean predicate $f\colon \{0,1\}^{O(1)} \to \{0,1\}$ can be written as a 3-CNF formula φ with clauses (and some additional variables z) such that for any assignment x of Boolean values to variables of f, $f(x) = 1$ iff $\varphi(x, z)$ is satisfiable for some z. We have thus transformed the $O(1)$-query verfier with an arbitrary Boolean acceptance predicate to a 3-query verifier with acceptance predicate an OR of the 3 variables (or their negations). We thus have.

Claim 2.2.2. *If every NP language L has a constant query verifier that uses only logarithmic number of coin tosses and queries Q proof symbols, such that*

- *if $x \in L$, then there exists a proof that is always accepted;*
- *if $x \notin L$, then for any proof the probability to err and accept is at most A.*

then $[A', 1]$-gap-MAX-3SAT is NP-hard for some $A' < 1$.

Thus, we have shown that the problem of proving NP-hardness of gap-MAX-3SAT is equivalent to the problem of constructing constant query verifiers for NP. But do such verifiers exist?

2.2.4 The PCP Theorem

Following a long sequence of work, Arora and Safra and Arora, Lund, Motwani, Sudan and Szegedy in the early 90's constructed local verifiers for NP:

Theorem 2.2.3 (PCP Theorem (. . . ,[AS98],[ALM+98]))**.** *Every NP language L has a probabilistically checkable proof (PCP). More precisely, there is an efficient randomized verifier that uses only logarithmic number of coin tosses and queries $O(1)$ proof symbols, such that*

- *if $x \in L$, then there exists a proof that is always accepted;*
- *if $x \notin L$, then for any proof the probability to accept it is at most $1/2$.*

The proof in [AS98, ALM+98] is algebraic and uses properties of low-degree polynomials. There is a more recent alternate combinatorial proof for the theorem due to Dinur [Din07]. We will later in the workshop see some of the elements that go into the construction.

The PCP Theorem shows that it is NP-hard to approximate MAX-3SAT to within *some constant factor*. The natural further question is: can we improve this constant to 7/8 (to match the trivial approximation algorithm from Claim 2.1.2)? A positive answer to this question would yield a *tight* 7/8-hardness for approximation of MAX-3SAT.

2.3 Projection Games

In 1995, Bellare, Goldreich, and Sudan [BGS98] introduced a paradigm for proving inapproximability results. Following this paradigm, Håstad [Hås01] established *tight* hardness for approximating MAX-3SAT, as well as many other problems.

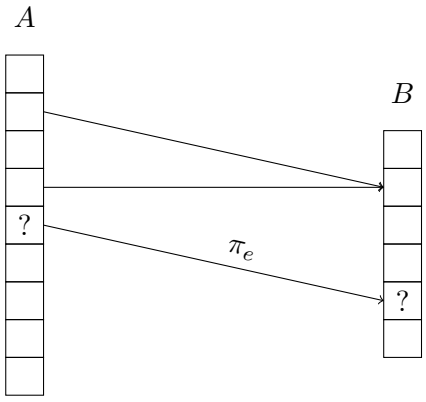
The paradigm is based on the hardness of a particular gap problem, called *Label-Cover*.

Definition 2.3.1. *An instance of a* projection game *(also called* label-cover*) is specified by bipartite graph* $G = (A, B, E)$*, two alphabets* Σ_A*,* Σ_B*, and* projections $\pi_e \colon \Sigma_A \to \Sigma_B$ *(for every edge* $e \in E$*).*

Given assignments $\mathcal{A} : A \to \Sigma_A$ *and* $\mathcal{B} : B \to \Sigma_B$*, an edge* $e = (a, b) \in E$ *is said to be satisfied iff* $\pi_e(\mathcal{A}(a)) = \mathcal{B}(b)$*. The* value *of this game is*

$$\max_{\mathcal{A},\mathcal{B}} \left(\Pr_{e \in E} [e \text{ is satisfied}] \right) .$$

In a label-cover problem*: given a projection game instance, compute the value of the game.*



The PCP-theorem could be formulated as a theorem about Label-Cover.

Theorem 2.3.2 (. . . , [AS98], [ALM+98])**.** *Label-Cover is NP-hard to approximate within* some constant*.*

Proof. The proof is by reduction to Label-Cover. Recall that we have a PCP verifier from the previous formulation. We construct a bipartite graph as follows. For each possible random string of the verifier, we have a vertex on the left-side. Since the verifier uses only a logarithmic number of bits, the number of such strings is polynomial. For each proof location, we have a vertex on the right-side. We add an edge between a left vertex (i.e., a random string) and a right vertex (i.e., a proof location), if the verifier queries this proof location on this random string. Thus, we defined a bipartite graph. We are now going to describe the labels and projections.

The labels for the left-side vertices are accepting verifier views and for the right-side are proof symbols. A projection is just a consistency check. For example, in case of MAX-3SAT, we have satisfying assignments of a clause on the left-side and values of variables on the right-side. □

However, this theorem is not strong enough to get tight hardness of approximation results. For this reason, we call this the weak projection games theorem. What we actually need is a low error version of this theorem, which improves the hardness in Theorem 2.3.2 from "some constant" to "any arbitrarily small constant".

Theorem 2.3.3 ((strong) Projection Games Theorem (aka Raz's verifier) [Raz98])**.** *For every constant $\varepsilon > 0$, there exists a $k = k(\varepsilon)$ such that is is NP-hard to decide if a given projection game with labels of size at most k (ie., $|\Sigma_A|, |\Sigma_B| \leq k$) has value 1 or at most ε.*

The PCP construction in the strong projection games theorem is commonly refered to as Raz's verifier as the theorem follows from Raz's *parallel repetition theorem* applied to the construction in Theorem 2.3.2 [Raz98]. Recently, Moshkovitz and Raz [MR08] gave an alternate proof of this theorem that allows the error ε to be sub-constant.

Why does the label size k depend on ε in the above theorem? This is explained by the following claim, which implies that k must be at least $1/\varepsilon$.

Claim 2.3.4. *There is an efficient $1/k$-approximation algorithm for projection games on labels of size k (i.e., $|\Sigma_A|, |\Sigma_B| \leq k$).*

We will later in this tutorial see how the projection games theorem implies tight hardness of approximation for 3Sat.

We remark that for many other problems, like Vertex-Cover or Max-Cut, we do not know of tight hardness of approximation results based on the projection games theorem. To handle such problems, Khot formulated the *unique games conjecture* [Kho02]. This conjecture postulates the hardness of *unique label cover*, where the projections π_e on the edges are permutations. More on that – later in the tutorial.

Lecture 3

Approximation Algorithms for Network Problems

Matthew Andrews

Scribe: Gwen Spencer

20 July, 09

Lectures 3 and 4 will be on network/flow problems, the known approximation algorithms and inapproximability results. In particular, this lecture will serve as an introduction to the different types of network/flow problems and a survey of the known results, while the follow-up lecture by Lisa Zhang will deal with some of the techniques that go into proving hardness of approximation of network/flow problems.

3.1 Network Flow Problems

Network/Flow problems are often motivated by industrial applications. We are given a communication or transportation network and our goal is to move/route objects/information though these networks.

The basic problem that we shall be considering is defined by a graph $G = (V, E)$ and a set of (source,destination) pairs of nodes which we'll denote (s_1, t_1), (s_2, t_2), etc. We will sometimes call these pairs "demand pairs." There are many variants of the problem:

- **Only Connectivity is required.** The question is one of feasibility: "Is it possible to select a subset of the edge set of G that connects every (s_i, t_i) pair?"
- **Capacities must be respected.** Each edge has a capacity, and each (s_i, t_i) pair has some amount of demand that must be routed from s_i to t_i. Observe that this problem is infeasible if there exists a cut in the graph which has less capacity than the demand which must cross it. Imagine variations on this problem in which more

capacity can be purchased on an edge at some cost (that is, the capacities are not *strict*): the question becomes: "What is the minimum amount of capacity that must be purchased to feasibly route all demand pairs?"

- **What solutions are "good" depends on the objective function.** Consider the difference between the objective of trying to minimize the maximum congestion (where congestion is the total demand routed along an edge) and the objective of trying to minimize the total capacity purchased: it is not hard to find examples where a good solution with respect to the first objective is a bad solution with respect to the second objective and vice versa. The maximum congestion objective is often used to describe delay/quality of service.

- **Splittable vs. unsplittable flow.** In the unsplittable flow case all demand routed between s_i and t_i must travel on a single (s_i, t_i) path. In the splittable flow case each demand can be split so that it is routed on a set of paths between s_i and t_i.

- **Directed vs. Undirected.** Is the graph directed or undirected? As a rule of thumb, problems in which the graph is directed are more difficult.

Next we'll consider some specific problems and describe what positive and negative results exist for each of them:

3.1.1 Minimum Cost Steiner Forest

In this problem we are interested in simple connectivity. The input to the problem is a graph with edge costs and a set of (s_i, t_i) pairs. The goal is to connect each (s_i, t_i) pair via a set of edges which has the minimum possible total cost (the cost of a set of edges is just the sum of the costs of all edges in the set).

Notice that any feasible solution to this problem is a set of trees.

Both positive and negative results exist for this problem:

- **Positive:** 2-approximation (Agrawal-Klein-Ravi [AKR95], Goemans-Williamson [GW95])

- **Negative:** APX-hard, there exists ε such that no $1 + \varepsilon$ approximation algorithm exists for the problem unless P=NP.

3.1.2 Congestion Minimization (Fractional)

The input to this problem is a graph with edge capacities and a set of (s_i, t_i) pairs. The goal is to connect all (s_i, t_i) pairs fractionally (that is, for all i, to route a total of one unit of demand from s_i to t_i along some set of paths in the graph) in a way that minimizes the maximum congestion. The congestion on an edge is simply the total demand routed on that edge divided by the capacity of the edge. The maximum congestion is the maximum congestion taken over all edges in the graph.

This problem can be solved exactly in polynomial time via a linear program. We write the linear program as follows: let u_e denote the capacity of edge e, and have a decision variable $x_{p,i}$ which is the amount of demand i that is routed on path p:

$$\begin{array}{ll} \min & z \\ \text{s.t.} & \sum_p x_{p,i} = 1 \quad \forall i \\ & \sum_i \sum_{p:e\in p} x_{p,i} \le z u_e \quad \forall e. \\ & x_{p,i} \ge 0 \quad \forall p, i. \end{array}$$

The first set of constraints says that for each demand pair i, one unit of demand must be routed from s_i to t_i. The second set of constraints says that for each edge e, the sum of all demand routed on e must be less than z times the capacity of e. Since the objective is to minimize z, the optimal LP solution finds the minimum multiplicative factor z required so that the capacity of each edge is at least $1/z$ times the total demand routed on that edge (that is, the optimal z is the minimum possible maximum congestion).

Though this LP is not of polynomial size (the number of paths may be exponentially large) it can be solved in polynomial time, using an equivalent edge-based formulation whose variables $y_{e,i}$ represent the amount of flow from demand i routed through edge e. Hence we can obtain an exact solution to the problem.

3.1.3 Congestion Minimization (Integral)

Now consider the Congestion Minimization problem when we require the routing be **integral** (all demand routed from s_i to t_i must be routed on a single path). We can no longer solve this problem using the linear program above. The following results are known:

- **Positive:** A $O(\log n / \log\log n)$-approximation algorithm where n is the number of vertices due to Raghavan-Thompson [RT87]. This algorithm is based on the technique of *randomized rounding* which we describe below.
- **Negative:** Andrews-Zhang [AZ07]) show that there is no $(\log n)^{1-\varepsilon}$-approximation unless NP has *efficient* algorithms. More formally our result holds unless NP$\subseteq$ZPTIME($n^{polylog(n)}$), where ZPTIME($n^{polylog(n)}$) is the class of languages that can be recognized by randomized algorithms that always give the correct answer and whose expected running time is $n^{polylog(n)} = n^{\log^k n}$ for some constant k. The assumption that NP$\not\subseteq$ZPTIME($n^{polylog(n)}$) is not quite as strong an assumption as $NP \neq P$ but is still widely believed to be true.

Note that the gap between the positive and negative results here is large. We comment that for the directed version of the problem, a negative result has been proved that no $\Omega(\log n / \log\log n)$-approximation exists unless NP has efficient algorithms [CGKT07, AZ08].

We now describe the Raghavan-Thompson randomized rounding method for approximating the Integral Congestion Minimization Problem:

- Notice that the LP for the fractional problem is a linear relaxation of the IP we would write for the integral case. Thus, the optimal solution to the fractional version is a lower bound on the optimal value of the integral version: $OPT_{frac} \le OPT_{integral}$.

- Note that $\sum_p x_{p,i} = 1$ for all i. Treat the $x_{p,i}$ as a probability distribution: demand i is routed on path p with probability $x_{p,i}$. By linearity of expectation, the expected congestion of the resulting ranodmly rounded solution on each edge is at most OPT_{frac}.

  In any given rounding though, some edges will have more than their expected congestion. It is possible to show that for any *fixed* edge e, with large probability ($\geq 1 - \frac{1}{2n^2}$) the congestion on edge e is $O(\log n / \log\log n)OPT_{frac} = O(\log n / \log\log n)OPT_{integral}$. By a union bound this implies that with probability at least $\frac{1}{2}$ the maximum congestion of the randomly rounded solution *on any edge* is $O(\log n / \log\log n)OPT_{integral}$.

  For the directed case this gives the best achievable approximation. Whether something better exists for the undirected case is an open question.

3.1.4 Edge Disjoint Paths

The input is a graph with edge capacities and a set of (s_i, t_i) pairs. The goal is to connect every (s_i, t_i) pair integrally using disjoint paths (that is, to find a set of paths, one connecting each pair, such that the paths for two distinct pairs share no edges). The goal is to connect the maximum possible number of pairs.

The following results are known:

- **Positive:** A $O(m^{1/2})$-approximation where m is the number of edges, due to Kleinberg [Kle96].
- **Negative:** (undirected) No $(\log n)^{1/2-\varepsilon}$ -approximation exists unless NP has efficient algorithms (Andrews-Zhang [AZ06]).
- **Negative:**(directed) No $O(m^{1/2-\varepsilon})$-approximation exists unless P=NP (Guruswami-Khanna-Rajaraman-Shepherd-Yannakakis [GKR+03]).

We'll look at Kleinberg's $m^{1/2}$ approximation algorithm for this problem (m is the number of edges). Consider a greedy algorithm as follows:

1. Find the shortest path that connects two terminals.
2. Remove all the edges on that path from the graph.
3. Repeat until we cannot connect any more terminals.

Analysis. At all times we let G' be the subgraph of G that contains the remaining edges (i.e. the edges that have not been removed in Step 2). There are two cases in our analysis: either the shortest path p linking two terminals in the remaining graph G' has length at most $m^{1/2}$, or not:

- Suppose the shortest path p in G' has length $\leq m^{1/2}$. Each edge in p intersects at most one path from the optimal solution (since the paths in the optimal solution are disjoint), so p intersects at most $m^{1/2}$ paths from the optimal solution. Thus, when the algorithm removes p, at most $m^{1/2}$ paths are removed from the optimal solution.

Thus, the algorithm produces at least one path for every $m^{1/2}$ paths in the optimal solution.

- Suppose the shortest path p in G' has length strictly greater than $m^{1/2}$. Since the paths in the optimal solution are disjoint, and all must have length at least as long as p, the optimal solution has at most $m/((m^{1/2}) = m^{1/2}$ paths in G'. Thus, to get a $m^{1/2}$ approximation for G' the algorithm need only produce one path (so the algorithm can just use p).

We mention that for this problem we can't hope to do better with a linear programming relaxation method because the gap between the optimal IP solution and the optimal LP solution can be $m^{1/2}$.

3.1.5 Minimum Cost Network Design

The input is a graph, a set of (s_i, t_i) pairs and a cost function $f(c)$ for placing capacity on an edge. The goal is to route one unit of demand between each pair in a way that requires the minimum cost expenditure for capacity.

Commonly considered cost functions include:

1. Linear: shortest paths are optimal.
2. Constant: this results in the Steiner forest problem.
3. Subadditive: economies of scale and buy-at-bulk problems. This is a nice way of modelling how aggregating demand onto a core network is beneficial and arises in many industrial network design problems. For subadditive cost functions the following results are known:
   - **Positive:** a $O(\log n)$-approximation (Awerbuch-Azar [AA97], Bartal [Bar98], Fakcharoenphol-Rao-Talwar [FRT04]).
   - **Negative:** No $(\log n)^{1/4-\varepsilon}$ approximation exists unless NP has efficient algorithms (Andrews [And04]).

Summary

Approximation ratios vary widely for different types of network flow problems:

- Constant approximation: Steiner forest.
- $O(\log n)$-approximation: Congestion minimization, Buy-at-Bulk network design.
- $m^{1/2}$-approximation: Edge Disjoint paths.

Questions

Q: Are these algorithms actually what is used in practice?

Answer: Not exactly. Take the case of the randomized rounding that we covered: in practice this technique may not give the best congestion due to the Birthday Paradox. It is quite likely that there is some edge that gets higher congestion than the average by a logarithmic factor. Hence, practical algorithms typically apply heuristics to try and reduce the congestion. One technique that often works well is to sort the demands based on the distance between the terminals (from closest to farthest). We then go through the demands in order and try to greedily reroute them.

We remark that industrial networks often cost a huge amount of money and so tweaking a solution a little to save even a single percent can generate meaningful cost savings. In addition, a lot of these real applications are huge: cutting-edge computing power together with CPLEX are not even close to being able to solve these problems exactly. Approximation really is necessary.

Lecture 4

Hardness of the Edge-Disjoint Paths Problem

Lisa Zhang

Scribe: David Pritchard

20 July, 2009

4.1 Overview

The *edge-disjoint paths* problem (EDP) is the combinatorial optimization problem with inputs

- a (directed or undirected) graph G with n nodes and m edges
- a list of k pairs of (not necessarily distinct) nodes of G, denoted $(s_i, t_i)_{i=1}^k$

and whose output is

- a subset X of $\{1, \ldots, k\}$ representing a choice of paths to route
- s_i-t_i paths $\{P_i\}_{i \in X}$ so that the P_i are pairwise edge-disjoint

and

- the objective is to maximize $|X|$.

In these notes, the main results are: a simple proof that for any $\varepsilon > 0$ it is NP-hard to approximate the *directed* edge-disjoint paths problem to ratio $m^{1/2-\varepsilon}$ (Section 4.3); and a more complex proof that for any $\varepsilon > 0$, if we could approximate the *undirected* edge-disjoint paths problem to ratio $\log^{1/3-\varepsilon} n$, then there would be randomized quasi-polynomial time algorithms for NP (Section 4.4).

4.2 Literature

For directed EDP, there is a simple $\sqrt{m}$-approximation algorithm due to Kleinberg [Kle96] (see also Erlebach's survey [Erl06]), which nearly matches the $m^{1/2-\varepsilon}$-hardness result we will present (which is due to Guruswami et al. [GKR+03]). A $O(n^{2/3}\log^{2/3} n)$ approximation is also known [VV04].

For undirected EDP, Kleinberg's simple algorithm [Kle96] still gives a $\sqrt{m}$-approximation, but an improved $\sqrt{n}$-approximation was recently obtained by Chekuri et al. [CKS06]. The main technical ingredient in the proof we will present is the *high girth argument*, which was used first in 2004 by Andrews [And04] and subsequently in a variety of papers [ACG+07, AZ06, AZ07, AZ08, AZ09, CK06, GT06b], some of which have closed the approximability and inapproximability gaps of various problems up to constant factors. Many of these papers deal with *congestion minimization*, where all demands must be routed and the objective is to minimize the maximum load on any edge. Focusing on undirected graphs, the papers most closely related to what we will show are:

- [And04], which introduced the high girth argument and gave a polylog-hardness for buy-at-bulk undirected network design. For this problem, all demands must be routed, and the cost minimized. The types of "buy-at-bulk" edges used in the hardness construction were fixed-cost edges (which once bought, can be used to any capacity) and linear-cost edges (where you pay proportional to the capacity used). The paper reduced from a type of 2-prover interactive system, similar to PCPs.

- [AZ06], which gave the $\log^{1/3-\varepsilon} n$-hardness proof we will describe in these notes. The paper reduced from Trevisan's inapproximability results [Tre01] on the *bounded degree independent set* problem. In turn, those resuls rely on advanced PCP technology [ST00].

- [ACG+07] — a paper which was the culmination of merging several lines of work — which resulted in an improved $\log^{1/2-\varepsilon} n$-hardness proof for undirected EDP. This paper uses the hardness of *constraint satisfaction problems*, while the preliminary versions use the Raz verifier (parallel repetition) and directly-PCP based methods. This is so far the best inapproximability result known for undirected EDP, although it is very far from the best known approximation ratio of $\sqrt{n}$ [CKS06].

In more detail, the table below summarizes some results in the literature (lower bounds assume $\mathsf{NP} \not\subset \mathsf{ZPTIME}(n^{\text{polylog}n})$, and some constant factors are omitted). Stars ($\star$) denote results in which the high girth method is used.

Problem	Upper bound	Lower Bound
Undirected EDP	$m^{1/2}$ [Kle96], $n^{1/2}$ [CKS06]	$\star\log^{1/3-\varepsilon} m$ [AZ06], $\star\log^{1/2-\varepsilon} m$ [ACG+07]
Directed EDP	$m^{1/2}$ [Kle96], $n^{2/3}\log^{2/3} n$ [VV04]	$m^{1/2-\varepsilon}$ [GKR+03]
Undirected Congestion Minimization	$\log m/\log\log m$ [RT87]	$\star\log^{1-\varepsilon}\log m$ [AZ07], $\star\frac{\log\log m}{\log\log\log m}$ [RZ]
Directed Congestion Minimization	$\log m/\log\log m$ [RT87]	$\star\log^{1-\varepsilon} m$ [AZ08], $\star\frac{\log m}{\log\log m}$ [CGKT07]
Undirected Uniform Buy-at-Bulk	$\log m$ [AA97, FRT04]	$\star\log^{1/4-\varepsilon} m$ [And04]
Undirected Nonuniform Buy-at-Bulk	$\log^5 m$ [CHKS06]	$\star\log^{1/2-\varepsilon} m$ [And04]
Undirected EDP with Congestion c (with some restrictions on c)	$n^{1/c}$ [AR06, BS00, KS01]	$\star\log^{(1-\varepsilon)/(c+1)} m$ [ACG+07]
Directed EDP with Congestion c (with some restrictions on c)	$n^{1/c}$ [AR06, BS00, KS01]	$\star n^{\Omega(1/c)}$ [CGKT07]

If the number k of terminal pairs is fixed, the undirected EDP problem is exactly solvable in polynomial-time, using results from the theory of graph minors [RS95]. (As we will see in Theorem 4.3.2, the directed case behaves differently.)

4.3 Hardness of Directed EDP

In this section we prove the following theorem.

Theorem 4.3.1 ([GKR+03])**.** *For any $\varepsilon > 0$ it is* NP*-hard to approximate the directed edge-disjoint paths problem* (Dir-EDP) *to within ratio $m^{1/2-\varepsilon}$.*

Although it is very common for inapproximability proofs in the literature to reduce one approximation problem to another, this proof has the cute property that it reduces an exact problem to an approximation problem. Phrased differently, the complete proof does not rely on any PCP-like technology. Specifically, our starting point is the following theorem.

Theorem 4.3.2 ([FHW80])**.** *The following decision problem* (Dir-2EDP) *is* NP*-hard: given a directed graph G and four designated vertices s, s', t, t' in the graph, determine whether there are two edge-disjoint directed paths, one from s to t, and another from s' to t'.*

(Note, this immediately shows that it is hard to NP-hard to approximate (Dir-EDP) to a factor better than 2.)

The key to proving Theorem 4.3.1 is a construction which maps (G, s, s', t, t') to an instance $(H, (s_i, t_i)_{i=1}^k)$ of Dir-EDP where k is a parameter we will tune later. The construction is illustrated in Figure 4.1. The two important properties of this construction are the following:

(a) If G admits edge-disjoint s-t and s'-t' paths (say P and P'), then H has a solution of value k (i.e. all pairs s_i-t_i for $1 \le i \le k$ can be simultaneously linked by edge-disjoint paths). To see this, we utilize the copies of P and P' within each copy of G; then it's easy to see there are mutually disjoint paths s_i-t_i paths (the path leaving s_i goes up through $i-1$ copies of P, then right through $k-i$ copies of P', to t_i).

(b) If G does not admit edge-disjoint s-t and s'-t' paths, then there is no solution for H with value greater than 1. To see this, suppose for the sake of contradiction that there is an s_i-t_i path P_i and a s_j-t_j path P_j in H such that P_i, P_j are edge-disjoint.

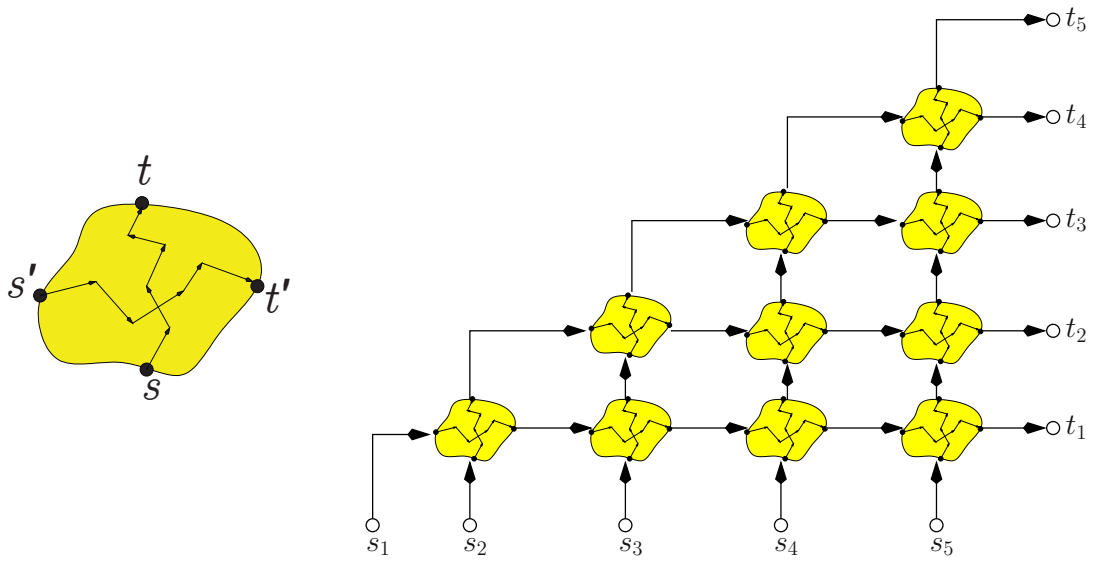


Figure 4.1: Left: the graph G in an instance of Dir-2EDP. Right: The construction of H for $k = 5$. Each small yellow object is a copy of G, and each other line is a directed edge. This illustration is adapted from Erlebach [Erl06].

> Without loss of generality $i < j$. Then a topological argument shows that there must be some copy of H such that P_i uses the copies of s' and t' and P_j uses the copies of s and t. This contradicts our assumption about the Dir-2EDP instance.

Facts (a) and (b) show that any algorithm that has approximation ratio better than $k/1$ on the Dir-EDP instance H also solves the Dir-2EDP instance.

Without loss of generality we assume G is (weakly) connected, then the encoding size of the Dir-2EDP instance is proportional to $|E(G)|$ and the encoding size of the Dir-EDP instance is $|E(H)| = O(k^2|E(G)|)$. In order to conclude that it is NP-hard to approximate the Dir-EDP instance to a factor better than k, we need $|E(H)|$ to be polynomial in $|E(G)|$. Thus we may take $k = |E(G)|^\alpha$ for any constant α. Going back to the analysis, we get $k = |E(H)|^{\frac{\alpha}{2\alpha+1}}$; hence by taking $\alpha \to \infty$, we get the desired result (that it is NP-hard to approximate Dir-EDP to a factor $k = m^{1/2-\varepsilon}$).

4.4 Hardness of Undirected EDP

In this section we sketch the proof of the following theorem.

Theorem 4.4.1 ([AZ06]). *For any $\varepsilon > 0$, if we can approximate the directed edge-disjoint paths problem* (Undir-EDP) *to within ratio $O(\log^{1/3-\varepsilon} m)$, then every problem in* NP *has a probabilistic always-correct algorithm with expected running time* $\exp(\text{polylog}(n))$, *i.e.* $\mathsf{NP} \subset \mathsf{ZPTIME}(\exp(\text{polylog}(n)))$.

We fully describe the construction of the proof and give intuition for the analysis, but skip some of the detailed parts and precise setting of parameters. The construction creates a

simple graph (i.e. one with no parallel edges) so the theorem also holds with $\log m$ replaced by $\log n$ since these are the same up to a factor of 2. The proof shows more precisely that $\mathsf{NP} \subset \mathsf{coRPTIME}(\exp(\mathrm{polylog}(n)))$, i.e. it gives a quasi-polynomial size, randomized reduction with one-sided error that is right at least (say) 2/3 of the time; then standard arguments[1] allow us to move to ZPTIME. Here is the proof overview.

- The starting point is the inapproximability of the *independent set* problem (IS) in bounded-degree graphs: find a set of mutually non-adjacent vertices (an *independent set*) with as large cardinality as possible. We denote the degree upper bound by Δ.
- As usual, our goal is to find a transformation from IS instances to Undir-EDP instances which preserves the "NP-hard-to-distinguish gap" in the objective function.
- We will create a new graph G in the following way. Roughly we "define a path" P_i for each vertex v_i of the IS instance so that $P_i \cap P_j \neq \varnothing$ iff v_i, v_j are adjacent. (It is easy to see this is possible, with the length of P_i proportional to the degree of v_i.) Then we define G to be the union of all P_i. (See Figure 4.2.)
- To get some intuition for the rest of the proof, define the terminals s_i, t_i of the Undir-EDP instance to be the endpoints of P_i. Then it is *almost* true that the Undir-EDP instance is isomorphic (in terms of feasible solutions) to the IS instance. The significant problem is that G necessarily also contains s_i-t_i paths other than P_i, which may be used for routing. (Such paths are obtained by using a combination of edges taken from different P_j's; see Figure 4.2.)
- To get around this problem, we transform G into a different graph H defined by two parameters x, c. Each intersection of two paths P_i, P_j is replaced by c consecutive intersections; and we replace each P_i with x images $\{P_{i,\alpha}\}_{\alpha=1}^{x}$. The construction of H has a lot of independent randomness, two consequences of which are that (i) when v_i, v_j are adjacent, we can lower-bound the probability that $P_{i,\alpha} \cap P_{j,\beta} \neq \varnothing$ and (ii) H has few short cycles. We call each $P_{i,\alpha}$ a *canonical path*; for each canonical path its endpoints define a terminal pair for the new Undir-EDP instance.
- The optimum of the Undir-EDP solution is at least x times the optimum of the IS instance. To get our hardness-of-approximation result, we also need that when the Undir-EDP optimum is "large", so is the IS optimum. This is done via a map from Undir-EDP solutions R on H to IS solutions S on G. The map is parameterized by a number $a \leq x$. We (1) throw away all non-canonical paths in R and (2) take v_i in S iff at least a out of the x paths $\{P_{i,\alpha}\}_{\alpha=1}^{x}$ are routed by Y. The final analysis uses the fact that the canonical paths have length $O(c\Delta)$ while most non-canonical paths are long; the latter depends on the fact that H has few short cycles.

[1] We insert Q into standard complexity class names to denote quasi-polynomial time. Suppose $\mathsf{NP} \subset \mathsf{coRQP}$. Then there is a $f(n)$-time algorithm for SAT with f quasi-polynomial. This also implies $\mathsf{NQP} \subset \mathsf{coRQP}$ since every NQP language with quasi-polynomial running time $g(n)$ is equivalent (by the Cook-Levin construction) to satisfiability of a formula of size $g(n)$, and it can be decided in time $f(g(n))$ which is quasi-polynomial. The definition of RQP immediately implies $\mathsf{RQP} \subset \mathsf{NQP}$ hence $\mathsf{RQP} \subset \mathsf{coRQP}$. Taking complements we deduce $\mathsf{RQP} = \mathsf{coRQP}$ and it is easy to show that $\mathsf{RQP} \cap \mathsf{coRQP} = \mathsf{ZPQP}$, hence $\mathsf{NP} \subset \mathsf{coRQP} = \mathsf{ZPQP}$. Alternatively, see Lemma 5.8 in [EH03] for a more efficient construction.

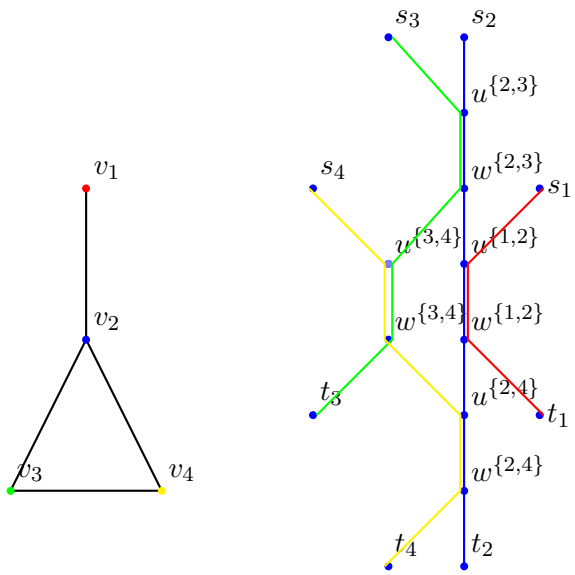


Figure 4.2: An illustration of how the independent set instance, G_0 (left), yields a UNDIR-EDP instance on graph G (right). The colour of v_i in G_0 corresponds to the colour of canonical path P_i in G. Note that in this example, even though $\{v_1, v_2\}$ is not an independent set, there are edge-disjoint s_1-t_1 and s_2-t_2 paths (P_1, and a non-canonical s_2-t_2 path).

4.4.1 Hardness of Bounded-Degree Independent Set

Trevisan [Tre01] showed that for any constant Δ, it is NP-hard to approximate the independent set problem within ratio $\Delta/2^{\sqrt{\log \Delta}}$ on graphs with degrees bounded by Δ. For our purposes, we need a version of this that works for super-constant Δ. By extending the framework in Trevisan [Tre01], Andrews and Zhang [AZ06] proved the following:

Theorem 4.4.2. *Consider the family of graphs with upper bound $\Delta = \log^b n$ on degree, where n is the number of nodes and b is a constant. If there is a $(\log^{b-\varepsilon} n)$-approximation algorithm for* IS *on these graphs for any $\varepsilon > 0$, then* $\mathsf{NP} \subset \mathsf{coRPTIME}(n^{O(\log \log n)})$.

4.4.2 The Graphs G and H

First we give a formal description of the graph G we sketched earlier. Let G_0 denote the IS instance, without loss of generality G_0 is connected. Each edge $e = v_i v_j$ of G_0 yields two vertices $u^{\{i,j\}}, w^{\{i,j\}} \in V(G)$ and each vertex v_i of G_0 yields two vertices s_i, t_i; these are all the vertices of G. Let the neighbours of v_i in G_0 in any order be $v_p, v_q, \ldots, v_r$, then we define the path $P_i := (s_i, u^{\{i,p\}}, w^{\{i,p\}}, u^{\{i,q\}}, w^{\{i,q\}}, \ldots, u^{\{i,r\}}, w^{\{i,r\}}, t_i)$. More precisely, for each adjacent pair of vertices in this list we define an edge of G; this constitutes all of the edges of G. Every edge of the form $u^{\{i,j\}} w^{\{i,j\}}$ appears in both P_i and P_j, while every other edge of G appears in exactly one P_i. The number of vertices and edges of G is $O(|E(G_0)|)$ and the number k of terminal pairs is $|V(G_0)|$.

There are two additional ideas needed to define H, one whose effect is to randomly replace P_i by x images $\{P_{i,\alpha}\}_{\alpha=1}^{x}$ and another whose effect is to increase the probability that two paths $P_{i,\alpha}$, $P_{j,\beta}$ intersect when $v_i v_j \in E(G_0)$.

Consider the following probabilistic operation f_x on graphs: replace every vertex v by x "copies" $\{v_\alpha\}_{\alpha=1}^{x}$, and replace every edge vv' with a random bipartite matching of $\{v_\alpha\}_{\alpha=1}^{x}$ to $\{v'_\alpha\}_{\alpha=1}^{x}$, where these matchings are chosen independently for all input edges vv'. Thus f_x multiplies the total number of vertices and edges by x. Note that if vv' is an edge of some graph K and $1 \le \alpha, \beta \le x$ then the probability that $u_\alpha v_\beta \in f_x(K)$ is exactly $1/x$; we will later use this fact, as well as the independence of the different random matchings, to show that $f_x(K)$ behaves like a random graph in terms of short cycles. We define the image of terminal pairs under f_x as follows. Define $P_{i,\alpha}$ as the unique path in $f_x(G)$ obtained by starting at $s_{i,\alpha}$ (which denotes $(s_i)_\alpha$) and following the images of edges of P_i. $P_{i,\alpha}$ does not necessarily end at $t_{i,\alpha}$, rather it ends at $t_{i,\beta} =: t'_{i,\alpha}$ for some uniformly random β. The terminal pairs of $f_x(G)$ are all pairs $(s_{i,\alpha}, t'_{i,\alpha})$. We call the $P_{i,\alpha}$ *canonical paths*.

At this point it is straightforward to compute the following: if $v_i v_j \in E(G_0)$ and α, β are fixed, the probability that the paths $P_{i,\alpha}$, $P_{j,\beta}$ intersect in $f_x(G)$ is exactly $1/x$. More generally, for subsets A, B of $\{1, \ldots, x\}$, the probability $\Pr[\{P_{i,\alpha}\}_{\alpha \in A} \cup \{P_{j,\beta}\}_{\beta \in B}$ are mutually edge-disjoint in $f_x(G)]$ can be expressed as some function $\delta(x, |A|, |B|)$[2]. We would like to decrease this (i.e. increase the probability some $P_{i,\alpha}$ intersects some $P_{j,\beta}$), and to do so, we consider a graph G' obtained similarly to G except, for $v_i v_j \in E(G_0)$, we force P_i and P_j to intersect c times. We give an informal but precise definition since the formal definition is lengthy. To construct G' from G, we perform the following for all edges $v_i v_j \in E(G_0)$:

[2]Explicitly, $\delta(x, |A|, |B|) = (x - |A|)!(x - |B|)!/(x - |A| - |B|)!x!$.

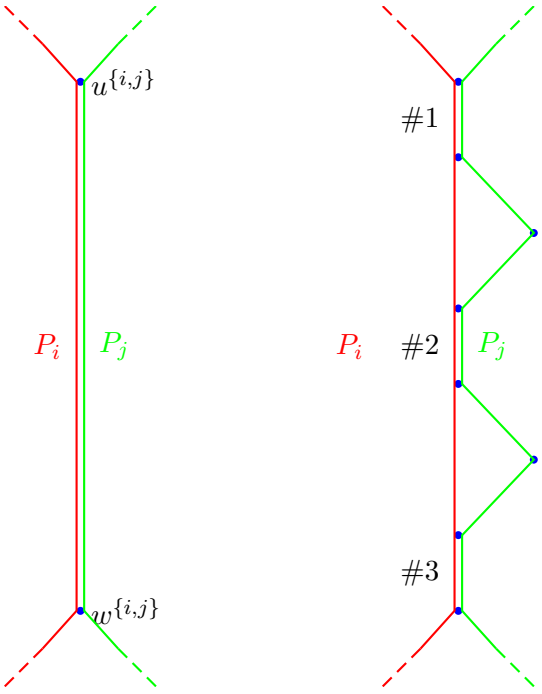


Figure 4.3: Illustrating, for $c = 3$, the operation used to transform G (left) into G' (right).

replace the intersection edge $u^{\{i,j\}}w^{\{i,j\}}$ with the gadget pictured in Figure 4.3, and simultaneously redefine P_i, P_j to follow the indicated paths. Not only will this cause the new P_i and P_j to intersect c times, but the images of these intersections under f_x will be *independent* in the sense that, for all subsets A, B of $\{1, \ldots, x\}$, the probability $\Pr[\{P_{i,\alpha}\}_{\alpha \in A} \cup \{P_{j,\beta}\}_{\beta \in B}$ are mutually edge-disjoint in $f_x(G')]$ decreases to $\delta^c(x, |A|, |B|)$.

Finally, H is defined to be $f_x(G')$, with canonical paths $P_{i,\alpha}$ and terminal pairs defined as for $f_x(G)$.

4.4.3 Small Cycles

The graph $H = f_x(G')$ we defined is not quite a "random graph" in the usual (Erdős-Renyi) sense, but it has enough randomness that it has a typical property of random graphs, namely that the number of small cycles can be bounded. This is done using the first moment method (Markov's inequality), analogous to the 1963 Erdős-Sachs theorem (e.g. that in the Erdős-Renyi model $G(n, d/n)$ the number of cycles of length g is at most d^g in expectation).

If C is any simple cycle in G', then it is not hard to see that the expected number of simple cycles in H that are "images" of C is 1. This is a good bound but it is not quite sufficient for our purposes, since cycles may exist in H whose inverse image in G' is not simple. To be precise about getting a bound, for each edge vv' of G', we say that it has x^2 corresponding *potential edges* $\{v_\alpha v'_\beta \mid 1 \leq \alpha, \beta \leq x\}$ in $H = f_x(G')$. (In any realization of H, exactly x of these edges are actually present.) Then it is not hard to see that we get the

following: conditioned on the existence of any $\kappa < x$ potential edges in H, the probability that any other potential edge is in H is at most $1/(x-\kappa)$. Then the first moment method allows us to show:

$$\text{if } g < x/2, \mathbb{E}[\#\text{ cycles in } H \text{ of length} \leq g] \leq O(n_0 c\Delta)^{g+1} \tag{4.4.1}$$

where we define $n_0 = |V(G_0)|$. Note that this bound is independent of x; roughly speaking this is because the factor of x in $|V(H)|$ cancels with the factor $1/x$ in the probability of H containing any given potential edge. Eventually, we will set g to be poly-logarithmic in n_0, and the right-hand side of Equation (4.4.1) will be quasi-polynomial in n_0.

It is not hard to argue that H has maximum degree 3, so each vertex has $O(2^g)$ vertices within distance g; combining this fact with Equation (4.4.1) gives us the following form of the "high-girth argument" that we use in the final proof:

$$\Pr[O(n_0 c\Delta)^{g+1} \text{ vertices in } H \text{ have distance} \leq g \text{ to a cycle of length} \leq g] \geq 9/10. \tag{4.4.2}$$

4.4.4 Analysis Sketch

As mentioned earlier, our reduction uses the following map $R \mapsto S$ from Undir-EDP routings on H to independent sets on G_0, parameterized by a number a: put v_i into S if at least a out of the x paths $P_{i,\alpha}$ for i are routed by R. To be exact, S is only an independent set with some probability, which we would like to make large. By applying simple bounds to δ, for any $A, B \subset \{1, \ldots, x\}$ with $|A|, |B| \geq a$ and for i, j adjacent in G_0, we have that

$$\Pr[\{P_{i,\alpha}\}_{\alpha \in A} \cup \{P_{j,\beta}\}_{\beta \in B} \text{ mutually edge-disjoint in } f_x(G')] \leq \exp(-ca^2/x). \tag{4.4.3}$$

For any two fixed adjacent vertices v_i, v_j in G_0, by a union bound, the probability that *any subsets* A, B with $|A|, |B| \geq a$ exist, such that A and B fail the event in (4.4.3) is at most $\binom{x}{a}\binom{x}{a}\exp(-ca^2/x)$. Therefore using another union bound,

$$\Pr[S \text{ independent}] \geq 1 - |E(G_0)|\binom{x}{a}\binom{x}{a}\exp(-ca^2/x). \tag{4.4.4}$$

The setting of parameters in the proof is then chosen so that $\Pr[S \text{ independent}]$ is at least $9/10$. This, along with Theorem 4.4.2 and Equation (4.4.2), are the three sources of error in the proof.

At a high level the analysis breaks the paths in R into four types,

(a) a canonical path $P_{i,\alpha}$ so that $\#\{\beta | P_{i,\beta} \in R\} \geq a$.

(b) a canonical path $P_{i,\alpha}$ so that $\#\{\beta | P_{i,\beta} \in R\} < a$

(c) a non-canonical $s_{i,\alpha}$-$t'_{i,\alpha}$ path where $s_{i,\alpha}$ has distance $\leq g$ to a cycle of length $\leq g$

(d) any other non-canonical $s_{i,\alpha}$-$t'_{i,\alpha}$ path

and applies the following analysis (recall $n = |E(G_0)|$):

- The number of paths of type (a) is at most $|S|x$.

- The number of paths of type (b) is at most $(n_0 - |S|)a$.
- The number of paths of type (c) is at most $O(n_0 c\Delta)^{g+1}$ by (4.4.2).
- To upper bound the number of paths of type (d), let P' denote one such path. The union of P' and $P_{i,\alpha}$ contains a simple cycle, but the length of that cycle is at least g. The length of $P_{i,\alpha}$ is fixed at $O(c\Delta)$ and hence the length of P' is at least $g - O(c\Delta)$. Since the type-(d) paths are disjoint, there are at most $|E(H)|/(g - O(c\Delta))$ of them.

This gives a lower bound on $|S|$ in terms of $|R|$. The proof is then completed by setting the parameters carefully. In detail, using the fact that the greedy algorithm for independent set on G_0 always gives a solution of value at least $n_0/(\Delta + 1)$, we can show that $|S|$ is an independent set with size at least a constant times $|R|/x$ provided that $g > \Delta^2 c$, $x > \Delta a$, $c > \frac{x}{a} \ln \frac{x}{a} + \frac{x \ln n_0}{a^2}$, $x > \Delta^2 c \cdot O(n_0 \Delta c)^g$ hold, where we have omitted some constant factors. (These conditions come from the relative contributions of the different types of paths, as well as the error bounds.) The ratio of inapproximability for Undir-EDP is then roughly Δ as a function of the input size $m = |E(H)| = O(n_0 c\Delta x)$, and it is not hard to show that Δ is roughly $\log^{1/3} m$ at maximum. (In [AZ06], a precise setting of parameters is given.)

Lecture 5

Proof of the PCP Theorem (Part I)

Prahladh Harsha

Scribe: Ashkan Aazami

20 July, 2009

In this lecture and the follow-up lecture tomorow, we will see a sketch of the proof of the PCP theorem. Recall the statement of the PCP theorem from Dana Moshkovitz's lecture earlier today. Dana had mentioned both a weak form (the original PCP Theorem) and a strong form (Raz's verifier or hardness of projective games). We need the strong form as it is the starting point of most tight inapproximability results. "Standard proofs" of the strong form proceed as follows: first prove the PCP Theorem [AS98, ALM+98] either using the original proof or the new proof of Dinur [Din07] and then apply Raz's parallel repetition [Raz98] theorem to it to obtain the strong form. However, since the work of Moshkovitz and Raz [MR08], we can alternatively obtain the strong form directly using the proof techniques in the orginal proof of the PCP Theorem along with the composition technique of Dinur and Harsha [DH09]. We will follow the latter approach in this tutorial.

5.1 Probabilistically Checkable Proofs (PCPs)

We first introduce the *probabilistically checkable proof* (PCP) and some variants of it.

Our goal is to construct a PCP for some NP-complete problem. We will work with the NP-complete problem Circuit-SAT. Let C be an instance of the Circuit-SAT problem. A PCP consists of a verifier V that is provided with a proof π of acceptance of the input instance C. The goal of the verifier is to check if the given proof is "valid". Given the input C, the verifier V generates a random string R and based on the input instance C and the random bits of R it generates a list Q of queries from the proof π. Next, the verifier V

queries the proof π at the locations of Q and based on the content of the proof in these locations the verifier either accepts the input C as an acceptable instance or rejects it. The content of π at the locations Q is called the *local view* of π and it is denoted by π_Q. We denote the *local predicate* that the verifier checks by φ; the verifier accepts if $\varphi(\pi_Q) = 1$ and it rejects otherwise. The verifier has the following properties:

Completeness: If C is satisfiable then there is a proof π such that the verifier always accepts with probability 1; i.e.,

$$\exists \pi : \text{Prob}\left[\varphi(\pi_Q) = 1\right] = 1.$$

Soundness: If C is not satisfiable then for every proof π the verifier accepts with probability at most δ (say $\delta = \frac{1}{3}$);

$$\forall \pi : \text{Prob}\left[\varphi(\pi_Q) = 1\right] \leq \frac{1}{3}.$$

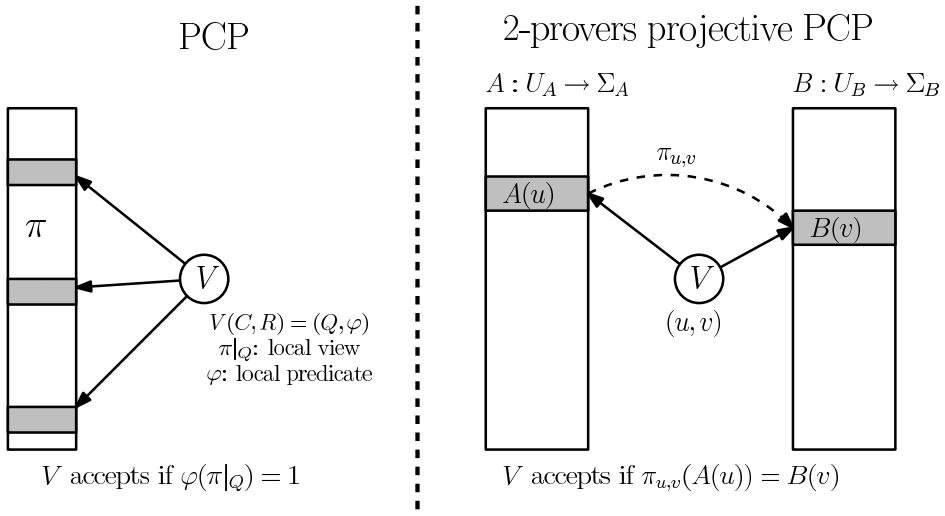


Figure 5.1: PCP and 2-queries projective PCP

The original PCP Theorem now can be stated formally as follows.

Theorem 5.1.1 (PCP Theorem [AS98, ALM+98])**.** CIRCUIT-SAT *has a PCP (with the above completeness and soundness properties) that uses* $|R| = O(\log n)$ *random bits and queries* $|Q| = O(1)$ *locations of the proof where* n *is the size of the input circuit.*

Note that the length of the proof π is polynomial in n, the size of the input instance C, since the length of the random string is $O(\log n)$ so the verifier can make $2^{O(\log n)}$ number of queries.

5.1.1 Strong Form of the PCP Theorem and Robust PCPs

Now we introduce a strong form of the PCP theorem, this is also called the 2-prover *projection game* theorem. In this type of PCPs, there are two non-communicating provers $A : U_A \to \Sigma_A$ and $B : U_B \to \Sigma_B$ and a verifier V. Given the input instance C, the verifier first generates a random string R of length logarithmic in the input size and then using the random string, it determines two locations u and v and generates a projection function $\pi_{u,v} : \Sigma_A \to \Sigma_B$. The verifier then queries the two provers U_A and U_B on locations u and v respectively and accepts the provers' answers if they are consistent with the projection function ie., $\pi_{u,v}(A(u)) = B(v)$.

We can now state the strong from of the PCP theorem as follows.

Theorem 5.1.2 (Strong form of PCP (aka Raz's verifier, hardness of projection games) [Raz98])**.** *For any constant $\delta > 0$, there exist alphabets Σ_A, Σ_B such that the* Circuit-SAT *has a 2-prover projection game with a verifier V such that*

Completeness: *If C is satsifiable ($C \in$ Circuit-SAT) then there exist two provers $A : U_A \to \Sigma_A, B : U_B \to \Sigma_B$ such that*

$$\text{Prob}\left[\pi_{u,v}(A(u)) = B(v)\right] = 1$$

Soundness: *If C is not satisiable ($C \notin$ Circuit-SAT) then for all pairs of provers $A : U_A \to \Sigma_A, B : U_B \to \Sigma_B$, we have*

$$\text{Prob}\left[\pi_{u,v}(A(u)) = B(v)\right] \leq \delta.$$

Now we introduce the notion of *robust* PCP. These PCPs have a stronger soundness property. In the ordinary PCPs, the soundness property says that if the input instance C is not an acceptable input, then the local predicate that the verifier checks is not satisfied with high probability. In the robust PCPs the local view is far from any satisfying assignment with high probability.

First for some notation. Given two codewords x and y, the agreement between x and y is defined as $agr(x, y) = \Pr_i[x_i = y_i]$. For a given set S of code-words, we define the agreement of S and x by $\text{agr}(x, S) = \max_{y \in S} \text{agr}(x, y)$. Let us denote the set of all satisfiable assignments to the local predicate φ by $SAT(Q)$.

The robust PCPs have the same completeness property as in the ordinary PCPs, but they have a stronger soundness property. More precisely, the following soundness property of regular PCPs is replaced by the stronger "robust soundness" property.

Soundness:

$$C \notin \text{Circuit-SAT} \Rightarrow \text{Prob}\left[\pi_Q \in SAT(\varphi)\right] \leq \delta$$

Robust Soundness:

$$C \notin \text{Circuit-SAT} \Rightarrow \mathbb{E}[\text{agr}(\pi_Q, SAT(\varphi))] \leq \delta$$

We call PCPs with the robust soundness property, robust PCPs.

5.1.2 Equivalence of Robust PCPs and 2-Provers Projection PCPs

Note that robust PCPs are just regular PCPs with a stronger soundness requirement. We now show that robust PCPs are equivalent to 2-provers projection PCPs. Given a robust PCP with the verifier V and the prover π, we construct a 2-prover projective verifier V' and two provers A, B as follows. The prover B is the same prover as π. For each possible random string R and the corresponding queries Q of the verifier V, the prover A has the local view π_Q at the location indexed by R; i.e., the prover A has all possible local views of the prover π. The verifier V' of the 2-prover projection PCP is as follows.

1. Generate a random string R and compute a set Q of queries as in the verifier V.
2. Query 1: Asks the prover A for the entire "accepting" local view (i.e., π_Q).
3. Query 2: Ask the prover B for a random location within the local view (i.e., $(\pi_Q)_i$).
4. Accept if the answer of the prover B is consistent with the answer of the prover A.

It is an easy exercise to check the following two facts. The constructed 2-provers PCP has the completeness property. Tthe robust soundness of the robust PCP translates into the soundness of the 2-provers PCP. A closer look at this transformation reveals that it is in fact, invertible. This demonstrates a syntactic equivalence between robust PCPs and 2-prover projection PCPs. Note that in this equivalence, the alphabet size of the left prover $|\Sigma_A|$ translates to query complexity of the robust PCP verifier (to be precise, free-bit complexity of robust PCP verifier). Given this equivalence, our goal to prove Theorem 5.1.2 can be equivalently stated as constructing for every constant δ, robust PCPs for Circuit-SAT with robust soundness δ and query complexity some function of δ (but independent of n).

5.2 Locally Checkable Codes

Our goal is to construct a robust PCP for the Circuit-SAT over a constant size alphabet with constant number of queries for arbitrarily small soundness error. To achieve this goal, we need to transform a NP-proof (or a certificate for an NP problem) to a proof that can be locally checked. To do this, we use locally checkable codes. There are two potential candidates for locally checkable codes.

1. The first one is the *Directed Product* code; the new proof of the PCP theorem by Dinur and the proof of the parallel repetition theorem of Raz are based on this encoding.
2. The second one is the *Reed-Muller* code which is based on the low-degree polynomials over a finite field $\mathbb{F}$, and the original proof of the PCP theorem is based on this encoding.

We use the Reed-Muller code in construction of the robust PCP.

A PCP, by definition, is a locally checkable encoding of the NP witness. In the rest of today's lecture, we shall construct locally checkable encodings of two very specific properties, namely "low-degreeness" and "being zero on a sub-cube". We will define these properties formally shortly, however it is worth noting that neither of these properties is a NP-complete property. In the next lecture, we will show how despite this, we can use the local checkability of these two properties to construct PCPs for all of NP.

5.2.1 Reed-Muller Code

Let $\mathbb{F}$ be a finite field, and let $\mathcal{P}_d^m$ be the set of all m-variate polynomials of degree at most d over $\mathbb{F}$. The natural way of specifying a function $f \in \mathcal{P}_d^m$ is to list the coefficients of f. It is easy to check that a m-variate polynomial of degree d has $\binom{m+d}{m}$ coefficients. The Reed-Muller encoding of f is the list of the evaluations of f on all $x \in \mathbb{F}^m$; the codeword at the position indexed by $x \in \mathbb{F}^m$ has value $f(x)$. The length of this codeword is $|\mathbb{F}^m|$.

This encoding is inefficient but there is an efficient "local test" to find out if a given codeword is close to a correct encoding of a low degree polynomial.

- Question: Given a function $f : \mathbb{F}^m \to \mathbb{F}$, how does one check if f is a Reed-Muller encoding: The straightforward way to do this is to interpolate the polynomial and check if it has degree at most d.
- Question: Given a function $f : \mathbb{F}^m \to \mathbb{F}$, how does one *locally* check if f is close to a Reed-Muller encoding. A test for this purpose was first suggested by Rubinfeld and Sudan [RS96] This test is based on the fact that a restriction of a low-degree polynomial (over $\mathbb{F}^m$) to a line (or any space with small dimension) is also a low-degree polynomial.

5.2.2 Low Degree Test (Line-Point Test)

Given the evaluations of function f on all points in $\mathbb{F}^m$. Our goal is to check if f is close to a m-variate polynomial of degree at most d; we do this by checking the values of the function f on a random line. A set $\{x + ty | t \in \mathbb{F}\}$, for some $x, y \in \mathbb{F}^m$, is called a line in $\mathbb{F}^m$.

Low Degree Test (LDT):

1. Pick a random line ℓ in $\mathbb{F}^m$; this can be done by picking two random points $x, y \in \mathbb{F}^m$.
2. Query the function f on all points of the line ℓ. Let $f|_\ell$ denote the restriction of f on the line ℓ (i.e., $f|_\ell(t) = f(x + ty)$).
3. Accept if $f|_\ell$ is an univariate low-degree polynomial (i.e., $f|_\ell \in \mathcal{P}_d^1$).

Clearly, if $f \in \mathcal{P}_d^m$, then $f|_\ell$ is an univariate polynomial of degree at most d. Hence, we have the perfect completeness.

Completeness: $f \in \mathcal{P}_d^m \Rightarrow \text{Prob}\,[\text{LDT accepts}] = 1$

Rubinfeld and Sudan [RS96] proved the following form of soundness for this test.

Soundness: $\forall \delta, \exists \delta' : \text{Prob}\,[\text{LDT accepts}] \geq 1 - \delta \Rightarrow f$ is $(1 - \delta')$-close to some low-degree polynomial (i.e., $\text{agr}(f, \mathcal{P}_d^m) \geq 1 - \delta'$).

We will actually need the following stronger soundness that was proven by Arora and Sudan [AS03].

Stronger Soundness: $\mathbb{E}[\text{agr}(f|_\ell, \mathcal{P}_d^1)] \geq \delta \Rightarrow \text{agr}(f, \mathcal{P}_d^m) \geq \delta - m\varepsilon$, where $\varepsilon = \text{poly}(m, d, \frac{1}{|\mathbb{F}|})$.

Raz and Safra [RS97] proved an equivalent statement (with better dependence of $|\mathbb{F}|$ on d) for the plane-point test as opposed to the line-point test.

5.2.3 Zero Sub-Cube Test

In this section, we introduce another test that is used in the construction of robust PCPs. Let f be a polynomial over $\mathbb{F}^m$ and let H be a subset of $\mathbb{F}$. We want to test if f is a low degree polynomial (i.e., $f \in \mathcal{P}_d^m$) and if it is zero on the sub-cube H^m (i.e., $f|_{H^m} \equiv 0$). Using the low degree test (LDT) we can check if $f \in \mathcal{P}_d^m$, but to test if f is zero on H^m it is not enough to pick few random points from H^m and test if f is zero on those points.

Before describing the correct test, we present two results about the polynomials.

Lemma 5.2.1 (Schwartz-Zippel)**.** *Let f be a m-variate polynomial of degree d over $\mathbb{F}^m$. If f is not a zero polynomial (i.e., $f \not\equiv 0$), then*

$$\Pr_x [f(x) = 0] \leq \frac{d}{|\mathbb{F}|}.$$

The above lemma shows that if a low degree polynomial over a sufficiently large field is not zero at every point, then it can only be zero on small fraction of points.

Proposition 5.2.2. *Let f be a polynomial of degree at most d over $\mathbb{F}^m$. The restriction of f to H^m is a zero polynomial (i.e., $f|_{H^m} \equiv 0$) if and only if there exist polynomials $q_1, \ldots, q_m$ of degree at most $d - |H|$ such that*

$$f(x) = \sum_{i=1}^{m} g_H(x_i) q_i(x), \tag{5.2.1}$$

where $g_H(x) = \prod_{h \in H} (x - h)$ is an univariate polynomial (of degree $|H|$).

Now we describe the Zero Sub-cube Test. In the LDT we assumed that the evaluation of f on all points are given in the proof table. By the above Proposition if the polynomial f of degree at most d is zero on H^m then there are polynomials $q_1, \ldots, q_m$ of degree at most $d - |H|$ that satisfy Equation (5.2.1). In the Zero Sub-Cube Test, we require that the proof table also contains the evaluations of $q_1, \ldots, q_m$ (in addition to the evaluation of f) on all points in $\mathbb{F}^m$.

Zero Sub-cube Test:

1. Choose a random line ℓ in $\mathbb{F}^m$.
2. For $f, q_1, \ldots, q_m$ check if $f|_\ell, q_1|_\ell, \ldots, q_m|_\ell$ is a low degree polynomial. In more detail, check if $f|_\ell$ has degree at most d, and for each $i = 1, \ldots, m$ check if $q_i|_\ell$ has degree at most $d - |H|$.
3. For each $x \in \ell$, check if $f(x) = \sum_{i=1}^{m} g_H(x_i) q_i(x)$.
4. Accept if each of the above tests passes, and reject otherwise.

Combining the soundness of the low-degree test and the above properties of polynomials, we can prove the following completeness and soundness of the **Zero Sub-cube Test**. Let $\mathcal{Z}_d^m$ denote the set of m-variate polynomials P of degree d such that $P|_{H^m} = 0$. Also for any line ℓ, let $\mathrm{acc}(\ell)$ denote the set of accepting local views of the **Zero Sub-cube Test** for the random line ℓ.

Completeness: If $f \in \mathcal{Z}_d^m$, then Prob [Zero Sub-cube Test accepts] $= 1$ or equivalently Prob $[(f|_\ell, q_1|_\ell, \ldots, q_m|_\ell) \in \mathrm{acc}(\ell)] = 1$.

Soundness: $\mathbb{E}[\mathrm{agr}((f|_\ell, q_1|_\ell, \ldots, q_m|_\ell), \mathrm{acc}(\ell))] \geq \delta \Rightarrow \mathrm{agr}(f, \mathcal{Z}_d^m) \geq \delta - m\varepsilon - d/|\mathbb{F}|$, where $\varepsilon = \mathrm{poly}(m, d, \frac{1}{|\mathbb{F}|})$.

Lecture 6

Proof of the PCP Theorem (Part II)

Prahladh Harsha

Scribe: Geetha Jagannathan & Aleksandar Nikolov

21 July, 2009

6.1 Recap from Part 1

Recall that we want to construct a robust PCP for the NP-Complete problem. I.e. for every n-sized instance x of the NP-complete problem L we want to construct a proof Π, which can be checked by a verifier using a random string R of length $\log n$ and a constant-size query Q. The verifier computes a local predicate φ of the local view $\Pi|_Q$ and accepts iff $\varphi(\Pi|_Q) = 1$. We want the construction to satisfy the following properties.

Completeness: If $x \in L$ then there exists a proof Π such that

$$\operatorname*{Prob}_R [\varphi(\Pi|_Q) = 1] = 1.$$

Soundness: If $x \notin L$ then for all proofs Π,

$$\mathbb{E}[agr(\Pi|_Q, sat(\varphi))] \leq \delta.$$

Recall further that in Part I we constructed a PCP with the parameters above not for any NP-complete property but for the specific "Zero on a Subcube" property. We say that a function $f : \mathbb{F}^m \to \mathbb{F}$ satisfies the "Zero on Subcube" property iff:

- f is a low-degree polynomial P.
- P vanishes on H^m, where $H \subseteq \mathbb{F}$.

6.2 Robust PCP for CIRCUIT-SAT

In this part of the proof we will show how to use the local test for Zero on a Subcube to construct a PCP for the CIRCUIT-SAT problem.

6.2.1 Problem Definition

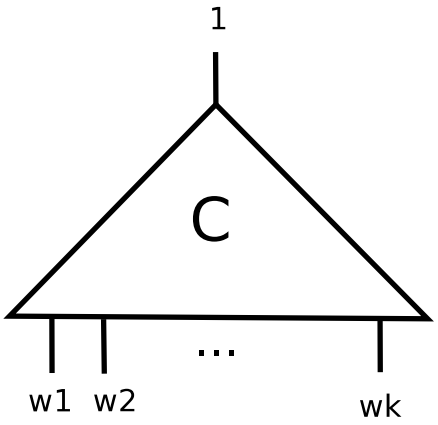


Figure 6.1: CIRCUIT-SAT's input

CIRCUIT-SAT is the following decision problem:

- **Input**: A circuit C with n gates (Figure 6.1); k of them are the input gates $w_1, \ldots, w_k$, and the rest are OR and NOT gates with fan in at most 2 and fanout at most 1. Let's associate variables $z_1, \ldots, z_n$ with each gate (including the input gates). Variable z_i is the output of gate i. The output gate outputs 1.
- **Output**: 1 iff there exists an assignment to $z_1, \ldots, z_n$ that respects the gate functionality, and 0 otherwise.

Note that a proof for this problem is an assignment to $z_1, \ldots, z_n$, and verifying the proof amounts to checking that the assignment respects gate functionality at each gate. To use our local Zero on a Subcube test for CIRCUIT-SAT we need to encode the assignment and the circuit C algebraically, so that an assignment satisfies C iff a related function is a low-degree polynomial that vanishes on a small subcube. Representing the assignment and the circuit algebraically is performed by a process known as arithmetization.

6.2.2 Arithmetization of the Assignment

First we will map an assignment to the gate variables $z_1, \ldots, z_n$ to a low-degree polynomial over an arbitrary field $\mathbb{F}^m$ so that the assignment is encoded by the polynomial.

Let $|H^m| = n$ and choose an arbitrary bijection $H^m \leftrightarrow [n]$. The assignment maps each gate to either 0 or 1, so it is equivalent to a function $A : H^m \to \{0, 1\}$. We choose H so that $\{0, 1\} \subseteq H \subseteq \mathbb{F}$, and we can write $A : H^m \to \mathbb{F}$.

The following (easy-to-prove) algebraic fact will be used in the arithmetization of the circuit.

Fact 6.2.1 (Low-Degree Extension (LDE)). *For any function $S : H^m \to \mathbb{F}$, there exists a polynomial $\hat{S} : \mathbb{F}^m \to \mathbb{F}$ such that $\hat{S}|_{H^m} \equiv S$ and the degree of $\hat{S}$ for each variable is at most $|H|$. Therefore the total degree of $\hat{S}$ is at most $m|H|$.*

Then $A : H^m \to \mathbb{F}$ is mapped by the low-degree extension to a polynomial $\hat{A} : \mathbb{F}^m \to \mathbb{F}$ and the degree of $\hat{A}$ is at most $m|H|$.

6.2.3 Arithmetization of the Circuit

Our goal is to derive a rule from the circuit C which maps any polynomial $\hat{A} : \mathbb{F}^m \to \mathbb{F}$ to a different low-degree polynomial $P_{\hat{A}} : \mathbb{F}^{3m+3} \to \mathbb{F}$, such that $P_{\hat{A}}|_{H^{3m+3}} \equiv 0$ if and only if $\hat{A}$ encodes a satisfying assignment. Note that the existence of such a rule is all we need to construct a PCP for CIRCUIT-SAT, as it reduces verifying a satisfying assignment to testing the Zero on a Subcube property.

We will specify the circuit in a slightly different fashion to enable the arithmetization. Consider a function $\bar{C} : H^{3m} \times H^3 \to \{0,1\}$ that takes three indexes $i_1, i_2, i_3 \in [n] = H^m$ and three bits $b_1, b_2, b_3 \in \{0,1\} \subseteq H$ and outputs a bit as follows based on the functionality of the gate whose input variables are z_{i_1} and z_{i_2} and output variable is z_{i_3}.

$$\bar{C}(i_1, i_2, i_3, b_1, b_2, b_3) = \begin{cases} 1, & \text{iff the assignment } z_{i_1} = \bar{b}_1,\ z_{i_2} = \bar{b}_2\ z_{i_3} = \bar{b}_3, \text{ where } i_1 \text{ and } i_2 \\ & \text{are input values to gate } i_3 \text{ and } i_3 \text{ is the output value is an} \\ & \text{INVALID configuration for the gate } i_3 \\ 0 & \text{otherwise.} \end{cases}$$

Figure 6.2 illustrates the meaning of the arguments of $\bar{C}$.

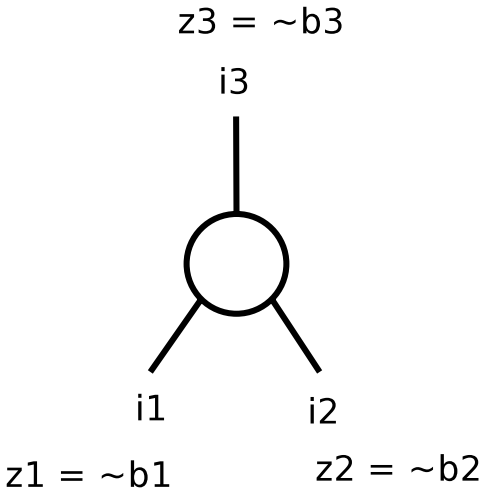


Figure 6.2: Setting of gate variables for $\bar{C}$

Now once again we can use the LDE to map $\bar{C} : H^{3m+3} \to \mathbb{F}$ to a low-degree polynomial $\hat{C} : \mathbb{F}^{3m+3} \to \mathbb{F}$.

We are ready to construct the rule we need. Given any $q : \mathbb{F}^m \to \mathbb{F}$ we define $P_{(q)} : \mathbb{F}^{3m+3} \to \mathbb{F}$ such that

$$P_{(q)}(\underbrace{x_1, \ldots, x_m}_{\mathbf{x}_1}, \underbrace{x_{m+1}, \ldots, x_{2m}}_{\mathbf{x}_2}, \underbrace{x_{2m+1}, \ldots, x_{3m}}_{\mathbf{x}_3}, z_1, z_2, z_3)$$
$$= \hat{C}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, z_1, z_2, z_3)(q(\mathbf{x}_1) - z_1)(q(\mathbf{x}_2) - z_2)(q(\mathbf{x}_3) - z_3).$$

Note that if q is low-degree, $P_{(q)}$ is also low-degree.

The motivation for defining $P_{(q)}$ in this way will become clear when we apply the definition to $\hat{A}$:

$$P_{(\hat{A})}(\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3, b_1, b_2, b_3) = \hat{C}(\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3, b_1, b_2, b_3)(\hat{A}(\mathbf{i}_1) - b_1)(\hat{A}(\mathbf{i}_2) - b_2)(\hat{A}(\mathbf{i}_3) - b_3). \quad (6.2.1)$$

It is now an easy case-analysis to observe the following.

Observation 6.2.2. *$P_{(\hat{A})}|_{H^{3m+3}} \equiv 0 \Leftrightarrow \hat{A}$ is a satisfying assignment.*

6.2.4 The PCP Verifier

Given a circuit C, the PCP proof consists of the oracles $\hat{A} : \mathbb{F}^m \to \mathbb{F}$ and $P_{\hat{A}} : \mathbb{F}^{3m+3} \to \mathbb{F}$.

The PCP verifier needs to make the following checks:

- $\hat{A}$ satisfies the low-degree test
- $P_{\hat{A}}$ satisfies the low-degree test
- $(P_{\hat{A}}, \hat{A})$ satisfies the rule described in (6.2.1).
- $P_{\hat{A}}$ is zero on the subcube H^m.

Given the low-degree test and zero-on-subcube test, it is straightforward to design a PCP that performs the above tests. The PCP verifier expects as proofs the oracles $\hat{A} : \mathbb{F}^m \to \mathbb{F}, P_{\hat{A}} : \mathbb{F}^{3m+3} \to \mathbb{F}, q_1 : \mathbb{F}^{3m+3} \to \mathbb{F}, \ldots, q_{3m+3} : \mathbb{F}^{3m+3} \to \mathbb{F}$. The oracles $q_1, \ldots, q_{3m+3}$ are the auxiliary oracles for performing the zero-on-subcube test. The verifier first picks a random line ℓ in $\mathbb{F}^{3m+3}$. It reads the value of all the oracles along the line ℓ. It checks that the restrictions of all the oracles to the line is low-degree. It then checks that for each point x on the line l, the "zero-on-subcube" test is satisified, namely

$$P_{\hat{A}}(x) = \sum_{i=1}^{3m+3} q_i(x) g_H(x_i).$$

It finally checks for each point on that (6.2.1) is satisified. This completes the description of the PCP verifier.

For want of time, we will skip the analysis of the Robust PCP (see [BGH+06] and [Har04] for details).

Let us now compute the parameters of the PCP verifier. Here $n = H^m$ is the input length. Let us assume $m = \log(n)/\log\log(n)$. We can choose $|\mathbb{F}| = \text{poly}(m|H|)$. The PCP verifier makes $O(|\mathbb{F}|) = \text{poly}\log n$ queries and the amount of randomness used is $O(m \log(|\mathbb{F}|)) = O(\log n)$. The above construction yields a robust PCP of the following form

Theorem 6.2.3. *CIRCUIT-SAT has a robust PCP that uses $O(\log n)$ randomness, makes $\operatorname{poly}\log n$ queries and has $(1/\operatorname{poly}\log n)$ robust soundness parameter.*

Observe that the robust PCP constructed in the above theorem has polylog query complexity and not constant, as we had originally claimed. In the next section, we give a high level outline of how to reduce number of queries (from polylog to constant) using a composition technique originally designed by Arora and Safra [AS98].

6.2.5 PCP Composition

In this section, we describe briefly the PCP composition (due to Arora and Safra [AS98]) that helps in reducing the number of queries made by the verifier form $\operatorname{poly}\log(n)$ to constant without affecting the other parameters too much. Recall that the PCP verifier on input x and an oracle access to a proof φ, tosses some random coins and based on the randomness queries some locations. Denote the query locations as the set I. The PCP verifier evaluates a CIRCUIT-SAT predicate $\varphi(\Pi|_I)$ and accepts or rejects based on the outcome of the predicate. The PCP constructed in the previous section achieves $O(\log(n))$ randomness and $O(\operatorname{poly}\log(n))$ number of queries. We would like to reduce the query complexity from $\operatorname{poly}\log n$ to $O(1)$. How do we do this? How does one check that $\Pi|_I$ satisfies φ without reading all of $\Pi|_I$ and only reading a constant number of locations in $\Pi|_I$. Arora and Safra suggested that use another PCP to recursively perform this check: $\varphi|_I = 1$. Let us denote the original PCP verifier as the *outer* verifier and the one that checks the predicate φ without reading the entire set I as the *inner* verifier. The *inner* verifier gets the circuit φ and $(\Pi|_I)$ as inputs. But if it reads all the input bits then the query complexity is not reduced. Instead, the *inner* gets as input the circuit and an oracle to the proof of $(\Pi|_I)$ and it now needs to check that the proof in this location satisfies φ. This requires some care as a simple recursion will not do the job. A composition in the context of robust PCPs (or equivalently 2-query projective PCPs) was first shown by Moshkovitz and Raz [MR08]. A more generic and simpler composition paradigm for was then shown by Dinur and Harsha [DH09]. For want of time, we will skip the details of the composition and conclude on the note that applying the composition theorem of Dinur and Harsha to the robust PCP constructed in Theorem 6.2.3, one can obtain the constant query PCP with arbitrarily small error as claimed before (see [DH09] for the details of this construction).

Lecture 7

Håstad's 3-Bit PCP

Subhash Khot

Scribe: Dev Desai

21 July, 2009

Previously, we saw the proof of the PCP theorem and its connection to proving inapproximability results. The PCPs that we have seen so far have constant number of queries, but over a large alphabet. Now we are interested in designing useful PCPs while keeping the number of query bits low. The purpose of this lecture is to present such a PCP construction, which leads to optimal inapproximability results for various problems such as MAX-3SAT and MAX-3LIN.

7.1 Introduction

The PCP theorem and Raz's parallel repetition theorem [Raz98] give the NP-hardness of a problem called LABEL COVER (which we will define shortly). This problem is the canonical starting point for reductions that prove inapproximability results. Such reductions can be broadly categorized into two:

Direct reductions. These have been successful in proving inapproximability for network problems, lattice based problems, etc.

Long code based reductions. Salient examples of such results can be found in the paper of Bellare, Goldreich, and Sudan [BGS98] and Håstad [Hås01].

Long code based reductions have been successful for many important problems like MAX-CUT, albeit with a catch: many of these results depend on conjectures like the Unique

Games Conjecture [Kho02], which will be the subject of the next lecture. The focus of this lecture is to prove the powerful result of Håstad's, which can be stated as follows:

Theorem 7.1.1 (Håstad's 3-bit PCP [Hås01])**.** *For every $\varepsilon, \eta > 0$, NP has a PCP verifier that uses $O(\log n)$ random bits, queries exactly 3 bits to the proof, evaluates a linear predicate on these 3 bits, and has completeness $1 - \varepsilon$ and soundness $1/2 + \eta$.*

We can view the bits in this PCP proof as boolean variables. Then the test of the verifier can be interpreted as a system of linear equations, each equation corresponding to the triplet of bits tested for a given random string. Thus, we immediately obtain the hardness for the MAX-3LIN problem (where an instance of MAX-3LIN is a system of linear equations modulo 2 with at most 3 variables per equation, and we are interested in maximizing the fraction of equations that can simultaneously be satisfied).

Corollary 7.1.2. *For every $\varepsilon, \eta > 0$, given an instance of MAX-3LIN, it is NP-hard to tell if $1 - \varepsilon$ fraction of the equations are satisfiable by some assignment or that no assignment satisfies more than $1/2 + \eta$ fraction of the equations.*

In other words, getting an efficient algorithm with approximation guarantee better than $(1/2 + \eta)/(1 - \varepsilon)$, which is $\approx 1/2$ (since we can choose ε and η to be arbitrarily small), for MAX-3LIN is impossible unless P = NP. This result is tight since a random assignment satisfies half of the equations in expectation. Note that the imperfect completeness in the result is essential if P $\neq$ NP, since Gaussian elimination can be used to efficiently check whether any system of linear equations can be completely satisfied.

We now go on to the proof of Theorem 7.1.1. Here, the concepts of Proof composition, Long codes and Fourier analysis play a pivotal role. We will start with a high-level picture of the PCP and work our way down to the actual 3-bit test.

7.2 Proof Composition

The standard method to construct a Long code based PCP is by composing an Outer PCP with an Inner PCP. These two concepts are explained below.

The Outer PCP

The Outer PCP is based on a hard instance of the LABEL COVER problem.

Definition 7.2.1. *A LABEL COVER problem $\mathcal{L}(G(V, W, E), [m], [n], \{\pi_{vw} | (v, w) \in E\})$ consists of:*

1. *A bipartite graph $G(V, W, E)$ with bipartition V, W.*
2. *Every vertex in V is supposed to get a label from a set $[m]$ and every vertex in W is supposed to get a label from a set $[n]$ $(n \geq m)$.*
3. *Every edge $(v, w) \in E$ is associated with a projection $\pi_{vw} : [n] \mapsto [m]$.*

We say that a labeling $\varphi : V \mapsto [m]$, $\varphi : W \mapsto [n]$ satisfies an edge (v, w) if $\pi_{vw}(\varphi(w)) = \varphi(v)$. The goal is to find a labeling that maximizes the number of satisfied edges.

Let us define $\text{opt}(\mathcal{L})$ to be the maximum fraction of edges that are satisfied by any labeling. As mentioned earlier, the hardness of LABEL COVER is obtained by combining the PCP theorem [FGL+96, AS98, ALM+98] with Raz's parallel repetition theorem [Raz98].

Theorem 7.2.2. *For every $\delta > 0$, there exist m and n such that given a* LABEL COVER *instance $\mathcal{L}(G, [m], [n], \{\pi_{vw}\})$, it is* NP*-hard to tell if $\text{opt}(\mathcal{L}) = 1$ or $\text{opt}(\mathcal{L}) \leq \delta$.*

It is useful to think of the above theorem as a PCP that makes 2 queries over the constant (but large) alphabets of size m and n. The verifier just picks an edge at random, queries the labels of the endpoints of this edge and accepts if and only if these labels satisfy the edge. This PCP, which is based on a hard instance of LABEL COVER forms our Outer PCP.

The Inner PCP

The Outer PCP verifier expects some labels as the answers to its two queries. We will *compose* this verifier with an Inner PCP verifier, which expects the proof to contain some encoding (in our case, the Long Code) of the labels, rather than the labels themselves. We choose to have an encoding of the labels so that we can check just a few bits of the proof and tell with a reasonable guarantee whether the labeling is valid.

Thus, the Inner verifier expects large bit strings (supposed to be encodings) for each vertex in G. Let the edge picked by the Outer verifier be (v, w). Then the Inner verifier checks 1 bit from g_v (the supposed encoding of the label of v) and 2 bits from f_w (the supposed encoding of the label of w). Note that the Inner verifier is trying to simulate the Outer PCP. It needs to check the following two things in one shot:

Codeword Test The strings f_w and g_v are correct encodings of *some* $j \in [n]$ and $i \in [m]$.

Consistency Test These i and j satisfy $\pi(j) = i$.

We can therefore convert a PCP which asked 2 large queries to a PCP which asks 3 'bit' queries. We now need to show the following two implications:

1. (*Completeness*) $\text{opt}(\mathcal{L}) = 1 \implies \exists$ Proof $\quad \Pr[\mathsf{acc}] \geq 1 - \varepsilon$.
2. (*Soundness*) $\text{opt}(\mathcal{L}) \leq \delta \implies \forall$ Proofs $\quad \Pr[\mathsf{acc}] < \frac{1}{2} + \eta$.

The completeness follows by the design of the PCP and should be regarded as a sanity check on the construction. The soundness will be proved by contraposition. We will assume that $\Pr[\mathsf{acc}] \geq 1/2 + \eta$ and then decode the labels to satisfy a lot of edges.

7.3 The Long Code and its Test

Håstad's Inner PCP verifier does the codeword test and consistency test in one shot. For clarity, let us first analyze just the codeword test. We will see how to incorporate the consistency test in the next section. For the codeword test, we need to look at the particular encoding that the Inner PCP will use: the Long Code. It is defined below.

Definition 7.3.1. *The Long Code encoding of $j \in [n]$ is defined to be the truth table of the boolean dictatorship function on the jth coordinate, $f : \{\pm 1\}^n \mapsto \{\pm 1\}$ such that $f(x_1, \ldots, x_n) = x_j$.*

Some observations are in order. Note that we are representing bits by $\{\pm 1\}$ and not $\{0, 1\}$. This is done just for the sake of clarity, since the calculations done with $\{\pm 1\}$ are less messy. Also note that the Long Code is huge. An element $j \in [n]$ will require $\log n$ bits to represent, but the Long Code of j requires 2^n bits, a doubly-exponential blowup! We can get away with this because n, the alphabet size, is a constant.

Now for a short aside on Fourier analysis. Recall that for each $S \subseteq [n]$, the Fourier character $\chi_S : \{\pm 1\}^n \mapsto \{\pm 1\}$ is defined as

$$\chi_S(x) = \prod_{i \in S} x_i.$$

These characters form an *orthonormal basis* for the set of Boolean functions $f : \{\pm 1\}^n \mapsto \{\pm 1\}$. All such functions can therefore be written in terms of this basis, called the *Fourier expansion*, as

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \chi_S(x).$$

where $\widehat{f}(S)$ is called the *Fourier coefficient* of set S. For Boolean functions, these coefficients satisfy Parseval's identity, namely $\sum_{S \subseteq [n]} \widehat{f}(S)^2 = 1$.

Back to the Long Code. In terms of Fourier expansion, the Long Code of element j is the same as the function $\chi_{\{j\}}$. Thus it is simple to write down, since the only non-zero coefficient is $\widehat{f}(\{j\}) = 1$. The Long Code then fits into a general class of functions which have *high Fourier coefficients of low order*. This fact will be useful in the soundness analysis of the test.

On to the codeword test. This will essentially be a linearity test, that is, we will check whether

$$f(x + y) = f(x) + f(y).$$

Since we are in the $\{\pm 1\}$ domain, this linearity translates to checking whether

$$f(xy) = f(x)f(y).$$

We will also introduce a small randomized perturbation in the linearity test. This is done to improve the overall soundness of the test.

Definition 7.3.2. *An ε-perturbation vector is a string of ± 1 bits, where each bit is independently set to -1 with probability ε and 1 with probability $1 - \varepsilon$.*

The final codeword test is described below. It is a randomized 3-bit linear test that checks whether the input function is close to a Long Code.

Long Code Test.

Input: Function $f : \{\pm 1\}^n \mapsto \{\pm 1\}$ and error parameter ε.

Test: Pick $x, y \in \{\pm 1\}^n$ at random. Pick an ε-perturbation vector $\mu \in \{\pm 1\}^n$ and let $z = xy\mu$. Accept if and only if

$$f(z) = f(x)f(y).$$

Let us analyze the completeness and soundness of this test. We have the following theorem.

Theorem 7.3.3. *Given a truth table of a function $f : \{\pm 1\}^n \mapsto \{\pm 1\}$ and $\varepsilon > 0$, the following are true for the Long Code Test:*

1. *If $f = \chi_{\{j\}}$ for some j, then $\Pr[\mathsf{acc}] = 1 - \varepsilon$.*
2. *If $\Pr[\mathsf{acc}] \geq 1/2 + \eta$, then f "resembles" a Long Code in the following sense: $\exists\, S \subseteq [n]$ such that $|\widehat{f}(S)| \geq \eta$ and $|S| \leq O((1/\varepsilon)\log(1/\eta))$, in other words, there exists a large Fourier coefficient of low order.*

Proof. To prove the completeness part, assume that $f = \chi_{\{j\}}$ for some $j \in [n]$, that is, it is a Long Code. Then the test will "Accept" if and only if

$$z_j = x_j y_j \iff x_j y_j \mu_j = x_j y_j \iff \mu_j = 1$$

which happens with probability $1 - \varepsilon$.

For the soundness analysis, assume that $\Pr[\mathsf{acc}] \geq 1/2 + \eta$. We can write this probability in terms of the test as

$$\Pr[\mathsf{acc}] = \mathbb{E}_{x,y,\mu}\left[\frac{1 + f(z)f(x)f(y)}{2}\right].$$

This is a standard PCP trick that is used often in analyzing such tests. Substituting for $\Pr[\mathsf{acc}]$ and using the Fourier expansion of f,

$$\begin{aligned}
\frac{1}{2} + \eta &\leq \frac{1}{2} + \frac{1}{2}\,\mathbb{E}_{x,y,\mu}[f(xy\mu)f(x)f(y)] \\
2\eta &\leq \mathbb{E}_{x,y,\mu}\left[\left(\sum_{S\subseteq[n]} \widehat{f}(S)\chi_S(xy\mu)\right)\left(\sum_{T\subseteq[n]} \widehat{f}(T)\chi_T(x)\right)\left(\sum_{U\subseteq[n]} \widehat{f}(U)\chi_U(y)\right)\right] \\
&= \sum_{S,T,U} \widehat{f}(S)\widehat{f}(T)\widehat{f}(U)\,\mathbb{E}_{x,y,\mu}\left[\chi_S(xy\mu)\chi_T(x)\chi_U(y)\right] \\
&= \sum_{S,T,U} \widehat{f}(S)\widehat{f}(T)\widehat{f}(U)\,\mathbb{E}_{x,y,\mu}\left[\chi_S(x)\chi_S(y)\chi_S(\mu)\chi_T(x)\chi_U(y)\right] \\
&= \sum_{S,T,U} \widehat{f}(S)\widehat{f}(T)\widehat{f}(U)\,\mathbb{E}_{x}\left[\chi_S(x)\chi_T(x)\right]\mathbb{E}_{y}\left[\chi_S(y)\chi_U(y)\right]\mathbb{E}_{\mu}\left[\chi_S(\mu)\right].
\end{aligned}$$

We can simplify the last expression by using orthonormality of the χ's to argue that

$$\mathbb{E}_x[\chi_S(x)\chi_T(x)] = \mathbb{E}_x\left[\prod_{i\in S} x_i \prod_{j\in T} x_j\right] = \prod_{i\in S\Delta T} \mathbb{E}_x[x_i] = \begin{cases} 1 & \text{if } S = T, \\ 0 & \text{otherwise} \end{cases}$$

where $S\Delta T$ stands for the symmetric difference between sets S and T. Thus the terms in the summation will vanish unless $S = T = U$. We then get

$$2\eta \leq \sum_S \widehat{f}(S)^3\,\mathbb{E}_\mu[\chi_S(\mu)]. \tag{7.3.1}$$

As an aside, the Long Code Test without perturbations was analyzed long before Håstad by Blum, Luby and Rubinfeld [BLR93]. In that case, we just get

$$\sum_S \widehat{f}(S)^3 \geq 2\eta \implies |\widehat{f}_{\max}| \sum_S \widehat{f}(S)^2 \geq 2\eta \implies \exists \text{ large } |\widehat{f}| \quad (\text{since } \sum \widehat{f}^2 = 1).$$

Now, with perturbation, we have

$$\mathbb{E}_\mu \left[\chi_S(\mu)\right] = \mathbb{E}_\mu \left[\prod_{i \in S} \mu_i\right] = [1(1-\varepsilon) + (-1)\varepsilon]^{|S|} = (1-2\varepsilon)^{|S|}.$$

Substituting this in Inequality (7.3.1), we get

$$2\eta \leq \sum_S \widehat{f}(S)^3 (1-2\varepsilon)^{|S|}.$$

We can again think of the above sum as a convex combination (since $\sum \widehat{f}^2 = 1$). This implies that there exists an S such that $\widehat{f}(S)(1-2\varepsilon)^{|S|} \geq 2\eta$. Thus we have

$$\Pr[\mathsf{acc}] \geq \frac{1}{2} + \eta \implies \exists\, S : \quad |\widehat{f}(S)| \geq 2\eta \quad \text{and} \quad |S| \leq O\left(\frac{1}{\varepsilon} \log \frac{1}{\eta}\right). \qquad \square$$

The Long Code test can be thought of as an analog of the concept of *gadget* in NP reductions. In particular, Theorem 7.3.3 is very important and is the crux of the PCP. We will prove similar results in the analysis of later tests.

7.4 Incorporating Consistency

Let us restate what we want from our 3-bit test. Recall that the inputs to the Inner PCP verifier are two supposed Long codes, g and f, of two vertices v and w, and the projection π between them. We have to check two things in one shot:

1. g and f are Long codes of some $i \in [m]$ and $j \in [n]$.
2. $\pi(j) = i$.

We are going to do this by reading 1 bit from g and 2 bits from f and applying a 3-bit linear test similar to the Long code test.

Consistency Test.

Input: Functions $g : \{\pm 1\}^m \mapsto \{\pm 1\}$ and $f : \{\pm 1\}^n \mapsto \{\pm 1\}$, projection $\pi : [n] \mapsto [m]$, and error parameter ε.

Test: Pick $x \in \{\pm 1\}^m$, $y \in \{\pm 1\}^n$ at random. Pick an ε-perturbation vector $\mu : \{\pm 1\}^n$. Let $z = (x \circ \pi) y \mu$. Accept if and only if

$$f(z) = g(x) f(y).$$

In the above test, the vector $(x \circ \pi)$ is defined as $(x \circ \pi)_j = x_{\pi(j)} \quad \forall\ 1 \le j \le n$. Such a definition is needed because x and y are vectors of different sizes.

The analysis of the above test is similar to that of the Long Code test. Hence, we will skip a rigorous proof and state only the important details. The completeness is simple to analyze.

For the soundness analysis, we can imagine a restricted case where $n = m$, $\pi = \mathsf{id}$ (identity permutation), and $f = g$. Then the test is exactly the 3-bit Long Code test that we saw in the previous section and we get

$$\Pr[\mathsf{acc}] \ge \frac{1}{2}+\eta \xrightarrow{\text{Theorem 7.3.3}} \sum_{\substack{S\subseteq[n]\\ |S|\le O(\frac{1}{\varepsilon}\log\frac{1}{\eta})}} \widehat{f}(S)^3 \ge \eta \xrightarrow{\text{Cauchy-Schwarz}} \sum_{\substack{S\subseteq[n]\\ |S|\le O(\frac{1}{\varepsilon}\log\frac{1}{\eta})}} \widehat{f}(S)^4 \ge \eta^2.$$

Now if we have different f and g, one can verify by analysis similar to that in Theorem 7.3.3 that instead of the above inequality, we would get

$$\sum_{\substack{S\subseteq[n]\\ |S|\le O(\frac{1}{\varepsilon}\log\frac{1}{\eta})}} \widehat{g}(S)^2\widehat{f}(S)^2 \ge \eta^2.$$

Further, now if $\pi \neq \mathsf{id}$ and $n \neq m$, then we would end up with the inequality in the following theorem.

Theorem 7.4.1. *The following are true for Consistency Test(g, f, π, ε):*

1. *If $f = \chi_{\{j\}}$, $g = \chi_{\{i\}}$ and $\pi(j) = i$, then $\Pr[\mathsf{acc}] = 1 - \varepsilon$.*
2. *If $\Pr[\mathsf{acc}] \ge 1/2 + \eta$, then f and g are correlated in the following sense:*

$$\sum_{\substack{S\subseteq[m],T\subseteq[n]\\ |S|,|T|\le O(1/\varepsilon\log(1/\eta))\\ S,T\text{ correlated by }\pi}} \widehat{g}(S)^2\widehat{f}(T)^2 \ge \eta^2$$

where "S, T correlated by π" means that there exist $i \in S$, $j \in T$ such that $\pi(j) = i$.

We are now ready to describe Håstad's composed PCP verifier.

Håstad's 3-bit PCP.

Input: Hard instance of LABEL COVER, $\mathcal{L}(G(V, W, E), [m], [n], \{\pi_{vw}\})$ *(given by Theorem 7.2.2) and error parameter ε.*

Verifier: Pick edge $(v, w) \in E$ at random. Let g_v and f_w be the supposed Long codes of the two vertices. Run Consistency Test$(g_v, f_w, \pi_{vw}, \varepsilon)$.

The following theorem gives the completeness and soundness of the verifier.

Theorem 7.4.2. *Given a hard instance of* LABEL COVER, $\mathcal{L}$, *Håstad's PCP guarantees*

1. *(Completeness) If $\mathrm{opt}(\mathcal{L}) = 1$, then there exists a proof for which $\Pr[\mathsf{acc}] \ge 1 - \varepsilon$.*

2. *(Soundness) If* $\Pr[\mathsf{acc}] \geq 1/2 + 2\eta$*, then* $\mathrm{opt}(\mathcal{L}) \geq \varepsilon^2\eta^3/\log^2(1/\eta)$*, that is, there is a labeling to* $\mathcal{L}$ *that satisfies at least* $\varepsilon^2\eta^3/\log^2(1/\eta)$ *fraction of edges.*

Proof. For the completeness part, we can assume that the proof contains correct encodings on correct labels. Then by Theorem 7.4.1, the Verifier accepts with probability $1-\varepsilon$ on every choice of edge. Therefore, the overall acceptance probability also remains $1-\varepsilon$.

Now for the soundness part. If $\Pr[\mathsf{acc}] \geq 1/2 + 2\eta$, then by an averaging argument, for at least η fraction of the edges $(v,w) \in E$, Consistency Test$(g_v, f_w, \pi_{vw}, \varepsilon)$ accepts with probability at least $1/2 + \eta$.

Fix any such *good* edge. Then by Theorem 7.4.1, we have

$$\sum_{\substack{|S|,|T| \leq O(1/\varepsilon \log(1/\eta)) \\ \exists\ i \in S, j \in T\ :\ \pi(j)=i}} \widehat{g}_v(S)^2 \widehat{f}_w(T)^2 \geq \eta^2. \tag{7.4.1}$$

We will now define labels for vertices v and w. First, we pick sets $S \subseteq [m]$ and $T \subseteq [n]$ with probability $\widehat{g}_v(S)^2$ and $\widehat{f}_w(T)^2$ respectively. Next, we pick labels $i \in S$ and $j \in T$ at random. This is a randomized labeling and by the probabilistic method, the argument goes through. In expectation, at least

$$\underbrace{\eta}_{\substack{\Pr[\text{Pick} \\ \text{good edge}]}} \cdot \underbrace{\eta^2}_{\substack{\Pr[\text{Pick correlated } S,T \\ \text{by Inequality 7.4.1}]}} \cdot \underbrace{\frac{\varepsilon^2}{\log^2(1/\eta)}}_{\substack{\Pr[\text{Pick } i \in S, j \in T \\ \text{s.t. } \pi(j)=i]}}$$

fraction of label-cover edges are satisfied. ∎

Finally, observe that if we set $\delta = c\varepsilon^2\eta^3/\log^2(1/\eta)$, then Theorem 7.2.2 and Theorem 7.4.2 together prove Theorem 7.1.1.

7.5 Concluding Remarks

In this lecture, we have seen and analyzed Håstad's powerful 3-bit PCP. There are a few subtleties in the construction of this PCP that are worth mentioning. The first subtlety, that we have already seen in the Introduction, is that we have to sacrifice perfect completeness if we want our Verifier to have linear predicates.

The second issue is that any linear test can be satisfied if everything is 0 (or +1 in our case). Moreover, in the soundness analysis of Theorem 7.4.2, the sets S and T that are chosen by the probability distributions $\{\widehat{g}_v(S)^2\}$ and $\{\widehat{f}_w(T)^2\}$ respectively should not be empty. This problem is taken care by an operation called *folding*. For more information, the reader can look at the references below.

Some good references for more information on Håstad's PCP are Chapter 22 in Arora and Barak's textbook [AB09], Khot's article on Long code based PCPs [Kho05] and Håstad's original paper [Hås01].

Lecture 8

Semidefinite Programming and Unique Games

Moses Charikar

Scribe: Alantha Newman

21 July, 2009

8.1 Unique Games

The topic of Unique Games has generated much interest in the past few years. The *Unique Games Conjecture* was posed by Khot [Kho02]. We will discuss the associated optimization problem and the algorithmic intuition and insight into the conjecture, as well as the limits of these algorithmic techniques. Finally, we mention the amazing consequences implied for many optimization problems if the problem is really as hard as conjectured.

We now define the Unique Games problem. The input is a set of variables V and a set of k labels, L, where k is the size of the *domain*. Our goal is to compute a mapping, $\ell : V \rightarrow L$, satisfying certain constraints that we now describe. Let E denote a set of pairs of variables, $\{(u, v)\} \subset V \times V$. For each $(u, v) \in E$, there is an associated constraint represented by π_{uv}, indicating that $\ell(v)$ should be equal to $\pi_{uv}(\ell(u))$; we assume that the constraint π_{vu} is the inverse of the constraint π_{vu} i.e, $\pi_{uv} = \pi_{vu}^{-1}$. Thus, our goal is to compute the aforementioned mapping, $\ell : V \rightarrow L$, so as to *maximize the number of satisfied constraints*.

Each constraint, π_{uv}, can be viewed as a permutation on L. Note that this permutation may be different for each pair $(u, v) \in E$. For a pair $(u, v) \in E$, if v is given a particular label from L, say $\ell(v)$, then there is only one label for u that will satisfy the constraint π_{uv}. Specifically, $\ell(u)$ should equal $\pi_{vu}(\ell(v))$. Hence, the "unique" in Unique Games. The practice of calling this optimization problem a unique "game" stems from the connection of this problem to 2-prover 1-round games [FL92]. The Unique Games problem is a special case of Label Cover (discussed in other lectures in the workshop), in which each constraint

forms a bijection from L to L. Having such a bijection turns out to be useful for hardness results.

8.2 Examples

We will refer to E as a set of edges, since we can view an instance of Unique Games as a graph $G = (V, E)$ in which each edge $(u, v) \in E$ is labeled with a constraint π_{uv}. We now give some specific examples of optimization problems that are special cases of Unique Games.

8.2.1 Linear Equations Mod p

We are given a set of equations in the form $x_i - x_j \equiv c_{ij} \pmod{p}$. The goal is to assign each variable in $V = \{x_i\}$ a label from the set $L = [0, 1, \ldots p-1]$ so as to maximize the number of satisfied equations. Note that each constraint is a bijection.

8.2.2 MAXCUT

Given an undirected graph $G = (V, E)$, the Max Cut problem is to find a bipartition of the vertices that maximizes the weight of the edges with endpoints on opposite sides of the partition.

We can represent this problem as a special case of Linear Equations mod p and therefore as a special case of Unique Games. For each edge $(i, j) \in E$, we write the equation $x_i - x_j \equiv 1 \pmod{2}$. Note that the domain size is two, since there are two possible labels, 0 and 1.

8.3 Satisfiable vs Almost Satisfiable Instances

If an instance of Unique Games is satisfiable, it is easy to find an assignment that satisfies all of the constraints. Can you see why? Essentially, the uniqueness property says that if you know the correct label of one variable, then you know the labels of all the neighboring variables. So we can just guess all possible labels for a variable; at some point your guess is correct and this propagates correct labels to all neighbors, and to their neighbors, and so on. This is a generalization of saying that if a graph is bipartite (e.g. all equations in the Max Cut problem are simultaneously satisfiable), then such a bipartition can be found efficiently. So when all constraints in an instance of Unique Games are satisfiable, this is an "easy" problem.

In contrast, the following problem has been conjectured to be "hard": If 99% of the constraints are satisfiable, can we satisfy 1% of the constraints? The precise form of the conjecture is known as the Unique Games Conjecture [Kho02]: For all small constants $\varepsilon, \delta > 0$, given an instance of Unique Games where $1 - \varepsilon$ of the constraints are satisfied, it is hard to satisfy a δ fraction of satisfiable constraints, for some $k > f(\varepsilon, \delta)$, where k is the size of the domain and f is some function of ε and δ.

How does f grow as a function of ε and δ? We claim that $f(\varepsilon, \delta) > 1/\delta$. This is because it can easily be shown that we can satisfy a $1/k$ fraction of the constraints: Randomly assigning a label to each variable achieves this guarantee. Thus, in words, the conjecture is

that for a sufficiently large domain size, it is hard to distinguish between almost satisfiable and close to unsatisfiable instances.

8.3.1 Almost Satisfiable Instances of MAXCUT

We can also consider the Max Cut problem from the viewpoint of distinguishing between almost satisfiable and close to unsatisfiable instances. However, for this problem, a conjecture as strong as that stated above for general Unique Games is clearly false. This is because we can always satisfy at least half of the equations. (See Sanjeev's lecture.) We now consider the problem of satisfying the maximum number of constraints given that a $(1-\varepsilon)$ fraction of the constraints are satisfiable. We write the standard semidefinite programming (SDP) relaxation in which each vertex u (with a slight abuse of notation) is represented by a unit vector, u.

$$\begin{aligned} & \max & \sum_{(u,v)\in E} \frac{1-u\cdot v}{2} \\ u\cdot u & = & 1 \quad \forall u \in V \\ u & \in & \mathbb{R}^n \quad \forall u \in V. \end{aligned}$$

For a fixed instance of the Max Cut problem, let OPT denote the fraction of constraints satisfied by an optimal solution, and let OPT_{SDP} denote the value of the objective function of the above SDP on this instance. If $OPT \geq (1-\varepsilon)|E|$, then $OPT_{SDP} \geq (1-\varepsilon)|E|$, since $OPT_{SDP} \geq OPT$. In Lecture 1 (Sanjeev's lecture), it was shown that using the random hyperplane rounding of Goemans-Williamson [GW95], we can obtain a .878-approximation algorithm for this problem. We will now try to analyze this algorithm for the case when OPT is large, e.g. at least $(1-\varepsilon)|E|$. From a solution to the above SDP, we obtain a collection of n-dimensional unit vectors, where $n = |V|$. We choose a random hyperplane, represented by a vector $r \in N(0,1)^n$ (i.e. each coordinate is chosen according to the normal distribution with mean 0 and variance 1). Each vector $u \in V$ has either a positive or a negative dot product with the vector r, i.e $r \cdot u > 0$ or $r \cdot u < 0$. Let us now analyze what guarantee we can obtain for the algorithm in terms of ε.

As previously stated, we have the following inequality for the SDP objective function:

$$\sum_{(u,v)\in E} \frac{(1-u\cdot v)}{2} \quad \geq \quad (1-\varepsilon)|E|.$$

Let θ'_{uv} represent the angle between vectors u and v, i.e. $\arccos(u\cdot v)$. Let θ_{uv} denote the angle $(\pi - \theta'_{uv})$. Then we can rewrite the objective function of the SDP as:

$$\sum_{(u,v)\in E} \frac{1+\cos(\theta_{uv})}{2}.$$

Further rewriting of the objective function results in the following:

$$\begin{aligned}
\sum_{(u,v)\in E} \frac{1+\cos(\theta_{uv})}{2} &= \sum_{(u,v)\in E} 1 - \frac{1-\cos(\theta_{uv})}{2} \\
&= |E| - \sum_{(u,v)\in E} \frac{1-\cos(\theta_{uv})}{2} \\
&= |E| - \sum_{(u,v)\in E} \sin^2\left(\frac{\theta_{uv}}{2}\right) \\
&\geq |E| - \varepsilon|E|.
\end{aligned}$$

We say that vertices u and v are "cut" if they fall on opposite sides of the bipartition after rounding.

$$\Pr[u \text{ and } v \text{ cut}] = \frac{\theta'_{uv}}{\pi} = 1 - \frac{\theta_{uv}}{\pi}.$$

The expected size of S—the number of edges cut in a solution—is:

$$\begin{aligned}
\mathrm{E}[S] &= \sum_{(u,v)\in E} 1 - \frac{\theta_{uv}}{\pi} \\
&= |E| - \sum_{(u,v)\in E} \frac{\theta_{uv}}{\pi}.
\end{aligned}$$

Assume for all $(u,v) \in E$ that $\sin^2\left(\frac{\theta_{uv}}{2}\right) = \varepsilon$. Then $\sin\left(\frac{\theta_{uv}}{2}\right) = \sqrt{\varepsilon}$. For small θ, we have that $\sin(\theta) \approx \theta$. Therefore, $\theta_{uv}/2 \approx \sqrt{\varepsilon}$.

Thus, the expected value $\mathrm{E}[S] \geq |E|(1 - c\sqrt{\varepsilon})$ for some constant c. In other words, if we are given a Max Cut instance with objective value $(1-\varepsilon)|E|$, we can find a solution of size $(1 - c\sqrt{\varepsilon})|E|$. In other words, an almost satisfiable instance can be given an almost satisfying assignment, although the assignment has a weaker guarantee.

8.4 General Unique Games

What happens for a large domain? How do we write an SDP for this problem? Before we had just one vector per vertex. Now for each variable, we have k values. So we have a vector for each variable and for each value that it can be assigned. First, we will write a $\{0,1\}$ integer program for Unique Games and then we relax this to obtain an SDP relaxation.

8.4.1 Integer Program for Unique Games

Recall that L is a set of k labels. For each variable u and each label $i \in L$, let u_i be an indicator variable that is 1 if u is assigned label i and 0 otherwise. Note that the expression in the objective function is 1 exactly when a constraint π_{uv} is satisfied.

$$\begin{aligned}
&\max \quad \sum_{(u,v)\in E} \sum_{i\in L} u_i \cdot v_{\pi_{uv}(i)} \\
\sum_{i\in L} u_i &= 1 \quad \forall u \in V.
\end{aligned}$$

Now we move to a vector program. The objective function stays the same, but we can add some more equalities and inequalities to the relaxation that are valid for an integer program. Below, we write quadratic constraints since our goal is ultimately to obtain a quadratic program.

$$\begin{aligned} \sum_{i \in L} u_i \cdot u_i &= 1 \quad \forall u \in V,\ i \in L, \\ u_i \cdot u_j &= 0 \quad \forall u \in V,\ i \neq j \in L. \end{aligned}$$

Additionally, we can also add triangle-inequality constraints on triples of vectors, $\{u_i, v_j, w_h\}$ for $u, v, w \in V$ and $i, j, h \in L$:

$$||u_i - w_h||^2 \leq ||u_i - v_j||^2 + ||v_j - w_h||^2, \tag{8.4.1}$$

$$||u_i - v_j||^2 \geq ||u_i||^2 - ||v_j||^2. \tag{8.4.2}$$

These constraints are easy to verify for 0/1 variables, i.e. for integer solutions. Note that these constraints are not necessary for the integer program, but they make the SDP relaxation stronger.

8.4.2 Trevisan's Algorithm

We now look at an algorithm due to Trevisan [Tre08]. Recall that if we know that every constraint in a given instance is satisfiable, then we can just propagate the labels and obtain a satisfiable assignment. The algorithm that we discuss is roughly based on this idea.

How can we use a solution to the SDP relaxation to obtain a solution that satisfies many constraints? Suppose that OPT is $|E|$ and consider two vertices u and v connected by an edge. In this case, the set of k vectors corresponding to u is the same constellation of k vectors corresponding to vertex v, possibly with a different labeling. If OPT is $(1-\varepsilon)|E|$, then although these two constellations may no longer be identical, they should be "close". The correlation of the vectors corresponds to the distance, i.e. high correlation corresponds to small distance. Thus, we want to show that the vector corresponding to the label of the root vertex r is "close" to other vectors, indicating which labels to assign the other vertices.

An Algorithm for Simplified Instances

Consider the following "simplified instance". Recall that the constraint graph consists of a vertex for each variable and has an edge between two variables if there is a constraint between these two variables. Suppose the constraint graph has radius d: there exists a vertex r such that every variable is a distance at most d from vertex r. The following lemma can be proved using the ideas discussed above.

Lemma 8.4.1. *If every edge contributes $1 - \varepsilon/8(d+1)$ to the SDP objective value, then it is possible to efficiently find an assignment satisfying a $(1-\varepsilon)$-fraction of the constraints.*

We now give the steps of the rounding algorithm.

Rounding the SDP

(i) Find root vertex, r, such that every other vertex is reachable from r by a path of length at most d.

(ii) Assign label i to r with probability $||r_i||^2$.

(iii) For each $u \in V$, assign u label j, where j is the label that minimizes the quantity $||u_j - r_i||^2$.

As mentioned earlier, the intuition for this label assignment is that u_j is the vector that is "closest" to r_i. We now prove the following key claim: For each edge (u, v), the probability that constraint π_{uv} is satisfied is at least $1 - \varepsilon$. In particular, recall that edge (u, v) is mislabeled if $\ell(v) \neq \pi_{uv}(\ell(u))$. Thus, we want to show that the probability that edge (u, v) is mislabeled is at most ε.

Since r is at most a distance d from all other vertices, a BFS tree with root r has the property that each u has a path to r on the tree of distance at most d. Fix a BFS tree and consider the path from r to u: $r = u^0, u^1, u^2, \ldots, u^{t-1}, u^t = u$, where $t \leq d$. Let π_{u^1} denote the permutation π_{u^0,u^1}, and recursively define π_{u^k} as the composition of permutations $(\pi_{u^k,u^{k-1}}) \cdot (\pi_{u^{k-1}})$. Let $\pi_v = (\pi_{uv}) \cdot (\pi_u)$. We now compute the probability that vertex u is assigned label $\pi_{u(i)}$ and that vertex v is assigned label $\pi_{v(i)}$, given that r is assigned label i. Note that if both these assignments occur, then edge (u, v) is satisfied. (Since edge (u, v) may also be satisfied with another assignment, we can think of our calculation as possibly being an *underestimate* on the probability that edge (u, v) is satisfied.)

Let $A(u)$ denote the label assigned to vertex u by the rounding algorithm. We will show:

$$\Pr[A(u) = \pi_u(i)] \;\geq\; 1 - \frac{\varepsilon}{2} \quad \text{and} \quad \Pr[A(v) = \pi_v(i)] \;\geq\; 1 - \frac{\varepsilon}{2}.$$

This implies that the probability that constraint π_{uv} is satisfied is at least $1 - \varepsilon$. Now we compute the probability that $A(u) \neq \pi_u(i)$. Suppose that u_j for $j \neq \pi_u(i)$ is closer to vector r_i than $u_{\pi_u(i)}$ is. In other words, suppose:

$$||u_j - r_i||^2 \;\leq\; ||u_{\pi_u(i)} - r_i||^2. \tag{8.4.3}$$

Let B_u be the set of labels such that if r is assigned label $i \in B_u$, then u is not assigned label $\pi_u(i)$. Note that label j belongs to B_u iff inequality (8.4.3) holds for j. Thus, the probability that u is not labeled with $\pi_u(i)$ is exactly:

$$\Pr[A(u) \neq \pi_u(i)] \;=\; \sum_{i \in B_u} ||r_i||^2.$$

One can verify that if there is some label j such that inequality (8.4.3) holds, then the quantity $||r_i||^2$ is at most $2||r_i - u_{\pi_u(i)}||^2$. This proof makes use of inequalities from the SDP, (8.4.1) and (8.4.2), as well as inequality (8.4.3). (See Lemma 8.6.1 from [Tre08], which we include in the Appendix.) Recall that each edge in the graph (and thus each edge on the path from r to u in the BFS tree) contributes at most $1 - \varepsilon/8(d + 1)$ to the

objective value. By triangle inequality, this implies that $\sum_{i \in L} ||r_i - u_{\pi_u(i)}||^2 \leq \varepsilon/4$. Thus, we conclude:

$$\begin{aligned}
\Pr[A(u) \neq \pi_u(i)] &= \sum_{i \in B_u} ||r_i||^2 \\
&\leq 2 \sum_{i \in B_u} ||r_i - u_{\pi_u(i)}||^2 \\
&\leq 2 \sum_{i \in L} ||r_i - u_{\pi_u(i)}||^2 \\
&\leq \frac{\varepsilon}{2}.
\end{aligned}$$

Similarly, we conclude that $\Pr[A(v) \neq \pi_v(i)] \leq \varepsilon/2$, which implies that the probability that constraint π_{uv} is not satisfied is at most ε.

Shift Invariant Instances

In the case of Linear Equations mod p, we can add more constraints to the SDP relaxation, which allow for a simplified analysis of the rounding algorithm. For any assignment of labels, we can shift each of the labels by the same fixed amount, i.e, by adding a value $k \in L$ to each label, and obtain an assignment with the same objective value. This property of a solution has been referred to as *shift invariance*. In these instances, the following are valid constraints. Note that $p = |L|$.

$$\begin{aligned}
||u_i||^2 &= \frac{1}{p} \quad u \in V,\ i \in L, \\
u_i \cdot v_j &= u_{i+k} \cdot v_{j+k} \quad u, v \in V,\ i, j, k \in L.
\end{aligned}$$

In this case, we obtain a stronger version of Lemma 8.4.1.

Lemma 8.4.2. *In a shift invariant instance in which every edge contributes more than $1 - 1/2(d+1)$ to the SDP objective value, it is possible to efficiently find an assignment that satisfies all of the constraints.*

We will show that in this case, the vector r_i is closer to vector $u_{\pi_u(i)}$ than to vector u_j for any label $j \neq \pi_u(i)$. In other words, $r_i \cdot u_{\pi_u(i)} > r_i \cdot u_j$ for all $j \in L$. If each edge contributes more than $1 - 1/2(d+1)$ to the objective value, then $||r_i - u_{\pi_u(i)}||^2 < 1/p$. This implies that $r_i \cdot u_{\pi_u(i)} > 1/2p$. By triangle inequality, we have:

$$\begin{aligned}
||u_j - u_{\pi_u(i)}||^2 &\leq ||u_j - r_i||^2 + ||r_i - u_{\pi_u(i)}||^2 \\
\frac{2}{p} &\leq \frac{2}{p} - 2 r_i \cdot u_j + \frac{1}{p} \quad \Rightarrow \\
r_i \cdot u_j &\leq \frac{1}{2p}.
\end{aligned}$$

Assuming that vector u_j is closer to r_i than vector $u_{\pi_u(i)}$, we obtain the following contradiction:

$$\frac{1}{2p} < r_i \cdot u_{\pi_u(i)} \leq r_i \cdot u_j \leq \frac{1}{2p}.$$

Note that in the case of shift invariance, r is assigned each label from L with equal probability. Because of shift invariance, it does not actually matter which label r is assigned. Thus, we can just assign r a label i arbitrarily (we no longer need randomization) and then proceed with the rest of the SDP rounding algorithm.

Extension to General Instances

Applying this SDP rounding to general graphs may not yield such good results as in Lemmas 8.4.1 and 8.4.2, since the radius of an arbitrary graph can be large, and the objective values of the SDP relaxation would therefore have to be very high for the lemmas to be applicable. In order to apply these lemmas, we break the graph into pieces, each with a radius of no more than $O(\log n/\varepsilon)$. Doing this requires throwing out no more than an ε-fraction of the constraints. The following lemma is originally due to Leighton and Rao [LR99] and can also be found in [Tre08].

Lemma 8.4.3. *For a given graph $G = (V, E)$ and for all $\varepsilon > 0$, there is a polynomial time algorithm to find a subset of edges $E' \subseteq E$ such that $|E'| > (1-\varepsilon)|E|$, and every connected connected component of E' has diameter $O(\log |E|/\varepsilon)$.*

Using this lemma, we obtain the following guarantee for general instances: Given an instance for which OPT is at least $(1 - c\varepsilon^3/\log n)|E|$, we can efficiently find a labeling satisfying a $1-\varepsilon$ fraction of the constraints. Note that c is an absolute constant. For shift invariant instances, we can satisfy $(1-\varepsilon)|E|$ of the constraints for an instance where OPT is at least $(1 - c\varepsilon^2/\log n)|E|$.

Given a graph, we remove the $\frac{\varepsilon}{3}$ fraction of constraints that contribute the least to the objective value. This leaves us with at least $(1-\varepsilon/3)|E|$ constraints that each contributes at least $1-3c\varepsilon^2/\log n$ (or $1-3c\varepsilon/\log n$ for shift invariant instances) to the objective value. We can apply Lemma 8.4.1 (or Lemma 8.4.2) with $d = \log n/\varepsilon$, satisfying at least $(1-2\varepsilon/3)|E|$ constraints (or $(1-\varepsilon/3)|E|$ constraints).

8.5 Improving the Approximation Ratio

Algorithms with improved approximation guarantees for Unique Games have been presented in [GT06a, CMM06]. The latter work gives an algorithm with the following guarantee: Given an instance of Unique Games with a domain size k for which OPT is at least $(1-\varepsilon)|E|$, the algorithm produces a solution that satisfies at least $\max\{1-\sqrt{\varepsilon \log k},\ k^{-\varepsilon/(2-\varepsilon)}\}$ fraction of the constraints. Furthermore, it has been shown that the existence of an efficient algorithm that can distinguish between instances in which $(1-\varepsilon)|E|$ constraints can be satisfied and those at which less than $k^{-\varepsilon/2}$ constraints can be satisfiable, would disprove the Unique Games Conjecture [KKMO07]. Moreover, it is sufficient to refute the conjecture if this algorithm works only for the special case of Linear Equations mod p. Thus, focusing on shift invariant instances is a reasonable approach.

Additionally, the Unique Games problem has been studied for cases in which the constraint graph is an expander; in an instance in which OPT is at least $(1-\varepsilon)|E|$, one can efficiently find a solution satisfying at least $1 - O(\frac{\varepsilon}{\lambda})$ fraction of the constraints, where λ is a function of the expansion of the graph [AKK+08, MM09].

8.6 Consequences

The interest in the Unique Games Conjecture has grown due to the many strong, negative consequences that have been proved for various optimization problems. Assuming the Unique Games Conjecture, it has been shown that the Goemans-Williamson algorithm for Max Cut (presented in Sanjeev's lecture) achieves the optimal approximation ratio [KKMO07]. More surprisingly, there are many other NP-complete optimization problems for which the best-known approximation guarantees are obtained via extremely simple algorithms. Nevertheless, no one has been able to find algorithms with improved approximation guarantees, even when resorting to sophisticated techniques such as linear and semidefinite programming. Such optimization problems include the Minimum Vertex Cover problem and the Maximum Acyclic Subgraph problem, for which the best-known approximation factors are 1/2 and 2, respectively. If the Unique Games Conjecture is true, then these approximation ratios are tight [KR08, GMR08]. This phenomena has been investigated for several other optimization problems as well. A recent result shows that for a whole class of constraint satisfaction problems, which can be modeled using a particular integer program, the integrality gap of a particular SDP relaxation is exactly equal to its approximability threshold under the Unique Games Conjecture [Rag08].

Appendix

We include the following lemma from [Tre08] and its proof:

Lemma 8.6.1. *Let* $\mathbf{r}, \mathbf{u}, \mathbf{v}$ *be vectors such that: (i)* $\mathbf{u} \cdot \mathbf{v} = 0$*, (ii)* $||\mathbf{r} - \mathbf{u}||^2 \geq ||\mathbf{r} - \mathbf{v}||^2$*, and (iii) the vectors* $\mathbf{r}, \mathbf{u}, \mathbf{v}$ *satisfy the triangle inequality constraints from the SDP. Then* $||\mathbf{r} - \mathbf{u}||^2 \geq \frac{1}{2}||\mathbf{r}||^2$.

Proof. There are three cases:

1. If $||\mathbf{u}||^2 \leq \frac{1}{2}||\mathbf{r}||^2$, then by (8.4.2), we have:

$$||\mathbf{r} - \mathbf{u}||^2 \geq ||\mathbf{r}||^2 - ||\mathbf{u}||^2 \geq \frac{1}{2}||\mathbf{r}||^2.$$

2. If $||\mathbf{v}||^2 \leq \frac{1}{2}||\mathbf{r}||^2$, then by (8.4.1), and subsequently (8.4.2), we have:

$$||\mathbf{r} - \mathbf{u}||^2 \geq ||\mathbf{r} - \mathbf{v}||^2 \geq ||\mathbf{r}||^2 - ||\mathbf{v}||^2 \geq \frac{1}{2}||\mathbf{r}||^2.$$

3. If $||\mathbf{u}||^2, ||\mathbf{v}||^2 \geq \frac{1}{2}||\mathbf{r}||^2$, then from (8.4.1) and assumption (ii), we have:

$$||\mathbf{v} - \mathbf{u}||^2 \leq ||\mathbf{v} - \mathbf{r}||^2 + ||\mathbf{r} - \mathbf{u}||^2 \leq 2||\mathbf{r} - \mathbf{u}||^2.$$

   By Pythagoras theorem and by orthogonality of $\mathbf{u}$ and $\mathbf{v}$ (assumption (i)), we have:

$$||\mathbf{v} - \mathbf{u}||^2 = ||\mathbf{v}||^2 + ||\mathbf{u}||^2.$$

Finally, we have:

$$||\mathbf{r} - \mathbf{u}||^2 \;\geq\; \frac{1}{2}||\mathbf{v} - \mathbf{u}||^2 \;=\; \frac{1}{2}||\mathbf{v}||^2 + \frac{1}{2}||\mathbf{u}||^2 \;\geq\; \frac{1}{2}||\mathbf{r}||^2.$$

□

Lecture 9

Unique Games Hardness for MAXCUT

Subhash Khot

Scribe: Igor Gorodezky

21 July, 2009

9.1 Introduction: MAXCUT and Unique Games

In this lecture we sketch the proof of one of the more remarkable consequences of the Unique Games Conjecture: MAX-CUT is inapproximable to any constant better than α, where

$$\alpha = \min_{-1 \leq \rho \leq 1} \frac{2}{\pi} \frac{\arccos(\rho)}{1-\rho} \approx .87856 \tag{9.1.1}$$

is the approximation ratio of the Goemans-Williamson algorithm.

9.1.1 The Goemans-Williamson algorithm

Recall that given a graph $G = (V, E, w)$ with edge weights $w_{ij} \geq 0$, the MAX-CUT problem asks for $S \subseteq V$ that maximizes $\sum_{i \in S, j \notin S} w_{ij}$ (we call this the *weight* of the cut induced by S). We will write $\mathrm{mc}(G)$ for the maximum weight of a cut in G.

MAX-CUT is NP-hard. The best known approximation algorithm for MAX-CUT is due to Goemans and Williamson [GW95] and is as follows: given $G = (V, E, w)$, first solve the MAX-CUT SDP

$$\begin{aligned} \max \sum_{ij} w_{ij} \frac{1 - v_i \cdot v_j}{2} & \\ ||v_i||^2 = 1, & \qquad i = 1, \ldots, n \\ v_i \in \mathbb{R}^n, & \qquad i = 1, \ldots, n \end{aligned} \tag{9.1.2}$$

to get a set of unit vectors $v_1, \ldots, v_n$ in $\mathbb{R}^n$. Then, uniformly sample a hyperplane through the origin and define $S \subseteq V$ to be the set of i such that v_i lies "above" this hyperplane.

We saw in Sanjeev's lecture (Lecture 1) that the Goemans-Williamson algorithm gives, in expectation, an $(\alpha - \varepsilon)$-approximation for any $\varepsilon > 0$ (this additive error of ε stems from the fact that semidefinite programs must be solved to within some fixed, arbitrarily small accuracy). The algorithm can be derandomized (see [MH99]) to yield a deterministic $(\alpha - \varepsilon)$-approximation to MAX-CUT.

A series of subsequent hardness results culminated in Håstad's PCP-based proof in [Hås01] that MAX-CUT is NP-hard to approximate to within $16/17 \approx .941$. Then, roughly a decade after the publication of the Goemans-Williamson algorithm, Khot, Kindler, Mossel and O'Donnell proved in [KKMO07] that, assuming the Unique Games Conjecture, it is NP-hard to approximate MAX-CUT to within any factor greater than α. This suggests, as Khot et al. note, that the geometric nature of the Goemans-Williamson algorithm is intrinsic to the MAX-CUT problem.

9.1.2 Label Cover and Unique Games

The inapproximability of MAX-CUT is conditional on the Unique Games Conjecture, which we state in this section.

A *unique game* $\mathcal{L}$ is a bipartite graph with left-side vertex set V, right-side vertex set W, edge set E, and a set of labels of size M. Each edge (v, w) has an associated constraint function $\pi_{v,w}$ which is a permutation of $[M]$ (i.e. a bijection from the set of labels to itself). We will sometimes refer to a unique game with these parameters in the longhand $\mathcal{L}(V, W, E, [M], \{\pi_{v,w}\})$.

A *labeling* of a unique game $\mathcal{L}$ is an assignment of a label from $[M]$ to each vertex of $\mathcal{L}$. A labeling *satisfies* the edge (v, w) if $\pi_{v,w}$ maps the label of w to the label of v. We define

$$\text{opt}(\mathcal{L}) = \max\{\ r \mid \text{there exists a labeling of } \mathcal{L} \text{ that satisfies } r|E| \text{ edges}\}.$$

The *unique Label Cover problem with parameter* δ is the problem of deciding, given a unique game $\mathcal{L}(V, W, E, [M], \{\pi_{v,w}\})$, whether $\text{opt}(\mathcal{L}) \geq 1 - \delta$ or $\text{opt}(\mathcal{L}) \leq \delta$. That is, given $\mathcal{L}$, we are asked to decide whether there exists a labeling that satisfies nearly all edge constraints, or whether no labeling can satisfy more than a tiny fraction of them. We will write this decision problem as $\text{ULC}(\delta)$.

Intuition tells us that computing $\text{opt}(\mathcal{L})$ should be a hard problem, but what about $\text{ULC}(\delta)$? That is, if we are asked not to compute $\text{opt}(\mathcal{L})$ but simply to decide whether it is very large or very small, does the problem become easier? The Unique Games Conjecture claims that the answer is no.

The following is (a slightly weakened form of) the conjecture, first stated by Khot in [Kho02].

Conjecture 9.1.1 (Unique Games Conjecture, [Kho02])**.** *For any $\delta > 0$ there exists a constant M such that it is NP-hard to decide ULC(δ) on instances with a label set of size M.*

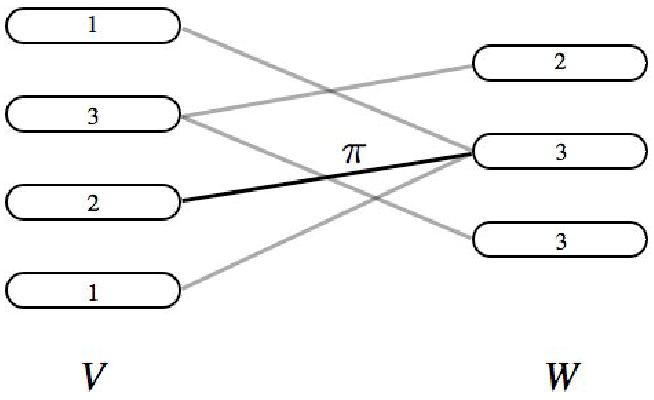


Figure 9.1: A unique game with $M = 3$. The constraint π is associated with the highlighted edge. The edge is satisfied by the labeling in the figure if $\pi(3) = 2$.

9.1.3 The Main Result

We are ready to present the main result of [KKMO07]. This result can be stated in the form of a PCP for the ULC(δ) problem, by which we mean a probabilistically checkable proof system such that given a unique game $\mathcal{L}$ and a proof, if $\mathrm{opt}(\mathcal{L}) \geq 1 - \delta$ then the verifier accepts with high probability c (completeness), while if $\mathrm{opt}(\mathcal{L}) \leq \delta$ then the verifier accepts with low probability s (soundness). As usual, the verifier only uses $O(\log n)$ random bits on an instance of size n.

Theorem 9.1.2. *For every $\rho \in (-1, 0)$ and $\varepsilon > 0$ there exists $\delta > 0$ such that there is a PCP for ULC(δ) in which the verifier reads two bits from the proof and accepts iff they are unequal, and which has completeness*

$$c \geq \frac{1-\rho}{2} - \varepsilon$$

and soundness

$$s \leq \frac{1}{\pi} \arccos(\rho) + \varepsilon.$$

Before sketching the proof of this theorem in Section 9.3, let us see how it implies the inapproximability of MAX-CUT. Recall that $\mathrm{mc}(G)$ is the maximum weight of a cut in G.

Corollary 9.1.3. *For every $\rho \in (-1, 0)$ and $\varepsilon > 0$ there exists $\delta > 0$ and a polynomial-time reduction from an instance $\mathcal{L}$ of ULC(δ) to an instance $G = (V, E, w)$ of MAX-CUT such that*

$$\begin{aligned} \mathrm{opt}(\mathcal{L}) \geq 1 - \delta \quad &\Longrightarrow \quad \mathrm{mc}(G) \geq \frac{1-\rho}{2} - \varepsilon \\ \mathrm{opt}(\mathcal{L}) \leq \delta \quad &\Longrightarrow \quad \mathrm{mc}(G) \leq \frac{1}{\pi} \arccos(\rho) + \varepsilon. \end{aligned}$$

Proof. Given ρ and ε, let δ be the same as in Theorem 9.1.2. Given an instance $\mathcal{L}$ of ULC(δ), consider the PCP given by that theorem. Define the graph G to have the bits of

the proof as vertices,[1] and create an edge between two bits if there is a non-zero probability of that pair of bits being sampled by the verifier. Finally, set w to be the trivial weight function that is 1 on all edges.

Observe that a proof, which is an assignment of a value in $\{-1,1\}$ to the bits, corresponds to a cut in G, and the number of edges crossing this cut is precisely the probability that this proof is accepted by the verifier. The claim now follows from the completeness and soundness of the PCP. □

Assuming the Unique Games Conjecture, for any $\delta > 0$ there is some constant M such that it is NP-hard to decide ULC(δ) on instances with a label set of size M. Now, by standard arguments, Corollary 9.1.3 implies that it is NP-hard to approximate MAX-CUT to within

$$\frac{\arccos(\rho)/\pi + \varepsilon}{(1-\rho)/2 - \varepsilon} > \frac{\arccos(\rho)/\pi}{(1-\rho)/2}$$

for any $\rho \in (-1,0)$. Therefore, MAX-CUT is hard to approximate to within any constant larger than

$$\min_{-1\le\rho\le 0} \frac{2}{\pi}\frac{\arccos(\rho)}{1-\rho} = \min_{-1\le\rho\le 1} \frac{2}{\pi}\frac{\arccos(\rho)}{1-\rho} = \alpha$$

which is the promised inapproximability result.

Let us turn our attention, then, to proving Theorem 9.1.2. The proof will rely on a highly nontrivial result in boolean Fourier analysis that we state in the next section.

9.2 Majority is Stablest

The proof of Theorem 9.1.2 makes crucial use of the Majority is Stablest (MIS) theorem, which is an extremal result in boolean Fourier analysis. In this section we state this theorem after defining the necessary concepts.

We will use the common convention that bits take value in $\{-1,1\}$ rather than $\{0,1\}$ (in particular, we identify $x \in \{0,1\}$ with $y \in \{-1,1\}$ using the bijection $y = (-1)^x$). Thus, a *boolean function* is a map $f : \{-1,1\}^n \to \{-1,1\}$. In what follows we will assume familiarity with the basic concepts of boolean Fourier analysis, as we lack the space for a thorough review of the subject; a reader seeking such a review is directed to the survey [O'D08].

We begin with several definitions. Given $f : \{-1,1\}^n \to \{-1,1\}$, define the *influence* of x_i on f to be the probability over all n-bit strings that f changes value when the ith bit is flipped:

$$\mathrm{Inf}_i(f) = \operatorname*{Prob}_{x\in\{-1,1\}^n}\left[f(x) \neq f(x_1,\ldots,x_{i-1},-x_i,x_{i+1},\ldots,x_n)\right].$$

It is not hard to show that

$$\mathrm{Inf}_i(f) = \sum_{S|i\in S} \hat{f}(S)^2 \tag{9.2.1}$$

[1] As we will later see, if $\mathcal{L}$ is of the form $\mathcal{L}(V,W,E,[M],\{\pi_{v,w}\})$ then there will be $|W|2^M$ vertices in G.

and indeed, equation (9.2.1) can be used to *define* the influence of a variable on a non-boolean function $f : \{-1,1\}^n \to \mathbb{R}$.

Given a bit string x and some $\rho \in (-1,1)$, let us define a distribution over $y \in \{-1,1\}^n$ by setting

$$y_i = \begin{cases} x_i \text{ with probability } \frac{1+\rho}{2} \\ -x_i \text{ with probability } \frac{1-\rho}{2} \end{cases}$$

We write $y \sim_\rho x$ to mean a y sampled from such a distribution. Now, given a boolean function f and $\rho \in (-1,1)$, define the *noise sensitivity* of f at rate ρ to be

$$\mathrm{NS}_\rho(f) = \operatorname*{Prob}_{x,\, y \sim_\rho x} [f(x) \neq f(y)]$$

where x is sampled uniformly and $y \sim_\rho x$. It can be shown that

$$\mathrm{NS}_\rho(f) = \frac{1}{2} - \frac{1}{2} \sum_S \hat{f}(S)^2 \rho^{|S|}. \tag{9.2.2}$$

As before, equation (9.2.2) serves as the *definition* of noise sensitivity for non-boolean functions.

We observe that if f is a *dictator function*, i.e. $f(x_1, \ldots, x_n) = x_i$ for some i, then $\mathrm{NS}_\rho(f)$ is exactly $(1-\rho)/2$ since $\hat{f}(S) = 0$ when $S \neq \{x_i\}$ and is 1 otherwise. If f is the majority function (the boolean function whose value on a bit string is equal to the value of the majority of the bits), then it can be shown (see [KKMO07] for references) that

$$\mathrm{NS}_\rho(f) = \frac{1}{\pi} \arccos(\rho) + o(1).$$

Observe that when $\rho \in [0,1)$, the noise sensitivity of a dictator function is lower than that of the majority function. The MIS theorem (proven in [MOO05]) tells us that if we disqualify dictators by restricting our attention to functions in which no coordinate has large influence, then the majority function achieves the smallest possible noise sensitivity.

Theorem 9.2.1 (Majority is Stablest, [MOO05])**.** *For every $\rho \in [0,1), \varepsilon > 0$ there exists δ such that if $f : \{-1,1\}^n \to [-1,1]$ with $\mathbb{E}[f] = 0$ and $\mathrm{Inf}_i(f) \leq \delta \ \forall i$, then*

$$\mathrm{NS}_\rho(f) \geq \frac{1}{\pi} \arccos \rho - \varepsilon.$$

Note the additional requirement that $\mathbb{E}[f] = 0$; such functions are called *balanced*. Our application requires the following corollary from [KKMO07]. It states that if we choose a negative rather than positive ρ, the majority function becomes the *least* stable.

Corollary 9.2.2. *For every $\rho \in (-1,0), \varepsilon > 0$ there exists δ such that if $f : \{-1,1\}^n \to [-1,1]$ with $\mathrm{Inf}_i(f) \leq \delta \ \forall i$, then*

$$\mathrm{NS}_\rho(f) \leq \frac{1}{\pi} \arccos \rho + \varepsilon.$$

Observe that f is no longer required to be balanced.

Looking ahead to the proof of Theorem 9.1.2, the test that the verifier will perform on the PCP in the theorem will be, in a way, a noise-sensitivity test on a boolean function. Thus, we will be able to bound the soundness of this test by appealing to the MIS theorem.

9.3 Proving Theorem 9.1.2

In this section we sketch the construction of the PCP whose existence is claimed in Theorem 9.1.2. Recall that the PCP is for the problem ULC(δ), so the verifier is given a unique game $\mathcal{L}(V, W, E, [M], \{\pi_{v,w}\})$ and a proof that is accepted with high probability if $\text{opt}(\mathcal{L}) \geq 1 - \delta$ and accepted with low probability if $\text{opt}(\mathcal{L}) \leq \delta$. The PCP will be parameterized by $\rho \in (-1, 0)$ and $\varepsilon > 0$.

It follows from the results of [KR08] that given $\mathcal{L}$, we may assume with no loss of generality that all $v \in V$ have the same degree. Thus, uniformly sampling $v \in V$ and then uniformly sampling a neighbor $w \in W$ of v yields a uniformly random edge (v, w). Therefore, if we define a proof to be a labeling of $\mathcal{L}$ that maximizes the proportion of satisfied edge constraints, and define the verifier's test to be uniformly sampling (v, w) and checking if this labeling satisfies $\pi_{v,w}$, then we would have a proof with completeness $1 - \delta$ and soundness δ. However, such a test involves sampling $\Omega(\log M)$ bits, and we require a test that samples only 2.

We therefore look for a way to encode elements of the label set $[M]$ in a way that will allow such a test. To this end, we will encode labels using the Long Code, which we first saw in Subhash's lecture (Lecture 7) on Håstad's 3-bit PCP.

9.3.1 Motivation: the Long Code

Recall that in the Long Code, the codeword encoding $i \in [M]$ is the truth-table for the dictator function $f(x_1, \ldots, x_n) = x_i$. In our PCP, the proof will be a labeling of $\mathcal{L}$ with each label encoded using the Long Code. It remains to design a test for the verifier with the properties specified in Theorem 9.1.2. Before explicitly stating the test in the next section, we use this section to motivate its construction.

Given a boolean function f, let us say that f is *far from a dictator* if all coordinates have negligible influence. It is not hard to design a 2-bit test that, given a truth-table for a function f, accepts with high probability if f is a dictator and with low probability if f is far from a dictator. The test required for our PCP must clearly do more than this, but for the moment let us warm up with this simpler problem.

Consider the following noise-sensitivity test: sample $x \in \{-1, 1\}^n$ uniformly, sample $y \sim_\rho x$ as in Section 9.2, and accept iff $f(x) \neq f(y)$. By definition, the probability of accepting is $\text{NS}_\rho(f)$, the noise sensitivity of f.

The completeness of this test is thus exactly the noise-sensitivity of a dictator function, which is $(1 - \rho)/2$. On the other hand, we can use Corollary 9.2.2 to bound the soundness: if f is far from a dictator its noise-sensitivity is at most $\arccos(\rho)/\pi + \varepsilon$.

Returning to our PCP, we require a 2-bit noise-sensitivity test with (almost) exactly these parameters that instead of testing whether a boolean function is a dictator or far from it, tests whether an encoded labeling of $\mathcal{L}$ (which consists of many boolean functions) satisfies many edge constraints, or far from it.

9.3.2 The Test

In this section we describe our PCP's verifier test. First, some notation. For $x \in \{-1, 1\}^M$ and a bijection $\pi : [M] \to [M]$, let $x \circ \pi$ denote the string $(x_{\pi(1)}, \ldots, x_{\pi(M)})$. Given a unique

game $\mathcal{L}(V, W, E, [M], \{\pi_{v,w}\})$ and the associated PCP proof (which the verifier expects is the Long Code encoding of a labeling), let f_v be the Long Code encoding of the label given to $v \in V$, and define f_w for $w \in W$ analogously.

The 2-bit verifier test:

1. Given $\mathcal{L}(V, W, E, [M], \{\pi_{v,w}\})$ and a proof, sample $v \in V$ uniformly, then sample two of its neighbors $w, w' \in W$ uniformly and independently.
2. Let $\pi = \pi_{v,w}$ and $\pi' = \pi_{v,w'}$ be the constraints for edges (v, w) and (v, w').
3. Sample $x \in \{-1, 1\}^M$ uniformly and sample $y \sim_\rho x$.
4. Accept if $f_w(x \circ \pi) \neq f_{w'}(y \circ \pi')$.

Completeness. Assume that $\text{opt}(\mathcal{L}) \geq 1 - \delta$ and that the proof given to the verifier encodes all labels correctly (i.e. as dictator functions). Let the labels of v, w, w' be $i, j, j' \in [M]$, respectively. With probability at least $1 - 2\delta$, both (v, w) and (v, w') are satisfied by the labeling, which implies $\pi(j) = \pi'(j') = i$. Conditioning on this event, we have

$$f_w(x \circ \pi) = x_{\pi(j)} = x_i \quad \text{and} \quad f_{w'}(y \circ \pi') = y_{\pi'(j')} = y_i.$$

Since $x_i = y_i$ with probability $(1 - \rho)/2$, the test accepts with the same probability. We conclude that the completeness is at least $(1 - 2\delta)(1 - \rho)/2$. Tweaking our choice of ρ, we conclude that completeness is at least $(1 - \rho)/2 - \varepsilon$, as desired.

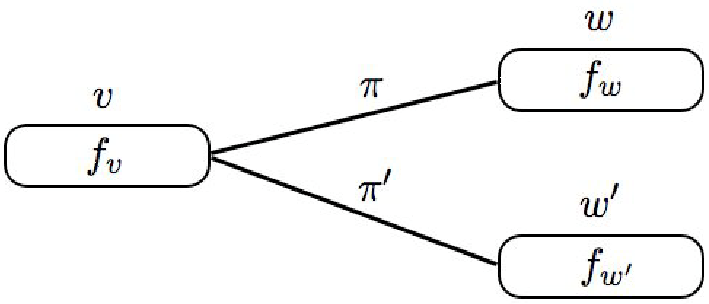


Figure 9.2: The 2-bit verifier test samples $v \in V$, two neighbors $w, w' \in W$, then compares f_w and $f_{w'}$ on certain inputs.

Soundness. As usual, bounding the soundness is the difficult part; we only sketch the argument. The proof is in the contrapositive direction: assuming that the test accepts with probability greater than $\arccos(\rho)/\pi + \varepsilon$, we prove the existence of a labeling that satisfies many edges by exploiting the resulting Fourier-analytic properties of the boolean functions encoded in the proof (which is where Corollary 9.2.2 comes in).

The proof is as follows. Given $v \in V$, let p_v be the probability that the test accepts after choosing v from V. Then

$$\begin{aligned} p_v &= \mathop{\mathbb{E}}_{w,w',x,y\sim_\rho x}\left[\frac{1 - f_w(x\circ\pi)f_{w'}(y\circ\pi')}{2}\right] \\ &= \frac{1}{2} - \frac{1}{2}\mathop{\mathbb{E}}_{w,w'}\left[\mathop{\mathbb{E}}_{x,y\sim_\rho x}\left[f_w(x\circ\pi)f_{w'}(y\circ\pi')\right]\right]. \end{aligned}$$

Standard Fourier-analytic arguments can be used to show that

$$\mathop{\mathbb{E}}_{x,y\sim_\rho x}\left[f_w(x\circ\pi)f_{w'}(y\circ\pi')\right] = \sum_S \hat{f}_w(S)\hat{f}_{w'}(S)\rho^{|S|}$$

from which it follows that

$$\begin{aligned} p_v &= \frac{1}{2} - \frac{1}{2}\mathop{\mathbb{E}}_{w,w'}\left[\sum_S \hat{f}_w(S)\hat{f}_{w'}(S)\rho^{|S|}\right] \\ &= \frac{1}{2} - \frac{1}{2}\sum_S \mathop{\mathbb{E}}_{w,w'}\left[\hat{f}_w(S)\hat{f}_{w'}(S)\right]\rho^{|S|} \\ &= \frac{1}{2} - \frac{1}{2}\sum_S \mathop{\mathbb{E}}_{w\sim v}\left[\hat{f}_w(S)\right]\mathop{\mathbb{E}}_{w'\sim v}\left[\hat{f}_{w'}(S)\right]\rho^{|S|} \end{aligned}$$

where the last equality follows from the independence of w and w' (and $w \sim v$ means that w is a neighbor of v). If we define a function $g_v : \{-1,1\}^n \to [-1,1]$ by

$$g_v(z) = \mathop{\mathbb{E}}_{w\sim v}\left[f_w(z\circ\pi_{v,w})\right]$$

then it is not hard to show that

$$\hat{g}_v(S) = \mathop{\mathbb{E}}_{w\sim v}\left[\hat{f}_w(S)\right].$$

Therefore, returning to p_v, we have

$$\begin{aligned} p_v &= \frac{1}{2} - \frac{1}{2}\sum_S \mathop{\mathbb{E}}_{w\sim v}\left[\hat{f}_w(S)\right]\mathop{\mathbb{E}}_{w'\sim v}\left[\hat{f}_{w'}(S)\right]\rho^{|S|} \\ &= \frac{1}{2} - \frac{1}{2}\sum_S \hat{g}_v(S)^2\rho^{|S|} \\ &= \mathrm{NS}_\rho(g_v) \end{aligned}$$

where the last equality is by equation (9.2.2).

Recall that we assumed that the test is accepted with probability at least $\arccos(\rho)/\pi+\varepsilon$. Standard averaging arguments tell us that $p_v \geq \arccos(\rho)/\pi+\varepsilon/2$ for at least an $\varepsilon/2$ fraction of $v \in V$. By the above, we have

$$\mathrm{NS}_\rho(g_v) \geq \arccos(\rho)/\pi + \varepsilon/2$$

for such v. Now we conclude by Corollary 9.2.2 (having tweaked ε as necessary) that for such a v, g_v has an influential coordinate. This fact can be used to show that for a constant fraction of neighbors w of v, f_w has a small set of influential coordinates. These various influential coordinates can be used to define a labeling that satisfies a large fraction of constraints; we direct the reader to [KKMO07] for the gory details.

A final caveat: technically, if we want to prove that f_w has a small set of influential coordinates for a constant fraction of neighbors w of v, it is not enough to assume that g_v has a coordinate with large influence. What is required is for g_v to have a coordinate with large *low-degree* influence, which is defined, in analogy to equation (9.2.1), as

$$\mathrm{Inf}_i^{\leq k}(f) = \sum_{S \,|\, i \in S, |S| \leq k} \hat{f}(S)^2$$

for some constant k. If we define low-order noise sensitivity in obvious analogy to equation (9.2.2), it is possible to prove low-order analogues of the Majority is Stablest theorem and Corollary 9.2.2, which can then be used to formalize the argument that we have sketched.

This completes the description of the PCP test and the proof of Theorem 9.1.2.

9.4 The Big Picture

The past few years have seen a flurry of powerful inapproximability results conditional on the Unique Games Conjecture. In [Rag08], Raghavendra exhibits a canonical semidefinite programming relaxation of an arbitrary CSP whose integrality gap, assuming UGC, is precisely equal to the best possible approximation ratio for that CSP. In FOCS 2009, Raghavendra and Steurer presented an efficient rounding scheme for these SDPs that achieves the integrality gap.

UGC has been used to prove that Vertex Cover is conditionally inapproximable to within $2-\varepsilon$ (see [KR08]). This proof utilizes a theorem on the influence of boolean functions due to Friedgut. In addition, it was independetly shown by Khot and Vishnoi [KV05] and Chawla et al. [CKK+06] that UGC implies the hardness of approximating Sparsest Cut to within any constant. These proofs, as expected, use theorems on the influence of boolean functions due to Kahn-Kalai-Linial and Bourgain.

Underlying these results are surprising and fruitful connections between unique game reductions, semidefinite programming relaxations of CSPs, extremal problems in Fourier analysis, and isoperimetric problems in geometry. The reader is directed to Section 5 of [KKMO07] for an insightful high-level discussion of how these connections are manifested in the particular case of MAX-CUT.

Bibliography

[AA97] Baruch Awerbuch and Yossi Azar. *Buy-at-bulk network design.* In *Proc. 38rd IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 542–547. IEEE, 1997. doi:10.1109/SFCS.1997.646143.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[ACG+07] Matthew Andrews, Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, Kunal Talwar, and Lisa Zhang. *Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs.* Technical Report TR07-113, Electronic Colloquium on Computational Complexity, 2007. eccc:TR07-113.

[AKK+08] Sanjeev Arora, Subhash Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K. Vishnoi. *Unique games on expanding constraint graphs are easy: extended abstract.* In *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, pages 21–28. ACM, 2008. doi:10.1145/1374376.1374380.

[AKR95] Ajit Agrawal, Philip N. Klein, and R. Ravi. *When trees collide: An approximation algorithm for the generalized steiner problem on networks.* SIAM J. Computing, 24(3):440–456, 1995. (Preliminary version in *23rd STOC*, 1991). doi:10.1137/S0097539792236237.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. *Proof verification and the hardness of approximation problems.* J. ACM, 45(3):501–555, May 1998. (Preliminary Version in *33rd FOCS*, 1992). eccc:TR98-008, doi:10.1145/278298.278306.

[And04] Matthew Andrews. *Hardness of buy-at-bulk network design.* In *Proc. 45th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 115–124. IEEE, 2004. doi:10.1109/FOCS.2004.32.

[AR06] Yossi Azar and Oded Regev. *Combinatorial algorithms for the unsplittable flow problem.* Algorithmica, 44(1):49–66, 2006. (Preliminary version in *IPCO*, 2001). doi:10.1007/s00453-005-1172-z.

[Aro03] Sanjeev Arora. *Approximation schemes for NP-hard geometric optimization problems: a survey.* Mathematical Programming, 97(1–2):43–69, 2003. doi:10.1007/s10107-003-0438-y.

[AS98] Sanjeev Arora and Shmuel Safra. *Probabilistic checking of proofs: A new characterization of NP.* J. ACM, 45(1):70–122, January 1998. (Preliminary Version in *33rd FOCS*, 1992). doi:10.1145/273865.273901.

[AS03] Sanjeev Arora and Madhu Sudan. *Improved low-degree testing and its applications.* Combinatorica, 23(3):365–426, 2003. (Preliminary Version in *29th STOC*, 1997). eccc:TR97-003, doi:10.1007/s00493-003-0025-0.

[AZ06] Matthew Andrews and Lisa Zhang. *Logarithmic hardness of the undirected edge-disjoint paths problem.* J. ACM, 53(5):745–761, 2006. (Preliminary version in *37th STOC*, 2005). doi:10.1145/1183907.1183910.

[AZ07] ———. *Hardness of the undirected congestion minimization problem.* SIAM J. Computing, 37(1):112–131, 2007. (Preliminary version in *37th STOC*, 2005). doi:10.1137/050636899.

[AZ08] ———. *Almost-tight hardness of directed congestion minimization.* J. ACM, 55(6), 2008. (Preliminary version in *38th STOC*, 2006). doi:10.1145/1455248.1455251.

[AZ09] ———. *Complexity of wavelength assignment in optical network optimization.* IEEE/ACM Trans. Netw., 17(2):646–657, 2009. (Preliminary version in *25th INFOCOM*, 2006). doi:10.1145/1552193.1552215.

[Bar98] Yair Bartal. *On approximating arbitrary metrices by tree metrics.* In *Proc. 30th ACM Symp. on Theory of Computing (STOC)*, pages 161–168. ACM, 1998. doi:10.1145/276698.276725.

[BGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. *Robust PCPs of proximity, shorter PCPs and applications to coding.* SIAM J. Computing, 36(4):889–974, 2006. (Preliminary Version in *36th STOC*, 2004). eccc:TR04-021, doi:10.1137/S0097539705446810.

[BGS98] Mihir Bellare, Oded Goldreich, and Madhu Sudan. *Free bits, PCPs, and nonapproximability—towards tight results.* SIAM J. Computing, 27(3):804–915, June 1998. (Preliminary Version in *36th FOCS*, 1995). eccc:TR95-024, doi:10.1137/S0097539796302531.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. *Self-testing/correcting with applications to numerical problems.* J. Computer and System Sciences, 47(3):549–595, December 1993. (Preliminary Version in *22nd STOC*, 1990). doi:10.1016/0022-0000(93)90044-W.

[BS00] Alok Baveja and Aravind Srinivasan. *Approximation algorithms for disjoint paths and related routing and packing problems.* Math. Oper. Res., 25(2):255–280, 2000. doi:10.1287/moor.25.2.255.12228.

[CGKT07] Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, and Kunal Talwar. *Hardness of routing with congestion in directed graphs.* In *Proc. 39th ACM Symp. on Theory of Computing (STOC)*, pages 165–178. ACM, 2007. doi:10.1145/1250790.1250816.

[CHKS06] Chandra Chekuri, Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R. Salavatipour. *Approximation algorithms for non-uniform buy-at-bulk network design.* In *Proc. 47th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 677–686. IEEE, 2006. doi:10.1109/FOCS.2006.15.

[Chr76] Nicos Christofides. *Worst-case analysis of a new heuristic for the travelling salesman problem.* Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.

[CK06] Julia Chuzhoy and Sanjeev Khanna. *Hardness of directed routing with congestion.* Technical Report TR06-109, Electronic Colloquium on Computational Complexity, 2006. eccc:TR06-109.

[CKK+06] Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. *On the hardness of approximating multicut and sparsest-cut.* Computational Complexity, 15(2):94–114, 2006. (Preliminary version in *20th IEEE Conference on Computational Complexity*, 2005). doi:10.1007/s00037-006-0210-9.

[CKS06] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. *An $o(sqrt(n))$ approximation and integrality gap for disjoint paths and unsplittable flow.* Theory of Computing, 2(1):137–146, 2006. doi:10.4086/toc.2006.v002a007.

[CMM06] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. *Near-optimal algorithms for unique games.* In *Proc. 38th ACM Symp. on Theory of Computing (STOC)*, pages 205–214. ACM, 2006. doi:10.1145/1132516.1132547.

[DH09] Irit Dinur and Prahladh Harsha. *Composition of low-error 2-query PCPs using decodable PCPs.* In *Proc. 50th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 472–481. IEEE, 2009. eccc:TR09-042, doi:10.1109/FOCS.2009.8.

[Din07] Irit Dinur. *The PCP theorem by gap amplification.* J. ACM, 54(3):12, 2007. (Preliminary Version in *38th STOC*, 2006). eccc:TR05-046, doi:10.1145/1236457.1236459.

[EH03] Lars Engebretsen and Jonas Holmerin. *Towards optimal lower bounds for clique and chromatic number.* Theoretical Comp. Science, 299(1–3):537–584, 2003. doi:10.1016/S0304-3975(02)00535-2.

[Erl06] Thomas Erlebach. *Approximation algorithms for edge-disjoint paths and unsplittable flow.* In *Efficient Approximation and Online Algorithms*, LNCS, chapter 4, pages 97–134. Springer, 2006. doi:10.1007/11671541_4.

[FGL+96] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. *Interactive proofs and the hardness of approximating cliques*. J. ACM, 43(2):268–292, March 1996. (Preliminary version in *32nd FOCS*, 1991). doi:10.1145/226643.226652.

[FHW80] Steven Fortune, John E. Hopcroft, and James Wyllie. *The directed subgraph homeomorphism problem*. Theoretical Comp. Science, 10(2):111–121, 1980. doi:10.1016/0304-3975(80)90009-2.

[FL92] Uriel Feige and László Lovász. *Two-prover one-round proof systems: Their power and their problems (extended abstract)*. In *Proc. 24th ACM Symp. on Theory of Computing (STOC)*, pages 733–744. ACM, 1992. doi:10.1145/129712.129783.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. *A tight bound on approximating arbitrary metrics by tree metrics*. J. Computer and System Sciences, 69(3):485–497, 2004. (Preliminary version in *35th STOC*, 2003). doi:10.1016/j.jcss.2004.04.011.

[GKR+03] Venkatesan Guruswami, Sanjeev Khanna, Rajmohan Rajaraman, F. Bruce Shepherd, and Mihalis Yannakakis. *Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems*. J. Computer and System Sciences, 67(3):473–496, 2003. (Preliminary version in *31st STOC*, 1999). doi:10.1016/S0022-0000(03)00066-7.

[GMR08] Venkatesan Guruswami, Rajsekar Manokaran, and Prasad Raghavendra. *Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph*. In *Proc. 49th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 573–582. IEEE, 2008. doi:10.1109/FOCS.2008.51.

[GT06a] Anupam Gupta and Kunal Talwar. *Approximating unique games*. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 99–106. SIAM, 2006. doi:10.1145/1109557.1109569.

[GT06b] Venkatesan Guruswami and Kunal Talwar. *Hardness of low congestion routing in directed graphs*. Technical Report TR06-141, Electronic Colloquium on Computational Complexity, 2006. eccc:TR06-141.

[GW95] Michel X. Goemans and David P. Williamson. *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*. J. ACM, 42(6):1115–1145, 1995. (Preliminary version in *26th STOC*, 1994). doi:10.1145/227683.227684.

[Har04] Prahladh Harsha. *Robust PCPs of Proximity and Shorter PCPs*. Ph.D. thesis, Massachusetts Institute of Technology, September 2004.

[Hås01] Johan Håstad. *Some optimal inapproximability results*. J. ACM, 48(4):798–859, July 2001. (Preliminary Version in *29th STOC*, 1997). doi:10.1145/502090.502098.

[Kho02] Subhash Khot. *On the power of unique 2-prover 1-round games*. In *Proc. 34th ACM Symp. on Theory of Computing (STOC)*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

[Kho05] ———. *Guest column: inapproximability results via long code based PCPs*. SIGACT News, 36(2):25–42, 2005. doi:10.1145/1067309.1067318.

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. *Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?* SIAM J. Computing, 37(1):319–357, 2007. (Preliminary version in *45th FOCS*, 2004). eccc:TR05-101, doi:10.1137/S0097539705447372.

[Kle96] Jon M Kleinberg. *Approximation algorithms for disjoint paths problems*. Ph.D. thesis, Massachusetts Institute of Technology, May 1996.

[KR08] Subhash Khot and Oded Regev. *Vertex cover might be hard to approximate to within 2-ε*. J. Computer and System Sciences, 74(3):335–349, 2008. (Preliminary Version in *18th IEEE Conference on Computational Complexity*, 2003). doi:10.1016/j.jcss.2007.06.019.

[KS01] Stavros G. Kolliopoulos and Clifford Stein. *Approximation algorithms for single-source unsplittable flow*. SIAM J. Computing, 31(3):919–946, 2001. (Preliminary version in *38th FOCS*, 1997). doi:10.1137/S0097539799355314.

[KV05] Subhash Khot and Nisheeth K. Vishnoi. *The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into l_1*. In *Proc. 46th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 53–62. IEEE, 2005. doi:10.1109/SFCS.2005.74.

[LR99] Frank Thomson Leighton and Satish Rao. *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*. J. ACM, 46(6):787–832, 1999. doi:10.1145/331524.331526.

[MH99] Sanjeev Mahajan and Ramesh Hariharan. *Derandomizing approximation algorithms based on semidefinite programming*. SIAM J. Computing, 28(5):1641–1663, 1999. (Preliminary version in *36th FOCS*, 1995). doi:10.1137/S0097539796309326.

[MM09] Konstantin Makarychev and Yury Makarychev. *How to play unique games on expanders*, 2009. arXiv:0903.0367.

[MOO05] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. *Noise stability of functions with low in.uences invariance and optimality*. In *Proc. 46th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 21–30. IEEE, 2005. arXiv:math/0503503, doi:10.1109/SFCS.2005.53.

[MR08] Dana Moshkovitz and Ran Raz. *Two query PCP with sub-constant error*. In *Proc. 49th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 314–323. IEEE, 2008. eccc:TR08-071, doi:10.1109/FOCS.2008.60.

[O'D08] Ryan O'Donnell. *Some topics in analysis of Boolean functions*. In *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, pages 569–578. ACM, 2008. eccc:TR08-055, doi:10.1145/1374376.1374458.

[Rag08] Prasad Raghavendra. *Optimal algorithms and inapproximability results for every CSP?* In *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, pages 245–254. ACM, 2008. doi:10.1145/1374376.1374414.

[Raz98] Ran Raz. *A parallel repetition theorem*. SIAM J. Computing, 27(3):763–803, June 1998. (Preliminary Version in *27th STOC*, 1995). doi:10.1137/S0097539795280895.

[RS95] Neal Robertson and Paul D. Seymour. *Graph minors. XIII. The disjoint paths problem*. Journal of Combinatorial Theory, Series B, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

[RS96] Ronitt Rubinfeld and Madhu Sudan. *Robust characterizations of polynomials with applications to program testing*. SIAM J. Computing, 25(2):252–271, April 1996. (Preliminary Version in *23rd STOC*, 1991 and *3rd SODA*, 1992). doi:10.1137/S0097539793255151.

[RS97] Ran Raz and Shmuel Safra. *A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP*. In *Proc. 29th ACM Symp. on Theory of Computing (STOC)*, pages 475–484. ACM, 1997. doi:10.1145/258533.258641.

[RT87] Prabhakar Raghavan and Clark D. Thompson. *Randomized rounding: a technique for provably good algorithms and algorithmic proofs*. Combinatorica, 7(4):365–374, 1987. doi:10.1007/BF02579324.

[RZ] Satish Rao and Shuheng Zhou. (unpublished).

[ST00] Alex Samorodnitsky and Luca Trevisan. *A PCP characterization of NP with optimal amortized query complexity*. In *Proc. 32nd ACM Symp. on Theory of Computing (STOC)*, pages 191–199. ACM, 2000. doi:10.1145/335305.335329.

[Tre01] Luca Trevisan. *Non-approximability results for optimization problems on bounded degree instances*. In *Proc. 33rd ACM Symp. on Theory of Computing (STOC)*, pages 453–461. ACM, 2001. doi:10.1145/380752.380839.

[Tre08] ———. *Approximation algorithms for unique games*. Theory of Computing, 4(1):111–128, 2008. (Preliminary version in *46th FOCS*, 2005). eccc:TR05-034, doi:10.4086/toc.2008.v004a005.

[VV04] Kasturi R. Varadarajan and Ganesh Venkataraman. *Graph decomposition and a greedy algorithm for edge-disjoint paths*. In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 379–380. SIAM, 2004. doi:10.1145/982792.982846.