GEORGIA TECH FALL'04
CS 8002 : PCPS AND HARDNESS OF APPROXIMATION

Lecture 1: Introduction

Lecturer: *Subhash Khot* Scribe:*Nikhil R. Devanur*

We assume throughout this course that the reader is familiar with basic complexity classes like **P**, **NP**, **BPP**, notion of poynomial time reducability and **NP**-completeness. A good reference is [11].

1 Approximation Algorithms and Hardness Results

Consider a minimization problem $\mathcal{P}$. Let I be an instance of the problem and $n = |I|$ be the size of the instance. $OPT(I)$ be the value of the optimal solution to that instance. For example, if $\mathcal{P}$ is the Traveling Salesman Problem (TSP), an instance I is a weighted graph and $OPT(I)$ is the length of the shortest tour. For an algorithm A, let $A(I)$ be the value of the solution found by A.

DEFINITION 1 *A polynomial time algorithm A for a (minimization) problem $\mathcal{P}$ is an $\alpha(n)$-* Approximation Algorithm *if for all instances I of $\mathcal{P}$ with $|I| = n$, $A(I) \leq \alpha(n) \cdot OPT(I)$.*

In the above definition $\alpha(n) \geq 1$ is a function of input size. An approximation algorithm for a maximization problem can be defined analogously with $\alpha(n) \leq 1$ and the guarantee $A(I) \geq \alpha(n) \cdot OPT(I)$. We call $\alpha(n)$ as approximation *ratio*, or the approximation *factor*. Desigining approximation algorithms for NP-hard problems is a highly successful field (a nice reference is [14]).

DEFINITION 2 Hardness of Approximation: *An optimization problem $\mathcal{P}$ is said to be hard to approximate (or inapproximable) within a factor of $\alpha(n)$ if the existence of an $\alpha(n)$-approximation algorithm for $\mathcal{P}$ implies* $\mathbf{P} = \mathbf{NP}$ *(or a similar complexity theoretic implication that is regarded as highly unlikely).*

This course will focus on proving hardness of approximation results for many fundamental **NP**-hard optimization problems. Since it is widely believed that $\mathbf{P} \neq \mathbf{NP}$, a hardness result rules out possibility of a better approximation. Apart from this obvious motivation, the area has deep connections to several areas in computer science and mathematics, including combinatorics, cryptography, proof checking, coding theory, property testing, and Fourier analysis of boolean functions. Excellent surveys on inapproximability (see [13, 1]) are available.

2 Some NP-Hard Optimization Problems

Let us enlist some important **NP**-hard optimization problems and the approximation algorithms known for them. We also mention the best inapproximability results known, many of which will be covered in the course.

Definition 3 MAX-3SAT: *Given a set of 3-CNF clauses $\{C_1, \ldots, C_m\}$, find an assignment that satisfies the maximum number of clauses. Every clause is guaranteed to contain three distinct literals.*

There exists a trivial 7/8-approximation algorithm for MAX-3SAT : Just assign TRUE/FALSE at random to every variable. Out of 8 possible assignments to every clause, 7 are satisfying assignments. Therefore a random assignment satisfies 7/8 fraction of the clauses. It is easy to derandomize the algorithm as well. It would come as a great surprise to know that MAX-3SAT has a matching hardness result. Håstad [9] showed that for any $\epsilon > 0$, MAX-3SAT is hard to approximate within ratio better than $7/8 + \epsilon$!

Definition 4 Vertex Cover: *Given a graph G, find a minimum set of vertices $C \subseteq V(G)$ such that for each edge $e \in E(G)$, there is a vertex $v \in C$ that is adjacent to e.*

There is a trivial 2-approximation algorithm for Vertex Cover (see [14]) and surprisingly, this remains the best algortihm known. The best inapproximability ratio known is 1.36 due to Dinur and Safra [4] and perhaps it is **NP**-hard to achieve $2 - \epsilon$ approximation for every $\epsilon > 0$.

Definition 5 Set Cover: *Given a ground set U with $|U| = n$ and a collection of subsets of U, $\{S_1, \ldots, S_m\}$, find a minimum sub collection whose union is U.*

There is a $\ln n$ (greedy) approximation algorithm for Set Cover. Again a surprise ! A matching $(1 - \epsilon) \ln n$ hardness result is known for every $\epsilon > 0$, due to Feige [5].

Definition 6 Max Clique: *Given a n-vertex graph G, find the maximum size clique (a subgraph that is a complete graph).*

A trivial algorithm that outputs a single vertex as Clique qualifies as a n-approximation algorithm. Yet again, a surprise ! It is unlikely that one would do better. Håstad showed $n^{1-\epsilon}$ hardness for every $\epsilon > 0$ (see [8]).

Definition 7 Bin Packing: *Given rationals $x_1, x_2, \ldots, x_n \in [0, 1]$, find the minimum k such that there is a partition $P_1, \ldots, P_k$ of $\{1, 2, \ldots, n\}$ such that for all P_i, $\sum_{j \in P_i} x_j \leq 1$.*

Bin-Packing has a very good approximation algortihm. On an instance with optimum $OPT(I)$, the algorithm finds a packing using only $(1 + \epsilon)OPT(I) + 1$ bins where $\epsilon > 0$ is arbitrary. We say that Bin-Packing has an (asymptotic) PTAS.

Definition 8 *A problem $\mathcal{P}$ is said to have PTAS if for every $\epsilon > 0$, the problem has a polynomial time $(1 \pm \epsilon)$-approximation algoithm.*

3 Gap Problems

The main technique used in proving hardness results is by reducing an **NP**-complete problem to the *gap version* of the target problem.

Definition 9 *For a (minimization) problem $\mathcal{P}$, its gap version Gap-$\mathcal{P}_{g(n)}$ is a promise problem such that for some function $h(n)$,*

- *The YES instances are instances I of $\mathcal{P}$ such that $OPT(I) \leq h(n)$ and*
- *The NO instances are instances I of $\mathcal{P}$ such that $OPT(I) \geq g(n)h(n)$.*

The function $g(n) \geq 1$ is call the gap. A similar definition can be made for maximization problems.

Now suppose that there is a polytime reduction from an NP-complete language L to Gap-$\mathcal{P}_{g(n)}$ such that the YES (resp. NO) instances of L are mapped to YES (resp. NO) instances of Gap-$\mathcal{P}_{g(n)}$. Then $g(n)$-approximation algorithm for $\mathcal{P}$, if exists, can be used to decide the NP-complete language L in polytime. It follows that it is NP-hard to approximate $\mathcal{P}$ within factor $g(n)$. This is a typical way of proving inapproximability results. In the following section, we use this strategy to prove inapproximability results for two problems.

4 Hardness of TSP and Edge Disjoint Paths

Hardness for TSP

We prove the hardness of approximating TSP within a factor α, for any constant $\alpha > 1$. An instance of TSP is a edge-weighted graph and the goal is to find a tour that visits every vertex exactly once and has minimum length.

We reduce the Hamiltonian Cycle problem to Gap-TSP_α, with $h(n) = n$. That is, the YES instances of Gap-TSP_α have $OPT(I) \leq n$, and the NO instances have $OPT(I) > \alpha n$. Given a graph G, the Hamiltonian Cycle problem asks whether it has a simple cycle that visits all vertices. Construct a graph H with $V(H) = V(G)$. For all $e \in E(G)$, let its weight in H be $w_H(e) = 1$. Otherwise let $w_H(e) = \alpha n$ where $n = |V(G)|$. Clearly if G has a Hamiltonian cycle, then $OPT(H) = n$ and otherwise any tour in H must include a non-edge of G and therefore $OPT(H) \geq \alpha n$. Thus Gap-TSP_α is **NP**-Hard and TSP is hard to approximate within a factor α. Note that α could be made to depend on n and could be as large as $2^{n^{\Omega(1)}}$. Clearly, this inapproximability result is not very meaningful.

Edge Disjoint Paths

Definition 10 Edge Disjoint Paths (EDP) *: Given a directed graph G, and source-sink pairs (s_i, t_i) for $i = 1, \ldots, t$, find the maximum number of edge disjoint paths $\{P_{i_1}, P_{i_2}, \ldots, P_{i_l}\}$ where path P_{i_j} connects s_{i_j} to t_{i_j} and the indices i_j are distinct for $1 \leq j \leq l$.*

The special case of EDP when $t = 2$ is called 2 Disjoint Paths (2DP). Given a directed graph and two source-sink pairs, the problem is to decide whether there exist disjoint paths connecting the two sources to corresponding sinks. It is known that 2DP is **NP**-Complete [6]. For EDP, $\sqrt{m}$-approximation algorithm was given by [10], where $m = |E(G)|$. Here we sketch the proof of $m^{1/2-\epsilon}$ factor hardness result for any $\epsilon > 0$. This result is due to Guruswami et al. [7]. We reduce 2DP to Gap-EDP_n, where n is chosen so that $n = m^{1/2-\epsilon}$.

Consider an instance of 2DP, a directed graph H and two pairs (x_1, y_1) and (x_2, y_2). Create a graph G with source sink pairs (s_i, t_i) for $i = 1, \ldots, n$ as shown in Figure 1. There is a directed path from every s_i to t_i (directed upwards and then to the left). The set of

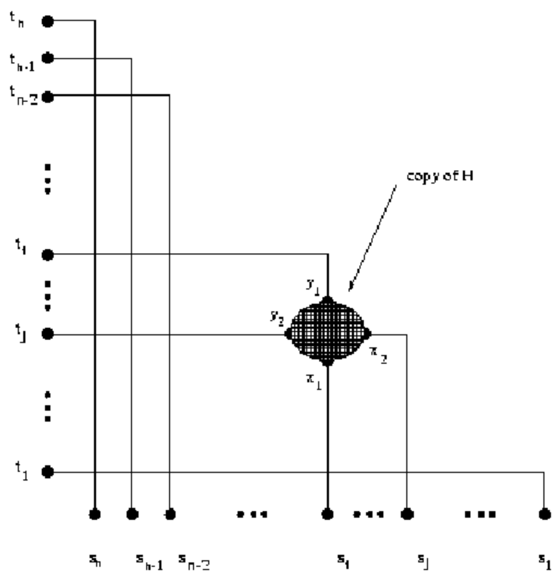

Figure 1: Reduction from 2DP to Gap-EDP_n.

these paths forms a grid. At every intersection point of the grid, we place a copy of H and the vertices x_1, y_1, x_2, y_2 are identified with the edges of the grid as shown.

It is easy to see that for the YES instances of 2DP, that is when H has 2 edge disjoint paths from x_1 to y_1 and x_2 to y_2, the resulting EDP instance has $OPT(G) = n$, i.e., all the n pairs in G can be connected by edge disjoint paths. This is done by taking the natural paths from s_i to t_i and within each copy of H, routing them in a edge-disjoint fashion.

Moreover the NO instances of 2DP are mapped to instances where $OPT(G) = 1$. Intuitively, it is clear that one can rout at most one pair (s_i, t_i). If one tries to rout two pairs, say i^{th} and j^{th} pairs, one is forced to find two disjoint paths in the copy of H at the intersection point of the i^{th} and j^{th} *natural* path. However a formal argument needs a little more work, which is left as an exercise. This gives us a reduction from 2DP to Gap-EDP_n and since 2DP is **NP**-complete, EDP is hard to approximate within a factor n.

Letr h be the number of edges in H. Then the number of edges in G is $m = n^2 h$. So by letting $n = h^{1/\epsilon}$, for any $\epsilon > 0$, we get that $n = m^{1/2-\epsilon}$. So EDP is hard to approximate within a factor of $m^{1/2-\epsilon}$ for any $\epsilon > 0$.

5 Complete Problems for MAX-SNP and PCP Theorem

Bin Packing, Knapsack and many of the scheduling problems have a PTAS, i.e. $(1+\epsilon)$ approximation for every $\epsilon > 0$. It was a perplexing question for long time whether other problems like MAX-3SAT and Vertex Cover also have a PTAS. An influential step towards tackling this question was taken by Papadimitriou and Yannakakis [12] who defined a class of problems called MAX-SNP. This class is inspired by Fagin's characterization of **NP** as the set of graph-properties expressible in existential second-order logic. All problem in MAX-SNP have a constant factor approximation. Papadimitriou and Yannakakis also defined the notion of complete problems for this class and proved that if any of the MAX-SNP-complete problems had a PTAS, then every problem in MAX-SNP would also have a PTAS. Several well studied problems were shown to be MAX-SNP-complete, e.g. MAX-3SAT, Metric TSP, Max Clique on bounded degree graphs etc.

The major open problem, whether MAX-SNP-complete problems have PTAS, was resolved in the negative by the PCP Theorem (due to Arora, Safra [3] and Arora et al. [2]). Let us state the PCP Theorem :

THEOREM 1
There is a polynomial time reduction from SAT to Gap-MAX-3SAT$_c$*, i.e. a polytime reduction that maps instances* ϕ *of SAT to instances* ψ *of MAX-3SAT such that*

- *If* ϕ *is satisfiable, so is* ψ*, i.e.* $OPT(\psi) = 1$.
- *If* ϕ *is unsatisfiable, then* $OPT(\psi) \leq c$ *for some absolute constant c. In other words, there is no assignment to* ψ *that satisfies more than a fraction c of clauses.*

In particular, MAX-3SAT and therefore every MAX-SNP-complete problem has no PTAS unless **P**=**NP**.

Discovery of this theorem has led to many breakthrough results in inappximability theory. This course would cover the proof of PCP Theorem as well as many of the hardness results.

In the first part of the course, we will assume the PCP Theorem as a black-box result and use it to prove hardness results. The proof of The PCP Theorem itself is very sophisticated and long; it will be covered in the second part of the course.

References

[1] Sanjeev Arora. *Probabilistic checking of proofs and the hardness of approximation problems.* Ph.D. Thesis, UC Berkeley, 1994.

[2] S. Arora, C. Lund, R. Motawani, M. Sudan, M. Szegedy, *Proof verification and the hardness of approximation problems*, Journal of the ACM, 45(3) : 501-555, 1998.

[3] S. Arora, S. Safra, *Probabilistic checking of proofs : A new characterization of NP*, Journal of the ACM, 45(1):70-122, 1998.

[4] I. Dinur and S. Safra. *The importance of being biased.* In Proc. 34th Annual ACM Symposium on Theory of Computing, pages 33-42, May 2002.

[5] U. Feige, *A threshold of ln n for approximating set cover*, Journal of the ACM, 45(4), 634–652, July 1998.

[6] S. Fortune, J. Hopcroft, and J. Wyllie. *The directed subgraph homeomorphism problem.* Theoretical Computer Science, 10(2):111–121, 1980.

[7] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis. *Near optimal hardness results and approximation algorithms for edge-disjoint paths and related problems.* In Proc. 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 19–28.

[8] J. Håstad, *Clique is hard to approximate within* $n^{1-\epsilon}$. In Proc. 37th Annual IEEE Symposium on Foundations of Computer Science, 627-636, 1996.

[9] J. Håstad, *Some optimal inapproximability results.* To appear in the Journal of the ACM. In Proc. 29th Annual ACM Symposium on Theory of Computing, 1-10, 1997.

[10] J. Kleinberg. *Approximation Algorithms for Disjoint Paths Problems.* PhD thesis, MIT, 1996.

[11] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[12] C.H. Papadimitriou and M. Yannakakis. *Optimization, approximization and complexity classes.* Journal of Computer and System Sciences, 43:425-440, 1991.

[13] L. Trevisan. *Inapproximability of Optimization Problems.* Survey, 2004. Available at http://www.cs.berkeley.edu/∼luca/pubs/inapprox.ps

[14] V. V. Vazirani. *Approximation Algorithms.* Springer Verlag, 2001.