Hardness of Approximation

Christopher M. Bourke

University Of Nebraska – Lincoln

Fall 2004

CSE 990 – Approximation Algorithms

Based on Chapter 29 of Vazirani

cbourke@cse.unl.edu

Deterministic Decision Problem World

- Assumption: $\mathrm{P} \neq \mathrm{NP}$
- $\Pi \in \mathrm{NPC}$ discourages efforts to find a P-time algorithm.
- Main tool: P-time reductions (not sufficient for optimization problems).
- Can we develop a similar tool to establish hardness of optimization problems?

Introduction

For example, can we establish the following: *There is no PTAS approximating Vertex Cover to within* $\alpha \cdot \frac{2}{3}|V|$ *of its optimal value.*

Approximation Algorithm World

- Assumption: P $\neq$ NP
- Hardness results for a certain factor discourage trying to find a PTAS
- Main tools: Gap Reductions, PCP Theorem

PTAS Definition

We already have what an α-factor (polynomial time) approximation algorithm is.

Definition

A *polynomial time approximation scheme* (PTAS) is a polynomial time running algorithm such that for any $\epsilon > 0$ is guaranteed to find a solution to within a $1 \pm \epsilon$ factor of OPT.

Gap Introducing Reduction

Definition

A *gap-introducing reduction* Γ from SAT (or any NPC problem) to an optimization problem, Π_1 (SAT $\leq_{gir} \Pi_1$) has two parameters: f, α. Let ϕ be a given boolean formula.

If Π_1 is a minimization problem, then Γ outputs, in polynomial time, an instance $x \in \Pi_1$ such that

- If $\phi \in$ SAT, $\mathrm{OPT}(x) \leq f(x)$
- If $\phi \notin$ SAT, $\mathrm{OPT}(x) > \alpha(|x|) \cdot f(x)$

If Π_1 is a maximization problem, then Γ outputs, in polynomial time, an instance $x \in \Pi_1$ such that

- If $\phi \in$ SAT, $\mathrm{OPT}(x) \geq f(x)$
- If $\phi \notin$ SAT, $\mathrm{OPT}(x) < \alpha(|x|) \cdot f(x)$

- For minimization problems, $\alpha(|x|) \geq 1$, for maximization problems, $\alpha(|x|) \leq 1$.
- f is a function of the input where as α is a function only of the *input size*
- α is the *gap*; i.e. the hardness factor that the gap-introducing reduction establishes ("there is no PTAS with optimality to within α")

Just as with regular reductions, gap-preserving reductions can be made transitive, but we need another special reduction.

Gap Preserving Reduction I

Definition

Let Π_1 and Π_2 be optimization problems. A *gap-preserving* reduction Γ, $(\Pi_1 \leq_{gpr} \Pi_2)$ has four parameters,

$$f_1, f_2, \alpha, \beta$$

For instances $x \in \Pi_1, y \in \Pi_2$ there are four cases to consider.

1. If Π_1 is a minimization problem, Π_2 a maximization problem then
   - $\mathrm{OPT}(x) \leq f_1(x) \Rightarrow \mathrm{OPT}(y) \geq f_2(y)$
   - $\mathrm{OPT}(x) > \alpha(|x|)f_1(x) \Rightarrow \mathrm{OPT}(y) < \beta(|y|)f_2(y)$
   - Note $\alpha(|x|) \geq 1, \beta(|y|) \leq 1$.
2. If Π_1 is a minimization problem, Π_2 a minimization problem then
   - $\mathrm{OPT}(x) \leq f_1(x) \Rightarrow \mathrm{OPT}(y) \leq f_2(y)$
   - $\mathrm{OPT}(x) > \alpha(|x|)f_1(x) \Rightarrow \mathrm{OPT}(y) > \beta(|y|)f_2(y)$

- Note $\alpha(|x|) \geq 1, \beta(|y|) \geq 1$.

3. If Π_1 is a maximization problem, Π_2 a maximization problem then
   - $\mathrm{OPT}(x) \geq f_1(x) \Rightarrow \mathrm{OPT}(y) \geq f_2(y)$
   - $\mathrm{OPT}(x) < \alpha(|x|) f_1(x) \Rightarrow \mathrm{OPT}(y) < \beta(|y|) f_2(y)$
   - Note $\alpha(|x|) \leq 1, \beta(|y|) \leq 1$.
4. If Π_1 is a maximization problem, Π_2 a minimization problem then
   - $\mathrm{OPT}(x) \geq f_1(x) \Rightarrow \mathrm{OPT}(y) \leq f_2(y)$
   - $\mathrm{OPT}(x) < \alpha(|x|) f_1(x) \Rightarrow \mathrm{OPT}(y) > \beta(|y|) f_2(y)$
   - Note $\alpha(|x|) \leq 1, \beta(|y|) \geq 1$.

The composition of a gap-introducing reduction with a gap-preserving reduction yields a gap-introducing function.

Example

Say

$$\text{SAT} \leq_{gir} \Pi_1 \leq_{gpr} \Pi_2 \Rightarrow \text{SAT} \leq_{gir} \Pi_2$$

then a $\beta(|x|)$-factor approximation algorithm for Π_2 yields a P-time Algorithm for SAT. Therefore, under the assumption P $\neq NP$, *there exists no $\beta(|x|)$-factor approximation algorithm for Π_2.*

Gap Preserving Reduction II

Notes

Note: whereas a (P-time) reduction from SAT to $\mathcal{P}$ and a P-time algorithm for $\mathcal{P}$ yields a P-time algorithm for SAT, the same doesn't happen under gap-preserving reductions; i.e. say SAT $\leq_{gpr} \Pi$; just because we may have a PTAS for Π doesn't give us one for SAT. A more stringent, *approximation factor preserving reduction* is required.

Probabilistically Checkable Proof Systems

The Complexity class PCP is a probabilistic characterization of NP.

- NP Protocol: A prover forms a polynomially large *proof* (or *witness*) and a verifier, V, checks the entire proof (in polynomial time). An instance is accepted if there exists a valid proof.
- PCP Protocol: A prover forms a *proof* and the verifier randomly checks *portions* of the proof.
  - If $x \in L$, then $\exists$ a proof such $\Pr[V(x) \text{ accepts}] = 1$
  - If $x \notin L$, $\Pr[V(x) \text{ rejects}] \geq 1/2$.

This protocol allows the verifier to consider super-polynomially long proofs or a very small portion (a constant number of bits) of a polynomially long proof.

PCP Parameters

The $1/2$ in the definition can be made arbitrarily close to 1 by repeated executions of the protocol.

PCP also has two parameters: $\text{PCP}[r(n), q(n)]$ where

- $r(n)$ is the number of random bits available to V
- $q(n)$ is the number of queries to the prover (i.e. the number of bits in the proof that V is allowed to examine)

By definition,

$$\text{NP} = \text{PCP}[0, \text{poly}(n)]$$

PCP Theorem

Theorem (The PCP Theorem)

$$\text{NP} = \text{PCP}[\log n, 1]$$

In other words, for every NP problem there exists a protocol such that a verifier can use a logarithmic sized random string to randomly access a constant number of bits (still a function of the input and random string) to be able to accept (or reject).

Note: this is *not* a PCP protocol (why?)

Illustrative Example

Let m be the number of clauses of a 3SAT formula ϕ. A verifier V does the following:

1. V uses $\mathcal{O}(\log m)$ random bits which forms an index i; it then queries the proof using i to get a random clause C_i from ϕ.
2. V examines the proof (a truth assignment $\{0,1\}^n$) and accepts if C_i is satisfiable.

If ϕ is satisfiable, then there obviously exists a proof since C_i must be satisfiable. If ϕ is not satisfiable, then

$$\Pr[V(\phi) \text{ accepts}] \leq 1 - \frac{1}{m}$$

A proof provided by a PCP protocol involves a complex algebraic construction that ensures each part of the *proof* depend on every bit of the input.

For example, an NP witness for SAT is simply a string $x \in \{0,1\}^n$. But a PCP witness for SAT is a re-mapping, $y \in \{0,1\}^{f(n)}$ where $f(n)$ may be a polynomial or even exponential.

The key tool of the PCP Theorem is that for any $\mathcal{P} \in$ NP, we know there *exists* a PCP$[\log n, 1]$ protocol. For our purposes it will not be necessary to explicitly construct one.

Problem (Maximize Accept Probability)

Let V *be a* PCP$[\log, 1]$ *verifier for* SAT*.*

Given *a boolean formula,* ϕ

Output *a proof (i.e. a satisfying assignment) that maximizes the probability of acceptance of* V

Claim

There is no factor $1/2$ factor approximation algorithm (i.e. no PTAS) for MAP

Assume to the contrary that there is a PTAS, $\mathcal{A}$ that solves MAP. If ϕ is satisfiable, then simply use $\mathcal{A}$ to simulate V for all random strings of length $\mathcal{O}(\log n)$ (there are $2^{\mathcal{O}(\log n)} = \text{poly}(n)$ number of them).

Thus $\mathcal{A}$ can calculate the acceptance probability p in polynomial time.

If $p \geq 1/2$, then we accept, otherwise reject. We have now constructed a P-time algorithm deciding SAT (why? recall the definition) $\square$

MAX3SAT

$\mathrm{MAX3SAT}$ is the restriction of SAT where each clause has *at most* three literals; we wish to maximize the number of true literals in a satisfying assignment

Theorem

For every constant $\epsilon > 0$ *there exists a gap-introducing reduction from an instance* ϕ *of* SAT *to an instance* ψ *of* $\mathrm{MAX3SAT}$ *such that*

- *If* $\phi \in$ SAT, $\mathrm{OPT}(\psi) = m$ *and*
- *If* $\phi \notin$ SAT, $\mathrm{OPT}(\psi) < (1 - \epsilon)m$

Thus, there is no PTAS for $\mathrm{MAX3SAT}$

We prove in two steps:

$$\text{SAT} \leq_{gir} \text{MAX } k\text{-FUNCTION SAT} \leq_{gpr} \text{MAX3SAT}$$

Problem (MAX k-FUNCTION SAT)

Given *n boolean variables, $x_1, x_2, \ldots, x_n$ and m boolean functions, $f_1, f_2, \ldots f_m$ ($f_j : \{0,1\}^k \rightarrow \{0,1\}$ of a constant k number of the literals).*
Output *a truth assignment that maximizes the number of functions satisfied.*

The Reduction

Theorem

There is a reduction,

$$\text{SAT} \leq_{gir} \text{MAX } k\text{-FUNCTION SAT}$$

that transforms an instance ϕ *of* SAT *to an instance* I *of* MAX k-FUNCTION SAT *such that*

- *If* $\phi \in \text{SAT}$, $\text{OPT}(I) = m$ *and*
- *If* $\phi \notin \text{SAT}$, $\text{OPT}(I) < \frac{1}{2}m$

The Reduction

Proof I

Let V be a PCP$[c \log n, q]$ verifier for SAT $(c, q$ constants) and ϕ a boolean formula. For each $r \in \{0,1\}^{c \log n}$, V reads q bits for a total of qn^c bits of the proof[1]. We associate a new boolean variable for each bit, $B = \{x_1, x_2, \ldots, x_{qn^c}\}$.

Now define functions $f_1, f_2, \ldots, f_r, \ldots, f_{n^c}$ $(\{0,1\}^q \rightarrow \{0,1\})$.

- If $\phi \in$ SAT, there exists a proof such that $f_r = 1 \forall r$
- If $\phi \notin$ SAT then $\Pr[V(x) \text{ accepts}] < 1/2 \forall x \in \{0,1\}^n$ thus $< \frac{1}{2} n^c$ functions are satisfied (note here, $m = n^c$)

[1] Recall the contrast between an NP witness and a PCP witness—the former is of length n, the latter in this case is of length qn^c

The Reduction
Proof II

Now we reduce to 3SAT. Each f_r can be represented as a SAT formula, ψ_r with at most 2^q clauses each having at most q literals. Define

$$\psi = \bigwedge_{1 \leq r \leq n^c} \psi_r$$

- If $\phi \in$ SAT, $\psi \in$ SAT
- If $\phi \notin$ SAT then at least 1 clause in each f_r is not satisfied so $> \frac{1}{2}n^c$ clauses in ψ are not satisfied, so $\psi \notin$ SAT.

We now convert ψ to an instance of 3SAT, ψ' in the usual manner. ψ' contains at most $n^c 2^q (q-2)$ clauses, choosing $\epsilon = 1/(2^{q+1}(q-2))$ gives the theorem. $\square$

Hardness Results Even for Bounded Occurrences of Literals

Even if we bound the number of times a variable v_i occurs in a formula ϕ, the hardness results still apply. Formally define this problem as $\mathrm{MAX3SAT}(k)$.

Theorem

There is a reduction $\mathrm{MAX3SAT} \leq_{gpr} \mathrm{MAX3SAT}(29)$ *that transforms a boolean formula* ϕ *to* ψ *such that*

- *If* $\mathrm{OPT}(\phi) = m$ *then* $\mathrm{OPT}(\psi) = m'$ *and*
- *If* $\mathrm{OPT}(\phi) < (1-\epsilon)m$ *then* $\mathrm{OPT}(\psi) < (1-\epsilon')m'$

where m, m' *are the number of clauses in* ϕ, ψ *and* $\epsilon' = \epsilon/43$.

Proof I

Claim

There exists an efficient algorithm $\mathcal{A}$ and a constant N_0 such that for every $N \geq N_0$, $\mathcal{A}$ constructs a degree 14 expander graph on N vertices. An expander graph is one in which all vertices have the same degree and for any nonempty $S \subsetneq V$, $|E(S, \overline{S})| > \min\{|S|, |\overline{S}|\}$ where $E(S, \overline{S})$ denotes edges crossing the *cut* across $(S, \overline{S})$

Proof II
Graph Widget

Expanders allow us to build the following *graph widget*: given a set of $k \geq N_0$ boolean variables, let $k \geq N_0$ and $G_x = (V, E)$ be a degree 14 expander graph on k vertices, each labeled with a unique variable, $x_1, \ldots x_k$. We construct a boolean formula as follows. For each edge $(x_i, x_j) \in E$, add the clauses $(\overline{x}_i \vee x_j)$ and $(x_i \vee \overline{x}_j)$ to ψ_x.

Proof III

Observe that ψ_x is consistent if and only if

$$x_1 = x_2 = \ldots = x_k = b \in \{0, 1\}$$

An *inconsistent* assignment partitions the edges in G_x into $S, \overline{S}$.

For each edge in the cut $(S, \overline{S})$, ψ_x will have an unsatisfied clause. Therefore, the number of unsatisfied clauses is at least $\min\{|S|, |\overline{S}|\} + 1$;

$$|E(S, \overline{S})| > \min\{|S|, |\overline{S}|\} + 1$$

Proof IV

Let $B = \{x_1, x_2, \ldots, x_n\}$ be the variables in a $\mathrm{MAX3SAT}$ formula ϕ (w.l.o.g. assume that that every variable $x \in B$ occurs at least N_0 times in ϕ). For each $x_i \in B$ do the following.

1. Let $V_{x_i} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$
2. Let G_{x_i} be a degree 14 expander graph using V_x for labels to obtain a formula ψ_{x_i}.
3. Replace each occurrence of x_i in ϕ with a unique variable from B.

Proof V

Now construct a new graph

$$V = \bigcup_{x_i \in B} V_{x_i}$$

which yields a formula ϕ'. Define

$$\psi = \phi' \wedge \psi_{x_1} \wedge \psi_{x_2} \wedge \ldots \wedge \psi_{x_n}$$

For each $x_i \in B$, each x_{i_j} occurs 28 times in ψ_{x_i} and once in ϕ', so ψ is an instance of $\mathrm{MAX3SAT}(29)$

Proof VI

Claim

An optimal truth assignment must satisfy every clause in

$$\bigwedge_{x_i \in B} \psi_{x_i}$$

.

Assume to the contrary that τ is an optimal assignment that is not consistent for V_{x_i}. τ partitions V_{x_i} into $S, \overline{S}$ (say S is smaller). Now flip each variable in S; some clauses in ϕ' may now be unsatisfied. If they are, they must contain a variable of S so the number of such clauses is at most $|S|$. However, we get at least $|S| + 1$ new satisfied clauses corresponding to the edges in the cut $(S, \overline{S})$, thus the new assignment satisfies more clauses than τ, so contradiction, τ is optimal.

Proof VII

Let m, m' be the number of clauses in ϕ, ψ. The total number of occurrences of all variables in ϕ is $\leq 3m$ (because its a $3\mathrm{SAT}$ with m clauses). Each of these participates in 28 2-literal clauses giving a total of

$$28\frac{3m}{2} = 42m$$

clauses, with m additional clauses in ϕ', so a total of $43m$ clauses, $\therefore m' \leq 43m$.

If $\phi \in$ SAT then ψ is satisfiable. Consider $\mathrm{OPT}(\phi) < (1-\epsilon)m$, then $> \epsilon m$ clauses of ϕ are unsatisfied on *any* truth assignment and so $> \epsilon m \geq \epsilon m'/43$ of the clauses in ψ are unsatisfied. $\square$

Vertex Cover

For $d \in \mathbb{Z}^+$ let $\mathrm{VC}(d)$ be the restriction of vertex cover to graphs with vertex degree *at most* d.

Theorem

There is a gap preserving reduction, $\mathrm{MAX3SAT}(29) \leq_{gpr} \mathrm{VC}(30)$, *such that*

- *If* $\mathrm{OPT}(\phi) = m$ *then* $\mathrm{OPT}(G) \leq \frac{2}{3}|V|$ *and*
- *If* $\mathrm{OPT}(\phi) < (1-\epsilon)m$ *then* $\mathrm{OPT}(G) > (1-\epsilon)\frac{2}{3}|V|$

We first pad out ϕ to ensure that every clause has *exactly* three literals.

- For each clause, $(x_i \vee x_j \vee x_k)$ define three labeled vertices, thus $|V| = 3m$
- For each clause, include 3 edges connecting each literal in each clause.
- For each $u, v \in V$, if the label of u is the negation of v then include the edge $(u, v) \in E$.

The degree of each edge is bounded by $2 + (29 - 1) = 30$, so this is now an instance of $\mathrm{VC}(30)$.

Claim

The size of a maximum independent set in $G = \mathrm{OPT}(\phi)$.

Let τ be an optimal truth assignment and select one vertex corresponding to a satisfied literal from *each* satisfied clause, this forms an independent set. Conversely, an independent set I in G corresponds to an optimal truth assignment—just set the labeled literals to true, any extension of τ must satisfy at least $|I|$ clauses.

Fact

The compliment of a maximum independent set in G is a minimum vertex cover; i.e. for $I \subseteq V$, $\overline{I}$ is a minimum vertex cover.

Therefore,

$$\mathrm{OPT}(\phi) = m \Rightarrow \mathrm{OPT}(G) = 2m = \frac{2}{3}|V|$$

and

$$\mathrm{OPT}(\phi) < (1-\epsilon)m \Rightarrow \mathrm{OPT}(G) > (2+\epsilon)m = (1-\epsilon)\frac{2}{3}|V|$$

□

Steiner Tree

Theorem

There is a reduction, $\mathrm{VC}(30) \leq_{gpr} \mathrm{STP}$ *which transforms an instance* $G = (V, E)$ *to* $H = (R, S, cost)$ *where* R, S *are the required and Steiner vertices of* H *such that*

- *If* $\mathrm{OPT}(G) \leq \frac{2}{3}|V|$ *then* $\mathrm{OPT}(H) \leq |R| + \frac{2}{3}|S| - 1$ *and*
- *If* $\mathrm{OPT}(G) > (1+\epsilon)\frac{2}{3}|V|$ *then* $\mathrm{OPT}(H) > (1+\epsilon)(|R| + \frac{2}{3}|S| - 1)$

Proof I

We first recall that the Steiner tree problem. Given a graph $H = (R, S, E)$ with a *required* vertex set R, *Steiner* vertex set S and weighted edges, find a minimum cost tree that spans R and any subset of S.

We construct a graph $H = (R, S, \mathrm{cost})$ such that $G = (V, E)$ has a vertex cover of size c iff H has a Steiner tree of cost $|R| + c - 1$.

Proof II

- For each $e \in E$ let $r_e \in R$ (i.e. a *vertex*)
- For each $v \in V$ let $s_v \in S$
- An edge between to required vertices has cost $\text{cost}(r_e, r'_e) = 2$
- An edge between two Steiner vertices has cost $\text{cost}(s_v, s'_v) = 1$
- An edge

$$(r_e, s_v) = \begin{cases} 1 & \text{if } e \text{ is incident on } v \\ 2 & \text{otherwise} \end{cases}$$

Proof III

($\Rightarrow$) Let S_c be the set of Steiner vertices in H corresponding to the c vertices in the vertex cover. Thus, there is a tree in H covering $R \cup S_c$ using only cost 1 edges since every $e \in E$ must be incident on a vertex in the cover. The total cost is $|R| + c - 1$.

($\Leftarrow$) Let T be a Steiner tree in H of cost $|R| + c - 1$. We transform T into an equivalent Steiner tree that only uses cost 1 edges, thus it corresponds to a vertex cover of size c.

Proof IV

Let (u, v) be an edge of weight 2. If u is a Steiner vertex, remove (u, v) from T yielding two components. Select another edge incident on v and another required vertex into T.

Now (u, v) is an edge of weight 2 and both vertices are Steiner. Consider the corresponding e_v, e_u in G. Since G is connected, there exists a path p from one endpoint to the other. Removing (u, v) disconnects T into two components, R_1 and R_2, of required vertices.

Proof V

Note that $u \in R_1, v \in R_2$ so p must have two adjacent edges, (a, b) and (b, c) such that the corresponding vertices, $w \in R_1, w' \in R_2$. Now let s_b be the Steiner vertex in H corresponding to b. The edges (s_b, w) and (s_b, w') connect R_1 and R_2, both are of unit cost.

- If $\mathrm{OPT}(G) \leq \frac{2}{3}|V|$ then $\mathrm{OPT}(H) > |R| + \frac{2}{3}|S| - 1$
- If $\mathrm{OPT}(G) > (1+\epsilon)\frac{2}{3}|V|$ then $\mathrm{OPT}(H) > |R| + (1+\epsilon)\frac{2}{3}|S| - 1$

$\square$

Clique

MaxClique

A *clique* of a graph $G = (V, E)$ is a subset $C \subseteq V$ that induces a complete subgraph. MaxClique is the problem that, given a graph G, find a clique of maximum cardinality.

Claim

For any $\epsilon > 0$, there is no $\frac{1}{n^\epsilon}$-factor approximation algorithm for MaxClique; i.e. there is no PTAS for MaxClique

Clique I

Lemma

For constants b, q there is a reduction, SAT $\leq_{gir}$ $\textsc{MaxClique}$ *transforming a boolean function ϕ to a graph $G = (V, E)$ where $|V| = 2^q n^b$ such that*

- *If* $\phi \in$ SAT *then* $\mathrm{OPT}(G) \geq n^b$
- *If* $\phi \notin$ SAT *then* $\mathrm{OPT}(G) < \frac{1}{2} n^b$

Proof Idea: Use a $F \in$ PCP$[q, b \log n]$ verifier; for each $r \in \{0,1\}^{b \log n}$ and each truth assignment $\tau \in \{0,1\}^q$, define vertices $v_{r,\tau}$. Connect vertices according to queries to F: $(v_{r_1,\tau_1}, v_{r_2,\tau_2}) \in E$ if F *accepts* given r_1 (and r_2) and reads τ_1 (and τ_2) *and* τ_1 agrees with τ_2 whenever they overlap).

Clique II

This establishes that there is no $\frac{1}{2}$-factor approximation algorithm for $\textsc{MaxClique}$.

Unfortunately, it doesn't establish the inverse polynomial, another characterization of PCP has to be made.

PCP Generalization

We define two additional parameters to the PCP class: Completeness and Soundness so that $L \in \text{PCP}_{c,s}[r(n), q(n)]$ if there is a verifier V that on input x (with $|x| = n$) obtains a random string of length $O(r(n))$, queries $O(q(n))$ bits and satisfies:

- If $x \in L$ then there exists a proof y that makes V accept with probability $\geq c$ (Correctness)
- If $x \notin L$ then for every proof y, V accepts with probability $< s$ (Soundness)

By definition,

$$\mathsf{PCP}[r(n), q(n)] = \mathsf{PCP}_{1,\frac{1}{2}}[r(n), q(n)]$$

but the we can establish the following stronger result.

Theorem

$$\mathsf{NP} = \mathsf{PCP}_{1,\frac{1}{n}}[\log n, \log n]$$

Proof Idea

We can always improve soundness to any $\frac{1}{2^k}$. But to get inverse polynomial soundness would require $k \in \mathcal{O}(n)$, and thus we would use $r(n) \in \mathcal{O}(n \log n)$ random bits and $q(n) \in \mathcal{O}(n)$, which is not PCP.

Instead, we need $k \in \Omega(\log n)$ so $q(n) \in \mathcal{O}(\log n)$, but $r(n) \in \mathcal{O}(\log^2 n)$.

The trick is to again use expander graphs; specifically, expander graphs on n^b vertices with a unique $b \log n$ bit label. Random walks can then be constructed using $r(n) \in \mathcal{O}(\log n)$ bits. These walks simulate random strings. Though they are not random, the properties of expander graphs help to show that the probability of error drops exponentially for each run of the verifier.

Clique

Theorem

For constants b, q, there is a reduction, SAT $\leq_{gir}$ MaxClique that transforms a Boolean formula ϕ of in n variables to a graph $G = (V, E)$ where $|V| = n^{b+q}$ such that

- *If $\phi \in$ SAT then $\mathrm{OPT}(G) \geq n^b$*
- *If $\phi \notin$ SAT then $\mathrm{OPT}(G) < n^{b-1}$*

Corollary

There is no $\frac{1}{n^\epsilon}$-factor approximation algorithm for MaxClique. *I.e. there is no PTAS for* MaxClique

Set Cover

Recall SetCover: A universal set U ($|U| = n$), and a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_k\}$ with associated costs, find a sub-collection of $\mathcal{S}$ with minimum cost that covers U.

Recall the greedy algorithm for set-cover; simply take the most cost-effective $\left(\frac{\mathrm{cost}(S)}{|S-C|}\right)$ set S_i until you cover U.

This remains the best algorithm known; indeed it is tight up to a constant multiple.

The Verifier, V is allowed to query two provers, P_1 and P_2. A prover is trying to "cheat": P_1 and P_2 are attempting to convince V that $x \in L$ even though x is a *no* instance.

Since P_1 and P_2 cannot communicate, V can cross check the answers that they give and their ability to cheat is limited.

The two-prover one-round model comes with three parameters: Completeness, Soundness, and number of random bits.

$$\text{2P1R}_{c,s}(r(n))$$

We say $L \in \text{2P1R}_{c,s}(r(n))$ if there is a polynomial time verifier V that uses $O(r(n))$ random bits and satisfies

- For every $x \in L$ there is a pair of proofs, y_1, y_2 that makes V accept with probability $\geq c$
- For every $x \notin L$ and every pair of proofs y_1, y_2, V accepts with probability $< s$.

Theorem

There is a constant ϵ such that

$$\mathsf{NP} = \mathsf{2P1R}_{1,1-\epsilon}(\log(n))$$

Set Cover

Theorem

There is a constant $c > 0$ for which there is a randomized reduction SAT $\leq_{gir}$ $\textsc{SetCover}$ *running in time $n^{O(\log\log n)}$ transforming a Boolean formula ϕ to a set system $\mathcal{S}$ over a universal set of size $n^{O(\log\log n)}$ such that*

- *If $\phi \in$* SAT, $\mathrm{OPT}(\mathcal{S}) = 2n^k$
- *If $\phi \notin$* SAT, $\Pr[\mathrm{OPT}(G) > cn^k k \log n] > 1/2$

Here n is the length of the 2P1R *proofs, polynomial in the size of ϕ and $k \in O(\log\log n)$.*

Proof Idea

A *set system* is a collection,

$$(U, C_1, \ldots, C_m, \overline{C}_1, \ldots, \overline{C}_m)$$

A "good" covering is obviously $C_i \cup \overline{C}_i$, otherwise it is a bad covering.

Theorem

There exists a randomized polynomial algorithm exists which generates a set system for each m, l with $|U| = p(m, 2^l)$ such that with $\Pr \geq 1/2$, every bad cover is of size $> l$.

I.e. we can construct set systems where good and bad covers have *very* different sizes. This means choosing a good cover must be well-coordinated.

Summary

pgflastimage