Approximation hardness via Gap Reduction and PCP

Sunghyeon Jo

Seoul National University

Project-Hardness
Oct 25, 2022

We will look...

- What **Optimization problem** is
- Approximation algorithms
- Gap problems and Gap reduction
- PCP machinery
  - what is PCP?
  - Hard-to-Approximate problems that can be proved by PCP machinery

Optimization problem

Definition

An optimization problem consists of the following:

- Set of instances of the problem (e.g. set of graphs for 3COL)
- For each instance: the set of possible solutions(e.g. the set of 3-coloring for graph instance)
- For each solution: a nonnegative weight(cost for min-problem, benefit for max-problem)
- An objective: either min or max(e.g. min in TSP, max in knapsack)

Goal of an optimization problem is to find a solution which achieves the objective(minimize cost / maximize benefit)

Approximation Algorithms

Definition

For an optimization problem A and an instance x, **OPT(x)** is the weight of the best solution (minimum cost for min-problem / maximum benefit for max-problem)

Definition

- Let A be a min-problem. algorithm alg is a **c-approximation algorithm for** A if for all valid instances x, $alg(x) \leq c \cdot OPT(x)$.
- Let A be a max-problem. algorithm alg is a **c-approximation algorithm for** A if for all valid instances x, $OPT(x) \leq c \cdot alg(x)$.

Examples of Approximation Algorithms

There are several approximation algorithms:

- $\frac{3}{2}$-approximation algorithm for Euclidean TSP
- $\frac{8}{7}$-approximation algorithm for MAX 3SAT. Note that the algorithm find a solution that satisfies at least $\frac{7}{8} \cdot OPT(x)$ clauses, but it is a $\frac{8}{7}$-approximation algorithm, not $\frac{7}{8}$-approximation algorithm.

PTAS

Definition

Let A be a min-problem. A **polynomial time approximation scheme(PTAS)** for A is a polynomial time algorithm that takes as input (x, ϵ) (x is an instance of A, $\epsilon > 0$ is a real number) and ouputs a solution that has weight $\leq (1+\epsilon)OPT(x)$.

Above definition is applied to max-problem in the same way.

Complexity classes

Definition

Assume that A be a min-problem.

- **PTAS**: set of all problems where PTAS exists.
- **APX**: $A \in$ APX if there is a constant $c \geq 1$ and a poly-time algorithm M such that $M(x) \leq c \cdot OPT(x)$
- **Log-APX**: $A \in$ Log-APX if there is a constant $c > 0$ and a poly-time algorithm M such that $M(x) \leq c \log |x| \cdot OPT(x)$
- **Poly-APX**: $A \in$ APX if there is a polynomial p and a poly-time algorithm M such that $M(x) \leq p(|x|) \cdot OPT(x)$

Above definition is applied to max-problem in the same way.
We will use both symbol n and $|x|$ for denoting size of input x.

Examples

- Euclidean TSP $\in$ APX since there is a $\frac{3}{2}$-approximation. In fact, Euclidean TSP $\in$ PTAS.
- MAX 3SAT $\in$ APX.

Basic Hard-to-Approximate Problems

Assuming P $\neq$ NP, then the followings hold:

- TSP $\notin$ Poly-APX.
- CLIQUE $\in$ Poly-APX $\setminus$ Log-APX.
- SET COVER $\in$ Log-APX $\setminus$ APX.
- MAX-3SAT $\in$ APX $\setminus$ PTAS.

Therefore, PTAS $\subsetneq$ APX $\subsetneq$ Log-APX $\subsetneq$ Poly-APX.

TSP is hard to approximate

Theorem

If TSP $\in$ Poly-APX, then P=NP.

Since HAM-CYCLE is NP-Complete, it is enough to show TSP $\in$ Poly-APX implies HAM-CYCLE is in P.
Assume that TSP $\in$ Poly-APX. Then there is a poly-time algorithm M for TSP and polynomial p such that $M(x) \leq p(|x|) \cdot OPT(x)$.
For instance of HAM-CYCLE $G = (V, E)$, create instance of TSP G' such that:

- if $e \in E$, then give e weight 1.
- If $e \notin E$, then give e weight $np(n) + 1$.

TSP is hard to approximate

- if $G \in$ HAM-CYCLE, $M(G') \leq p(n)OPT(G') = np(n)$
- if $G \notin$ HAM-CYCLE, $M(G') \geq OPT(G') \geq np(n) + 1$

so we can determine whether $G \in$ HAM-CYCLE or not, by running algorithm M on G'. Since M is poly-time algorithm for number of vertices n of G, it implies HAM-CYCLE $\in$ P.
Since HAM-CYCLE $\in$ NP-Complete, P=NP.

Gap problem

Let g be a max-problem (e.g. MAX 2SAT). Let $a(n), b(n)$ be functions from $\mathbb{N}$ to $\mathbb{N}$ such that $b(n) < a(n)$. Then:

Definition

GAP$(g, a(n), b(n))$
Instance: y which is guaranteed that $g(y) \geq a(|y|)$ or $g(y) \leq b(|y|)$.
Question: Determine which is the case.

Gap Lemma

Let A be NP-hard.

Lemma

If there exists a poly-time reduction that maps x to y such that

- *If $x \in A$ then $g(y) \geq a(|y|)$*
- *If $x \notin A$ then $g(y) \leq b(|y|)$*

Then,

- *GAP(g, a(n), b(n)) is NP-Hard*
- *If there is $\frac{a(n)}{b(n)}$-approximation poly-time algorithm for g, then P $=$ NP.*

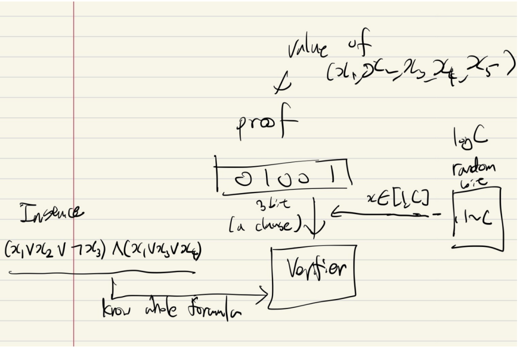
The proof is straightforward using the same technique we used for proving TSP $\in$ Poly APX implies P$=$NP.

PCP

PCP is hard to understand by definition, so I will introduce PCP using an example of 3SAT.

PCP

3SAT $\in$ PCP$(\log C, 3, \frac{C-1}{C})$ where C is number clauses. What does this mean?



PCP

3SAT $\in$ PCP($\log C, 3, \frac{C-1}{C}$) where C is number clauses. What does this mean?

- There is an instance(formula) φ, prover, verifier.
- A prover can submit an arbitrary evidence. Let the evidence as an assignment (values of $x_1, .., x_n$).
- Verifier rolls the dice and get $\log C$-bit random string. it is equivalent to a random number c from 1 to C.
- verifier see the instance (boolean formula) and verify that c-th clause is TRUE by read at most 3 variables of evidence.
- verifier accepts for prob 1 when the evidence is a solution. Otherwise, verifier accepts for prob $\leq \frac{C-1}{C}$.

Therefore, 3SAT $\in$ PCP($\log C, 3, \frac{C-1}{C}$)

PCP

$A \in \text{PCP}(r(n), q(n), \epsilon(n))$ iff:

- Given x, an instance of A. And There is a prover and a verifier.
- A prover can submit an arbitrary evidence.
- Verifier can read both instance and **part of** evidence and have to decide to accept or not, in poly-time.
- Verifier can access a $r(n)$-bit random string, which is equivalent to a random number between 1 and $2^{r(n)}$.
- Verifier can only read $q(n)$ bits of evidence. Verifier can decide the position of bits using above random string.
- If $x \in A$, there is an evidence y such that verifer always accepts.
- If $x \notin A$, acceptance rate should be at most $\epsilon(n)$ for every evidence.

Some Facts

- SAT $\in$ PCP$(0, n, 0)$. It can be achieved by evidence of n variables and checking by read all n variables.
- Some books omit $\epsilon(n)$ when $\epsilon(n) = \frac{1}{2}$. That is, PCP$(r(n), q(n))$ = PCP$(r(n), q(n), \frac{1}{2})$,
- PCP(0, poly(n), 0) = NP (by definition)
- PCP$(O(\log n), O(1), \frac{1}{2})$ = NP (**the PCP theorem**)
- SAT $\in$ PCP$(O(\log n), O(\log n), \frac{1}{n})$

Note: Definition of PCP in the book use the concept of **Oracle Turing Machine-bit access** and slightly different from prover-verifier scheme.

Problems that hard to approximate

In previous slides, we saw that $P \neq NP$ implies followings:

- TSP $\notin$ Poly-APX.
- CLIQUE $\in$ Poly-APX $\setminus$ Log-APX.
- SET COVER $\in$ Log-APX $\setminus$ APX.
- MAX-3SAT $\in$ APX $\setminus$ PTAS.

Using PCP theorem, we can prove them.

CLIQUE is hard to approximate

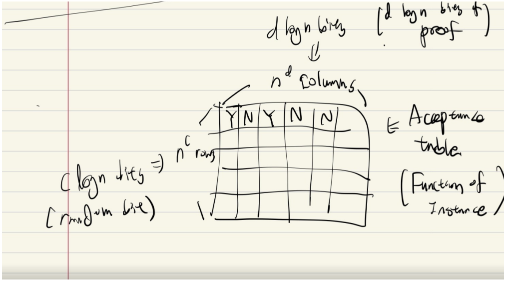
Proof is composed of following steps:

- There is an poly-time algorithm for CLIQUE which guarantees poly-scale approximation. Hence CLIQUE$\in$ Poly-APX (Fei04)
- Let A be a NP-Complete problem s.t. $A \in PCP(c \log n, d \log n, \frac{1}{n})$ for some c, d. There is a reduction that maps x ($|x| = n$) to a graph G on $N = n^{c+d}$ vertices such that:
  - If $x \in A$ then $OPT(G) \geq n^c = N^{c/(c+d)}$
  - If $x \notin A$ then $OPT(G) \leq n^{c-1} = N^{(c-1)/(c+d)}$
- If there is approximation algorithm for CLIQUE s.t. returns a clique of size at least $OPT(G)/N^{1/(c+d)}$, then P=NP. (Gap Lemma)
- Assuming P $\neq$ NP, CLIQUE $\in$ Poly-APX - Log-APX

Only the second step is not trivial. Let's see about it.

CLIQUE is hard to approximate

First, SAT $\in$ PCP$(O(\log n), O(\log n), \frac{1}{n})$ so c, d exists. Think of a verifier as a function of instance, then it can be seen as a table.



Connect $(r1, c1)$ and $(r2, c2)$ iff $r1 \neq r2$ and $(c1, c2)$ are not contradicting. Since $\epsilon(n) = 1/n$, there is no clique of size $n^(c-1)$ if $x \notin A$. Therefore, step 2 is proved.

Thank you!