Tuesday, May 29, 2018 8:30 PM

Probabilistically Checkable Proofs & Hardness of Approximation

Many hard problems (especially NP-hard problems) are optimization problems. (minimization or maximization)

e.g. smallest tour, smallest vertex cover, largest independent set.

"OPT" = value of the optimal solution.

If we can not compute OPT exactly, can it be approximated?

- many heuristics.
- want approximation algorithms w/ guarantees.

approximation ratio r

on every input x:

$$\frac{OPT}{r} \le \text{answer} \le OPT \quad \text{(maximization)}$$

$$OPT \le \text{answer} \le r \cdot OPT \quad \text{(minimization)}$$

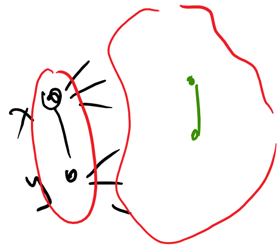
Example: Vertex Cover

Input: Graph $G = (V, E)$

What is the smallest subset of the vertices that touches every edge?

Decision version of VC is NP-complete:

Does G have a VC of size $\le k$?



Here's an approximation algorithm:

Pick an edge (x,y), add $x+y$ to VC

Discard all edges incident to x or y.

Continue until no edges remain.

Approximation ratio for this simple algorithm $= 2$.

For each edge — we add 2 vertices.

Edges considered do not share any endpoints.

OPT $\geq$ 1 vertex for each edge considered.

$$\frac{(\text{our VC})}{2} \leq \text{OPT}$$

$\leq \text{OPT} \leq (\text{our VC}) \cdot 2$

There is a diverse array of ratios achievable:

Vertex Cover: 2

Max-3-SAT: $8/7$ (find the assignment that satisfies the most clauses.)

Set Cover: $\log n$

Knapsack: $(1+\epsilon)$ for any $\epsilon > 0$

Clique: $\frac{n}{\log^2 n}$.

Best Case for approximations:

Poly-time approximation scheme (PTAS)

For every $\epsilon > 0$

$(1+\epsilon)$ can be achieved

Approximation runs in poly(n) but may be exponential in $1/\epsilon$.

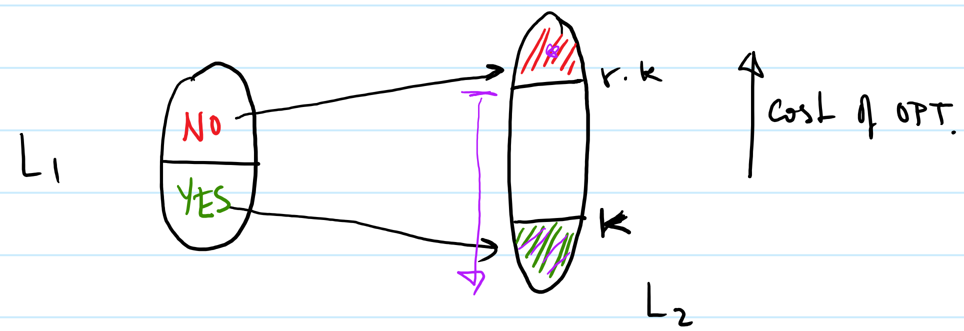
Tuesday, May 29, 2018 8:30 PM

How do we prove failure to improve on these?

Is it NP-hard to achieve a particular approximation ratio?

In order to prove a statement like that, we need a "gap producing" reduction from L_1 to L_2:

Minimization:



r-gap producing reduction: f: poly-time computable

$$x \in L_1 \longrightarrow OPT(f(x)) \leq k.$$
$$x \notin L_1 \longrightarrow OPT(f(x)) > r \cdot k.$$

(k can depend on x).

The target problem is not a language,
It is a promise problem.
Inputs = $\underline{\underline{Yes}} \cup \underline{\underline{No}}$ → not all strings!

opt $\geq rk$ opt $\leq k$

(Such as: Graphs w/ Independent set $\leq n/10$ or $> n/2$).

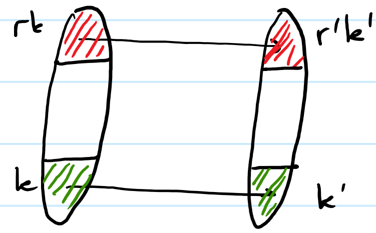
The algorithm is promised that the input comes from the set of Yes instances or No instances.

Tuesday, May 29, 2018 8:30 PM

Purpose: r-approximation alg. for L_2 distinguishes between f(yes) + f(no). This can be used to decide L_1.

Note that if it is NP-hard to compute with a promise, it is also hard to compute without it.

Gap producing reductions are difficult
Gap preserving reductions are easier



$r = \frac{1}{1-\epsilon}$

For example the standard reduction of K-SAT to 3-SAT is gap preserving.

MAX-k-Sat w/ gap ϵ $\propto$ Max 3-SAT w/ gap ϵ'

If it's hard to approximate MAX-k-SAT to within ϵ, then it's hard to approximate MAX-3-SAT to within ϵ'. ($r = \frac{1}{1-\epsilon}$)

MAXSNP (Papadimitriou + Yannikakis) a set of problems reducible to each other in this way.

A PTAS for a MAXSNP-complete problem gives a PTAS for all the problems in MAXSNP.

Missing piece: first gap-producing reduction.

Tuesday, May 29, 2018 8:30 PM

Consider Max-k-SAT w/ promise gap ϵ.

Input: Instance ϕ k-CNF

$\geq m$ YES: Some assignment satisfies all of the clauses

$\leq (1-\epsilon)m$ NO: No assignment satisfies more than a fraction of $(1-\epsilon)$ of the clauses.

max # clauses satisfied $= m$ or $\leq (1-\epsilon)m$.

Let's look at a proof system for this problem.

Suppose there is a reduction for an NP-hard problem to MAX-k-SAT w/ promise gap ϵ.

Then the following protocol will solve the NP-hard problem:

Given x, compute reduction to k-SAT ϕ_x

The Verifier expects that the proof is a satisfying assignment to ϕ_x

The verifier picks a random clause ("local test") and checks that it is satisfied by the assignment.

$$x \in L \Rightarrow \Pr[V \text{ accepts}] = 1$$

$$x \notin L \Rightarrow \Pr[V \text{ accepts}] < 1-\epsilon$$

Repeat $O(1/\epsilon)$ times to get error $< 1/2$.

Tuesday, May 29, 2018 8:30 PM

Note: Prover commits to the whole proof
Verifier only looks at part of the proof
Verifier does only local tests
An ϵ fraction of the tests will cause
Verifier to notice a mistake.
want $\epsilon \sim 1/\text{poly}(n)$

PCP Probabilistically Checkable Proofs
Novel way of verifying a proof.
V gets $O(r(n))$ random bits.
queries the proof in $O(q(n))$ locations.
Accept if the proof passes the test.

$PCP[r(n), q(n)]$ set of languages with a verifier V that has (r, q) restricted access to proof.

Completeness: $x \in L \Rightarrow \exists$ proof s.t.
$\Pr[V(x, \text{proof}) \text{ accepts}] = 1$
$x \notin L \Rightarrow \forall$ proof*
$\Pr[V(x, \text{proof}^*) \text{ accepts}] \leq 1/2$

Observations: $PCP[1, \text{poly}(n)] = NP$ Verifier sees entire proof.

$PCP[\log n, 1] \subseteq NP$

Run through all possible random strings and calculate prob of acceptance explicitly $\Rightarrow$ Verifier becomes deterministic.

Tuesday, May 29, 2018 8:30 PM

PCP Theorem: $PCP[\log n, 1] = NP$

Any problem in NP has a $[\log n, 1]$ probabilistically checkable proof.

How does this relate to hardness of approximation?

Corollary: MAX-K-SAT is hard to approximate to within some constant ϵ.

Unless P = NP there is no alg that can satisfy

$$(1-\epsilon)\,OPT \le \#\text{ satisfied clauses} \le OPT$$

Proof: Use $PCP[\log n, 1]$ protocol for some NP-hard problem (say VC)

$\Rightarrow$ Enumerate all $2^{O(\log n)} = poly(n)$ sets of queries.
$\Rightarrow$ Each set of queries consists of $O(1)$ (say k) queries.

Verifier acceptance is a function of the k-bits that are read.

For a particular $i \in \{0,1\}^{O(\log n)}$

$\phi_i(x_{i1}, \dots, x_{ik})$ = whether verifier accepts

Express ϕ_i as k-CNF w/ $\le 2^k$ clauses.

Wednesday, May 30, 2018 9:15 AM

"Yes" instance of VC $\Rightarrow$ all clauses satisfied.
(prob of acceptance = 1)

"No" instance of VC $\Rightarrow$ Every assignment fails to satisfy $\leq 1/2$ of the ϕ_i

ϕ_1 not satisfied $\Rightarrow$ ≥ 1 unsat clause

$$\# \text{ unsat clauses} \geq \underbrace{\frac{1}{2}(2^{-k})}_{\epsilon}.$$