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

\[ \text{e.g. Smallest tour, smallest VC, largest I.S.} \]

"opt" = value of the optimal solution.

Is approximating opt good enough?

- There are lots of heuristics.
- We want an approximation algorithm that guarantees some approximation ratio \( r \).

\[ \text{on every input } x \]
\[ \text{opt} \leq \text{answer} \leq \text{opt} \times r \text{ (for maximization)} \]
\[ \text{opt} \leq \text{answer} \leq r \times \text{opt} \text{ (for minimization)} \]

**Vertex Cover**: Input: undirected graph \( G = (V, E) \).

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

**Vertex Cover (VC)** is NP-complete.

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.

Approx ratio for this simple algorithm is 2:

- For each edge considered, we add 2 vertices.
- Edges considered to not share any endpoints.

\[ \text{opt} \geq 1 \text{ vertex per edge considered.} \]
\[ \text{opt} \geq (\text{opt VC})/2. \]
There is a diverse array of ratios achievable.

**Vertex Cover:** 2.

**MAX-3-SAT:** \(8/7\) (full assignment that satisfies the most clauses).

**Set Cover:** \(\log n\).

**Knap sack:** \((1+\varepsilon)\) for any \(\varepsilon > 0\).

**Clique:** \(n/\log n\).

**Best Case:** Poly-time approximation scheme (PTAS)

For every \(\varepsilon > 0\)

\((1+\varepsilon)\) ratio can be achieved

The approx algorithm runs in poly(n) where n is input size.

But might depend exponentially on \(1/\varepsilon\).

How do we explain the failure to improve some of these?

(How to explain the failure to find a poly-time alg for SAT?)

Is it NP-hard to improve an approximation ratio?

In order to prove a statement like this, we need a “gap-producing” reduction from \(L_1\) to \(L_2\): 

**Minimization:**

\[ L_1 \rightarrow \text{cost of optimal solution} \]

**R-gap producing reduction:** \(f\): poly-time computable.

\(x \in L_1 \quad \Rightarrow \quad \text{opt}(f(x)) \leq k.\)

\(x \notin L_1 \quad \Rightarrow \quad \text{opt}(f(x)) > rk.\)
The target problem is not a language. It is a promise problem. (Yes vs No instances are not all shipped)

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

Purpose: $k$-approximation alg for $L_2$ distinguishes between $f(\text{yes}) + f(\text{no})$. This can be used to decide $L_2$.

Note that if it is NP-hard to compute with the promise, then it's also hard to compute without it.

Gap producing reductions are difficult.
Gap preserving reductions are easier.

Many typical reductions are not gap preserving.

For example, the standard reduction of $k$-SAT to $3$-SAT is gap preserving. $\text{Max-}k\text{-Sat} \leq_g \text{gap } \varepsilon$ reduces to $\text{Max-3-Sat} \leq_g \text{gap } \varepsilon'$.

If it's hard to approximate $\text{Max-}k\text{-Sat}$ to within $\varepsilon$, then it's hard to approximate $\text{Max-3-Sat}$ to within $\varepsilon'$.

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

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

Missing piece: first gap-producing reduction.
Consider max-$k$-SAT with promise gap $\epsilon$.

Input: instance $\phi$ in $k$-CNF.

Yes: some assignment satisfies all the clauses.

No: no assignment satisfies more than $\frac{\alpha}{k} \cdot (1-\epsilon)$ of the clauses.

Let's look at a proof system view of this problem.

Suppose there is a reduction from an NP-hard problem to max-$k$-SAT with promise gap $\epsilon$.

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

Given $x$, compute reduction to $k$-SAT $\phi_x$.

The verifier expects that the proof is a satisfying assignment to $\phi_x$.

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

$x \notin L \Rightarrow \Pr[\text{Verifies}] = 1$

$x \in L \Rightarrow \Pr[\text{Verifies}] < 1-\epsilon$.

This can be repeated $O(1/\epsilon^2)$ times for error $< \frac{1}{2}$.

Note: Prover commits to the whole proof.

Verifier only looks at part of the proof.

$\Rightarrow$ verifier does only local test. An $\epsilon$ fraction of these tests will notice the invalidity.

$\Rightarrow$ looking for $\epsilon = \frac{1}{p_{\phi}(x)}$

PCP: Probabilistically Checkable Proof

Novel way of verifying a proof.

- pick a random local test
- query proof in specified $k$ locations (chosen by random bits)
PCP \([r(n), g(n)]\) set of languages is \(\text{p.p.t.} \text{ verifiable}\) that has \((r, g)\) restricted access to the proof.

- \(V\) tosses \(O(r(n))\) coins.
- \(V\) access the proof in \(O(g(n))\) locations \((\text{4th})\).

(Completeness) \(\forall x \in L \Rightarrow \exists\ \text{ proof } s.t.
\Pr [V(x, \text{ proof}) \text{ accepts}] = 1.\)

(Soundness) \(\forall x \notin L \Rightarrow \forall\ \text{ proof}
\Pr [V(x, \text{ proof}) \text{ accepts}] \leq 1/2.\)

Observations

- \(\text{PCP} \ [1, \text{poly}(n)] = \text{NP}\)
- \(\text{PCP} \ \lbrack \log n, 1 \rbrack \subseteq \text{NP}\)

 verifier runs the entire poly-size proof.

PCP Theorem

\(\text{PCP} \ \lbrack \log n, 1 \rbrack = \text{NP}\)

Any problem in NP has a \(\lbrack \log n, 1 \rbrack\) prob. checkable proof.

How does this relate to hardness of approximations?

Corollary: Max-k-SAT is hard to approximate to within some constant \(\varepsilon\).

Proof: Use PCP \(\lbrack \log n, 1 \rbrack\) protocol for some NP-hard problem.

- Verifier in \(\phi:\)
  \[x_1, \ldots, x_m\]
  bits of the proof.
- \(m = 2^{O(\log n)} = \text{poly}(n)\) bits of queries.

Enumerate all \(2^{O(\log n)} = \text{poly}(n)\) sets of queries.

Construct a \(k\)-CNF \(\phi_i\) for Verifier's test.

On each \(\text{ query } i \in \{0, 1\}^n\) \(\Rightarrow \phi_i\) accepts as a fan of the k-bit response.
there are $\leq 2^k$ clauses. They are all satisfied iff Verifier accepts.

"Yes" instance of VC $\Rightarrow$ all clauses satisfied.

"No" instance of VC $\Rightarrow$ every assignment fails to satisfy

$\leq \frac{1}{2}$ of the $\phi_i$.

$\phi_i$ not satisfied $\Rightarrow$ $\geq 1$ unsatisfied clause.

$\# \text{ unsat clauses} \geq \frac{1}{2}(2^{-k}) \frac{\text{factor m}}{\varepsilon}$.