Status of NP-complete problems w.r.t. approximation

Harder
- $O(\log n)$-factor (Set Cover)
- Constant factor (e.g., Vertex Cover, Euclidean TSP)
- PTAS (e.g., packing unit squares, bin packing)

Easier
- FPTAS (e.g., knapsack)

Positive results - give approx. alg.
Negative results - "hardness"

An easy example:
Poly-time 2-approx for general TSP $\Rightarrow$ P = NP
This is in notes for Lecture 15, and was covered today
There are also reductions that preserve good approximation
examples
- Poly-time $k$-approx for Ind. Set $\Rightarrow$ poly-time $k$-approx.
  for clique - on Assign 6
- PTAS for Ind. Set $\Rightarrow$ PTAS for Clique (same pf.)
- Constant factor approx for Clique $\Rightarrow$ PTAS for Clique
  - on current assignment.

Another example:
Thm: PTAS for Ind. Set $\Rightarrow$ PTAS for Max 3-SAT.
Recall:
Max 3-SAT $= (x \lor y \lor z) \land (a \lor z \lor b \lor c) \land \cdots$
Given 3-SAT formula, find truth value assignment maximizing # clauses
satisfied.
Proof of Thm uses standard reduction
3-SAT to Ind. Set.
Recall that reduction
\[ C = (x_1 v \overline{x_2} v x_3) \] becomes \( \triangle \)

\[ x_1 \quad \overline{x_2} \quad x_3 \]

put edge \((x, \overline{x})\) for all occurrences.

Graph \( G \) with \( 3m \) vertices, \( m = \# \) clauses.

- truth value assignment
- satisfies \( k \) clauses

\[ \text{maps to} \]

Ind. Set of \( k \) vertices.

in particular \( \text{OPT}_{\text{Max3Sat}}(F) = \text{OPT}_{\text{Ind Set}}(G) \)

and poly-time approx alg for Ind. Set that gives

\[ \text{A}_{\text{Ind Set}}(G) \geq \alpha \cdot \text{OPT}_{\text{Ind Set}}(G) \]

implies poly-time approx alg for Max 3-SAT that gives

\[ \text{A}_{\text{Max3Sat}}(F) \geq \alpha \cdot \text{OPT}_{\text{Max3Sat}}(F) \]

Then poly-time \((1 + \varepsilon)\)-approx for Vertex Cover

implies poly-time \((1 - 5\varepsilon)\)-approx for Max 3-SAT.

\( \alpha \) PTAS for V.C. \( \Rightarrow P = NP \).

This theorem can be proved somewhat like the one above (a bit harder)
Breakthrough Result 192

PTAS for Max 3-SAT $\implies P = NP$.

Ex. Go through results above to see implications

- e.g. constant factor approx for Clique $\implies P = NP$.

Today - idea of proof of breakthrough result.

- involves new characterization of NP.
Recall

NP - decision problems that can be verified in poly-time
  given a certificate of poly-size.

  e.g. Ham. cycle
  Max 3-SAT.

Think of this as a game between

  Prover P - all powerful, finds a certificate
  Verifier V - computationally limited - poly-time,
    - check certificate.

Generalize

  - allow Verifier to use randomness
  - allow interaction.

Interactive Proof Systems.

Example: Graph Isomorphism.

Given 2 graphs — can you relabel to get same graph.

[Diagram of two graphs showing isomorphism]

Graph Isom. is in NP.

in P? in NP-complete? OPEN.

in co-NP? i.e. can we verify if \( G_1 \not\cong G_2 \) given certificate? not isom.
An interactive proof protocol to verify \( G_1 \not\cong G_2 \)

Verifier:
- pick \( G_1 \) or \( G_2 \) at random
- randomly relabel
- ask Prover — was it \( G_1 \) or \( G_2 \)?

If \( G_1 \not\cong G_2 \) Prover can answer correctly.
If \( G_1 \cong G_2 \) Prover can't do better than \( 50\% \) right.

Verifier runs many trials & verifies \( G_1 \not\cong G_2 \) with high prob.

Probabilistically Checkable Proofs

Given a statement (e.g. \( G \) has a Ham. cycle)
- the Prover writes "proof"
- the Verifier is a randomized alg. that "checks" the proof & answers YES or NO

Conditions on correctness
- if statement is TRUE there is a "proof" that makes \( V \) answer YES (always)
- if statement is FALSE then \( V \) must answer NO with Prob \( \geq 3/4 \) no matter what "proof" is given.

Limiting \( V \)'s resouces
- poly. time
- \( O(f(n)) \) random bits
- \( O(g(n)) \) bits of proof.

more restricted proof system
- no rounds
PCP \([f, g] \) — class of decision problems with \textbf{Probabilistically Checkable Proof} where \(V\) uses \(O(f(n))\) random bits and \(O(g(n))\) bits of proof.

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

\textbf{Thm} \textit{"PCP theorem" [Arora, Lund, Motwani, Sudan, Szegedy '92]}

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

\(V\) looks at only \(O(1)\) bits of the proof!

\(V\) uses random bits to choose where to look at the proof as addresses into "proof".

easy direction of proof:

\[
\text{PCP}[\log n, 1] \leq \text{NP} = \text{PCP}[0, \text{poly}(n)]
\]

must eliminate randomness (increase \#bits proof)

Verifier tries all possible random strings of \(O(\log n)\) bits.

\[
2^{k \log n} = 2^{\log^k n} = n^k
\]

Verifier looks at \(n^k\) bits of proof.

other direction is hard.
Implications of PCP theorem to hardness of approximation.

**Theorem:** PTAS for Max 3-SAT \( \implies \) \( P = \text{NP} \).

**Proof Idea:** Use \( \text{NP} = \text{PCP}[\log n, 1] \)

Take any problem in \( \text{NP} \) (will give poly-time alg.)

Take the \( \text{PCP}[\log n, 1] \) verifier's alg. for it.

The alg. depends on:
- \( x \), the input, bits \( x_1 \ldots x_n \)
- \( y \), the "proof", bits \( y_1 \ldots y_t \) \( t \in O(\text{poly}(n)) \) (as above).
- \( r \), the random bits, \( r_1 \ldots r_k \) \( k \approx O(\log n) \)

Any alg. \( \implies \) Boolean 3-SAT formula (as in first NP-completeness proof)

Use variables for \( y_1 \ldots y_t \)

Formulate \( F(x, y, r) \) captures verifier's alg.

Let \( F(x, y) = \bigwedge_{r} F(x, y, r) \) — poly-size.

If \( x \) is YES input \( \implies \exists \) \( y \) s.t. all \( F(x, y, r) \) satisfied.

\( \implies F(x, y) \) satisfied.

If \( x \) is NO input \( \implies \) for any \( y \) at most \( \frac{1}{4} \) of the \( F(x, y, r) \) are satisfied.

\( \implies \) at most a fraction of the clauses of \( F(x, y) \) can be satisfied.

This gives a gap that we can detect with good approx alg. — just like for TSP.

\( \therefore \) PTAS for Max 3-SAT \( \implies \) \( P = \text{NP} \).