Keywords

The following terms should be well-known to you.

1. P, NP, NP-hard, NP-complete.
2. (polynomial-time) reduction.
3. Cook-Levin theorem.
4. NPO problems. Instances. Solutions.

Approximation Complexity Theory: historical remarks

In general, proving the NP-hardness of an optimization problem involves a *reduction* from SAT (or any other NP-complete problem) to the problem.

To prove the hardness of approximation, this reduction must produce a *gap* in the value of the optimum.

For instance, proving the NP-hardness of approximating the *maximum clique problem* within a factor g requires coming up with a reduction from SAT to CLIQUE that maps satisfiable formulae to graphs with clique number at least K (for some K), and unsatisfiable formulae to graphs with clique number at most K/g.

For a long time it was unclear how to construct such *gap producing* reductions for CLIQUE and many other important optimization problems.

The Cook-Karp-Levin technique seemed more suited for proving the hardness of decision problems (for example, non-optimization problems such as SAT or TILING) than of optimization problems.

Recent work has yielded a fairly general technique for constructing gap-producing reductions. This new technique, relies upon new probabilistic characterizations of the NP class in terms of interactive proof systems. The most well-known such characterization is the so-called PCP Theorem, written as NP = PCP$(\log n, 1)$.

What's PCP?

Remember Interactive Proof Systems?

A verifier is $(r(n), q(n))$-*restricted* if on each input of size n it uses at most $O(r(n))$ random bits for its computation and queries at most $O(q(n))$ bits of the P-to-V communication tape.

A language L is in PCP$(r(n), q(n))$ if there is an $(r(n), q(n))$-restricted verifier M that probabilistically checks membership proofs for L.

The proof of the PCP Theorem involves complicated algebraic techniques from complexity theory, and will not be given here.

Luckily, understanding the proof is not a prerequisite for using the theorem in inapproximability results.

In particular, we will look at few major inapproximability results, but the reader will only need some familiarity with the basic notions of NP-completeness (the first few chapters of the well-known book by Garey and Johnson provide all necessary background).

In our treatment we will divide problems into two broad classes, based on the approximation ratio that is provably hard to achieve. These approximation ratios are, respectively, $1+\epsilon$ for some fixed $\epsilon > 0$, and $\omega(n)$. Inapproximability results for problems within a class (sometimes also across classes) share common ideas. Note that problems in Class I cannot belong to PTAS, unless P = NP.

MAX 3-SAT

The input is a 3CNF boolean formula $\phi(x_1, x_2, x_3)$, like

$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$

The goal is to find an assignment that maximises the number of satisfied clauses.

Solutions are truth assignments, i.e. assignments of values in {TRUE, FALSE} to the variables in the formula. For instance one could set x_1 and x_2 to FALSE and x_3 to TRUE. We will denote truth-assignments by Greek letters like α and β.

Cost of a truth assignment is the *fraction* of clauses that it satisfies. In the example $c(\phi, \alpha) = 3/4$.

We let $\text{opt}_{\text{MAX3SAT}}(\phi)$ denote the maximum value of the objective function for a 3CNF formula ϕ.

Plan

Reductions from MAX 3-SAT will be used to prove all inapproximability results in this notes. Thus Max 3-SAT plays a role in the theory of inapproximability analogous to the one played by 3SAT in the classical theory of NP-completeness.

First we need to know that MAX 3-SAT itself is NP-hard to approximate! (analogue of the Cook-Levin Theorem).

A very important result

There is a fixed $\epsilon > 0$ and a polynomial time reduction h from SAT to MAX 3-SAT such that for every boolean formula ϕ:

$$\phi \in \text{SAT} \Rightarrow \text{opt}_{\text{MAX3SAT}}(h(\phi)) = 1$$
$$\phi \notin \text{SAT} \Rightarrow \text{opt}_{\text{MAX3SAT}}(h(\phi)) < \frac{1}{1+\epsilon}$$

In other words achieving a ratio $1+\epsilon$ for MAX 3-SAT is NP-hard.

The proof of this result is much beyond the scope of this module. We will however repeatedly invoke this result to prove further complexity results.

Gap-preserving reductions

Let Π and Π' be two maximization problems. A *gap-preserving reduction from* Π *to* Π' *with parameters* $(c, \rho), (c', \rho')$ is a polynomial-time algorithm f. For each instance x of Π, algorithm f generates an instance $x' = f(x)$ of Π' such that the optima of x and x' satisfy the following property.

$$\text{opt}(x) \geq c \Rightarrow \text{opt}(x') \geq c',$$

$$\text{opt}(x) < \tfrac{c}{\rho} \Rightarrow \text{opt}(x') < \tfrac{c'}{\rho'}.$$

Here c and ρ are (positive) values that depend on $|x|$, the size of instance x, and c', ρ' are positive values depending on $|x'|$. Also, $\rho, \rho' \geq 1$.

Remarks

(1) Suppose we wish to prove the inapproximability of problem Π'. If we have a polynomial time reduction h from SAT to Π that ensures, for every boolean formula ϕ, that

$$\phi \in \text{SAT} \Rightarrow \text{opt}(h(\phi)) \geq c, \text{ and}$$

$$\phi \notin \text{SAT} \Rightarrow \text{opt}(h(\phi)) < \tfrac{c}{\rho},$$

then composing this reduction with the gap-preserving reduction gives an algorithm that translates instances of SAT into instances of Π' such that:

$$\phi \in \text{SAT} \Rightarrow \text{opt}(f(h(\phi))) \geq c', \text{ and}$$

$$\phi \notin \text{SAT} \Rightarrow \text{opt}(f(h(\phi))) < \tfrac{c'}{\rho'}.$$

This algorithm shows that achieving an approximation ratio ρ' for Π' is NP-hard. This idea of composing reductions underlies our inapproximability results.

(2) Gap-preserving reductions map solutions to solutions in an obvious way. For instance, given a solution x' of value at least c', a solution to x of value at least c can be produced in polynomial time.

(3) The above definition can be modified in an obvious way when one (or both) of the optimization problems involve minimization

> Let P and Q be two minimisation problems. A *gap-preserving reduction* from P to Q with parameters $(c, \rho), (c', \rho')$ is a polynomial time algorithm f translating each instance x of P into $f(x)$ so that
>
> $$\text{opt}(x) \leq c \Rightarrow \text{opt}(f(x)) \leq c'$$
>
> $$\text{opt}(x) > \rho c \Rightarrow \text{opt}(f(x)) > \rho' c'$$

(4) The gap-preserving reduction could behave arbitrarily on an instance x for which $c/\rho \leq \text{opt}(x) < c$. Thus its "niceness" holds only on a partial domain.

> Papadimitriou and Yannakakis (1991) (see the chapters on approximation algorithms in Papadimitriou textbook) defined an alternative notion of reduction (L-reduction), whose "niceness" can be maintained on *all* instances of Π. An L-reduction, coupled with an approximation algorithm for Π', yields an approximation algorithm for Π. This statement is false for a gap-preserving reduction. On the other hand, for exhibiting merely the hardness of approximation, it suffices (and is usually easier) to find gap-preserving reductions.

(5) The name "gap-preserving" is a bit inaccurate, since the new gap ρ' could be much bigger or much smaller than the old gap ρ.

PTAS vs APX

We start this section with a "familiar" example of a hard problem: CLIQUE. This proof of hardness will be used later to show a stronger inapproximability result for CLIQUE.

For every $\epsilon > 0$, there is a gap-preserving reduction from MAX 3-SAT to CLIQUE that has parameters $(c, 1+\epsilon), (cN/3, 1+\epsilon)$ where N is the number of vertices in the new graph. In other words, CLIQUE does not have a PTAS.

Argument

Let ϕ be a 3-CNF formula in variables $x_1, \ldots, x_n$ with m clauses.

By replicating literals within clauses we can ensure that each clause has 3 literals (e.g., if the clause is x_i then change it to $x_i \wedge x_i \wedge x_i$).

Construct a graph $h(\phi)$ on $3m$ vertices as follows. Represent each clause $(\ell_1 \wedge \ell_2 \wedge \ell_3)$ by a triplet of vertices, one for each literal.

Put no edge between vertices within the same triplet.

If u, v are vertices in two different triplets, put an edge between them iff the literals they stand for are NOT the negations of each other.

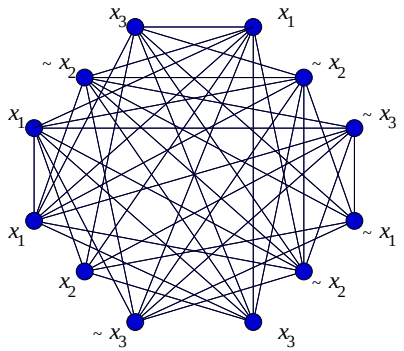
A clique in this graph can contain only one vertex per triple.

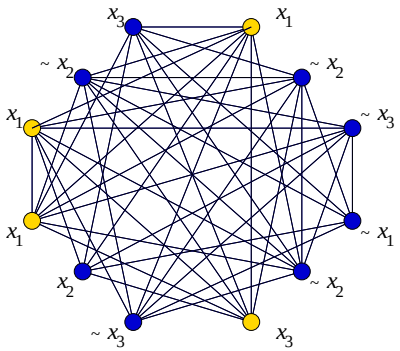
Furthermore, it cannot contain two vertices representing literals that are negations of each other.

In other words, by looking at the literals represented in the clique we can write in a natural way a partial assignment that satisfies as many clauses as there are vertices in the clique. Thus,

$$\text{opt}_{\text{MAX3SAT}}(\phi) = c \Rightarrow \omega(h(\phi)) = cm$$

$$\text{opt}_{\text{MAX3SAT}}(\phi) < \tfrac{c}{1+\epsilon} \Rightarrow \omega(h(\phi)) < \tfrac{cm}{1+\epsilon}$$





15-11

Vertex cover

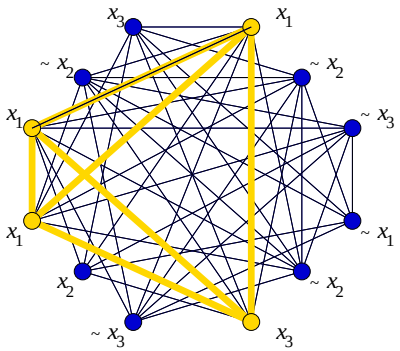
We already know the following relationships

- $\omega(G) = \alpha(\overline{G})$
- $\tau(G) = |V(G)| - \alpha(G)$

Here is the main inapproximability result for vertex cover.

> For every $\epsilon > 0$, there is a gap-preserving reduction from MAX 3-SAT to VERTEX COVER that has parameters $(c, 1+\epsilon), ((3-c)m, 1+\epsilon)$ where m is the number of clauses of the given formula. In other words, VERTEX COVER does not have a PTAS.

16



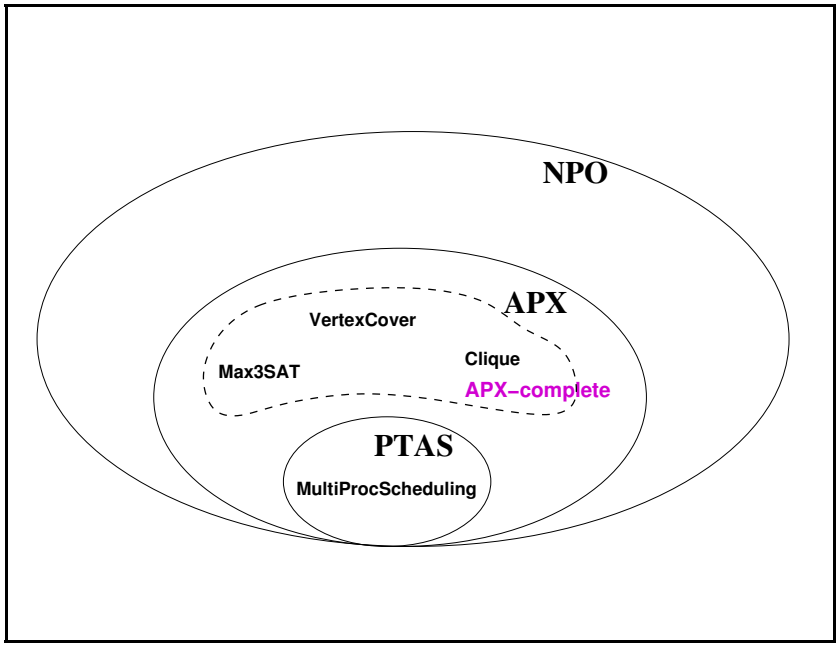
15-12

Argument

Given an instance of the MAX 3-SAT problem ϕ on n variables and m clauses, define a graph G_ϕ whose vertices are ϕ's literals and whose edges join pairs of literals in the same clause and pairs of complementary literals in different clauses (this graph has $3m$ vertices).

Assume now that there exists an assignment α satisfying at least cm clauses of ϕ, then we can pick a distinct literal in G_ϕ from each triplet corresponding to a satisfied clause. Let U be this set. The set $V(G_\phi) \setminus U$ is a vertex cover in G_ϕ of size at most $3m - cm$. Conversely if there exists an assignment α satisfying at most $\frac{cm}{1+\epsilon}$ clauses of ϕ, then the resulting cover (built in exactly the same way) will have size at least $(3 - \frac{c}{1+\epsilon})m$ which is at least $\frac{3-c}{1+\epsilon}m$.

17


NPO
APX
VertexCover
Max3SAT
Clique
APX−complete
PTAS
MultiProcScheduling