Applications of the PCP Theorem to Hardness of Approximation

See lecture slides for definitions of approximation algorithm and approximation ratio.

Lemma 1 *Let L be NP-complete. Suppose that there is a polynomial time mapping from instances x of L to instances x' of a maximization problem Π such that*

$$\begin{aligned} x \in L \Rightarrow &\quad \textit{Opt}(x') = g(x) \textit{ and} \\ x \notin L \Rightarrow &\quad \textit{Opt}(x') < (1-c)g(x), \end{aligned}$$

where $g(x) \in \mathbb{N}$, g is polynomial-time computable, and $0 < c < 1$. If Π has a polynomial-time approximation algorithm A with approximation ratio $1 + c/(1-c)$, then NP=P.

Proof We use A to construct a polynomial time algorithm A_L for L. Given instance x of L, A_L computes x' and runs A on x'. If the value of the solution found by A, namely $A(x')$, is at least $(1-c)g(x)$ then A_L accepts x, otherwise A_L rejects x.

To see that A_L is correct, note that if $x \in L$ then $\text{Opt}(x') = g(x)$. Since A_L has approximation ratio $1 + c/(1-c)$, from the definition of approximation ratio it must be that

$$\text{Opt}(x')/A(x') = g(x)/A(x') \leq 1 + c/(1-c).$$

From this and a little algebra it follows that $A(x') \geq (1-c)g(x)$. In contrast, if $x \notin L$ then by hypothesis of the lemma it must be that $A(x') < (1-c)g(x)$. Thus, A_L is correct.
$\square$

Lemma 2 *(a) Corresponding to any Boolean function of q variables is an equivalent q-CNF formula (i.e. a formula in conjunctive normal form in which each clause has at most q literals) with at most 2^q clauses.*

(b) Corresponding to any q-CNF formula ϕ is a 3CNF formula ϕ' such that ϕ is satisfiable if and only if ϕ' is. Moreover, the number of clauses in ϕ' is at most q times the number of clauses in ϕ.

Proof We first prove part (a). Let F be a Boolean function of q variables $v_1, v_2, \ldots, v_q$. Create a q-DNF formula that has one term for each assignment a of the variables that sets F to 1: the term is $l_1 \wedge l_2 \wedge \ldots \wedge l_q$ where $l_i = v_i$ if $v_i = 1$ in assignment a and $l_i = \bar{v}_i$ if $v_i = 0$ in assignment a. For example, if $q = 3$ and $F(1,0,0) = 1$ then the q-DNF formula contains the term $v_1 \wedge \bar{v}_2 \wedge \bar{v}_3$.

Convert this q-DNF formula into a q-CNF formula inductively as follows. If the q-DNF formula has just one term, it is already in q-CNF form. If it has k terms, inductively convert the formula consisting of just the first $k-1$ terms into q-CNF form to obtain $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ and let $D = l_1 \wedge l_2 \wedge \ldots \wedge l_q$ be the kth term in the q-DNF formula. We need to convert

$$(C_1 \wedge C_2 \wedge \ldots \wedge C_m) \vee (l_1 \wedge l_2 \wedge \ldots \wedge l_q)$$

into q-CNF form. To do this, we will apply one of the distributive laws for Boolean algebras, namely

$$(\phi_1 \wedge \phi_2) \vee \phi_3 \equiv (\phi_1 \vee \phi_3) \wedge (\phi_2 \vee \phi_3).$$

On the first application, we obtain

$$(C_1 \vee (l_1 \wedge l_2 \wedge \ldots \wedge l_q)) \wedge (C_2 \vee (l_1 \wedge l_2 \wedge \ldots \wedge l_q)) \wedge \ldots \wedge (C_m \vee (l_1 \wedge l_2 \wedge \ldots \wedge l_q)).$$

Applying the law m further times, for $1 \leq i \leq m$ we convert the term $C_i \vee (l_1 \wedge l_2 \wedge \ldots \wedge l_q)$ to

$$(C_i \vee l_1) \wedge (C_i \vee l_2) \wedge \ldots \wedge (C_i \wedge l_q).$$

The resulting formula, say ϕ, is in CNF form. To obtain a formula in q-CNF form, if any literal occurs twice in the same clause of ϕ then remove one occurrance of the literal, and if both v_i and $\bar{v}_i$ occur in the same clause then remove both v_i and $\bar{v}_i$. Both of these rules result in an equivalent formula ϕ'.

Finally, we can remove redundant clauses from ϕ' as follows: if the literals of some clause C of ϕ' are a subset of the literals of another clause C', then we can remove C to obtain an equivalent formula. Applying this rule until there are no redundant clauses, we obtain a formula ϕ'' with at most 2^q clauses.

Next we prove part (b). The method for doing this is exactly as used in constructing a 3CNF formula in the proof of the Cook-Levin Theorem. That is, for any clause $(l_1 \vee l_2 \vee \ldots \vee l_r)$ with $r \geq 4$ literals, introduce new variables $y_1, y_2, \ldots y_{r-3}$ to convert the clause to

$$(l_1 \vee l_2 \vee y_1) \wedge (\bar{y}_1 \vee l_3 \vee y_2) \ldots \wedge (\bar{y}_{r-4} \vee l_{r-2} \vee y_{r-3}) \wedge (\bar{y}_{r-3} \vee l_{r-1} \vee l_r).$$

□

Theorem 1 *For some constant $c > 1$ if there is a polynomial time algorithm for Max 3SAT with approximation ratio c then NP=P.*

Proof We will show that for any language L in NP, there is a polynomial time reduction f from L to Max 3SAT and a polynomial time computable function g such that

$$\begin{aligned} x \in L \Rightarrow &\quad \text{Opt}(f(x)) = g(x) \text{ and} \\ x \notin L \Rightarrow &\quad \text{Opt}(f(x)) < (1-c)g(x). \end{aligned}$$

Then from Lemma 1, unless NP=P, there is no polynomial time approximation algorithm for Π with approximation ratio $1 + c/(1-c)$ and the theorem follows.

Let L be in NP and let V be a PCP verifier for L that uses q queries and $r(|x|) = O(\log |x|)$ random bits on any instance x of L. Let the length of the proof provided to V on inputs of length n be $l(n)$ (we can assume without loss of generality that the length of the proof depends only on the input length). We describe a reduction f that, given x, produces a 3-CNF formula ϕ_x that has one variable π_i for for each position i of the proof, $1 \leq i \leq l(|x|)$, plus some additional variables.

For any fixed string τ of random bits of V on x, let $b_{\tau,1}, b_{\tau,2}, \ldots, b_{\tau,q}$ be the positions of the proof that V queries when its random string is τ. Let f_τ be the Boolean formula on q variables that evaluates to 1 on a given assignment to the q variables if and only if the verifier accepts on that assignment of values to its queries, when the verifier's random bit string is τ. By Lemma 2 (a), f_τ can be represented as a q-CNF formula with 2^q clauses over the variables $\pi_{b_{\tau,1}}, \pi_{b_{\tau,2}}, \ldots \pi_{b_{\tau,q}}$. Let ϕ_τ be the 3-CNF formula obtained from f_τ as in Lemma 2 (b). Let ϕ_x be the conjunction of the ϕ_τ for all τ. Each of the formulas ϕ_τ has constant size that depends only on q and not on the length of x. Since the total number of random strings τ is $2^{r(|x|)} = 2^{O(\log(|x|))}$, the size of ϕ_x is polynomial in $|x|$. Moreover, ϕ_x can be constructed in time polynomial in $|x|$ using simulations of the verifier V and the constructions of Lemma 2.

If x is in L then there is a proof π that causes the verifier V to accept with probability 1. Thus there is a truth assignment to the variables $\pi_1, \pi_2, \ldots, \pi_{l(|x|)}$ that can be extended to a satisfying assignment of ϕ_x. Thus if $x \in L$, Opt$(f(x))$ equals the number of clauses in ϕ_x.

If x is not in L then for all proofs π, the verifier V accepts with probability at most $1/2$. Thus any extension of any truth assignment to the variables $\pi_1, \pi_2, \ldots \pi_{l(|x|)}$ satisfies at most half of the formulas ϕ_τ. Therefore, for any truth assignment to ϕ_x, at least one clause in at least half of the ϕ_τ is not satisfied. Since each ϕ_τ has at most $q2^q$ clauses, at most a fraction $1 - 1/(2q2^q)$ of the clauses of ϕ_x are simultaneously satisfiable. Thus if $x \notin L$, Opt$(f(x))$ is at most $1 - 1/(2q2^q)$ times the number of clauses in ϕ_x.

In summary, we have

$$\begin{aligned} x \in L \Rightarrow &\quad \text{Opt}(f(x)) = \text{ number of clauses in } \phi_x \text{ and} \\ x \notin L \Rightarrow &\quad \text{Opt}(f(x)) < (1-c)(\text{ number of clauses in } \phi_x), \end{aligned}$$

where c is any constant less than $1/(2q2^q)$. This completes the proof.
□

Theorem 2 *If there is a polynomial time algorithm for Max Clique with approximation ratio $2 - \epsilon$ for any $\epsilon > 0$, then NP=P.*

Proof The proof has a similar structure to that of Theorem 1. We will show that for any language L in NP, there is a polynomial time reduction f from L to Max Clique and a polynomial time computable function g from positive integers to positive integers such that

$$\begin{aligned} x \in L \Rightarrow &\quad \text{Opt}(f(x)) = g(x) \text{ and} \\ x \notin L \Rightarrow &\quad \text{Opt}(f(x)) < (1-c)g(x). \end{aligned}$$

Then from Lemma 2, unless NP=P, there is no polynomial time approximation algorithm for Π with approximation ratio $1 + c/(1-c)$, and the theorem follows.

Let L be in NP and let V be a PCP verifier for L that uses q queries and $r(|x|) = O(\log |x|)$ random bits on any instance x of L. Let the length of the proof provided to V on inputs of length n be $l(n)$. We describe a reduction f that, given x, produces a graph G_x.

For any given string τ of random bits of V on x, let $b_{\tau,1}, b_{\tau,2}, \ldots, b_{\tau,q}$ be the positions of the proof that V queries when its random string is τ. The nodes of G_x are of the form $(\tau, v_1 v_2 \ldots v_q)$ where each v_i is a bit and the verifier V accepts on instance x and random string τ when the bits of the proof π that are queried are $v_1, \ldots, v_q$, that is, when $\pi_{b_{\tau,1}} = v_1$, $\pi_{b_{\tau,2}} = v_2$, ..., $\pi_{b_{\tau,q}} = v_q$.

Example: Suppose that for language L the verifier makes three queries (on any random string and instance). Fix an instance x of L and suppose that the verifier V uses two random bits on instances of length $|x|$. Suppose furthermore that when the string of random bits is $\tau = 01$, the bits of the proof that are queried by the verifier are at positions $b_{\tau,1} = 2$, $b_{\tau,2} = 7$ and $b_{\tau,3} = 21$. Let the decision of the verifier on random string $\tau = 01$, for each of the eight possible assignments to the bits of the proof that are queried, be given by the following truth table:

proof at position 2	proof at position 7	proof at position 21	verifier's decision on random string $\tau = 01$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Then the nodes $(\tau = 01, 001), (\tau = 01, 100)$, and $(\tau = 01, 111)$ are in the graph.

We say that two nodes $(\tau, v_1 v_2 \dots v_q)$ and $(\tau', v'_1 v'_2 \dots v'_q)$ are *compatible* if, when the verifier queries the same position of the proof on random strings τ and τ' then the value at that position is the same. Formally, nodes $(\tau, v_1 v_2 \dots v_q)$ and $(\tau', v'_1 v'_2 \dots v'_q)$ are compatible if, whenever $b_{\tau,i} = b_{\tau',j}$ then $v_i = v'_j$. The graph G_x has an edge between every pair of compatible nodes. Note that there are no edges between two nodes for the same random string τ.

Example: Continuing with the previous example, suppose that in addition to the nodes $(\tau = 01, 001), (\tau = 01, 100)$, and $(\tau = 01, 111)$, the nodes $(\tau' = 11, 100), (\tau' = 11, 101)$, and $(\tau' = 11, 110)$ are also in graph G_x. Suppose furthermore that when the string of random bits is $\tau' = 11$, the bits of the proof that are queried by the verifier are at positions $b_{\tau',1} = 3$, $b_{\tau',2} = 4$ and $b_{\tau',3} = 7$. Then on both τ and τ', position 7 is queried. Since $b_{\tau,2} = b_{\tau',3} = 7$, there is an edge between nodes $(\tau = 01, v_1 v_2 v_3)$ and $(\tau = 11, v'_1 v'_2 v'_3)$ if and only if $v_2 = v'_3$. Thus, in our example, the following edges are in the graph G_x:

$$\begin{array}{l}
((\tau = 01, 001), (\tau' = 11, 100)), \\
((\tau = 01, 100), (\tau' = 11, 100)), \\
((\tau = 01, 001), (\tau' = 11, 110)), \\
((\tau = 01, 100), (\tau' = 11, 110)), \text{ and} \\
((\tau = 01, 111), (\tau' = 11, 101)).
\end{array}$$

Suppose that $x \in L$. Then there is a proof that causes the verifier to accept with probability 1. Therefore, there is a set of nodes of size $2^{r(|x|)}$ that form a clique in G_x.

Suppose that $x \notin L$. Then all proofs cause the verifier to accept with probability at most 1/2. Therefore, the largest clique in G_x has size at most $2^{r(|x|)-1}$; if there were a larger clique, we could find a proof on which the verifier accepts with probability greater than 1/2.

In summary,

$$\begin{array}{ll}
x \in L \Rightarrow & \text{Opt}(f(x)) = 2^{r(|x|)} \text{ and} \\
x \notin L \Rightarrow & \text{Opt}(f(x)) \geq 2^{r(|x|)-1}.
\end{array}$$

Then from Lemma 2, unless NP=P, there is no polynomial time approximation algorithm for Π with approximation ratio $1 + (1/2)/(1 - (1/2)) - \epsilon = 2 - \epsilon$ for any $\epsilon > 0$ and the theorem follows.
□