E0 224 Computational Complexity Theory

Lecture 26

Lecturer: Chandan Saha
Scribe: Sandip Sinha and Sabareesh R

November 12, 2014

In this lecture, we will state two views of the **PCP** Theorem and show that they are equivalent. We will also show a hardness of approximation result for MAX-IND-SET.

We begin by mentioning a couple of results which led to the PCP Theorem.
Result: (Arora and Safra 1992)
$\mathbf{NP} = \mathbf{PCP}(\log n, \sqrt{\log n})$

Result: (Arora, Lund, Motwani, Sudan and Szegedy 1992)
$\mathbf{NP} = \mathbf{PCP}(\log n, 1)$.

1 Two versions of PCP Theorem:

Theorem 1.1. $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$.

Theorem 1.2. There exists a constant $0 < \rho < 1$ such that for every language L in **NP**, there is a deterministic polynomial-time computable function f that maps strings in $\{0,1\}^*$ to representations of 3CNF formulae such that:

$$x \in L \Rightarrow \text{val}(f(x)) = 1$$
$$x \notin L \Rightarrow \text{val}(f(x)) < \rho$$

Since the output of f on x is a 3CNF formula, we shall denote $f(x)$ by ϕ_x.
We will show in this lecture that Theorem 1.1 is equivalent to Theorem 1.2.
Definition: (q-CSP)
An instance of a q-CSP (CSP is an acronym for Constraint Satisfiability Problem), denoted by ϕ, is a tuple of clauses $\phi = \{\phi_1, \phi_2, ..., \phi_m\}$ such that every clause ϕ_i depends on at most q variables.
In a q-CSP, a clause is not necessarily an $\vee$ of literals. It is a general boolean formula.
Note: In this lecture, we will assume that q is a constant.

Definition: val(ϕ), where ϕ is an instance of a q-CSP, is defined to be the maximum fraction of clauses (i.e. the ϕ_i's) that can be satisfied (by any assignment to the variables).

ϵ-gap q-CSP Problem:
Given a q-CSP ϕ with the promise that:

- either $\text{val}(\phi) = 1$,
- or $\text{val}(\phi) < \epsilon$ (where ϵ is a constant),

find out whether $\text{val}(\phi) = 1$ or $\text{val}(\phi) < \epsilon$.

Given a ϵ-gap q-CSP Problem, we can define languages L_Y and L_N in the natural way:
$L_Y = \{\phi : \phi \text{ is a } q\text{-CSP instance and } \text{val}(\phi) = 1\}$
$L_N = \{\phi : \phi \text{ is a } q\text{-CSP instance and } \text{val}(\phi) < \epsilon\}$
We shall call a q-CSP ϕ a Yes-instance of the ϵ-gap q-CSP Problem if $\phi \in L_Y$ (i.e., if $\text{val}(\phi) = 1$) and a No-instance if $\phi \in L_N$ (i.e., if $\text{val}(\phi) < \epsilon$).

Note: The input size of an ϵ-gap q-CSP Problem is the size of ϕ, which is $\mathcal{O}(m(q \log n)2^q)$, where n is the number of variables and m is the number of clauses in ϕ. Since q is assumed to be a constant, the input size is polynomial in m and n.

Theorem 1.3. There exist constants $\epsilon > 0$ and $q > 0$, such that ϵ-gap q-CSP problem is **NP**-hard.

The meaning of the theorem is as follows:
There exist constants $\epsilon > 0$ and $q > 0$, such that for any language $L \in \mathbf{NP}$, there is a polynomial-time computable function f that maps $x \in \{0,1\}^*$ to an instance of q-CSP, such that f maps $x \in L$ to a string in L_Y, i.e., a Yes-instance of ϵ-gap q-CSP and $x \notin L$ to a string in L_N, i.e., a No-instance of ϵ-gap q-CSP.
We will show that Theorem 1.1 and Theorem 1.2 are equivalent by showing that they are both equivalent to Theorem 1.3.

Theorem 1.4. Theorem 1.1 is equivalent to Theorem 1.3.

Proof. **Theorem 1.1** $\Rightarrow$ **Theorem 1.3**:
Since 3-SAT is **NP**-complete, it is enough to reduce 3-SAT to ϵ-gap q-CSP for some ϵ and q. By Theorem 1.1, 3-SAT has a $(c \log n, d)$-**PCP** verifier V for some constants c and d. Thus, V uses $c \log n$ random coins and makes d queries to the proof y. We fix the input x to be a 3CNF instance of size n. Then the proof size is at most dn^c. For a fixed input x and fixed $r \in \{0,1\}^{c \log n}$, V reads some fixed d bits of the proof y (say $i_1, i_2, ..., i_d$) and its computation is deterministic and takes polynomial time. Hence the output of V on x and r can be thought of as a boolean function in $y_{i_1}, ..., y_{i_d}$. We denote the corresponding boolean formula $\phi_{x,r}$. Now we consider the following reduction:

$$x \to \{\phi_{x,r}\}_{r \in \{0,1\}^{c \log n}}$$

We denote the output of this reduction by ϕ. Clearly, ϕ is a d-CSP instance since for each r, $\phi_{x,r}$ depends on at most d variables. ϕ has size polynomial in n since there are at most n^c choices for r, each of which corresponds to a clause $\{\phi_{x,r}\}$. Moreover, since V runs in polynomial time, this reduction can be computed in polynomial time. Finally, by the completeness and soundness properties of the **PCP**

verifier V, we have:

$$x \in 3SAT \Rightarrow val(\phi) = 1$$
$$x \notin 3SAT \Rightarrow val(\phi) < \frac{1}{2}$$

Corollary: $\frac{1}{2}$-gap d-CSP is **NP**-hard. Thus we have shown Theorem 1.1 implies Theorem 1.3 with $\epsilon = \frac{1}{2}, q = d$.

Theorem 1.3 $\Rightarrow$ Theorem 1.1:
Theorem 1.3 implies that there exist constants $\epsilon > 0, q > 0$ such that ϵ-gap q-CSP is **NP**-hard. This means there is a polynomial time reduction from 3SAT to ϵ-gap q-CSP

$$x \to \phi_x = \{\phi_{x,1}, ..., \phi_{x,m}\}$$

which has the following property:

$$x \in 3SAT \Rightarrow \text{val}(\phi_x) = 1$$
$$x \notin 3SAT \Rightarrow \text{val}(\phi_x) < \epsilon$$

Each clause $\phi_{x,i}$ depends on at most q variables. Further, since the reduction is computable in polynomial time, the number of clauses m is polynomial in size of x.
Let x be a 3CNF formula of size n. We describe a $(\mathcal{O}(\log n), q)$-**PCP**-verifier V for 3SAT:

1. V applies the reduction to ϵ-gap q-CSP.
2. V picks a clause uniformly at random from $\{\phi_{x,1}, ..., \phi_{x,m}\}$, where m is the number of clauses in ϕ_x. It expects the proof to be an assignment to the variables of ϕ_x.
3. Suppose $\phi_{x,i}$ depends on $y_{i_1}, ..., y_{i_q}$. Then the verifier reads q locations $i_1, ..., i_q$ of the proof and finds out if $\phi_{x,i}$ is satisfied by the corresponding assignment to $y_{i_1}, ..., y_{i_q}$. If $\phi_{x,i}$ is satisfied then V accepts x, otherwise V rejects x.

Observation: V is a $(\mathcal{O}(\log n), \mathcal{O}(1))$-**PCP**-verifier for 3SAT.
Since the number of clauses is polynomial in n, $\mathcal{O}(\log n)$ random bits are enough to pick a clause in ϕ_x. Also, the verifier reads q (constantly many) locations of the proof. Finally, V runs in polynomial time since the reduction is computable in polynomial time.

If $x \in 3SAT$, then $\text{val}(\phi_x) = 1$, hence the clause $\phi_{x,i}$ chosen randomly by V must be satisfied. If $x \notin 3SAT$, V will accept x with probability at most ϵ. Since the error is one-sided, this probability can be reduced below $\frac{1}{2}$ by just repeating the process independently a few times. □

Theorem 1.5. Theorem 1.2 is equivalent to Theorem 1.3

Proof. **Theorem 1.2 $\Rightarrow$ Theorem 1.3**:
This is obvious since a 3CNF formula is a special case of a q-CSP instance with $q = 3$ and each clause being an $\vee$ of literals. Let $L \in$ **NP**. By Theorem 1.2, there is a polynomial-time computable reduction from L to 3SAT

$$x \to \phi_x$$

such that

$$x \in L \Rightarrow \text{val}(\phi_x) = 1$$
$$x \notin L \Rightarrow \text{val}(\phi_x) < \rho$$

Since ϕ_x is a 3CNF formula, it is a $3 - CSP$ instance. Therefore, Theorem 1.2 implies Theorem 1.3 if we take $q = 3$ and $\epsilon = \rho$.

Theorem 1.3 $\Rightarrow$ Theorem 1.2:
Suppose ϵ-gap q-CSP is **NP**-Hard.
Consider any $L \in$ **NP**. Then there is a reduction from L to ϵ-gap q-CSP. Let the reduction give m boolean functions, where m is polynomial in the size of the input.
$x \to \phi_x = \{\phi_{x_1}, \phi_{x_2}, \ldots, \phi_{x_m}\}$

$x \in L \implies \text{val}(\phi_x) = 1$
$x \notin L \implies \text{val}(\phi_x) < \epsilon$

Each ϕ_{x_i} is a boolean function that depends on at most q variables. These q variables can be assigned 0 or 1 in 2^q different ways. Consider those assignments at which the function evaluates to 0. Suppose the assignment is $x_1 = a_1, x_2 = a_2, \ldots, x_q = a_q$, then consider the clause given by the disjunction of the variable x_i if $a_i = 0$ and $\bar{x_i}$ if $a_i = 1$. This clause evaluates to true iff the assignment to the variables is not $a_1, \ldots, a_q$. Similarly a clause can be constructed for all the assignments on which the function evaluates to 0. There are at most 2^q clauses for each ϕ_i. Consider the conjunction of all these clauses. This evaluates to 1 iff the assignment to the variables is different from each of those assignments on which the function evaluates to 0. Thus the q-CNF formula computes the function ϕ_i. For each ϕ_i such a formula is constructed and their conjunction is considered. The size of the CNF formula is at most $\mathcal{O}(2^q m)$ which is polynomial in input size if q is a constant. Let π_x be this CNF formula.
If $\text{val}(\phi_x) = 1$, then there is an assignment on which all ϕ_i evaluate to 1 and hence all clauses in π_x are satisfied. So $\text{val}(\pi_x) = 1$.
If $\text{val}(\phi_x) < \epsilon$, then for any assignment at least $(1 - \epsilon)m$ functions in ϕ are not satisfied. So in π_x at least one clause corresponding to each of these ϕ_i is not satisfied. Hence at least $(1 - \epsilon)m$ clauses are not satisfied. The total number of clauses is less than $2^q m$. So at least $\frac{(1-\epsilon)}{2^q}$ of the clauses are not satisfied. Thus $\text{val}(\pi_x) < 1 - \frac{1-\epsilon}{2^q}$. Let $\alpha = \frac{1-\epsilon}{2^q}$.
Now we convert the q-CNF formula π_x to a 3SAT formula ψ_x. Let $\psi_x = c_1 \wedge c_2 \wedge \ldots \wedge c_r$ and $c_i = x_{i1} \vee x_{i2} \vee \ldots \vee x_{ik}$.
We introduce a new variable z_i. Let $c_{i1} = x_{i1} \vee x_{i2} \vee \ldots \vee x_{i(k-2)} \vee z_i$ and
$c_{i2} = x_{i(k-1)} \vee x_{ik} \vee \bar{z_i}$
Now if c_i has a solution, then one of the x_{ij} must be 1. Setting the z_i term in the same clause to 0, both the clauses will be satisfied. Also if the clause is not satisfiable, then all the x_{ij} must be 0 and z_i terms prevent both formulas from being satisfied simultaneously. Hence c_i is satisfiable if and only if $c_{i1} \wedge c_{i2}$ is satisfiable. We can proceed inductively to reduce the clause to a conjunction of clauses with at most 3 literals. Proceeding similarly the entire formula π_x can be written in the 3-CNF form. In each of the new clauses, by construction, at least one of the original literals exists. Thus the new number of clauses is at most $q2^q m$.
Each clause in π_x is represented by atmost q clauses in ψ_x. If $\text{val}(\pi_x) = 1$, then there is a satisfying

assignment for π_x and that assignment alongwith a suitable assignment for the new variables satisfies every clause of ψ_x. Thus $\text{val}(\psi_x) = 1$.
Suppose the number of clauses in π_x is r. If $\text{val}(\pi_x) < 1 - \alpha$, then for every assignment at least αr clauses are not satisfied. Thus atleast one of the q clauses corresponding to each of the αr clauses is not satisfied. The total number of clauses in ψ_x is at most qr. So at least $\frac{\alpha}{q}qr$ clauses are not satisfied or a fraction $\frac{\alpha}{q}$ of the clauses are not satisfied. Hence $\text{val}(\psi_x) < 1 - \frac{\alpha}{q}$. Also, since $\epsilon < 1$, $\alpha > 0$. Let $\rho = 1 - \frac{1-\epsilon}{q2^q}$.
$x \in L \implies \text{val}(\psi_x) = 1$
$x \notin L \implies \text{val}(\psi_x) < \rho$

Thus the membership problem for the language L is reduced to the problem of ρ-approximation of 3SAT. This is true for all languages $L \in \mathbf{NP}$. Hence there is a constant ρ such that the ρ approximation of 3SAT is **NP**-Hard.

□

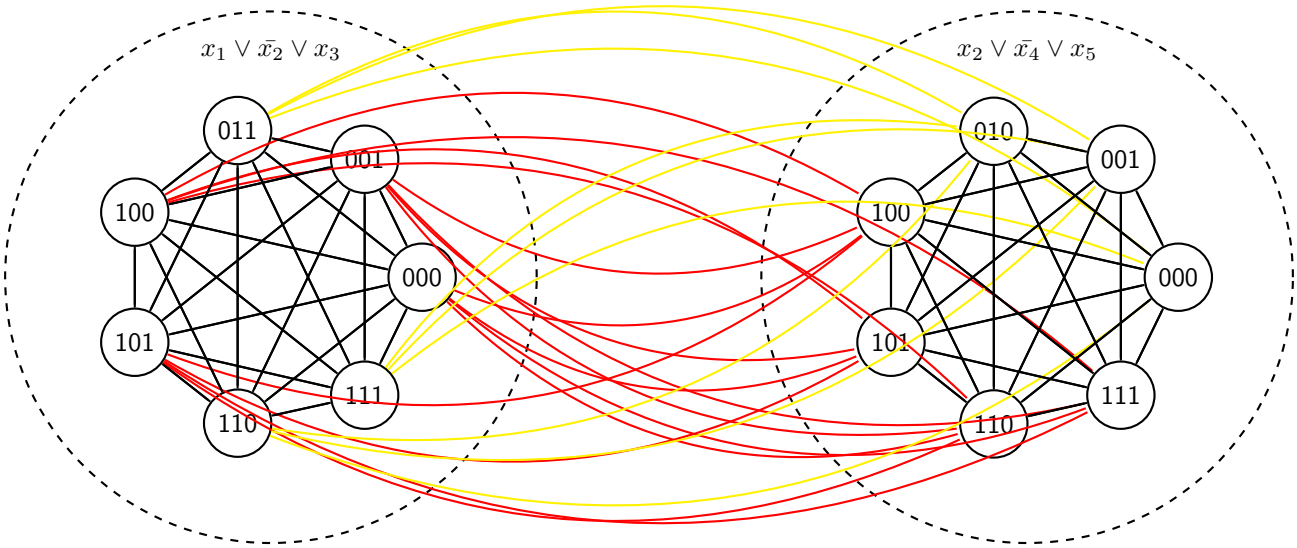
2 Hardness of Approximation

Approximation algorithms are one means of coping with **NP**-complete problems. In problems where in a certain optimum value is to be determined, such as the size of the minimum vertex cover of a graph, there might not be any known efficient algorithms to compute the optimum value. However, we might be able to compute a value that is an approximation of the optimum value. For a maximization problem, we try to find a value that may be less than the optimum but greater than ρ times the optimum value ($\rho < 1$). Similarly, for a minimization problem, we try to find a value that may be more than the optimum but less than ρ times the optimum value ($\rho > 1$).
Now we try to see how good an approximation can we make to the optimum value.

Theorem 2.1 (Hardness of approximation for MAX-IND-SET)**.** There exists a constant $\alpha < 1$ such that an α approximation of MAX-IND-SET is **NP**-Hard.

The theorem can also be stated as, if there is a polynomial time α-approximation algorithm for MAX-IND-SET, then **P**=**NP**.

Proof. Consider ϕ an instance of the 3-CNF SAT problem. Then $\phi = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_m$ where each ϕ_i is of the form $\phi_i = x_{i1} \vee x_{i2} \vee x_{i3}$. Consider the graph G_ϕ constructed from this formula ϕ. There are $7m$ vertices in the graph G_ϕ. The $7m$ vertices are in m clusters of 7 vertices each. Each cluster corresponds to one of the clauses ϕ_i. The 7 vertices in that cluster represent the 7 different assignments of the three variables in the clause ϕ_i that make ϕ_i evaluate to true. There is an edge between any two vertices within the cluster. Also, if two vertices represent two partial conflicting assignments, then there is an edge between them. Two partial assignments are conflicting if they assign the same variable two different values. The graph for the formula $\phi = (x_1 \vee \bar{x_2} \vee x_3) \wedge (x_2 \vee \bar{x_4} \vee \bar{x_5})$ is as follows:



Given the formula ϕ, we can construct the graph by storing along with each vertex the variables in the corresponding clause and the assignment. The $7m$ vertices can be determined in time $\mathcal{O}(m)$ since each clause has at most 8 assignments possible. Between any two vertices, we only need to check if they have a conflicting assignment and to determine if there is an edge between them. Thus the graph can be constructed in time polynomial in the size of the input. Let the number of vertices in the graph be $t = 7m$.

Claim 1. val(ϕ)= |MAX-IND-SET(G_ϕ)|

Proof. Consider any assignment y to the variables in ϕ. Consider all clauses ϕ_i that are satisfied by this assignment. Each of these clauses have a specific value assigned to its variables. Consider those vertices in the graph that correspond to these assignments. There is at most one vertex from each cluster. Also, since this is a valid assignment, no two vertices have a conflicting assignment and hence have no edges between them. Thus this forms an independent set. The size of this independent set is equal to the number of clauses satisfied in the formula ϕ. Let y_o be the assignment that maximises the number of clauses. Corresponding to this assignment also there is an independent set in the graph of size equal to number of satisfied clauses. Hence val(ϕ) $\leq$ |MAX-IND-SET(G_ϕ)|.
Consider the maximum independent set I of the graph G_ϕ. No two vertices of I can lie in the same cluster since there is an edge between any two vertices in the same cluster. Also, no two conflicting vertices in the graph have an edge between them. So every variable in ϕ is assigned only one value. Thus these vertices give an assignment of values for some of the variables in ϕ. The unassigned variables are all set to 0. Consider ϕ with this assignment. All the clauses corresponding to the vertices in the graph are satisfied by this assignment. Hence the number of clauses satisfied is at least the size of the independent set. Thus val(ϕ) $\geq$ |MAX-IND-SET(G_ϕ)|.
Thus val(ϕ)= |MAX-IND-SET(G_ϕ)|. This completes proof of the claim. □

Consider any language $L \in$ **NP**. Then by **PCP** Theorem, we have a reduction from L to 3-CNFSAT given by $x \to \phi_x$ such that
$x \in L \implies \text{val}(\phi_x) = 1$
$x \notin L \implies \text{val}(\phi_x) < \rho$
This reduction is computable in time polynomial in size of the input and the 3-SAT formula is also of size polynomial in the size of the input. Let m be the number of clauses in the formula.

Given the formula ϕ, we can construct the graph G_ϕ in time polynomial in the size of the formula, and hence in time polynomial in size of the input. Also,
$\text{val}(\phi) = 1 \implies |\text{MAX-IND-SET}(G_\phi)| = m$
$\text{val}(\phi) < \rho \implies |\text{MAX-IND-SET}(G_\phi)| < \rho m$

Combining the two relations above:
$x \in L \implies |\text{MAX-IND-SET}(G_\phi)| = m$
$x \notin L \implies |\text{MAX-IND-SET}(G_\phi)| < \rho m$

Suppose there is a TM M that gives a ρ-approximation for MAX-IND-SET problem. Then
$\text{val}(\phi) = 1 \implies |\text{MAX-IND-SET}(G_\phi)| > \rho m$
$\text{val}(\phi) < \rho \implies |\text{MAX-IND-SET}(G_\phi)| < \rho m$

Consider a TM N that takes an instance x of the language L, converts it to the 3SAT formula and then to the corresponding graph in polynomial time. It then simulates M on the graph constructed. If the output of M is greater than ρm, it outputs 1 and otherwise 0. Thus $x \in L \iff M(x) = 1$. The running time for the machine N is polynomial in the size of the input since each step takes polynomial time. Hence this implies that $L \in \mathbf{P}$. This is true for all $L \in \mathbf{NP}$.
Thus if there is a ρ-approximation for MAX-IND-SET, then $\mathbf{P} = \mathbf{NP}$.
Hence there is a constant α such that an α approximation of MAX-IND-SET is **NP**-Hard.

□

Theorem 2.2. For any constant $\alpha \leq 1$, α-approximation of MAX-IND-SET is **NP**-Hard.

Proof. From the previous theorem we have that a ρ-approximation of MAX-IND-SET is **NP**-Hard.
Given a graph G=(V,E) consider the map $G \to G^k$. The vertices in G^k correspond to all the subsets of vertices of G of size k. So G^k has $\binom{|V|}{k}$ vertices. Let the vertices be labelled $S_1, S_2, ...$ There is an edge between S_i and S_j iff the set $S_i \vee S_j$ is not an independent set.

Claim 2. $|\text{MAX-IND-SET}(G^k)| = \binom{|\text{MAX-IND-SET}(G)|}{k}$ where $\binom{i}{j} = 1$ if $i < j$.

Proof of claim: Suppose I is a maximum independent set of the graph G. Then any k sized subset of this set is also an independent set. Union of two such subsets is also independent. Thus the $\binom{|I|}{k}$ sets form an independent set in G^k. If $|I| < k$, then all sets of k vertices are dependent. So, $|\text{MAX-IND-SET}(G^k)| = 1 = \binom{|I|}{k}$. Thus, $|\text{MAX-IND-SET}(G^k)| \geq \binom{|\text{MAX-IND-SET}(G)|}{k}$.
Suppose $J = \{J_1, J_2, ..., J_r\}$ is a maximum independent set of G^k. Suppose $r = 1$, then all $k+1$ size sets of vertices are dependent. So $|\text{MAX-IND-SET}(G)| < k+1$. So, $|\text{MAX-IND-SET}(G^k)| = 1 = \binom{|\text{MAX-IND-SET}|}{k}$.
Suppose $r > 1$. Let $V = J_1 \cup J_2 \cup \ldots \cup J_r$. Suppose V is not an independent set in G. Suppose there is an edge between $u \in V$ and $v \in V$. $u \in J_i$ and $v \in J_j$ for some i and j. If $i \neq j$, then $J_i \cup J_j$ is not independent and hence J is not an independent set in G^k. If $i = j$, $J_i \cup J_k$ is not independent for any $k \leq r, k \neq i$ and so J is not an independent set in G^k. Since the set J is given to be independent in G^k, V is also an independent set in G. Let L be the set of all k sized subsets of V. Then $|L| = \binom{|V|}{k}$. Also all vertices in J are a subset of L by the definition of V. So $|J| \leq |L|$. Also, $|V| \leq |\text{MAX-IND-SET}|(G)$.
Thus $|\text{MAX-IND-SET}(G^k)| \leq \binom{|\text{MAX-IND-SET}(G)|}{k}$.
Hence, $|\text{MAX-IND-SET}(G^k)| = \binom{|\text{MAX-IND-SET}(G)|}{k}$.
This completes the proof of the claim.

Let $L \in \mathbf{NP}$. From the reduction used in the proof of the previous theorem we have that,
$x \in L \implies |\text{MAX-IND-SET}(G)| = t$
$x \notin L \implies |\text{MAX-IND-SET}(G)| < \rho t$
where $7t$ is the number of vertices in G.
So, $x \in L \implies |\text{MAX-IND-SET}(G^k)| = \binom{t}{k}$
$x \notin L \implies |\text{MAX-IND-SET}(G^k)| < \binom{\rho t}{k}$

$$\begin{aligned}
\frac{\binom{\rho t}{k}}{\binom{t}{k}} &= \frac{(\rho t)!\; k!\; (t-k)!}{k!\; (\rho t - k)!\; t!} \\
&= \frac{(\rho t)!\; (t-k)!}{(\rho t - k)!\; t!} \\
&= \frac{(\rho t)\; (\rho t - 1)\; (\rho t - 2) \ldots (\rho t - k + 1)}{t\; (t-1)\; (t-2) \ldots (t-k+1)} \\
&= \rho^k \frac{(t)\; (t - \frac{1}{\rho})\; (t - \frac{2}{\rho}) \ldots (t - \frac{k-1}{\rho})}{t\; (t-1)\; (t-2) \ldots (t-(k-1))} \\
&\leq \rho^k \qquad \text{(as } t - \frac{i}{\rho} < t - i, \text{ since } \rho < 1)
\end{aligned}$$

Hence, $x \notin L \implies |\text{MAX-IND-SET}(G^k)| < (\rho)^k \binom{t}{k}$.
Thus the problem has been reduced to the ρ^k-approximation of MAX-IND-SET.
For any constant α, there is a constant k such that $\rho^k < \alpha$. An α-approximation is also a ρ^k approximation. Thus the α-approximation of MAX-IND-SET is **NP**-Hard for all constants α.

□

References

[1] S. Arora and B. Barak, "Computational Complexity: A Modern Approach", Cambridge University Press, 2009