MIP = NEXP and the Nonapproximability of CLIQUE

This is the proof that was presented in class on November 4, 2010. Throughout, points that you are encouraged to think through and justify in detail are marked by **"(WHY?)."**

Recall that "multi-prover interactive proof systems" recognize the same set of languages as "oracle proof systems." More precisely, $L \in$ MIP if and only if there is a probabilistic polynomial-time oracle Turing Machine M such that:

$$x \in L \Rightarrow \exists O \text{ s.t. } M^O(x) = 1 \text{ with probability 1, and}$$
$$x \notin L \Rightarrow \forall O \; M^O(x) = 1 \text{ with probability less than } \frac{1}{4}.$$

Next, recall that the *clique number* of a graph G is the largest integer k such that G contains a complete subgraph on k vertices. Computing the clique number is, of course, an NP-hard problem, because the set $\{(G, k)$ such that G contains a clique of size $k\}$ is NP-complete. A function f is said to *c-approximate* the function g if, for all x, $g(x)/c \leq f(x) \leq c \cdot g(x)$.

Theorem: If the clique number function can be 2-approximated in polynomial time, then EXP = NEXP.

Let L be an arbitrary language in NEXP. Because NEXP = MIP, there is a probabilistic polynomial-time machine M that serves as the "verifier" in an oracle proof system as above. We use the following notion of a proof-system *transcript* to construct a graph $G_{M,x}$ for each $x \in \{0,1\}^*$ that may or may not be in L. Let r be a random coin-toss sequence that M uses on input x; because this is an oracle (i.e., non-adaptive) proof system, x and r uniquely determine the sequence $(q_1, a_1, \ldots, q_t, a_t)$ of M's oracle queries and the answers to these queries. **(WHY?)** Similarly, M's output $b \in \{\text{ACCEPT}, \text{REJECT}\}$ is uniquely determined by x, r, and $(q_1, a_1, \ldots, q_t, a_t)$. A transcript of M's execution on input x is a sequence of the form $(r, q_1, a_1, \ldots, q_t, a_t, b)$. We say that $(r, q_1, a_1, \ldots, q_t, a_t, b)$ is a *valid transcript* if x and r determine the queries, answers, and output $(q_1, a_1, \ldots, q_t, a_t, b)$. Note that, for $x \in \{0,1\}^n$, there is an upper bound $m = \text{poly}(n)$ on the length of r, as well as an upper bound $N = \text{poly}(n)$ on the total length of a transcript.

Consider the following deterministic exponential-time reduction from the decision problem "Is x in L?" to the optimization problem "what is the clique number of $G_{M,x}$?" The vertices of $G_{M,x}$ are the $2^N = 2^{\text{poly(n)}}$ possible transcripts of M's execution on input x. Let $S^1 = (r^1, q_1^1, a_1^1, \ldots, q_t^1, a_t^1, b^1)$ and $S^2 = (r^2, q_1^2, a_1^2, \ldots, q_t^2, a_t^2, b^2)$. The edge $\{S^1, S^2\}$ is in $E(G_{M,x})$ if and only if both are valid transcripts, $b^1 = b^2 =$ ACCEPT, and the two transcripts are *consistent* in that $a_i^1 = a_i^2$ whenever $q_i^1 = q_i^2$.

If $x \in L$, the fact that there is an O such that $M^O(x) = 1$ with probability 1 implies that the clique number of $G_{M,x}$ is 2^m. **(WHY?)** If $x \notin L$, the fact that there is no O such that $M^O(x) = 1$ with probability at least 1/4 implies that the clique number of $G_{M,x}$ is less than 2^{m-2}. **(WHY?)**

Thus, if there were a deterministic, polynomial-time algorithm that 2-approximated the clique-number function, we would have NEXP = EXP; the following deterministic, exponential-time algorithm would decide membership in L, where L is an arbitrary language in NEXP. To determine whether $x \in L$, first construction $G_{M,x}$; this can be done in deterministic exponential time. **(WHY?)** If the clique number of $G_{M,x}$ is less than 2^{m-1}, output "no"; if the clique number of $G_{M,x}$ is at least 2^{m-1}, output "yes."