ON UNAPPROXIMABLE VERSIONS OF NP-COMPLETE PROBLEMS*

DAVID ZUCKERMAN†

Abstract.
We prove that all of Karp's 21 original NP-complete problems have a version that's hard to approximate. These versions are obtained from the original problems by adding essentially the same, simple constraint. We further show that these problems are absurdly hard to approximate. In fact, no polynomial-time algorithm can even approximate $\log^{(k)}$ of the magnitude of these problems to within any constant factor, where $\log^{(k)}$ denotes the logarithm iterated k times, unless NP is recognized by slightly superpolynomial randomized machines. We use the same technique to improve the constant ϵ such that MAX CLIQUE is hard to approximate to within a factor of n^{ϵ}. Finally, we show that it is even harder to approximate two counting problems: counting the number of satisfying assignments to a monotone 2-SAT formula and computing the permanent of -1,0,1 matrices.

Key words. NP-complete, unapproximable, randomized reduction, clique, counting problems, permanent, 2SAT

AMS subject classifications. 68Q15, 68Q25, 68Q99

1. Introduction.

Previous Work. The theory of NP-completeness was developed in order to explain why certain computational problems appeared intractable [10, 14, 12]. Yet certain optimization problems, such as MAX KNAPSACK, while being NP-complete to compute exactly, can be approximated very accurately. It is therefore vital to ascertain how difficult various optimization problems are to approximate.

One problem that eluded attempts at accurate approximation is MAX CLIQUE. This is the problem of finding $\omega(G)$, the size of a largest clique in the graph G. There was no explanation for this until Feige, et.al. [11] showed that for all $\epsilon > 0$ no polynomial-time algorithm can approximate $\omega(G)$ to within a factor of $2^{(\log n)^{1-\epsilon}}$, unless $N\tilde{P} = \tilde{P}$, where $\tilde{P}$ denotes quasi-polynomial time, or $TIME(2^{polylog})$. This was based on the proof that MIP=NEXP [5]. Recently, there have been several improvements, culminating in the result that approximating $\omega(G)$ to within a factor of $n^{1/4-o(1)}$ is NP-complete [4, 3, 7].

A New Role for Old Reductions. It is natural and important to identify other NP-complete problems that are hard to approximate. In the original theory of NP-completeness, polynomial-time reductions were used. Yet these reductions might not preserve the quality of an approximation well, so researchers focused on reductions that preserved the quality of approximation very closely [18, 17]. Using such reductions, Panconesi and Ranjan [17] defined a class RMAX(2) of optimization problems, of which MAX CLIQUE is one natural complete problem. The intractability of approximating MAX CLIQUE implies that the other RMAX(2)-complete problems are intractable to approximate. Recently, Lund and Yannakakis

* Based on "NP-Complete Problems Have a Version That's Hard to Approximate" by D. Zuckerman, which appeared in *Proceedings of the 8th IEEE Conference on Structure in Complexity Theory*, San Diego, 1993, pp. 305-312.

† Dept. of Computer Sciences, The University of Texas at Austin, Austin, TX 78712, `diz@cs.utexas.edu`. Most of this research was done while the author was supported by an NSF Postdoctoral Fellowship at MIT. At UT Austin, the author was partially supported by NSF NYI Grant No. CCR-9457799.

[16] used an approximation-preserving reduction to show the intractability of approximating CHROMATIC NUMBER.

Here we show that the reductions can have a much more general form and still yield unapproximability results. This is essentially because the factors are so huge, namely n^ϵ, so polynomial blow-ups will only change this to $n^{\epsilon'}$. We show that Karp's original reductions can be modified to have this general form, and hence the original 21 NP-complete problems presented in [14] all have a version that's hard to approximate. This gives evidence that all NP-complete problems have a version that's hard to approximate.

The Small Jump to Unapproximability. What does it mean that an NP-complete problem has a version that's hard to approximate? Indeed, an NP-complete problem is a language recognition problem, and may not even have a corresponding optimization problem; moreover, many corresponding optimization problems are easy to approximate. Intuitively, however, it seems reasonable that by adding sufficiently many constraints to an optimization problem it becomes "harder," i.e. hard to approximate. Quite surprisingly, we show that by adding one, simple constraint that is essentially the same for every NP-complete problem, all of Karp's original NP-complete problems become unapproximable.

What is this constraint? Usually we can only form maximization problems when our NP language L is of the form $(x, k) \in L \iff (\exists y, f(y) \geq k)p(x, y)$ for some polynomial-time predicate p and function f. The corresponding optimization problem is then taken to be $\max_{y:p(x,y)} f(y)$. Of course, if L is of the same form except has $f(y) \leq k$, then we end up with the minimization problem $\min_{y:p(x,y)} f(y)$. For example, with the NP-complete language VERTEX COVER, x is a graph and y is a set of vertices; p is the predicate that y forms a vertex cover in x and $f(y)$ is the size of y. Thus the language is "does there exist a vertex cover of size k?" while the optimization problem is to find the size of a minimum vertex cover.

Although we do not prove a general theorem for all NP-complete languages, the constraint we add makes sense for any NP language. This is because we use the basic representation of an NP language L as $x \in L \iff (\exists y \in \{0,1\}^m)p(x, y)$, where m is polynomial in the length of x. Note that for languages that can be expressed as in the previous paragraph, the x here is what was previously the ordered pair (x, k), and p here is what was previously $p(x, y) \wedge (f(y) \geq k)$ (or with minimization problems, $f(y) \leq k$). The constraint we add is as follows. Using the natural correspondence between $\{0,1\}^m$ and subsets of $\{1, \ldots, m\}$, we view y as a subset of $\{1, \ldots, m\}$. We have as an additional input a subset $S \subseteq \{1, \ldots, m\}$, and the output should be $\max_{y \in \{0,1\}^m : p(x,y)} |S \cap y|$. We also insist that there be a y such that $p(x, y)$; otherwise, by taking $S = \{1, \ldots, m\}$, deciding whether the maximum is 0 or not is equivalent to deciding whether there exists a y such that $p(x, y)$. We even give such a y for free, as an additional input.

Thus, from the easily approximable minimization problem VERTEX COVER, we obtain the unapproximable constrained maximization version: given a graph $G = (V, E)$, $S \subseteq V$, a positive integer k, and a vertex cover of size at most k, find the maximum of $|S \cap C|$ over vertex covers C of size at most k. This can be interpreted as follows: the set S represents the important vertices; we can only afford a vertex cover of size at most k, but wish to use as many important vertices as possible.

For HAMILTONIAN CIRCUIT, this constrained version becomes: given a graph $G = (V, E)$, a Hamiltonian circuit in G, and $S \subseteq E$, find the maximum of $|S \cap C|$ over Hamiltonian circuits C. This has a natural interpretation: a salesman has to visit all cities, and certain trips between cities are scenic or free, so he wants to maximize the number of such trips.

Hyper-Unapproximability Results. Not only are these versions hard to approximate to within a factor of n^ϵ, but it is also hard to have any idea about the order of magnitude of the solutions to these optimization problems. More specifically, we show that any iterated logarithm of any of the above versions is hard to approximate within a constant factor, unless NP is recognized by slightly-superpolynomial randomized machines (slightly-superpolynomial will be made precise in the next paragraph). The proof also does not rely on the fact that the iterated logarithm may become 0 (or negative); we can assume the iterated logarithm is at least 1. This result extends the result in [22] that the logarithm of $\omega(G)$ is hard to approximate to within any constant factor, unless $N\tilde{P} = \tilde{P}$.[1]

In order to state our results precisely, define

$$\log^{(k)} n = \underbrace{\log\log\ldots\log}_{k} n,$$

$$p_{e,k}(n) = 2^{2^{\cdot^{\cdot^{2^{e\log^{(k)} n}}}}} \Big\} k \text{ 2's},$$

$$P_k = \bigcup_{e>0} TIME(p_{e,k}(n)),$$

with analagous definitions for NP_k, RP_k, $co\text{-}RP_k$, and $ZPP_k = RP_k \cap co\text{-}RP_k$. Note that $P_1 = P$ is polynomial-time, $P_2 = \tilde{P}$ is quasi-polynomial time, and P_k for $k > 2$ are other measures of slightly-superpolynomial time. Also let $F\mathcal{C}$ denote the functional version of the complexity class $\mathcal{C}$. In particular, $FZPP_k$ corresponds to functions computable by zero-error (Las Vegas) randomized algorithms that run in the appropriate expected time. For a function f, the notation $f(MAXCLIQUE)$ denotes the problem of finding $f(\omega(G))$, and similarly for other optimization problems. We show that if $NP_k \neq ZPP_k$, then no function in $FZPP_k$ approximates $\log^{(k)}$ of any of the above versions of NP-complete problems (e.g. MAX CLIQUE) to within any constant factor.

These techniques can also be used to improve the constant ϵ such that MAX CLIQUE cannot be approximated to within a factor n^ϵ. Suppose c answer bits are required by a PCP protocol to achieve error 1/2. We show that if $NP \neq ZPP$, then for all $\epsilon < 1/(c+1)$ there is no Las Vegas algorithm running in expected polynomial time which approximates MAX CLIQUE to within a factor n^ϵ. Recently much effort has been devoted towards improving the constant (see e.g. [6, 7]), and they all use this lemma or an extension of it.

We point out that similar results may be obtained by using the randomized graph product method of Berman and Schnitger [8]. However, such results would be under the stronger assumption that $NP_k \neq BPP_k$. The reason for this is that we look at the proof-theoretic construction of the graphs in question, while Berman and Schnitger use a straight reduction. We therefore need only deal with the error in the "easy" direction, while Berman and Schnitger need to worry about the error in both directions.

[1] This does not entirely improve upon [22]; here we show that $\log\omega(G)$ is hard to approximate unless $NP = ZPP$, a condition which as far as we know does not imply $N\tilde{P} = \tilde{P}$.

This difference also manifests itself in the derandomization: more work is needed to derandomize the randomized graph product construction [2] than the basic tool used to derandomize the proof-theoretic construction [1].

Implications for Counting Problems. We further show that under the same assumption that $NP_k \neq ZPP_k$, $\log^{(k+1)}$ of the number of satisfying assignments to a monotone 2-SAT formula is hard to approximate to within any constant factor. That this is hard to approximate may seem surprising, because finding a satisfying assignment is trivial. In the case of a DNF-formula, where finding a satisfying assignment is also easy, approximating the number of satisfying assignments is in randomized polynomial-time [15].

As a corollary, we use Valiant's reduction [21] to observe that approximating $\log^{(k+1)}$ of the permanent of a matrix with entries in $\{-1, 0, 1\}$ is hard under the same assumption as above. We can assume the matrix has positive permanent, because conceivably the problem of deciding if the permanent is 0 is NP-hard, which would make the corollary uninteresting. This result should be contrasted with the subexponential algorithm to approximate the permanent of 0,1-matrices [13].

2. The Iterated Log of Max Clique is Hard to Approximate. In this section we show that it is hard to approximate any iterated logarithm of the size of the maximum clique. We first define

DEFINITION 2.1. Approximating $g(x)$ to within a factor $a(n)$ is in $FZTIME(t(n))$ *if there is a zero-error (Las Vegas) randomized algorithm which, on input x, runs in expected time $t(|x|)$ and outputs $\tilde{g}$ such that $g(x)$ is in the half-open interval $I = [\tilde{g}, a(|x|)\tilde{g})$. Here $|x|$ denotes the length of x.*

Thus the algorithm can distinguish between x and y, $|x| = |y| = n$, if $g(x) \geq a(n)g(y)$ or $g(y) \geq a(n)g(x)$.[2]

Our proofs closely follow the proofs of [11, 4, 3], building on the work of [5]. First some definitions from [4]:

A verifier is a probabilistic polynomial-time probabilistic Turing Machine M given access to the input x, random bits y, and a proof Π. The verifier's goal is to decide whether Π is a valid proof that x is in some language L. We define the predicate $M^{\Pi}(x, y)$ to be true iff M accepts x given the proof Π and random bits y.

DEFINITION 2.2. *A verifier is $(r(n), c(n))$-restricted if on an input of size n it uses at most $r(n)$ random bits and queries at most $c(n)$ bits of the proof.*

DEFINITION 2.3. *A language L is in the complexity class $PCP(r(n), c(n))$ iff there is an $(r(n), c(n))$-restricted verifier such that*

$$\begin{aligned} x \in L &\Rightarrow (\exists \Pi) Pr_y[M^{\Pi}(x, y)] = 1 \\ x \notin L &\Rightarrow (\forall \Pi) Pr_y[M^{\Pi}(x, y)] < 1/2. \end{aligned}$$

Arora et.al. [3] give the following improvement of [5]:

THEOREM 2.4. *[3] $NP = PCP(O(\log n), O(1))$.*

Using this, they follow [11] and construct a graph G_x which has a large clique iff $x \in L$. In order to do this, they define transcripts and a notion of consistency among

[2] Of course, our definition is also equivalent to the algorithm always outputting a number $\tilde{g}$ such that $g(x)/b(|x|) \leq \tilde{g} < b(|x|)g(x)$, where $b(n) = \sqrt{a(n)}$. One might think it is natural to define this as approximating to within a factor $b(n)$; however, we want the property that the algorithm can distinguish between g values that differ by an $a(n)$ factor. Otherwise, we will get $b(n)^2$ terms instead of $a(n)$ terms. Our definition is also like the ones used in [11, 3].

them. A transcript is basically a set of queries to locations of the proof and the bits that are found in these locations. Two transcripts are consistent if there is one proof that can correspond to both transcripts.

DEFINITION 2.5. *[11] We say* $< y, q_1, a_1, \ldots, q_c, a_c >$ *is an* (r, c)-transcript *for verifier* M *on* x *if* $|y| = r$*, and on input* x *and random string* y*, for every* i M *queries bit location* q_i *after receiving the answers* a_j *to queries* q_j *for* $j < i$*. The transcript is* accepting *if on input* x*, random string* y*, and history of communication (questions and answers)* $< q_1, a_1, \ldots, q_c, a_c >$*,* M *accepts* x*.*

DEFINITION 2.6. *[11] We say that two transcripts* $< y, q_1, a_1, \ldots, q_c, a_c >$ *and* $< \hat{y}, \hat{q_1}, \hat{a_1}, \ldots, \hat{q_c}, \hat{a_c} >$ *are consistent if for every* i*,* $q_i = \hat{q_i}$ *implies* $a_i = \hat{a_i}$*.*

To decide whether x is in some NP language L, we construct a graph G_x based on the $(r(n) = O(\log n), c(n) = O(1))$-restricted verifier M for L. The vertices of G_x are all accepting $(r(n), c(n))$-transcripts of M on x, and two nodes are connected iff the corresponding transcripts are consistent. Thus G_x has at most $2^{r(n)+c(n)}$ vertices. It is also not hard to see:

LEMMA 2.7. *[11]* $\omega(G_x) = \max_\Pi Pr_y[M^\Pi(x, y)] \cdot 2^{r(n)}$*.*

In other words, $\omega(G_x)$ is the maximum over all proofs Π of the number of random strings on which M accepts x. Thus, if $x \in L$ then $\omega(G_x) = 2^{r(n)}$, and if $x \notin L$ then $\omega(G_x) < 2^{r(n)}/2$.

In order to get a wider separation in the clique sizes, Feige et.al. constructed the graph G'_x corresponding to a protocol M'. M' runs $\log^{O(1)} n$ independent iterations of M on x. This reduces the error probability if $x \notin L$ and therefore produces a wider separation in the clique sizes.

Yet once we fix a proof Π, M^Π basically corresponds to a *co-RP* machine: always accepting when $x \in L$ and usually rejecting if $x \notin L$. Thus it is natural to use pseudo-random strings that efficiently amplify the success probability of an *RP* (or *co-RP*) algorithm. Indeed, this was the idea used in [22] to show that approximating $\log \omega$ is hard. Arora, et.al. [3] later used this idea to achieve their result as well.

But since we will cycle through all possibilities of the random seeds, the pseudo-random strings do not have to be constructible in the usual sense. In fact, the best amplification schemes are given by random graphs, which are so-called "dispersers" with high probability. Thus, our plan will be to use a random amplification scheme.

DEFINITION 2.8. *An* (R, r, d)*-amplification scheme is a bipartite graph* $H = (\{0,1\}^R \cup \{0,1\}^r, E)$*, where* $\{0,1\}^R$ *and* $\{0,1\}^r$ *are independent sets, and the degree of every node in* $\{0,1\}^R$ *is* d*.*

An (R, r, d)-amplification scheme defines a pseudo-random generator that takes as input an R-bit string z and outputs the d r-bit neighbors of z. A good amplification scheme has been called a disperser [9]:

DEFINITION 2.9. *An* (m, n, d, a, b)*-disperser is a bipartite graph with* m *nodes on the left side, each with degree* d*, and* n *nodes on the right side, such that every subset of* a *nodes on the left side has at least* b *neighbors on the right.*

We pick an (R, r, d)-amplification scheme uniformly at random by choosing independently, for each $u \in \{0,1\}^R$, d uniformly random elements from $\{0,1\}^r$ as the neighbors of u. Santha [19] and Sipser [20] have shown that a random amplification scheme is a disperser. In order to get the optimal ϵ in the n^ϵ results we use an extremely minor modification of their arguments:

LEMMA 2.10. *The probability that a uniformly random* $(R, r, R+2)$*-amplification scheme is a* $(2^R, 2^r, R+2, 2^r, 2^{r-1})$*-disperser is greater than* $1 - 2^{-2^r}$*.*

Proof. We basically follow [20]. For $S \subseteq \{0,1\}^R$, $T \subseteq \{0,1\}^r$, let $A_{S,T}$ be the event that all neighbors of S are in T. Then the probability that the amplification scheme is not the desired disperser equals

$$Pr\Big[\bigcup_{\substack{|S|=2^r\\|T|=2^{r-1}-1}} A_{S,T}\Big] \leq \sum_{\substack{|S|=2^r\\|T|=2^{r-1}-1}} Pr[A_{S,T}] = \binom{2^R}{2^r}\binom{2^r}{2^{r-1}-1}\left(\frac{2^{r-1}-1}{2^r}\right)^{(R+2)2^r}$$

$$< 2^{R2^r} \cdot 2^{2^r} \cdot 2^{-(R+2)2^r} = 2^{-2^r}.$$

□

The unapproximability of the iterated log will follow from the following lemma.

LEMMA 2.11. *Let $L \in PCP(r(n) = O(\log n), c)$. Let $R = R(n)$ be a function of n, and let $N = N(n) = 2^{R+(R+2)c}$. If there is a Las Vegas algorithm running in expected time $t(N)$ which can correctly determine whether a graph with at most N nodes has a clique of size at least 2^R or at most 2^r, then $L \in$ co-$RTIME(t(N(n)) + p(N(n)))$ for some polynomial p.*

Proof. We describe a randomized algorithm A to recognize L. Let x be the input. Let M be the $(r = r(n), c)$-restricted verifier accepting L. A picks a uniformly random $(R, r, R+2)$ amplification scheme H. Define the verifier V as the machine which picks a uniformly random R-bit string, determines its neighbors $y_1, \ldots, y_{R+2}$ in H, and simulates M on x with the random strings $y_1, \ldots, y_{R+2}$. V accepts if and only if all $R+2$ runs of M accept. Note that V uses R random bits and $dc = (R+2)c$ answer bits. A constructs G'_x corresponding to V. Then G'_x has at most $N = 2^{R+(R+2)c}$ vertices. Observe that if $x \in L$ then $\omega = \omega(G'_x) = 2^R$. Now consider if $x \notin L$. Let Π be an arbitrary proof. With overwhelming probability (in particular, at least 5/6), H is a $(2^R, 2^r, d, 2^r, 2^{r-1})$-disperser. Since less than 2^{r-1} r-bit strings cause M to accept, by the disperser property of H at most 2^r R-bit strings cause V to accept. Thus $\omega \leq 2^r$.

Let B be a Las Vegas algorithm running in expected time $t(N)$ which correctly determines whether $\omega \geq 2^R$ or $\omega \leq 2^r$. A simulates B for $3t(N)$ steps (in which time B fails to halt with probability at most 1/3). If B doesn't halt or determines that $\omega \geq 2^R$, then A accepts. Otherwise A rejects. Thus A always accepts if $x \in L$, and with probability $\geq 5/6 - 1/3$ rejects if $x \notin L$. A runs in time $O(t(N) + p(N))$, where p is a polynomial depending on the running time of M. □

We also make use of a complexity-theoretic lemma:

LEMMA 2.12. *For any positive integer k, $NP \subseteq co - RP_k$ if and only if $NP_k = ZPP_k$.*

Proof. One direction is easy: if $NP_k = ZPP_k$, then $NP \subseteq NP_k = ZPP_k \subseteq co - RP_k$. Now assume $NP \subseteq$ co-RP_k. Since $p_{d,k}(p_{e,k}(n)) \leq p_{de,k}(n)$, this implies that $NP_k \subseteq$ co-RP_k. Taking complements, we get co-$NP_k \subseteq RP_k$. But then $NP_k \subseteq$ co-$RP_k \subseteq$ co-$NP_k \subseteq RP_k \subseteq NP_k$. Thus all containments are equalities and $NP_k = RP_k =$ co-RP_k; hence, $NP_k = ZPP_k$. □

We can now show:

THEOREM 2.13. *If for any constant a approximating $\log^{(k)} \omega$ to within a factor a is in $FZPP_k$, then $NP_k = ZPP_k$.*

Proof. Let $L \in PCP(r(n) = O(\log n), c)$ be NP-complete. Set $R = p_{a,k-1}(r)$ and apply Lemma 2.11. Suppose there is an algorithm A approximating $\log^{(k)} \omega$ to within a factor a. Since $\log^{(k)} 2^R = a \cdot \log^{(k)} 2^r$, A can determine whether $\omega \geq 2^R$

or $\omega \leq 2^r$ in a graph on $N = 2^{R+(R+2)c}$ vertices. For n large enough so $R \geq 2c$, $N \leq 2^{(c+2)R} = 2^{(c+2)p_{a,k-1}(r)} \leq 2^{(c+2)p_{a,k-1}(\log n)} \leq p_{a(c+2),k}(n)$. Thus, for some constant e, A runs in time $p_{e,k}(N) \leq p_{ea(c+2),k}(n)$ on inputs of length n. Lemma 2.11 now implies the theorem, except that the conclusion is $NP \subseteq$ *co-RP*$_k$ instead of $NP_k = ZPP_k$. Lemma 2.12 shows that these conclusions are equivalent. ☐

Similarly, we improve the constant ϵ in the n^ϵ of the MAX CLIQUE unapproximability results of [3]:

THEOREM 2.14. *Let c be a constant such that some NP-complete language is in $PCP(O(\log n), c)$ (which exists by Theorem 2.4). Then for any constant $\epsilon < 1/(c+1)$, there is no Las Vegas algorithm running in expected polynomial time that approximates MAX CLIQUE to within a factor n^ϵ, unless $NP = ZPP$.*

Proof. Choose k large enough so that $(k-1)/(k+(k+2)c) \geq \epsilon$, and let $R = kr$. By Lemma 2.11, ω cannot be approximated to within a factor of 2^{R-r} in a graph with $N = 2^{R+(R+2)c}$ vertices, unless $NP \subseteq$ *co-RP*. By our choice of R this factor is at least N^ϵ. Moreover, by Lemma 2.12, $NP \subseteq$ *co-RP* is equivalent to $NP = ZPP$. ☐

3. Unapproximable Versions of NP-Complete Problems. We now modify Karp's list of 21 NP-complete problems to obtain versions that are hard to approximate. Problems 4 and 11 had previously been shown to be as difficult to approximate as MAX SAT [17].

THEOREM 3.1. *For each of the following maximization problems A, there exists a constant $\epsilon > 0$ such that A cannot be approximated to within a factor n^ϵ in polynomial time unless $P = NP$. For any positive constant c, positive integral k, and any of the following maximization problems A, approximating $\log^{(k)}(A)$ to within a factor of c is not in $FZPP_k$ unless $NP_k = ZPP_k$.*

1. **CONSTRAINED MAX SAT**
   See MAX 2-ANLSAT.
2. **MAX 0-1 INTEGER PROGRAMMING**
   Input: integer matrix C and integer vector d
   Output: The maximum, over all 0-1 vectors x such that $Cx \geq d$, of the number of 1's in x.
3. **MAX CLIQUE**
   Input: undirected graph G
   Output: The maximum number of vertices in a clique.
4. **MAX SET PACKING**
   Input: family of finite sets $\{S_j\}$, $j \in \{1, \dots, n\}$
   Output: The maximum, over all $I \subseteq \{1, \dots, n\}$ such that the $S_i, i \in I$ are disjoint, of $|I|$.
5. **CONSTRAINED MAX VERTEX COVER**
   Input: undirected graph $G = (V, E)$, $S \subseteq V$, and a vertex cover of size k
   Output: The maximum, over vertex covers C of size k, of $|C \cap S|$.
6. **CONSTRAINED MAX SET COVERING**
   Input: family of finite sets $\{S_i\}$, $i = 1, \dots, n$, $T \subseteq \{1, \dots, n\}$, and a subcover of size k, (i.e. $J \subseteq \{1, \dots, n\}$, $|J| = k$, such that $\cup_{j \in J} S_j = \cup_{i=1}^n S_i$)
   Output: The maximum, over subcovers C of size k, of $|C \cap T|$.
7. **CONSTRAINED MAX FEEDBACK NODE SET**
   Input: digraph $G = (V, E)$, $S \subseteq V$, and a feedback node set of size k (i.e. a subset $R \subseteq V$ of size k that contains a vertex of every directed cycle)
   Output: The maximum, over feedback node sets C of size k, of $|C \cap S|$.

8. **CONSTRAINED MAX FEEDBACK ARC SET**
Input: digraph $G = (V, E)$, $S \subseteq E$, and a feedback arc set of size k (i.e. a subset $R \subseteq E$ of size k that contains an edge of every directed cycle)
Output: The maximum, over feedback arc sets C of size k, of $|C \cap S|$.
9. **CONSTRAINED MAX DIRECTED HAMILTONIAN CIRCUIT**
Input: digraph $G = (V, E)$, $S \subseteq E$, and a Hamiltonian circuit in G
Output: The maximum, over Hamiltonian circuits $C \subseteq E$, of $|C \cap S|$.
10. **CONSTRAINED MAX HAMILTONIAN CIRCUIT**
Input: undirected graph $G = (V, E)$, $S \subseteq E$, and a Hamiltonian circuit in G
Output: The maximum, over Hamiltonian circuits $C \subseteq E$, of $|C \cap S|$.
11. **MAX 2-ANLSAT**
Input: 2-CNF formula F with all variables negated
Output: The maximum, over all satisfying assignment x, of the number of variables set to "true" in x.
12. **CONSTRAINED MAX CHROMATIC NUMBER**
Input: graph $G = (V, E)$, $v \in V$, $S \subseteq V$, and a k-coloring of G
Output: The maximum, over all k-colorings $\mathcal{C}$ of G, of $|C \cap S|$, where C is the set of vertices in the same color class as v in $\mathcal{C}$.
13. **CONSTRAINED MAX CLIQUE COVER**
Input: graph $G = (V, E)$, $v_0 \in V$, $S \subseteq V$, and a clique cover of G of size at most k, i.e. a representation of G as the union of at most k cliques
Output: The maximum, over all clique covers $\mathcal{C}$ of G of size at most k, of $|C \cap S|$, where C is the clique containing v in $\mathcal{C}$.
14. **CONSTRAINED MAX EXACT COVER**
Input: family of finite sets $\{S_i\}$, $i = 1, \ldots, n$, $T \subseteq \{1, \ldots, n\}$, and an exact cover, (i.e. $J \subseteq \{1, \ldots, n\}$ such that the S_j, $j \in J$ are disjoint, and $\cup_{j \in J} S_j = \cup_{i=1}^{n} S_i$)
Output: The maximum, over exact covers C, of $|C \cap T|$.
15. **CONSTRAINED MAX HITTING SET**
Input: family of subsets $\{S_i\}$ of $\{i = 1, \ldots, n\}$, $T \subseteq \{1, \ldots, n\}$, and a hitting set, (i.e. $W \subseteq \{1, \ldots, n\}$ such that for all i, $|W \cap \{1, \ldots, n\}| = 1$)
Output: The maximum, over hitting sets C, of $|C \cap T|$.
16. **CONSTRAINED MAX STEINER TREE**
Input: undirected graph $G = (V, E)$, $S \subseteq E$, and a Steiner tree of weight at most k with respect to $R \subseteq V$ and weighting function $w : E \to Z$ (i.e. a subtree of weight at most k containing the set of nodes in R)
Output: The maximum, over Steiner trees $T \subseteq E$ of weight at most k with respect to R and w, of $|T \cap S|$.
17. **CONSTRAINED MAX 3-DIMENSIONAL MATCHING**
Input: hypergraph $H = (V, F)$, $F \subseteq V \times V \times V$, $S \subseteq F$, and a 3-dimensional matching (i.e. $M \subseteq F$, $|M| = |V|$ and no two elements of M agree in any coordinate)
Output: The maximum, over 3-dimensional matchings N, of $|N \cap S|$.
18. **CONSTRAINED MAX KNAPSACK**
Input: $(a_1, a_2, \ldots, a_n) \in Z^n$, $T \subseteq \{1, \ldots, n\}$, and a knapsack of size b (i.e. an $S \subseteq \{1, \ldots, n\}$, $\sum_{j \in S} a_j = b$)
Output: The maximum, over knapsacks C of size b, of $|C \cap T|$.

19. **CONSTRAINED MAX JOB SEQUENCING**
Input: "execution time vector" $(T_1, \ldots, T_n) \in Z^n$
"deadline vector" $(D_1, \ldots, D_n) \in Z^n$
"penalty vector" $(P_1, \ldots, P_n) \in Z^n$
$S \subseteq \{1, \ldots, n\}$, and a schedule (permutation) π with penalty at most k, i.e.

$$\sum_j [\text{if } T_{\pi(1)} + \ldots + T_{\pi(j)} > D_{\pi(j)} \text{ then } P_{\pi(j)} \text{ else } 0] \leq k$$

Output: The maximum, over schedules with penalties of at most k, of the number of jobs in S completed by the deadline.

20. **CONSTRAINED MAX PARTITION**
Input: $(a_1, a_2, \ldots, a_n) \in Z^n$, $k \in \{1, \ldots, n\}$, $T \subseteq \{1, \ldots, n\}$, and an equal partition (i.e. an $S \subseteq \{1, \ldots, n\}$, $\sum_{j \in S} a_j = \sum_{j \notin S} a_j$)
Output: The maximum, over equal partitions C with $k \in C$, of $|C \cap T|$.

21. **CONSTRAINED MAX CUT**
Input: undirected graph $G = (V, E)$, $v \in V$, $T \subseteq V$, and a cut of weight at least W with respect to the weighting function $w : V \to Z$ (i.e. a set $S \subseteq V$ such that

$$\sum_{\{u,v\} \in E, u \in S, v \notin S} w(\{u, v\}) \geq W)$$

Output: The maximum, over cuts C of weight at least W with $v \in C$, of $|C \cap T|$.

Note that the above languages are all of the following similar form. Let p be a polynomial-time predicate corresponding to an NP language L, so that $x \in L$ iff $(\exists y \in \{0,1\}^m) p(x, y)$, where m is polynomial in n. Let $S \subseteq \{1, \ldots, m\}$, and view y as a subset of $\{1, \ldots, m\}$. Then the maximization problems above correspond to maximizing $|S \cap y|$ over y such that $p(x, y)$, given such a y.

We now consider when reductions between two such maximization problems preserve the difficulty of approximation.

Lemma 3.2. *Suppose L is a language of the above form, where approximating $\beta = \max\{|S \cap y|\}$, given some y such that $p(x, y)$, to within a factor n^ϵ is hard, for some $\epsilon > 0$, and approximating $\log^{(k)} \beta$ to within any constant factor is hard. Let q be a polynomial-time reduction such that $x \in L$ if and only if $x' = q(x) \in L'$; moreover, given y such that $p(x, y)$ in polynomial-time one can compute y' such that $p'(x', y')$. Suppose that there is an $S' \subseteq \{1, \ldots, m'\}$ such that*

$$\begin{aligned} & (\exists y) p(x, y) \text{ and } |S \cap y| = k \\ \iff \quad & (\exists y') p'(x', y') \text{ and } |S' \cap y'| = k. \end{aligned}$$

Then approximating $\beta' = \max\{|S' \cap y'|\}$, given some y' such that $p'(x', y')$, to within a factor $n^{\epsilon'}$ is hard for some $\epsilon' > 0$, and approximating $\log^{(k)} \beta'$ to within any constant factor is hard.

Proof. The lemma follows because $\beta' = \beta$ and $|x'| = poly(|x|)$. □

We can now prove the theorem. We first observe as in [17] that approximating MAX 2-ANLSAT is as hard as approximating MAX CLIQUE.

LEMMA 3.3. *[17] For any functions f, g, approximating $f(MAXCLIQUE)$ to within a factor $g(n)$ is polynomial time reducible to approximating $f(MAX\ 2\text{-}ANLSAT)$ to within a factor $g(n)$.*

Proof. The proof is contained in the proof of Theorem 4.1. □

Proof of Theorem 3.1. We basically use the sequence of reductions given by Karp [14] that the unconstrained versions of the above problems are NP-complete. Lemma 3.3 tells us that the constrained version of 2SAT is hard to approximate. Moreover, for 2SAT, we can easily compute a satisfying assignment if one exists. Next, for most of the problems above, we can look at the reductions in [14] and verify that they satisfy the conditions of Lemma 3.2. There are some reductions, however, for which the reductions in [14] will not work. For example, Karp reduces CLIQUE to VERTEX COVER by taking complements. This would yield a minimization problem. Instead, we use the reduction given in [12] which goes directly from 3SAT, and we can let S be the subset of vertices which Garey and Johnson call u_i.

To show the result for HAMILTONIAN CIRCUIT requires some care. We modify the reduction given in [12] reducing VERTEX COVER to HAMILTONIAN CIRCUIT. We briefly outline their reduction. Say we have an instance of VERTEX COVER: a graph $G = (V, E)$ and an integer k. They construct $G' = (V', E')$ as follows. V' consists of k "selector vertices" $A = \{a_1, \ldots, a_k\}$ plus other vertices corresponding to edges in G. E' is constructed in such a way that G' has a Hamiltonian circuit iff G has a vertex cover of size k. Each a_i has the same adjacency list, and there are no edges between any two a_i.

Our reduction is from MAX INDEPENDENT SET to CONSTRAINED MAX HAMILTONIAN CIRCUIT. Given an instance $G = (V, E)$, $|V| = n$, of MAX INDEPENDENT SET, construct G' using the reduction from VERTEX COVER above with the parameter $k = n$. Form G'' by adding the edges $\{a_i, a_j\}$ for each pair of selector vertices $(a_i, a_j), i < j$. Let S be the edges $\{a_i, a_j\}$ of this clique A. Since there is always a vertex cover of size n in G, there will always be a Hamiltonian circuit C in G' and hence in G''. The construction of [12] ensures that C can be found efficiently. The input to CONSTRAINED MAX HAMILTONIAN CIRCUIT is G'', S, and C.

We show that the output of CONSTRAINED MAX HAMILTONIAN CIRCUIT is α, the size of a maximum independent set in G. That is, we show that there is a Hamiltonian circuit passing through α edges of S, and no Hamiltonian circuit passing through $\alpha + 1$ edges of S. We use the fact that the size of a minimum vertex cover is $n - \alpha$. Since there is a vertex cover of size $n - \alpha$ in G, there is a Hamiltonian circuit in G'' which passes through α edges in S. Namely this is the Hamiltonian circuit in [12] with $a_{n-\alpha}$ replaced by the path $a_{n-\alpha}, a_{n-\alpha+1}, \ldots, a_n$. Note that we can make this replacement since each a_i is connected to the same vertices outside A. Conversely, suppose there is a Hamiltonian circuit in G'' passing through $\alpha + 1$ edges in S. Since each a_i has the same adjacency list outside A, by contracting these edges we see that there is a Hamiltonian circuit passing through $n - \alpha - 1$ selector vertices in the original construction of [12], and hence there is a vertex cover of size $n - \alpha - 1$, a contradiction. □

4. Two Unapproximable Counting Problems. In this section we show how difficult it is to approximate the number of satisfying assignments to a monotone 2CNF formula, or equivalently, a 2CNF formula where all variables are negated. As a corollary, we deduce the hardness of approximating the permanent of a matrix with $\{-1, 0, 1\}$ entries.

THEOREM 4.1. *There exists $\epsilon > 0$ such that if the log of the number of satisfying assignments to a monotone 2CNF can be approximated to within a factor of n^ϵ, then $NP = P$. If for some constant a approximating $\log^{(k+1)}$ of the number of satisfying assignments to a monotone 2CNF to within a factor a is in $FZPP_k$, then $NP_k = ZPP_k$.*

Proof. The proof extends the reduction in [17]. Let $G = (\{1, \ldots, n\}, E)$ be a graph with maximum clique size $\omega > 1$, and consider the formula $F = \bigwedge_{\{i,j\} \notin E}(\bar{x_i} \vee \bar{x_j})$. Viewing an assignment x as a subset S_x of $\{1, \ldots, n\}$, we see that x satisfies F iff S_x forms a clique in G. Thus the number N of satisfying assignments to F is equal to the number of cliques in G. Since any subset of the max clique is a clique, $N \geq 2^\omega$. Since each clique has size at most ω,

$$N \leq \binom{n}{\omega} + \binom{n}{\omega - 1} + \ldots + \binom{n}{0} \leq n^\omega.$$

Therefore $\omega \leq \lg N \leq \omega \lg n$, so $\lg N / \lg n \leq \omega \leq \lg N$. Thus if $\lg N$ can be approximated to within a factor a, then ω can be approximated to within a factor $a \lg n$. Observing that the additional $\lg n$ factor is negligible in the proof of Theorem 2.13 completes the proof. $\square$

As a corollary, using Valiant's reduction [21] we can show that computing the permanent of matrices with entries in $\{-1, 0, 1\}$ is hard.

COROLLARY 4.2. *If the log of the permanent of a matrix having positive permanent and entries in $\{-1, 0, 1\}$ can be approximated to within a factor of n^ϵ, then $NP = P$. If for some constant a approximating $\log^{(k+1)}$ of the permanent of a matrix having positive permanent and entries in $\{-1, 0, 1\}$ to within a factor a is in $FZPP_k$, then $NP_k = ZPP_k$.*

Proof. Valiant [21] showed that the number of satisfying assignments to a 3CNF formula, and hence 2CNF formula, can be expressed as the permanent of a -1,0,1 matrix. $\square$

Acknowledgements. I am grateful to Silvio Micali for several extremely useful discussions, and to Laci Babai for observing that the conclusion $NP \subseteq BPP$ appearing in the preliminary version could be strengthened to $NP = ZPP$. I also thank Shafi Goldwasser and Mario Szegedy for helpful discussions, and the anonymous referees for a careful reading and many valuable comments.

REFERENCES

[1] M. Ajtai, J. Komlos, and E. Szemeredi, *Deterministic Simulation in Logspace*, 19th Annual ACM Symposium on Theory of Computing, 1987, pp. 132-140.

[2] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman, *Derandomized graph products*, Computational Complexity, to appear.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, *Proof Verification and Intractibility of Approximation Problems*, 33rd Annual IEEE Symposium on Foundations of Computer Science, 1992, pp. 14-23.

[4] S. Arora and S. Safra, *Approximating Clique is NP-Complete*, 33rd Annual IEEE Symposium on Foundations of Computer Science, 1992, pp. 2-13.

[5] L. Babai, L. Fortnow, and C. Lund, *Non-Deterministic Exponential Time Has Two-Prover Interactive Protocols*, Computational Complexity, 1 (1991), pp. 16-25.

[6] M. Bellare, S. Goldwasser, C. Lund, and A. Russell, *Efficient probabilistically checkable proofs and applications to approximation*, 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 294-304.

[7] M. Bellare and M. Sudan, *Improved Non-approximability Results*, 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 184-193.

[8] P. Berman and G. Schnitger, *On the Complexity of Approximating the Independent Set Problem*, Information and Computation 96 (1992), pp. 77-94.

[9] A. Cohen and A. Wigderson, *Dispersers, Deterministic Amplification, and Weak Random Sources*, 30th Annual IEEE Symposium on Foundations of Computer Science, 1989, pp. 14-19.

[10] S.A. Cook, *The Complexity of Theorem-Proving Procedures*, 3rd Annual ACM Symposium on Theory of Computing, 1971, pp. 151-158.

[11] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, M. Szegedy, *Approximating Clique is Almost NP-Complete*, 32nd Annual IEEE Symposium on Foundations of Computer Science, 1991, pp. 2-12.

[12] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[13] M. Jerrum and U. Vazirani, *A Mildly Exponential Approximation Algorithm for the Permanent*, 33rd Annual IEEE Symposium on Foundations of Computer Science, 1992, pp. 320-326.

[14] R.M. Karp, *Reducibility Among Combinatorial Problems*, in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., 1972, pp. 85-103.

[15] R.M. Karp, M. Luby, and N. Madras, *Monte-Carlo Approximation Algorithms for Enumeration Problems*, J. of Algorithms, 10(3) (1989), pp. 429-448.

[16] C. Lund and M. Yannakakis, *On the Hardness of Approximating Minimization Problems*, 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 286-293.

[17] A. Panconesi and D. Ranjan, *Quantifiers and Approximation*, 22nd Annual ACM Symposium on Theory of Computing, 1990, pp. 446-456.

[18] C.H. Papadimitriou and M. Yannakakis, *Optimization, Approximation, and Complexity Classes*, Journal of Computer and System Sciences, 43 (1991), pp. 425-440.

[19] M. Santha, *On Using Deterministic Functions to Reduce Randomness in Probabilistic Algorithms*, Information and Computation, 74 (1987), pp. 241-249.

[20] M. Sipser, *Expanders, Randomness, or Time versus Space*, Journal of Computer and System Sciences, 36 (1988), pp. 379-383.

[21] L.G. Valiant, *The Complexity of Computing the Permanent*, Theoretical Computer Science, 8 (1979), pp. 189-201.

[22] D. Zuckerman, *Simulating BPP Using a General Weak Random Source*, Algorithmica, to appear. Preliminary version in the 32nd Annual IEEE Symposium on Foundations of Computer Science, 1991, pp. 79-89.