

The NP-Completeness Column: An Ongoing Guide

DAVID S. JOHNSON

AT&T Bell Laboratories, Murray Hill, New Jersey 07974

This is the 23rd edition of an irregularly appearing column that covers new developments in the theory of NP-completeness. The presentation is modeled on that used by M. R. Garey and myself in our book “Computers and Intractability: A Guide to the Theory of NP-Completeness,” W. H. Freeman & Co., New York, 1979 (hereinafter referred to as “[G&J]”; previous columns will be referred to by their dates). A background equivalent to that provided by [G&J] is assumed, and, when appropriate, cross-references will be given to that book and the list of problems (NP-complete and harder) presented there. Readers who have results they would like mentioned (NP-hardness, PSPACE-hardness, polynomial-time-solvability, etc.) or open problems they would like publicized, should send them to David S. Johnson, Room 2D-150, AT&T Bell Laboratories, Murray Hill, NJ 07974 (or to dsj@research.att.com). Please include details, or at least sketches, of any new proofs; full papers are preferred. If the results are unpublished, please state explicitly that you are willing for them to be mentioned. Comments and corrections are also welcome. For more details, see the December 1981 issue of this Journal.

THE TALE OF THE SECOND PROVER

This will be the first column to appear in four years, not counting a brief appearance in March 1990 that simply provided a table of contents to the 22 columns that had appeared by that time. I now hope to resume my regular appearances in these pages, starting out at a rate of two columns per year. There will be much to catch up on. In this issue I’d like to concentrate on a remarkable sequence of results that has grown out of the subject I covered in the last “real” column [September, 1988], a sequence that has so far resulted in at least two articles in *The New York Times* [45,46]. That four-year-old column was about complexity results related to the then-new concept of an “interactive proof.” Since then, surprise has followed surprise in this area, leading to unexpected consequences for many aspects of the theory of NP-completeness, from the complexity of approximation algorithms to the definition of NP itself. A major step in the story was what at first seemed a sheer flight of intellectual fancy: the introduction of a second prover into the model of interaction. It is this new character who gives this column its title.

But first we'll take a look at what became of the original single prover. Section 1 begins by quickly summarizing the state of our knowledge as described in that 1988 column and reminding the reader of the relevant definitions, results, and conjectures. It then describes how those conjectures were upset by a series of results (the first having nothing to do with interactive proofs) that lead to the conclusion that $IP = PSPACE$. The proof techniques used here, involving "arithmetization" and the "low-degree polynomial trick," laid the foundation for the much more complicated proofs that were to follow, and I shall say a bit about them.

Section 2 turns to the more general idea of a *multi-prover* interactive proof and its sister concept, the *probabilistically checkable proof*. Again there are connections to previously defined complexity classes, and I will explain these and cover ongoing research aimed at refining them. This is a very active area, and some of the most exciting results still exist only in manuscript form.

One reason this area is so active is the recent discovery that results about multi-prover interactive proofs can have important corollaries in the seemingly unrelated area of approximation algorithms. These corollaries resolve long-standing open problems (and, sad to say, render obsolete one oft-cited paper of G&J [31]). A first corollary is that for any constant c , it is as hard to find a clique whose size is within a factor c of optimal as it is to find an optimal clique itself (an NP-hard problem). A second corollary is that, assuming $P \neq NP$, no MAX-SNP-hard optimization problem can have a polynomial-time approximation scheme. Still further corollaries, all announced in the few weeks since I began writing this column, relate to graph coloring, set covering, and a general class of maximum subgraph problems. Section 3 will describe these results, and explain how the clique and MAX-SNP results are derived (as well as what, precisely, the latter result means).

1. THE SEARCH FOR IP

Recall the basic idea of an interactive proof: The cast of characters consists of an all-powerful prover P and a more computationally limited verifier V . Each is equipped with a secret source of unbiased random bits and a copy of a given statement whose truth is in question. (The names of P and V are often expanded to *Pat* and *Vanna* in honor of the hosts of *Wheel of Fortune*, a quiz show on North American television where a related interactive verification process takes place nightly.) In *our* type of interaction, Pat and Vanna take turns sending each other messages (strings of bits). The number and semantics of these messages are governed by some agreed-upon protocol. After a finite number of rounds the conversation halts, and Vanna is required to say *true* or *false*. It is the goal of the prover to make the verifier say *true*, no matter whether the given statement is true or not. The verifier wants to determine the truth of the statement but is willing to tolerate a small probability of being wrong. That is, some

sequences of her random bits may lead her into conversations in which she gets fooled, but such sequences should be rare.

This scenario is adapted to recognizing languages (and hence to defining complexity classes) as follows: An *interactive proof system* (prover P , verifier V) recognizes a language L if (a) for any string x in L the verifier V always says *true* when x is the input to the conversation, and (b) for any string x not in L , neither P nor any imposter P' can make V say *true* with probability greater than $1/4$. The class IP consists of all those languages recognizable by *polynomial-time* interactive proof systems, i.e., ones in which the overall computation time of the verifier is bounded by a polynomial in the input length. Note that this imposes similar bounds on the number of rounds, the lengths of the messages, and the number of random bits she sees. Note also that by repeating the conversation many times using new random bits each time, the verifier can reduce her probability of error to arbitrarily low levels (k repetitions reduces it to $1/4^k$.)

The class IP and the concept of interactive proof system were independently proposed in 1985 by Goldwasser, Micali, and Rackoff [37] and Babai [4,9]. (The original definition only required that the verifier be correct $3/4$ of the time when $x \in L$, but it was shown in [30] that requiring that she always be correct in this situation did not affect the set of languages she could recognize.) Examples of systems for recognizing particular languages are given in those references and in the [September, 1988] column.

A major question left open by those papers was the precise relationship between IP and the other classes known to be located in the vicinity of NP. Clearly IP contains NP, since the prover can simply send the verifier the traditional “short proof” of membership, which she can verify in polynomial time, with no need to send messages herself or use her random bits. Moreover, the containment appears to be proper. The GRAPH NON-ISOMORPHISM problem (given graphs G and H , is it the case that they are *not* isomorphic?) is not known to be in NP but does have a polynomial-time interactive proof system [36], as described in the [September, 1988] column. Just how much bigger IP was than NP remained a major open problem in 1988. The obvious question was the relationship between IP and the polynomial hierarchy $\text{PH} = \cup_{i=1}^{\infty} \Sigma_i^P$, where $\Sigma_1^P = \text{NP}$ and Σ_{i+1}^P is the set of all languages recognizable by a polynomial-time NDTM with an oracle for a Σ_i^P -complete language. There were oracles under which IP contained problems that weren't in PH, as well as oracles under which co-NP (and hence PH) contained problems not in IP [1,29]. This seemed to suggest that proofs of either $\text{IP} \subseteq \text{PH}$ or $\text{PH} \subseteq \text{IP}$ were beyond our reach (given the conventional wisdom that all our standard proof techniques for dealing with complexity classes continue to work if an oracle is present, and so only results that hold true for all oracles can be proved). The best one could say was that $\text{IP} \subseteq \text{PSPACE}$, since it could be shown that the “all-powerful” prover P need be no more powerful than a polynomial-space Turing Machine [27,55].

At this point in our story, a new and seemingly unrelated character wandered

onto the stage. It turns out that IP was not the only class trapped between NP and PSPACE whose relationship to PH remained undetermined. Consider the class $P^{\#P}$ consisting of all those languages recognizable by a polynomial-time Turing machine with an oracle for a function in #P. (Recall that a function is in #P if it corresponds to the number of accepting computations of some polynomial-time nondeterministic Turing machine (NDTM) [67].) It is easy to see that $NP \subseteq P^{\#P}$, since if one can count the number of accepting computations, one can certainly tell whether that number is greater than zero. Similarly, $P^{\#P} \subseteq PSPACE$, since polynomial space suffices to simulate all runs of a polynomial-time NDTM. Our knowledge in 1988 about the relation between $P^{\#P}$ and PH was similar to that for IP, in that for all we knew $P^{\#P}$ and PH were incomparable.

It was the $P^{\#P}$ question that fell first. In 1989, the Japanese graduate student Seinosuke Toda published the unexpected result that $PH \subseteq P^{\#P}$ [66]. In a future column, I hope to have more to say about this result and the recent flowering of complexity results about exact and approximate counting problems. For now, however, the key observation is that Toda's result provided a new angle from which to attack the IP versus PH question. How does IP relate to $P^{\#P}$?

Note that if our goal is to show that $PH \subseteq IP$, all we now need show is that some #P-complete function F can be computed using an interactive proof system. More precisely, all we need show is that there is an interactive proof system in which, given an element e of the domain of F , the prover sends the verifier the value of $F(e)$ and then proves it correct. Shortly after Toda's result was published, Lund, Fortnow, Karloff, and Nisan [52] did just this. Building on earlier work of Beaver and Feigenbaum [10], Lipton [49], Blum, Luby, and Rubinfeld [19], and others, they constructed an interactive proof system for the famous #P-complete problem of computing the *permanent* of a square 0-1 matrix [67]. I would like to say a bit about how this sort of construction works, as it will allow me to illustrate in simple form two basic techniques that lie at the core of the whole resulting line of research. For my purposes, however, it will be more useful to sketch an interactive proof system for a different #P-complete problem, #3SAT. This is the problem of computing the number of satisfying truth assignments for an instance of SATISFIABILITY with three literals per clause. The proof system is from [52], based on ideas in [6,64].

The first technique is the *low-degree polynomial trick*. It was first used in the context of interactive computation by Beaver and Feigenbaum [10] and is based on a well-known and simple mathematical fact: If two polynomials with degree d or less agree on more than d inputs, then they are the same polynomial. Thus if f and g are two different degree- d polynomials and p is a prime much larger than d , the probability that $f(a) = g(a)$ for a random element $a \in \mathbf{Z}_p$ is vanishingly small. It turns out that this fact can be used by the verifier to help keep the prover honest.

In order to apply this idea, however, one needs to have polynomials. These were present explicitly in the permanent problem, since the permanent is

defined as a sum of products. For #3SAT nothing so obvious exists. Here's where the second technique comes in: We obtain polynomials from an instance of #3SAT by a process called *arithmetization*. This is best illustrated by an example. Let E be an instance of #3SAT containing m clauses and n variables. Suppose one clause is $C = (x_1 \bar{x}_2 x_3)$. Consider the polynomial $f_C(x_1, x_2, x_3) = 1 - (1-x_1)(x_2)(1-x_3)$. Note that if we equate the value of 1 for x_i with TRUE and the value 0 with FALSE, then any truth assignment that satisfies the clause yields $f_C(x_1, x_2, x_3) = 1$, and any assignment that fails to satisfy the clause yields $f_C(x_1, x_2, x_3) = 0$. Multiplying together the polynomials for all the clauses in a conjunctive normal form expression E , one gets a polynomial $f_E(x_1, \dots, x_n)$ that takes on the value of 1 on all satisfying truth assignments and 0 on all non-satisfying assignments. The polynomials used in our proof are derived from f_E . For $0 \leq i \leq n$, let

$$f_i(x_1, \dots, x_i) = \sum_{x_{i+1}=0}^1 \sum_{x_{i+2}=0}^1 \cdots \sum_{x_n=0}^1 f_E(x_1, \dots, x_n)$$

Note that f_0 is a constant equal to the number of satisfying truth assignments for E . Moreover, for $1 \leq i \leq n$, f_i is a function of the first i variables and $f_i(x_1, \dots, x_i) = f_{i+1}(x_1, \dots, x_i, 0) + f_{i+1}(x_1, \dots, x_i, 1)$. Also, $f_n \equiv f_E(x_1, \dots, x_n)$ can be evaluated at any given point in polynomial time, and each f_i is a polynomial of degree at most $3m$.

Now we have all the basic machinery needed to design an interactive proof system for #3SAT. Suppose we are given an instance I . First, the prover picks a prime p lying in the interval $(2^m, 2^{2m})$ and sends p and a short proof of p 's primality to the verifier. (Such a proof exists by [60].) The verifier checks the primality proof and generates n random elements a_1, \dots, a_n of the field \mathbf{Z}_p , which she keeps secret for later use. The prover then sends the value g_0 that he claims is f_0 , along with the coefficients of a single-variate polynomial $g_1(x)$ that he claims is $f_1(x)$. (The coefficients of $f_1(x)$ are of polynomial-bounded length, so this is possible.) Note that the verifier can use the coefficients she receives to compute values of $g_1(x)$ in polynomial time.

The verifier checks that $g_0 = g_1(0) + g_1(1)$, as it must for consistency's sake. But now note the key point that if $g_0 \neq f_0$, then either $g_1(0) \neq f_1(0)$ or $g_1(1) \neq f_1(1)$, and so $g_1(x)$ will be a different polynomial from $f_1(x)$. Consequently, by our observation about low-degree polynomials and the fact that $\text{degree}(g_1) \leq 3m \ll 2^m < p$, the value $g_1(a_1)$ will almost surely not equal $f_1(a_1)$ unless the prover gave the correct value of f_0 in the first place. (Although the semantics of our polynomials f_i assume that the arguments are elements of $\{0, 1\}$, we are here viewing them as "extended" to the entire field \mathbf{Z}_p . Extending the domain is the key to applying the low-degree polynomial trick.) The verifier now reveals a_1 to the prover, asks him for the coefficients of a polynomial $g_2(x) = f_2(a_1, x)$, and iteratively proceeds as before, eventually asking the prover to send her the coordinates of a $g_n(x) = f_n(a_1, \dots, a_{n-1}, x)$. By an iteration of the low-degree polynomial argument, equality will be violated with high probability

unless the prover initially gave the correct value for f_0 , and now the verifier can check for equality directly. That, in brief, is the interactive proof system and why it works.

The proof that $\text{PH} \subseteq \text{IP}$ came as something of a surprise since, as mentioned before, there were oracles under which this containment did not hold. Thus a major result had been proved that did not “relativize.” Does this mean that the conventional wisdom about relativization is wrong? Or does it mean that a fundamentally new non-relativizing proof technique has been invented, one that might provide a major breakthrough on some of our other unresolved problems, such as the P versus NP question? In fact the proof technique is not new; we just had not been paying attention to it. In essence, it is just algorithm design. Let me explain. Note that most of our interesting complexity classes are defined in terms of some resource-bounded computational process, be it a nondeterministic Turing machine or an interactive proof. Note also that the completeness of a particular problem for a given class is *not* the kind of result that relativizes. (SATISFIABILITY is clearly not complete for the set of all languages recognizable by polynomial-time NDTM’s with an oracle for the halting problem.) Thus a simple non-relativizing way to prove that class X contains class Y is to show that a problem complete for class Y can be solved using the defining computational process for class X . This is what Lund et al. did for IP and P^{IP} . It is also what people have been trying to do for years in their (so far unsuccessful) attempts to prove $\text{P} = \text{NP}$, where the key claim always seems to be that a given algorithm solves some NP-complete problem in polynomial time.

Thus no fundamentally new “proof technique” had been created for the Lund et al. result, but an old, discredited approach had suddenly proved useful. And we did have new algorithmic techniques (arithmetization plus the low-degree polynomial trick) for use in the context of that approach. The obvious next step was to figure out how to use that technique to derive an interactive proof system for some PSPACE-complete problem, and hence conclude that IP contained all of PSPACE. A natural candidate for the PSPACE-complete problem was QUANTIFIED BOOLEAN FORMULA (QBF for short).

A simple version of this problem that is still PSPACE-complete is the following: Given an expression $E(x_1, \dots, x_{2n})$ in conjunctive normal form with 3 literals per clause (i.e., an instance of 3SAT having variables x_1 through x_{2n}), is the first order sentence $\forall x_1 \exists x_2 \forall x_3 \cdots \forall x_{2n-1} \exists x_{2n} E(x_1, \dots, x_{2n})$ true? A natural approach to arithmetization would be to start with the polynomial f_E defined above. Each $\exists x_i$ can then be replaced by $\sum_{x_i=0}^1$ and each $\forall x_i$ by $\prod_{x_i=0}^1$. Unfortunately, because of the nesting of the quantifiers in the above expression, this may lead to variables with exponential degree, rendering the low-degree polynomial trick unusable. Fortunately, there is a way around this difficulty (using a trick from the original proof of PSPACE-completeness for QBF [65]), as Shamir [64] demonstrated shortly after the Lund et al. results were announced. Consequently we do indeed have $\text{IP} = \text{PSPACE}$, and the end of a quest. In the next section, we

shall discuss the new quests that followed.

2. MULTIPLE PROVERS AND PROBABILISTICALLY CHECKABLE PROOFS

The fact that $IP = PSPACE$, being the surprise it was, received considerable publicity, even in the popular press. Much was made of the fact that this gave us a new way of proving difficult results, and there were even hypothetical discussions of business applications. We should keep in mind, however, that intriguing as this sort of proof is, it requires far more of the prover than simply the lucky guess at a short proof that is needed for problems in NP. Here the prover may well need to use the full power of polynomial space to determine his next message, and if the system is to be practical, he will have to be able to perform such computations in polynomial time. In other words, the fact that $IP = PSPACE$ can only be of practical importance if $PSPACE = P$ (in which case it becomes a trivial result).

As a theoretical contribution, however, $IP = PSPACE$ clearly has major importance, and more than just as another characterization of the latter class (joining such previous theorems as $PSPACE = \text{Nondeterministic } PSPACE$ [63], $PSPACE = \text{Probabilistic } PSPACE$ [35], $PSPACE = \text{Alternating } PTIME$ [21], and $PSPACE = \text{Stochastic Alternating } PTIME$ [55]). With $IP = PSPACE$, we have shown that a simple computational paradigm (interactive computation) has far more power than might at first be expected. The question thus becomes where else that insight might apply.

Here is where the second prover mentioned in the column's title arrived. Actually, he arrived somewhat earlier, and in the company of an entire coterie of fellow provers. In 1988, Ben-Or, Goldwasser, Kilian, and Wigderson introduced the idea of a *multi-prover interactive proof* [13]. Their initial purpose in introducing this new model was to allow perfect zero-knowledge proofs without cryptographic assumptions (see the [September 1988] column for an explanation). What interests us here, however, is the basic computational power of the model itself. In it there is still but one polynomial-time verifier with her source of random bits, and computations must satisfy the same validity conditions as before. Now, however, there are k all-powerful provers with whom the verifier can communicate. The only restriction on the provers is that they not be able to communicate with each other once the conversations with the verifier have begun. (It is easy to see that without this restriction they might as well just elect one representative and let him do all the work.) The question is whether, with this restriction on communication, the many provers can accomplish anything more than a single one.

Let MIP denote the set of all languages recognizable by multi-prover interactive proof systems, and let MIP_k denote the set of languages recognizable with just k provers. It is clear that $IP = MIP_1 \subseteq MIP_2 \subseteq MIP_3 \subseteq \dots \subseteq MIP$. The major result of interest to us in [13] is that $MIP = MIP_2$; it suffices to consider just two

provers. The major question left open is whether MIP is bigger than IP, and if so how much bigger?

In [28], Fortnow, Rompel, and Sipser showed that $\text{MIP} \subseteq \text{NEXPTIME}$, the set of all languages recognizable in nondeterministic time $O(2^{n^k})$ for some k (a result implicit in a much-earlier work by Peterson and Reif on multi-person games [58]). Once again (and as with IP), it turns out that the upper bound is the right answer. Babai, Fortnow, and Lund [8], in the same avalanche of papers that brought us $\text{IP} = \text{PSPACE}$, showed that in fact $\text{MIP} = \text{NEXPTIME}$. The proof is considerably more complicated than the one for $\text{IP} = \text{PSPACE}$, although it too is based on arithmetization and the low-degree polynomial trick. Now, however, the prover cannot be kept honest simply by evaluating a function at a random point. We have to make sequences of evaluations along lines in multi-dimensional space, so as to test whether a particular function provided by the provers is (approximately) *multilinear*. (A multivariate polynomial is multilinear if no individual variable occurs with degree exceeding 1.) Note that $\text{MIP} = \text{NEXPTIME}$ does not completely rule out the possibility that $\text{IP} = \text{MIP}$, since as yet we do not know how to rule out $\text{PSPACE} = \text{NEXPTIME}$. It at least suggests, however, that the second prover adds significant power to interactive proof systems.

Just what does this mean, however? To many theoreticians, myself included, the result seemed merely a *tour de force* in pure mathematics, concerning classes too high up to be of practical significance, and models too fanciful to apply to real computing. Fortunately, researchers in the area did not listen to us. Open questions persisted about the effect on multi-prover proof systems of restrictions other than simply limiting the number of provers. Two distinct research directions emerged, each of which has yielded surprising consequences for the world of approximation algorithms (the subject of the next section). The first direction concerns the effect of bounding the number of rounds in the multi-prover protocol.

In the case of one prover, it was known that if we allow only a constant number of rounds, then just one round suffices (one message from and to the verifier) [4,9,38]. Moreover, the power of the system will be significantly weakened: the class AM of languages recognizable by one-prover, constant-round interactive proof systems is contained in the class Π_2^P of the polynomial hierarchy [4,9], and so cannot equal all of IP unless $\Pi_2^P = \text{PSPACE}$ and the polynomial hierarchy collapses. Do analogous results hold in the multi-prover case?

As early as 1988, researchers were conjecturing that MIP was unlike IP, in that one or two rounds should suffice for any language in MIP. Let $\text{MIP}_{k,r}$ be the set of languages recognizable by k -prover proof systems using r or fewer rounds and having exponentially small error probability. Note that our standard requirement that the error probability be $1/4$ or less is not good enough for the case of bounded rounds—what we really want is minuscule error probability. This is easy to obtain when the number of rounds is not an issue, as we can just

repeatedly execute the protocol until the error probability is sufficiently reduced. Sequential repetition may dramatically increase the number of rounds, however. The obvious alternative is to perform the repetitions in parallel, sending k -tuples of messages instead of individual messages. Unfortunately this conceivably might open a loophole that cheating provers could exploit, and as yet no one has been able to rule out this possibility in general.

The first bounded-round result came in 1988, when Fortnow, Rompel, and Sipser [28] showed that if one *could* assume that parallel execution opened no loopholes, then MIP would equal $\text{MIP}_{3,2}$, i.e., every language in MIP would have a 3-prover, 2-round proof system, yielding exponentially small error probability. (The result of [13] that two provers suffice for all of MIP does not apply here, since the proof of that result required that the number of rounds be polynomial. Fewer rounds might require more provers.) The first bounded-round result not requiring an unproved hypothesis came in 1990, when Kilian [44] showed (without any parallelization assumptions) that MIP has 2-prover, 2-round proof systems for which the error probability is a constant $c < 1$. Results then came in a rush. In early 1991, Feige reduced the constant to $1/2 + \epsilon$ and the number of rounds to one [24]. Within months, Lapidot and Shamir [47] had made the jump to exponentially small error probabilities. Using the concept of a “quasi-oracle,” they proved that $\text{MIP} \subseteq \text{MIP}_{4,1}$ (again without resort to any general parallelization assumption). The presumed final chapter of this story is this year’s result by Feige and Lovasz [26] that indeed the strongest possible result of this form holds: $\text{MIP} \subseteq \text{MIP}_{2,1}$. Two provers and one round suffice.

The second direction of research deriving from the $\text{MIP} = \text{NEXPTIME}$ result concerns an alternative characterization of MIP first pointed out by Fortnow, Rompel, and Sipser [28]. Restating it in the more modern terminology of [3], this characterization is in terms of *probabilistically checkable proofs*. In this new model of “proof,” the verifier continues to have polynomial-time computing power and a source of random bits, but instead of a port for communicating with a prover, she has a random access port for accessing a *proof tape* of (possibly) exponential length (to determine the identity of the k th bit of the proof, she simply writes down the address k in binary notation). We say that a language L has probabilistically checkable proofs if there is a verifier V such that (a) for each input $x \in L$ there exists a proof P such that if V is given P on its proof tape, V always accepts, and (b) for each input $x \notin L$ there is no proof that can cause V to accept with probability greater than $1/4$. Fortnow, Rompel, and Sipser in essence show that the set of all languages having probabilistically checkable proofs is precisely MIP. This means that in a 2-prover protocol, the first prover might as well settle in advance on the answers to all possible questions the verifier might ask (these answers constitute the bits of the proof). The existence of a second prover, with whom the first cannot communicate, suffices to keep the first prover from later changing his mind about an answer.

This characterization of MIP proved useful in the process of obtaining the

results about bounded-round protocols mentioned above, but it also gives rise to interesting questions on its own. In particular, it suggests two new computational resources to restrict. The first is the number of proof bits the verifier can examine. The significance of this was first pointed out in early 1991 by Babai, Fortnow, Levin, and Szegedy [7]. They observed that the $MIP = NEXPTIME$ result meant that proofs of membership in $NEXPTIME$ -complete languages existed that could be probabilistically checked by looking at only a very few randomly chosen bits of the proof. In particular, the number of bits examined needed only to be polylogarithmic in the length of the proof. (In this case, a polynomial number of bits from an exponential length proof.) They then showed that any deterministic proof system could be converted to one that had this property. In particular, one could “scale down” the $MIP = NEXPTIME$ protocol of [8] to show that for any language in NP , there exist polynomial-length proofs of membership that can be probabilistically checked by looking at only a polylogarithmic number of bits.

They actually showed something much stronger, which in the case of NP implied not only that the verifier could restrict her attention to a polylogarithmic number of bits, but also that her overall running time need only be polylogarithmic in the sum of the lengths of the proof and the input. Note that although such a verifier can determine with high probability that the proof is correct, she does not have enough time to figure out what is being proved! Although the practical value of such proofs might be questioned, there are interesting scenarios under which they might be useful [7]. Moreover, their theoretical utility has already been demonstrated, as we shall shortly see. For our general notion of a probabilistically checkable proof, however, we shall continue to assume that, no matter how few proof bits are examined, the verifier still is allowed polynomial time for her overall computation.

A second resource one might want to conserve is the number of random bits used. Given that we as yet have no guaranteed source, physical or otherwise, of truly random bits, it is reasonable to consider such bits to be a scarce resource. Moreover, there is a simple theoretical reason for restricting the number used: Any process using k random bits and time T can be simulated deterministically in time $2^k T$ by trying all possibilities for the random bits and keeping statistics on the outcomes. Thus the cost of simulation grows exponentially with the number of random bits, a strong motivation indeed for restricting their number.

As a notation for keeping track of restrictions on the above two resources, let $PCP_{[f,g]}$ represent the class of all languages with probabilistically checkable proofs that use $O(f(n))$ random bits and look at $O(g(n))$ proof bits. The parameters f and g may either be specific functions, such as n^2 , or classes of functions, such as $poly(n)$, which stands for “any polynomial in n .” This notation was introduced just this year for the results of [3] but provides a convenient way of stating and distinguishing past results as well. For instance, the original $MIP = NEXPTIME$ result of [8] can be restated simply as $NEXPTIME =$

$\text{PCP}_{[\text{poly}(n), \text{poly}(n)]}$. On a more down-to-earth level, the results of [7] imply that $\text{NP} \subseteq \text{PCP}_{[\text{polylog}(n), \text{polylog}(n)]}$. A natural question to ask is just how far we can restrict f and g and still have $\text{NP} \subseteq \text{PCP}_{[f, g]}$. One limiting case derives from the standard definition of NP in terms of proofs that are checkable in deterministic polynomial time, which yields $\text{NP} = \text{PCP}_{[0, \text{poly}(n)]}$. We are more interested here in characterizations that substantially restrict *both* resources. A sequence of three major improvements on the Babai et al. result has now led to what might be characterized as the ultimate result of this form.

The first improvement came in 1991, shortly after the Babai et al. [7] results were announced. Feige, Goldwasser, Lovasz, Safra, and Szegedy [25] showed that $\text{NP} \subseteq \text{PCP}_{[\log(n) \log \log(n), \log(n) \log \log(n)]}$, a result they proved by a somewhat different scaling down of the $\text{MIP} = \text{NEXPTIME}$ protocol of [8], using a more efficient multilinearity test. In one way, however, their proof system was not as good as that of Babai et al. [7], for although they used fewer random bits and looked at fewer proof bits, the actual proofs they considered were of length $n^{\log \log(n)}$, as opposed to the polynomial-length proofs in the construction of Babai et al.

Early this year, Arora and Safra [3] tightened the result further (and got back to polynomial-length proofs) by showing that $\text{NP} \subseteq \text{PCP}_{[\log(n), \log(n)]}$. Note that now there are just a polynomial number of possibilities for the random bit string, and hence at most a polynomial number of positions in the proof can be examined over all possible choices of the random bits. Thus we may assume the proof is of polynomial length. Moreover, the total time to simulate the protocol on all possible random bit strings is also polynomial. Thus we get not just containment, but equality: $\text{NP} = \text{PCP}_{[\log(n), \log(n)]}$. (A key innovation in the proof of this result is a recursive application of the above-mentioned polylogarithmic-time checkable proofs of Babai et al. [7].)

The Arora-Safra characterization of NP in terms of $\text{PCP}_{[f, g]}$ was close to the tightest possible. Note that the number of random bits must be at least proportional to $\log(n)$ if we are to obtain NP. There remained some room, however, for tightening the bound on query bits. Arora and Safra [3] went part of the way, showing that their bound could be reduced from $O(\log(n))$ to $O(\sqrt{\log(n) \text{polyloglog}(n)})$. The ultimate tightening followed soon thereafter. Arora, Lund, Motwani, Sudan, and Szegedy [2], using an argument that builds on the proof of [3] while adding some new tricks of its own, have shown that for problems in NP, the verifier need only look at a *constant* number of bits, i.e., $\text{NP} = \text{PCP}_{[\log(n), 1]}$. The $O(\log(n))$ random bits merely provide the addresses of the bits to be examined.

This is an amazing result. No matter how large the instance and how long the corresponding proof, it is enough to look at a fixed number of (randomly chosen) bits of the proof in order to determine (with high probability) whether it is valid. In essence, the proof in question has to be extremely “holographic” (a term first suggested by Leonid Levin [5]). Recall that a hologram is a

photographic plate that displays a 3-dimensional image when illuminated by an appropriate light source. The relevant property here is that if you break off a tiny corner of a hologram, it can still be made to display the image contained in the full hologram, albeit somewhat more fuzzily. Similarly, most small collections of bits from the proofs in question here must in a sense reflect the entire proof, at least insofar as its correctness is concerned. (Babai et al. use a different optical allusion in [7], formally calling their type of probabilistically checkable proof a “transparent proof.”)

Probabilistically checkable proofs also partake of another property of holograms: After you break off the abovementioned small corner, the image produced by the remainder of the hologram is not significantly degraded. Similarly, altering a few bits in a probabilistically checkable proof will have only a small probability of changing the outcome of any particular run of the verifier. For instance, suppose we had a $PCP[\log(n), 1]$ proof system in which the verifier looks at 10 proof bits. Then even if 1% of the proof bits were altered by a malicious adversary, the probability that the verifier would look at any of the altered bits would still be less than $1/10$. Consequently, the probability of saying *false* to a true proof, although no longer zero, would be less than .10, and the probability of saying *true* to a false proof would be less than .35. Given this, the truth of the proof can still be ascertained with extremely high accuracy by iterating the protocol. Needless to say, traditional deterministically checkable proofs cannot readily be made so resilient.

This strong error-correction property is no accident. The proofs of the various results mentioned in this section make strong use of results about what are known as *self-testing* and *self-correcting* programs for the computation of low-degree polynomials. Here the idea is to use multiple calls on the program (plus the low-degree polynomial trick) to verify that the program’s output is correct for a given input (or else to deduce the correct value with high probability), assuming that the program gives correct answers for *most* of its possible inputs. Relevant references include [19,34,61]. In addition, the more traditional notion of an error-correcting code has also been exploited, both by Babai et al. [7], who explicitly use such codes in their constructions, and by Arora and Safra [3], who use a 20 year old patented decoding technique of Berlekamp and Welch [14].

How practical is all this? Well, probably not very. Given an ordinary proof of membership in an NP language, the corresponding probabilistically checkable proof can be constructed in time polynomial in the length of the original proof [2,51], but its length may itself grow as a polynomial, i.e., be cubed, or something worse (the precise degree of the polynomial has yet to be worked out). The blowup need not be so bad, however, if one is willing to look at more proof bits. For instance, for any $\epsilon > 0$, a traditional NP proof can be converted to a probabilistically checkable one of the Babai et al. [7] form whose length is only that of the original raised to the power $1+\epsilon$ [7]. (There is a trade-off however, in that the verifier will have to look at $\log^{O(1/\epsilon)}(n)$ bits of the proof.) Even if

there were no blow-up at all, however, there would still remain the problem of finding the original proof. For NP-complete problems, this remains a difficult task. Even for simpler problems, however, the theory requires that we start with a proof that is substantially more formal and detailed than the ones we typically produce. That is, it must be more like the record of a Turing machine computation than like our familiar sequences of definitions and lemmas, and will at the very least be much more laborious to generate.

Thus it may be that the main significance of these results (aside from the corollaries about approximation algorithms that we are about to discuss) is simply that they give us a new characterization of NP. This new characterization has a distinctly different flavor from previous ones and hence might conceivably be the key to resolving some of our longstanding open problems about the class (if not the P versus NP question, perhaps NP versus EXPTIME?)

3. LIMITS ON APPROXIMABILITY

The results of the last section, which I shall refer to collectively as “multi-prover results,” are certainly important in their own right. They would not have stirred up nearly so much excitement in the theoretical computer science community, however, were it not for their unexpected connection to the world of approximation algorithms. Many of the very first problems to be shown NP-hard in the early 1970’s were optimization problems (e.g, Vertex Cover, Graph Coloring, Clique, and the Knapsack and Traveling Salesman problems, etc.). Given the implication that algorithms for finding optimal solutions to such problems are likely to take superpolynomial time, many researchers instead started looking for algorithms that ran in polynomial time but only found solutions that were *near-optimal*. The term *approximation algorithm* (introduced in [41]) soon came to be applied to any algorithm that took this approach.

A standard theoretical metric for the quality of an approximation algorithm is its *worst-case ratio*. The nearness to optimality of a given solution can be expressed as the ratio of its value to that of an optimal solution, the closer to 1 the better. The worst-case ratio for an approximation algorithm tells just how far from 1 that ratio can be for a solution it generates. (We shall assume for uniformity in what follows that the numerator of the ratio is always the larger of the two solution values, so that worst-case ratios are always greater than or equal to 1, whether the optimization problem be a minimization or a maximization problem.) For many optimization problems, algorithms with small worst-case ratios were quickly found (e.g, 2 for Vertex Cover [33], $3/2$ for the Traveling Salesman problem under the triangle inequality [22], etc.). For others, fairly easy intractability results could be obtained (e.g., the result of Sahni and Gonzalez [62] that if the triangle inequality is *not* assumed in the TSP, then guaranteeing a worst-case ratio of c for *any* constant c was just as hard as finding an optimal solution). For many important problems, however, the situation was less clear,

and remained so for a decade and a half, until Feige et al. [25] first noticed an ingenious connection to the results about multiple provers.

The connection they discovered was to the Clique problem: Given a graph $G = (V, E)$, find a maximum-sized clique in G , where a clique is a set of vertices each pair of which is an edge in G . No approximation algorithm with constant worst-case ratio is known for this problem. Indeed, what few algorithms had been analyzed had been shown to have worst-case ratios that grew at least as quickly as $|V|^\epsilon$ for some ϵ [41], and the best upper bound known on a worst-case ratio is $O(|V|/\log^2|V|)$ [20]. As far as intractability results were concerned, no constant ratio had been ruled out, even assuming $P \neq NP$. (It was known, however, that if any constant worst-case ratio C , however large, could be guaranteed in polynomial time, then for any $c > 1$, however small, there was a polynomial-time approximation algorithm that had worst-case ratio less than c [31].) Feige et al. [25] made the connection between this problem and multiple provers by showing how any sufficiently good approximation algorithm for clique could be used to test whether probabilistically checkable proofs exist, and hence to determine membership in NP-complete languages. Here's how the construction goes.

Suppose we have a $PCP[f, g]$ proof system for 3SAT, and suppose that I is a 3SAT instance of length n . Let r be the actual number of random bits that the verifier uses on input I , and let b be the actual number of proof bits she queries. We may assume without loss of generality [2,3,7,8,25] that the verifier uses her r random bits simply to determine the addresses of the b proof bits, and does this before receiving any of the answers, i.e., in a nonadaptive fashion. Construct a graph $G = (V, E)$ as follows. The vertices of G correspond to pairs $\langle x, y \rangle$, where $x \in \{0, 1\}^r$, $y \in \{0, 1\}^b$, and the verifier would declare a proof to be true should her random bit string be x and the sequence of answers to her queries be y . Thus $|V| \leq 2^{r+b}$. Now let $A(x) \in \mathbf{Z}^b$ be the sequence of addresses that she would query given x as her random bit string. There is an edge between vertices $u = \langle x_u, y_u \rangle$ and $v = \langle x_v, y_v \rangle$ if and only if for any address that is in both sequences $A(x_u)$ and $A(x_v)$, the corresponding answers in y_u and y_v agree.

I claim that the maximum clique size for G is 2^r if instance I is satisfiable, and at most $2^{r/4}$ otherwise. First note that if I is satisfiable, then a proof must exist, and if the verifier is given that proof, then she must declare it to be true no matter what her random bit string might be. Fix some proof with this property. For each random bit string $x \in \{0, 1\}^r$, our clique will contain the vertex v with $x_v = x$ and y_v equal to the sequence of proof bits in the locations specified by $A(x)$. It is easy to see from the definition of G that all 2^r of these v 's will indeed be vertices and that there will be an edge between each pair of them. Thus the desired clique exists. On the other hand, suppose that I is not satisfiable and yet there is a clique V' of size larger than $2^{r/4}$. Construct a "proof" as follows. For each $v \in V'$, and each i , $1 \leq i \leq b$, assign the i th bit of y_v to the i th address in $A(x_v)$. By the definition of G , none of these assignments can conflict. Let the remaining

bits of the proof be assigned arbitrarily. Now observe that no two distinct vertices in V' can correspond to the same string $x \in \{0,1\}^r$ of random bits. (If they did their corresponding answer strings would have to differ, and there would be an address in $A(x)$ for which their answers would not agree.) Thus there are at least $|V'|$ random strings for which the verifier will declare this proof to be true, yielding a probability of $|V'|/2^r > 1/4$, contrary to the definition of a PCP.

Now suppose there were a polynomial-time approximation algorithm that was guaranteed to find a clique within a factor of two of the optimal size. Because of the factor-of-4 gap we just proved between the size of the optimal clique when I is satisfiable and the size when I is not, such an approximation algorithm could be used to determine whether I were satisfiable simply by running it on the graph G we constructed. Instance I is satisfiable if and only if the clique generated is of size $2^k/2$ or larger. What would be the running time? Using the Feige et al. [25] result that $\text{NP} \subseteq \text{PCP}[\log(n)\log\log(n), \log(n)\log\log(n)]$, the graph G has $2^{c\log(n)\log\log(n)+d\log(n)\log\log(n)} = n^{(c+d)\log\log(n)}$ vertices, and the time to construct it is clearly polynomial in that number, which would still be function of the form $n^{O(\log\log n)}$. The running time for our approximation algorithm on this graph would also have this form. Thus the overall running time for determining whether I is satisfiable would have this form, and by the NP-completeness of 3SAT, we could conclude that $\text{NP} \subseteq \text{DTIME}[n^{O(\log\log n)}]$, a weaker consequence than $\text{P} = \text{NP}$, but still one that few would think likely to be true.

Thus under the assumption NP is *not* contained in $\text{DTIME}[n^{O(\log\log n)}]$, Clique cannot have a polynomial-time approximation algorithm with worst-case ratio 2. Applying the result of [31] mentioned above, this implies that under this assumption no polynomial-time approximation algorithm can guarantee any constant ratio. This was a major result, but of course everyone would have preferred that a polynomial-time constant ratio guarantee imply $\text{P} = \text{NP}$. In fact, it does. This follows from Arora and Safra's stronger characterization of NP as equal to $\text{PCP}[\log(n), \log(n)]$. For PCP proofs of this form, the above construction leads to a graph with $2^{c\log n + d\log n} = n^{c+d}$ vertices, a polynomial number. Thus a polynomial-time factor-of-2 approximation algorithm, when applied to this graph, would yield a polynomial-time algorithm for SATISFIABILITY and so would imply $\text{P} = \text{NP}$.

If we now turn to the Arora et al. result that $\text{NP} = \text{PCP}[\log(n), 1]$, we get an even stronger result, one that comes much closer to matching the positive results mentioned earlier: There exists an $\epsilon > 0$ such that no polynomial-time approximation algorithm for clique can have worst-case ratio less than $|V|^\epsilon$ unless $\text{P} = \text{NP}$. The proof goes something like this. Suppose the $\text{PCP}[\log(n), 1]$ verifier were to go through her protocol $\log(n)$ times. Assuming that the results of these runs are all independent, this would reduce her probability of error to $(1/4)^{\log(n)} = 1/n^2$. Suppose also that we could obtain $\log(n)$ "essentially" independent runs by cleverly re-using $O(\log(n))$ random bits, rather than using the $O(\log^2(n))$ random bits that would normally be required. (Techniques for doing this can be

found in [40].) We would then have a $\text{PCP}[\log(n), \log(n)]$ protocol with error probability $1/n^2$. The corresponding clique construction would consequently give a graph with $|V| = O(n^c)$ and a multiplicative gap of size $n^2 = |V|^{2/c}$ between the sizes of the maximum clique under the assumption that I is or is not satisfiable. Hence no polynomial-time approximation algorithm could have worst-case ratio less than $|V|^{2/c}$ unless $P = NP$, and $2/c$ is certainly bigger than ϵ for some $\epsilon > 0$.

We can conclude that the clique problem is not at all amenable to approximation, assuming $P \neq NP$. Can we conclude anything more? In particular, can we extend this result to other problems? Initial optimism on this account was tempered by two facts. First, the above PCP construction was very specific to the clique problem, which seems to be ideally suited for it. Second, it is much harder to transfer complexity results about approximation from one problem to another than it is to translate complexity results about optimization. There were a few problems known to be equivalent to clique with respect to constant factor approximation (e.g., see [54]), but the correspondences were all more or less immediate. Thus some researchers began to fear that this connection between multi-prover results and approximation might be a happy but isolated coincidence.

The Arora et al. paper [2] shows that it is not. A significantly different type of approximation result is obtained based on their $\text{NP} = \text{PCP}[\log(n), 1]$ characterization, and this result propagates to a wide variety of problems. As an illustration, consider the optimization problems MAX- k SAT ($k \geq 2$): Given an instance I of SATISFIABILITY with at most k literals per clause, find a truth assignment that satisfies a maximum number of I 's clauses. (Note that this problem is NP-hard for $k = 2$, even though SATISFIABILITY itself is solvable in polynomial time if there are no more than 2 literals per clause [32].) Approximation algorithms for these problems were first studied in [41], where a simple greedy heuristic was shown to have a worst-case ratio of 2 for all $k \geq 2$. This has recently been improved upon. In [68], Yannakakis presents an algorithm with worst-case ratio $4/3$. A natural question to ask is what is the best possible ratio that can be guaranteed in polynomial time, assuming $P \neq NP$?

From an abstract theoretical point of view there are two main possibilities for the answer. Either there is an *approximation threshold* $c > 1$ such that no polynomial-time algorithm can guarantee a solution within a factor c of optimal unless $P = NP$, or for every $c > 1$ there exists a polynomial-time approximation algorithm with worst-case ratio c or less. In the latter case, we would say the problem had a *polynomial-time approximation scheme (PTAS)*. There are many examples of problems that have such schemes, for instance the knapsack problem and the restriction of the maximum independent set problem to planar graphs. (For a discussion and references, see Section 6.1 of [G&J].) For far more problems, however, we have polynomial-time approximation algorithms with constant worst-case ratios, but do not know whether a PTAS or an

approximation threshold exists. The MAX- k SAT problems are just one such example, but they turn out to play a key role.

In 1988, Papadimitriou and Yannakakis [56] showed that MAX- k SAT, $k \geq 2$ is complete for a class of optimization problems they called MAX-SNP. These problems all have polynomial-time approximation algorithms with constant worst-case ratios, but many of them are not known to have polynomial-time approximation schemes. The completeness of MAX- k SAT holds under a special type of reduction (called an L -reduction) under which upper bounds on worst-case ratios are preserved to within a constant factor. Hence, if any MAX- k SAT problem, $k \geq 2$, had a PTAS, then so would all problems in MAX-SNP. Equivalently, if for any problem X in MAX-SNP there were an approximation threshold $c > 1$, then a similar threshold would exist for each MAX- k SAT problem. Such a threshold would also exist for any problem Y to which a MAX- k SAT problem was L -reducible (i.e., to any MAX-SNP-hard problem), whether Y was in MAX-SNP or not. (MAX-SNP is technically restricted to maximization problems, and among maximization problems to ones that can be specified in a format derived from Fagin's characterization of NP in terms of 2nd-order logic [23].)

A surprising number of important NP-hard optimization problems turn out to be MAX-SNP-hard. Some of the more famous include Max Cut, Vertex Cover, and Dominating Set [56], the Steiner Tree and Traveling Salesman problems with edge weights restricted to 1 and 2 (and hence obey the triangle inequality) [16,57], and the Shortest Common Superstring problem [18]. Thus a proof that some MAX- k SAT problem had an approximation threshold $c > 1$ would have widespread consequences. This is just what Arora et al. [2] have proved.

Here is how the connection to PCP's is made. Suppose we have a PCP[$\log(n), 1$] proof system for 3SAT. Once again we may assume without loss of generality that the verifier uses her random bits at the beginning of the computation to generate the addresses of all the proof bits she intends to query, and from then on proceeds deterministically. We may also assume that she always queries precisely k proof bits, for some constant k . Let c be such that on an input of length n , the number of random bits used is always $c \log(n)$ or less. Note that given these constants, the verifier can never examine more than kn^c different proof bits on an instance of size n , so we may assume that kn^c is an upper bound on proof length. Suppose we are given an instance I of 3SAT of length n . We shall construct a corresponding instance I_k of MAX- k SAT. There are kn^c variables in I_k , with variable x_i standing for the statement "the bit in location i of the proof is 1." Let r be the precise number of random bits that the verifier uses for instance I . For each of the 2^r possible sequences x of r random bits, let $v_{x[1]}, \dots, v_{x[k]}$ be the k variables corresponding to the addresses the verifier examines given I and x , and let N_x be the set of k -tuples of values for these variables (assignments of bits to the addresses) that would cause the verifier to disbelieve the proof. Note that we must have $|N_x| \leq 2^k$.

Now suppose (b_1, \dots, b_k) is a tuple in N_x . The statement that the variables $v_{x[i]}$ do not take on this tuple of values is simply a single k SAT clause. For instance, if $k = 4$ and $(b_1, \dots, b_4) = (1, 0, 0, 1)$, then the clause would be $(\bar{v}_{x[1]} v_{x[2]} v_{x[3]} \bar{v}_{x[4]})$. Given x , the verifier will believe the proof if and only if the conjunction of such k SAT clauses, one for each tuple in N_x is true. Our k SAT instance is the conjunction, over all length r binary sequences x , of all the clauses corresponding to tuples in N_x (for a total of $N = \sum_x |N_x| \leq 2^{r+k}$ clauses). Note that the size of I_k (number of literals it contains) is at most $kN \leq k2^{r+k} \leq k2^{kn^c}$. Thus it is bounded by a polynomial in the size of I (and of course, so is the time to construct I_k).

Now if I is satisfiable, there must be a proof (i.e., a truth assignment for the variables v_i) such that *all* the clauses corresponding to each set N_x are satisfied. If I is not satisfiable, then for any truth assignment at least $2^{r/4}$ of the sets N_x must yield one or more unsatisfied clauses. The ratio between the maximum number of satisfiable clauses in the two cases is thus at least $N/(N - 2^{r/4}) > 1 + 2^{r/4N} \geq 1 + 1/2^{k+2}$. Thus no polynomial-time approximation algorithm for k SAT can have worst-case ratio less than $1 + 1/2^{k+2}$ unless $P = NP$.

So now we know that, assuming $P \neq NP$, every MAX-SNP-hard problem X has an approximation threshold $c_X > 1$. If one is also willing to assume that NP is not contained in R (the set of languages recognizable in *random* polynomial time—see the [September, 1984] column), the same argument implies that such thresholds exist even for *randomized* approximation algorithms, a possibility previously explored in [15]. In most cases we also get thresholds for *asymptotic* worst-case ratios, i. e., we can rule out guarantees of the form $a \cdot OPT + o(OPT)$ for sufficiently small $a > 1$. Do these results have practical significance? The answer of course depends on how big the thresholds are.

If a threshold is sufficiently close to the best worst-case ratio we know how to guarantee, it will probably be enough to convince us to stop the search for better approximation algorithms. If instead it were say 1.001, the consequences would only be of interest as theory. Unfortunately, the thresholds yielded for MAX 3-SAT by the Arora et al. construction seem more likely to fall in the second category [51] (and the thresholds derivable for other MAX-SNP-hard problems via L -reductions from MAX 3-SAT are likely to be substantially closer to 1 than this). Note, however, that this threshold is already substantially better than the threshold for MAX- k SAT provided by the simplified argument given above. The number of proof bits k that need to be examined in a $PCP[\log(n), 1]$ proof for 3SAT is well over 100 in the current Arora et al. construction [51], so that argument yields a threshold of only $1 + 2^{-102}$. Arora et al. obtain thresholds on the order of 1.001 by exploiting the details of their $NP = PCP[\log(n), 1]$ proof to derive k SAT expressions involving far fewer than 2^k clauses per random bit string. Phillips and Safra [59] have recently claimed improvements on the proof that may get the approximation thresholds up to as much as 1.01 for some problems. This is still quite small compared to the best ratios we currently know how to guarantee, however, so the question of practical significance must for now

remain open.

Nevertheless, the Arora et al. results certainly represent a major theoretical advance. Moreover, they apparently have given researchers the confidence needed to find further connections between the multi-prover results and approximation algorithms. During the past few weeks several long-standing open problems have been resolved.

First, consider Graph Coloring: Given a graph $G = (V, E)$, find the minimum number k for which there exists a function $f: V \rightarrow \{1, 2, \dots, k\}$ such that no two endpoints of the same edge are assigned the same number. Researchers have long worked at trying to design good approximation algorithms for this problem without significant success. In 1974 a simple greedy heuristic was shown to have worst-case ratio growing as $\Theta(|V|/\log|V|)$ [42], and although this has been improved several times, the worst-case ratio for the currently best polynomial-time approximation algorithm has only declined to $O(|V|(\log\log|V|)^2/(\log|V|)^3)$, a ratio that is worse than $|V|^\epsilon$ for every $\epsilon < 1$ [39]. Even for 3-colorable graphs, the best worst-case ratio known is $O(|V|^{3/8} \text{polylog}|V|)$ [17]. However, until now the only complexity result known was the 1976 result of [31] which showed that for any $\epsilon > 0$, no polynomial-time algorithm could have worst-case ratio $2-\epsilon$ unless $P = NP$. Now, by a clever approximation-preserving transformation from the clique problem (restricted to graphs like those arising in the above Feige et al. construction), Lund and Yannakakis [53] have been able to show that Graph Coloring is just as hard to approximate as the Arora et al. results say Clique is: There is an $\epsilon > 0$ such that no polynomial-time approximation algorithm can have worst-case ratio growing as $O(|V|^\epsilon)$ unless $P = NP$.

Other long-standing open problems about approximation have been successfully attacked using a different result from the multi-prover world, the Feige and Lovasz result [26] that $MIP = MIP_{2,1}$. Consider first the problem of Set Cover: Given a collection $\{S_1, S_2, \dots, S_k\}$ of sets whose union is a finite set U , find a minimum cardinality subcollection whose union is U . The simple greedy heuristic for this problem has a worst-case ratio of $\Theta(\log|U|)$, with the constant of proportionality lying between $1/3$ and $\log_2(e) \sim 1.44$ [41, 50]. For all we knew until a few weeks ago, however, a polynomial-time algorithm with worst-case ratio 2 or better might have existed. Using the $MIP = MIP_{2,1}$ result, Lund and Yannakakis have now shown that no polynomial-time approximation algorithm for Set Cover can have worst-case ratio less than $\log|U|/50$ unless $NP \subseteq DTIME[n^{\text{polylog}(n)}]$ (again a weaker consequence than $P = NP$, but still an unlikely one).

Assuming that NP is not contained in $DTIME[n^{\text{polylog}(n)}]$, Lund and Yannakakis have also shown that no polynomial-time approximation algorithm with constant worst-case ratio can exist for a wide variety of maximum induced subgraph problems, such as finding a maximum-sized set of vertices that induces a planar subgraph. (The results hold for any nontrivial property that is preserved under vertex deletion. See [48] for a survey of such problems and a proof that

the corresponding optimization problems are NP-hard.)

In addition to the above, there are also non-approximability results for the Longest Path problem, Quadratic Programming, and several others I don't have space to talk about [11,12,26,43,69]. Given all these results, it now appears that there is a fundamental and wide-ranging connection between the multi-prover model and the complexity of approximation. Each tightening of the results in the former model seems a potential source of new results in the latter. Given the rapid progress, and the three-month delay until the publication of this column, it would be a bit foolhardy to attempt an open problem list. (The open problems I originally intended to mention were all solved before I could get to them.) I shall thus conclude here, and promise a more leisurely survey of the approximation algorithm front (including positive results), once the dust raised by the second prover has had more time to settle.

REFERENCES

1. W. AIELLO, S. GOLDWASSER, AND J. HASTAD, On the power of interaction, in "Proceedings 27th Ann. Symp. on Foundations of Computer Science," pp. 368-379, IEEE Computer Society, Los Angeles, 1986.
2. S. ARORA, C. LUND, R. MOTWANI, M. SUDAN, AND M. SZEGEDY, Proof verification and intractability of approximation problems, unpublished manuscript (April, 1992).
3. S. ARORA AND S. SAFRA, Approximating clique is NP-complete, unpublished manuscript (February, 1992). Current version (June, 1992) has revised title: Probabilistic checking of proofs: A new characterization of NP.
4. L. BABAI, Trading group theory for randomness, in "Proceedings 17th Ann. ACM Symp. on Theory of Computing," pp. 421-429, Association for Computing Machinery, New York, 1985.
5. L. BABAI, personal communication (May, 1992).
6. L. BABAI AND L. FORTNOW, A characterization of #P by arithmetic straight line programs, in "Proceedings 31st Ann. Symp. on Foundations of Computer Science," pp. 26-34, IEEE Computer Society, Los Angeles, 1990. Journal version: Arithmetization: A new method in structural complexity theory, *Computational Complexity* **1** (1991), 41-66.
7. L. BABAI, L. FORTNOW, L. A. LEVIN, AND M. SZEGEDY, Checking computations in polylogarithmic time, in "Proceedings 23rd Ann. ACM Symp. on Theory of Computing," pp. 21-31, Association for Computing Machinery, New York, 1991.
8. L. BABAI, L. FORTNOW, AND C. LUND, Non-deterministic exponential time has two-prover interactive protocols, in "Proceedings 31st Ann. Symp. on Foundations of Computer Science," pp. 16-25, IEEE Computer Society, Los Angeles, 1990. Journal version (same title): *Computational Complexity* **1** (1991), 3-40.
9. L. BABAI AND S. MORAN, Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes, *J. Comput. System Sci.* **36** (1988), 254-276.
10. D. BEAVER AND J. FEIGENBAUM, Hiding instances in multioracle queries, in "Proc. 7th Symp. on Theoretical Aspects of Computing," *Lecture Notes in Computer Science* **415**, pp. 37-48, Springer, Berlin, 1990.
11. M. BELLARE, "Interactive proofs and approximation," Report No. RC 17969, IBM Research, Yorktown Heights, NY (May, 1992).

12. M. BELLARE AND P. ROGOWAY, "The complexity of approximating a nonlinear program," Report No. RC 17831, IBM Research, Yorktown Heights, NY (March, 1992).
13. M. BEN-OR, S. GOLDWASSER, J. KILIAN, AND A. WIGDERSON, Multi-prover interactive proofs: How to remove intractability assumptions, in "Proceedings 20th Ann. ACM Symp. on Theory of Computing," pp. 113-131, Association for Computing Machinery, New York, 1988.
14. E. BERLEKAMP AND L. WELCH, Error correction of algebraic block codes, U.S. Patent No. 4,633,470.
15. P. BERMAN AND G. SCHNITGER, On the complexity of approximating the independent set problem, *Information and Control* **96** (1992), 77-94.
16. M. BERN AND P. PLASSMAN, The Steiner problem with edge lengths 1 and 2, *Inform. Process. Lett.* **32** (1989), 171-176.
17. A. BLUM, Some tools for approximate 3-coloring, in "Proceedings 31st Ann. Symp. on Foundations of Computer Science," pp. 554-562, IEEE Computer Society, Los Angeles, 1990.
18. A. BLUM, T. JIANG, M. LI, J. TROMP, AND M. YANNAKAKIS, Linear approximation of shortest superstrings, in "Proceedings 23rd Ann. ACM Symp. on Theory of Computing," pp. 328-336, Association for Computing Machinery, New York, 1991.
19. M. BLUM, M. LUBY, AND R. RUBINFELD, Self-testing/correcting with applications to numerical programs, in "Proceedings 22nd Ann. ACM Symp. on Theory of Computing," pp. 73-83, Association for Computing Machinery, New York, 1990.
20. R. B. BOPPANA AND M. M. HALLDORSSON, Approximating maximum independent sets by excluding subgraphs, in "Proc. 2nd Scandinavian Workshop on Algorithmic Theory," *Lecture Notes in Computer Science* **447**, pp. 13-25, Springer, Berlin, 1990.
21. A. K. CHANDRA, D. C. KOZEN, AND L. J. STOCKMEYER, Alternation, *J. Assoc. Comput. Mach.* **28** (1981), 114-133.
22. N. CHRISTOFIDES, "Worst-case analysis of a new heuristic for the travelling salesman problem," Technical Report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.
23. R. FAGIN, Generalized first-order spectra and polynomial-time recognizable sets, in R. M. Karp (ed.) "Complexity of Computer Computations," pp. 43-73, American Mathematical Society, Providence, RI, 1974.
24. U. FEIGE, On the success probability of the two provers in one-round proof systems, in "Proceedings: Structure in Complexity Theory (6th Annual Conference)," IEEE Computer Society, Los Angeles, Calif., 1991, 116-123.
25. U. FEIGE, S. GOLDWASSER, L. LOVÁSZ, S. SAFRA, M. SZEGEDY, Approximating clique is almost NP-complete, in "Proceedings 32nd Ann. Symp. on Foundations of Computer Science," pp. 2-12, IEEE Computer Society, Los Angeles, 1991.
26. U. FEIGE AND L. LOVÁSZ, Two-prover one-round proof systems: Their power and their problems, in "Proceedings 24th Ann. ACM Symp. on Theory of Computing," pp. 733-744, Association for Computing Machinery, New York, 1992.
27. P. FELDMAN, The optimum prover lives in PSPACE, unpublished manuscript (1986).
28. L. FORTNOW, J. ROMPEL, AND M. SIPSER, On the power of multi-prover interactive protocols, in "Proceedings: Structure in Complexity Theory (3rd Annual Conference)," IEEE Computer Society, Los Angeles, Calif., 1988, 156-61. Erratum in in "Proceedings: Structure in Complexity Theory (5th Annual Conference)," IEEE Computer Society, Los Angeles, Calif., 1990, 318-319.
29. L. FORTNOW AND M. SIPSER, Are there interactive protocols for co-NP languages?, *Inform. Process. Lett.* **28** (1988), 249-251.
30. M. FÜRER, O. GOLDREICH, Y. MANSOUR, M. SIPSER, AND S. ZACHOS, On completeness and soundness in interactive proof systems, in "Advances in Computing Research 5: Randomness and Computation," pp. 429-442, S. Micali, ed., J.A.I. Press, Greenwich, 1989.

31. M. R. GAREY AND D. S. JOHNSON, The complexity of near-optimal graph coloring, *J. Assoc. Comput. Mach.* **23** (1976), 43-49.
32. M. R. GAREY, D. S. JOHNSON, AND L. STOCKMEYER, Some simplified NP-complete graph problems, *Theor. Comput. Sci.* **1** (1976), 237-267.
33. F. GAVRIL, personal communication (1974) cited in [G&J].
34. P. GEMMELL, R. LIPTON, R. RUBINFELD, M. SUDAN, AND A. WIGDERSON, Self-testing/correcting for polynomials and for approximate functions, in "Proceedings 23rd Ann. ACM Symp. on Theory of Computing," pp. 32-42, Association for Computing Machinery, New York, 1991.
35. J. GILL, Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6** (1977), 675-695.
36. O. GOLDBREICH, S. MICALI, AND A. WIGDERSON, Proofs that yield nothing but their validity and a methodology of cryptographic protocol design, in "Proceedings 27th Ann. Symp. on Foundations of Computer Science," pp. 174-187, IEEE Computer Society, Los Angeles, 1986. Journal version: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems, *J. Assoc. Comput. Mach.* **38** (1991), 691-729.
37. S. GOLDWASSER, S. MICALI, AND C. RACKOFF, The knowledge complexity of interactive proof-systems, *SIAM J. Comput.* **18** (1989), 186-208.
38. S. GOLDWASSER AND M. SIPSER, Private coins versus public coins in interactive proof systems, in "Advances in Computing Research 5: Randomness and Computation," pp. 73-90, S. Micali, ed., J.A.I. Press, Greenwich, 1989.
39. M. M. HALLDORSSON, "A still better guarantee for approximate graph coloring," Report No. 90-44, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1990.
40. R. IMPAGLIAZZO AND D. ZUCKERMAN, How to recycle random bits, in "Proceedings 30th Ann. ACM Symp. on Theory of Computing," pp. 48-25, Association for Computing Machinery, New York, 1989.
41. D. S. JOHNSON, Approximation algorithms for combinatorial problems, *J. Comput. System Sci.* **9** (1974), 256-278.
42. D. S. JOHNSON, Worst case behavior of graph coloring algorithms, in "Proceedings 5th South-eastern Conference on Combinatorics, Graph Theory, and Computing," pp. 513-527, Utilitas Mathematica Publishing, Winnipeg, Ont., 1974.
43. D. KARGER, R. MOTWANI, AND G. D. S. RAMKUMAR, On approximating the longest path in a graph, unpublished manuscript (April, 1992).
44. J. KILIAN, Strong separation models of multi prover interactive proofs, talk at *DIMACS Workshop on Cryptography* (October, 1990).
45. G. KOLATA, In a frenzy, math enters the age of electronic mail, *The New York Times Science Section* (June 26, 1990), B5.
46. G. KOLATA, New short cut found for long math proofs, *The New York Times Science Section* (April 7, 1992), C1.
47. D. LAPIDOT AND A. SHAMIR, Fully parallelized multi prover protocols for NEXP-time, in "Proceedings 32nd Ann. Symp. on Foundations of Computer Science," pp. 13-18, IEEE Computer Society, Los Angeles, 1991.
48. J. M. LEWIS AND M. YANNAKAKIS, The node-deletion problem for hereditary properties, *J. Comput. System Sci.* **20** (1980), 219-230.
49. R. J. LIPTON, New directions in testing, in "Distributed Computing and Cryptography," "DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 2," pp. 191-202, American Mathematical Society, Providence, 1991.
50. L. LOVÁSZ, On the ratio of optimal integral and fractional covers, *Discrete Math.* **13** (1975), 383-390.
51. C. LUND, personal communication (June, 1992).
52. C. LUND, L. FORTNOW, H. KARLOFF, AND N. NISAN, Algebraic methods for interactive proof systems, in "Proceedings 31st Ann. Symp. on Foundations of Computer Science," pp. 2-10, IEEE Computer Society, Los Angeles, 1989.

53. C. LUND AND M. YANNAKAKIS, personal communication (May, 1992).
54. A. PANCONESI AND D. RANJAN, Quantifiers and approximation, in "Proceedings 22nd Ann. ACM Symp. on Theory of Computing," pp. 446-456, Association for Computing Machinery, New York, 1990.
55. C. H. PAPANITRIOU, Games against nature, *J. Comput. System Sci.* **31** (1985), 288-301.
56. C. H. PAPANITRIOU AND M. YANNAKAKIS, Optimization, approximation, and complexity classes, *J. Comput. System Sci.* **43** (1991), 425-440.
57. C. H. PAPANITRIOU AND M. YANNAKAKIS, The traveling salesman problem with distances one and two, *Math. Oper. Res.*, to appear.
58. G. L. PETERSON AND J. H. REIF, Multiple-person alternation, in "Proceedings 20th Ann. Symp. on Foundations of Computer Science," pp. 348-363, IEEE Computer Society, Los Angeles, 1979.
59. S. PHILLIPS AND S. SAFRA, personal communication (May, 1992).
60. V. PRATT, Every prime has a succinct certificate, *SIAM J. Comput.* **4** (1975), 214-220.
61. R. RUBINFELD AND M. SUDAN, Testing polynomial functions efficiently over rational domains, in "Proceedings 3rd Ann. ACM-SIAM Symp. on Discrete Algorithms," pp. 23-32, Association for Computing Machinery, New York, and Society for Industrial and Applied Mathematics, Philadelphia, 1992.
62. S. SAHNI AND T. GONZALEZ, P-complete approximation problems, *J. Assoc. Comput. Mach.* **23** (1976), 555-565.
63. W. J. SAVITCH, Relationship between nondeterministic and deterministic tape classes, *J. Comput. System Sci.* **4** (1970), 177-192.
64. A. SHAMIR, $IP = PSPACE$, in "Proceedings 31st Ann. Symp. on Foundations of Computer Science," pp. 11-15, IEEE Computer Society, Los Angeles, 1990.
65. L. J. STOCKMEYER AND A. R. MEYER, Word problems requiring exponential time, in "Proceedings 5th Ann. ACM Symp. on Theory of Computing," pp. 1-9, Association for Computing Machinery, New York, 1973.
66. S. TODA, On the computational power of PP and P , in "Proceedings 30th Ann. Symp. on Foundations of Computer Science," pp. 514-519, IEEE Computer Society, Los Angeles, 1989. Journal version: PP is as hard as the polynomial-time hierarchy, *SIAM J. Comput.* **20** (1991), 865-877.
67. L. G. VALIANT, The complexity of computing the permanent, *Theor. Comput. Sci.* **8** (1979a), 189-201.
68. M. YANNAKAKIS, On the approximation of maximum satisfiability, in "Proceedings 3rd Ann. ACM-SIAM Symp. on Discrete Algorithms," pp. 1-8, Association for Computing Machinery, New York, and Society for Industrial and Applied Mathematics, Philadelphia, 1992.
69. D. ZUCKERMAN, NP-complete problems have a version that's hard to approximate, unpublished manuscript (April, 1992).