

The NP-Completeness Column: An Ongoing Guide

DAVID S. JOHNSON

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

This is the seventeenth edition of a (usually) quarterly 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 2C-355, AT&T Bell Laboratories, Murray Hill, NJ 07974 (or to dsj@btl.csnet). 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 If the results are unpublished, please state explicitly that you are willing for them to be mentioned.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 on the nature of the column and the form of desired submissions, see the December 1981 issue of this Journal.

1. COMPUTING WITH ONE HAND TIED BEHIND YOUR BACK

As a result of my six-month absence from these pages, I find that I have been scooped! While I was away, Dr. Crypton of *Science Digest* was busy interviewing all my sources, and now has beaten me into print by at least three months. His March 1986 column [18] provides an informative introduction to the topics I will be covering, as well as large color pictures of two of the protagonists, quotations from several others, and only one or two technical misunderstandings.

Our mutual subject is the recent spate of exciting new lower bound results. Although there continues to be a slow trickle of papers with provocative titles like " $P = NP$ " or " $P \neq NP$ " (at least one of each is currently in circulation), none has yet been found to lack its own "one or two technical misunderstandings." If we believe that $P \neq NP$ and restrict ourselves to the realm of verified proofs, there remains a substantial gap between the superpolynomial lower bounds we would like to prove and the lower bounds we have so far been *able* to prove for general models of computation. For instance, in the circuit complexity model the best lower bound known for a problem in NP has only

increased from $2.5n$ to $3n$ since [G&J] was published seven years ago [9].

Many researchers have thus turned their attention to machine models whose computational power is sufficiently limited that superpolynomial lower bounds may be easier to prove. Such a lower bound would imply only that the problem at hand is hard when your machine has one hand tied behind its back, but the techniques and insights developed might eventually be of use in attacking more general models. In any case, the mathematics involved offers its own challenges and rewards. This column will cover three popular ways of restricting the general Boolean circuit model and the surprising new developments that have recently occurred relating to each.

Section 2 considers the restriction to *monotone* circuits, i.e., circuits having only \wedge and \vee (“and” and “or”) gates, with no \neg (“not”) gates allowed. I will discuss key new results of A. A. Razborov [37,38], as well as recent improvements by others and some historical sidelights. Section 3 is devoted to circuits of bounded depth, where Andy Yao [53] is the source of the recent breakthrough (and where both Dr. Crypton and I were scooped by Gina Kolata in *Science* [27]). The bounded-depth circuit model is especially interesting because of its connection to questions about relativized versions of the polynomial hierarchy, and Yao’s results resolve two long-standing open problems in this area. Section 4 considers fixed width “branching programs,” a generalization of bounded depth circuits and another hotbed of lower-bound research. Here David Barrington [6] has greatly upset expectations by showing that a supposedly infinite hierarchy has at most five levels. In a brief concluding section, I say a few words about where these new results seem to be taking us, and what, if anything, they suggest about the quest for a proof that $P \neq NP$.

2. MONOTONE CIRCUIT COMPLEXITY

When theoreticians speak of “circuit complexity,” they normally refer to feedback-free (“combinational”) circuits in which the circuit elements are 2-input \wedge gates, 2-input \vee gates, and 1-input \neg gates, and where arbitrary fan-out is allowed from each gate (as well as from each input). The *size* of such a circuit is the number of circuit elements it contains. The *circuit complexity* $c(f)$ of a Boolean function f is the size of the smallest circuit that computes f .

We extend this definition to the traditional decision problems of complexity theory, which are defined for instances of arbitrary size, as follows. Suppose X is such a problem, and assume that its instances are encoded as binary strings. Let X_n be the restriction of X to inputs of length precisely n . Note that each X_n is a Boolean function and so $c(X_n)$ is defined for each n . The *circuit complexity* of X is simply $c(X_n)$ viewed as a function of n . (See [39] for a more detailed introduction to circuit complexity.)

It is easy to show that membership in P implies polynomial circuit complexity. The converse is false, however, as the circuits that realize the $c(X_n)$ lower

bounds might not be “uniform” enough to be constructed efficiently. (Observe that even undecidable problems can have polynomial circuit complexity.) Thus showing that a problem has super-polynomial circuit complexity may well be more difficult than merely showing that the problem is not in P. It is a natural approach to try, however. Unfortunately, although counting arguments show that almost all Boolean functions on n variables have circuit complexity exponential in n [42], the best bound we know for a problem in NP is the $3n$ bound of [9] mentioned above.

(Exponential bounds *are* known for problems not in NP, however. The first such result was proved in 1967 by Ehrenfeucht [19], and was for the problem of identifying the true sentences in the theory of integer arithmetic with all quantifiers bounded by constants in iterated exponential form (e.g., 3^{2^5}). Meyer generalized this, observing that exponential lower bounds on circuit complexity hold for *any* problem that is EXPSPACE-hard (under polynomial transformations that increase length at most linearly) [32]. Specializing to a fixed input size, Stockmeyer in [46] constructed a particular 3696 variable Boolean function with circuit complexity at least 10^{123} .)

Given our lack of success in proving even superlinear lower bounds on the circuit complexity of problems in NP, researchers wishing to prove *superpolynomial* bounds have turned to the study of less powerful variants on the general Boolean circuit model. Note that, just as the class P is immune to “reasonable” changes in the underlying machine model, the class of problems that have polynomial circuit complexity is immune to “reasonable” changes in the set of allowable circuits. For instance, we might limit fan-out by restricting it to a new circuit element that “fans out” a single input to two outputs, or we might replace the \wedge , \vee , and \neg gates by some other finite complete basis for Boolean logic. “Monotone” circuit complexity arises from an “unreasonable” restriction on the model.

The *monotone circuit complexity* $c^m(f)$ of a function f is the size of the smallest circuit that computes f and contains only \wedge and \vee gates. We call such circuits “monotone” because changing an input to a circuit element from 0 to 1 can never change the value of that element’s output from 1 to 0. Monotone circuits are “unreasonable” because \wedge ’s and \vee ’s do not by themselves form a complete basis, i.e., not all Boolean functions can be computed by monotone circuits. Only “monotone” functions can be computed, functions whose value can never be turned from 1 to 0 by changing an input variable from 0 to 1. (As a simple example of a non-monotone Boolean function, consider the *parity* function, which has value 1 if and only if an odd number of its inputs are 1.)

Fortunately, there are some interesting and non-trivial monotone problems. In particular, consider the NP-complete problem of determining whether a graph $G = (V, E)$ has a clique (i.e., complete subgraph) of size $\lfloor |V|/2 \rfloor$. Suppose instances are encoded as the lower triangle of the adjacency matrix of the graph, i.e., with one variable for each of the $\lfloor |V|(|V| - 1)/2 \rfloor$ potential edges in the graph. Then the

problem is monotone, since adding an edge to a graph can never decrease the size of the largest clique.

It was in connection with this problem that monotone circuit complexity first became linked to NP-completeness. The year was 1973, and the occasion was a suggestion by C. P. Schnorr [40] of a potential method for proving $P \neq NP$. It depended on the observation that $P \neq NP$ is jointly implied by the following two plausible conjectures:

- (a) If a monotone problem has superpolynomial monotone circuit complexity, then it has superpolynomial general circuit complexity.
- (b) Monotone circuits that test whether a graph has a clique of size $\lfloor \sqrt{V} \rfloor / 2$ must have sizes that grow superpolynomially in n .

Neither conjecture could be proved at the time, and they have remained open for over a decade. Doubts about (a) were expressed early, however, and received added impetus from a 1979 result of Valiant concerning monotone and non-monotone computations in arithmetic. The arithmetic analogues of \wedge , \vee , and \neg gates are the operations of \times , $+$, and $-$, i.e., multiplication, addition, and subtraction. (A more precise analogue for \neg would be the unary operation of multiplication by -1 , but this would still be enough to allow subtraction since $x - y$ can be rewritten as $x + (-1)y$.)

Before Valiant's work, the biggest gap known between monotone and non-monotone arithmetic computations seems to have been the one for the convolution of vectors, which can be computed in $O(n \log n)$ operations using $+$, \times , and $-$ (by the Fast Fourier Transform), but requires $\Omega(n^2)$ operations if subtractions are not allowed [41]. Valiant in [48] considered straight-line programs for computing polynomials, in which each step involved a single arithmetic operation. Exponential lower bounds on monotone arithmetic complexity had previously been proven within this model by Schnorr [41], but only for NP-hard problems. Valiant devised a sequence of monotonic polynomials (based on the perfect matchings in a special class of planar graphs) which require exponential length programs if only \times and $+$ is allowed, but which can be computed by polynomial length programs if even *one* subtraction is allowed.

This result did not carry over directly to the Boolean model, however. An exponential lower bound in the monotone arithmetic model need not imply a corresponding exponential lower bound on $c^m(X_n)$. (Jerrum and Snir proved an exponential lower bound on the monotone arithmetic complexity of the connectivity polynomial in [25], but connectivity can be recognized by polynomial-size monotone circuits [49].) Moreover, we knew far too little about how to take advantage of monotonicity in the Boolean setting to be able to prove the Boolean analogue of Valiant's result from scratch. The best monotone lower bound known for a problem in NP improved on the best general lower bound of $3n$, but only to $4n$ [47].

Given this sad state of affairs, the new results of Razborov, based on counting

arguments over lattices, are all the more impressive. In [37,38] he increases the best lower bound on $c^m(X_n)$ for a monotone problem in NP from $4n$ to $n^{c \log n}$. In doing so he resolves both the conjectures stated above. Conjecture (b) is proved true, since the clique problem it concerns is one of the problems to which Razborov's lower bounds apply [37]. Conjecture (a) is proved false, since Razborov proves the same superpolynomial lower bounds for a problem that is in P and hence has polynomial-size circuits when \neg gates are allowed [38]. The problem in P is that of telling whether a bipartite graph contains a perfect matching, and hence is closely related to the problem for which Valiant had proved his analogous exponential gap result for monotone arithmetic complexity in [48]. (Valiant had in fact at the time suggested perfect matching as a likely candidate for the Boolean result.)

By disproving Conjecture (a), Razborov appears to have put an end to hopes of proving $P \neq NP$ via monotone circuit complexity. However, although Razborov's arguments yield the same lower bounds for problems in P as for NP-complete problems, recent improvements suggest that some form of "monotone complexity gap" might still exist between the two classes. Although no one has been able to prove a lower bound exceeding $n^{c \log n}$ for the monotone circuit complexity of a problem in P, true "exponential" bounds have now been derived for problems in NP not known to be in P. For instance, Alon and Boppana [3] have now shown that the monotone circuit complexity of the NP-complete clique problem of Conjecture (b) is exponential in $\Omega(n^{1/6}/(\log n)^{1/3})$, where recall that for this problem the number n of input variables is $\lfloor V \rfloor (\lfloor V \rfloor - 1)/2$.

The best lower bound on $c^m(X_n)$ currently known for a problem in NP is exponential in $\Omega(n^{1/4}(\log n)^{1/2})$. This result was obtained by Alon and Boppana [3] for a rather baroque problem due to Andreev [4]. (Andreev proved a slightly weaker, though still exponential lower bound for it in [4].) To the best of my knowledge, the problem is not yet known to be NP-complete. If the problem should be in P, it would provide an example of a monotone problem with polynomial circuit complexity and exponential (rather than just superpolynomial) monotone circuit complexity. I shall thus present it as my "Open Problem of the Month."

ANDREEV'S PROBLEM

INSTANCE: Prime power q and a bipartite graph $G = (V \cup W, E)$, where $V = \{v_i : i \in GF(q)\}$, $W = \{w_j : j \in GF(q)\}$.

QUESTION: Is there a polynomial p over the finite field $GF(q)$ that has degree $d < s = (1/2)(q/\log q)^{1/2}$ and is such that every pair $\{v_i, w_{p(i)}\}$ is an edge in E .

Comment. In NP, but not known to be NP-complete, even if the bound s is given as part of the input, rather than as a function of q . Solvable in polynomial time if s is a fixed constant or if $s \geq q$. (In the latter case the answer is "yes")

unless some vertex in V has degree 0.) This is a monotone problem if the input is encoded as the $n = q \times q$ adjacency matrix for the graph. It is not clear which will be more difficult: finding an NP-completeness proof for this problem or finding a practical application.

Let me conclude this section by mentioning a class of problems for which Conjecture (a) *does* hold, and hence superpolynomial lower bounds on monotone circuit complexity do imply superpolynomial lower bounds on standard circuit complexity. These are problems based on “slice functions,” a concept introduced by Berkowitz [8]. A Boolean function f is a k -slice function if $f(x) = 0$ whenever input x has fewer than k 1’s and $f(x) = 1$ whenever x has more than k 1’s. (No restrictions are placed on $f(x)$ when x has exactly k 1’s.) Note that a slice function is monotone by definition. Berkowitz observes that any circuit for computing a slice function can be converted to a monotone one with only a polynomial blow-up in size. The trick is to replace $\neg x$ by a circuit that tests whether k inputs other than x equal 1. Such a “threshold” function is easy to implement with a polynomial size monotone circuit (say by dynamic programming [39]).

Although some natural slice functions associated with NP-hard problems turn out to be easy to solve [51], any intractable problem must have at least one hard slice. In fact, it is not difficult to construct NP-complete “slice problems,” such as “Given a graph $G = (V, E)$ with $|E| = |V|$, does G contain a clique of size $|V|^{1/2}$?” $P \neq NP$ would follow from a superpolynomial lower bound on the monotone circuit complexity for such a problem. Whether this observation will be any help in proving $P \neq NP$ is another question. As might be expected, Razborov’s techniques do not seem to extend to slice functions [11].

3. BOUNDED DEPTH COMPLEXITY

The second restricted model of computation we consider is another “unreasonable” variant on standard circuit complexity, the bounded-depth, unbounded fan-in circuit, a concept first promoted by Furst, Saxe, and Sipser [21] (see also [17]). To derive the relevant circuit model, let us first assume our circuits have *two* inputs for each variable x , the first providing the value of x and the second providing the value of $\neg x$. This increases the size of the circuit by at most a factor of 2, and allows us to assume that the rest of the circuit is monotone. (Any \neg gate can now be eliminated without further increase in size by repeated application of DeMorgan’s Rules: $\neg(x \wedge y) \Rightarrow (\neg x) \vee (\neg y)$ and $\neg(x \vee y) \Rightarrow (\neg x) \wedge (\neg y)$.) The “depth” of a circuit in this format is defined inductively as follows. The circuit inputs (and their negations) have depth 0, the depth of an \wedge or an \vee gate is one more than that of its maximum depth input, and the *depth* of a circuit is the depth of its maximum-depth circuit element.

“Unbounded fan-in” refers to the fact that we allow our \wedge and \vee gates to have

an arbitrary number of inputs. Note that allowing unbounded fan-in is necessary if we wish to say anything non-trivial about bounded-depth circuits, for if f is an n -variable Boolean function that depends on all its inputs, any circuit that computes f using the standard 2-input circuit elements must have depth at least $\log n$. Once we allow unbounded fan-in, depth 2 is sufficient to compute *any* Boolean function. (The trick is to use an \wedge gate for each of the at most 2^n input settings that yield function values of 1, and \vee all these results together.)

Furthermore, unbounded fan-in also allows us without loss of generality to restrict our attention to circuits consisting of alternating levels of \wedge and \vee gates. Thus such circuits can be viewed as yet another embodiment of the concept of “alternation” so beloved by theoreticians. (As we shall see later, the connection is more than superficial.) Finally, there even appears to be a “practical application” for this model. The “programmable logic array” or “PLA” found in so many of today’s VLSI chips can be viewed as a limited form of bounded depth, unbounded fan-in circuit [21]. Thus lower bounds within this model can justify claims from the VLSI “folklore” that certain functions cannot be realized by small PLA’s.

We define the *depth- d circuit complexity* $c_d(f)$ of a Boolean function f to be the size of the smallest circuit of depth d or less that computes f . By the above remarks, we know that this complexity measure is defined for all Boolean functions, so long as $d \geq 2$. It hence might be considered a more “reasonable” complexity measure than monotone circuit complexity, which is defined only for monotone functions. It is “unreasonable” in its own ways, however. Even relatively trivial problems X can be shown to have superpolynomial complexities $c_d(X_n)$.

The most famous example (and the subject of one of Yao’s two new results) is the “parity” function mentioned in the previous section, which asks if an odd number of the input variables are equal to 1. It is easy to compute parity with a linear number of gates if there is no bound on circuit depth. However, as Lupanov showed back in 1961 [30], any depth-2 circuit that computes parity must have an exponentially growing number of gates. Moreover, as Furst, Saxe, and Sipser [21] showed some 20 years later, no depth- d , polynomial-size circuit can compute parity for *any* fixed d . They proved this by relating the depth d circuit complexity for parity to that for depth $d - 1$ using a “probabilistic restriction” technique. In particular, they showed that $c_d(\text{parity}_n) = \Omega(n^{\log^{(3d-6)} n})$, where $\log^{(i)} n$ is defined inductively by $\log^{(1)} n = \log n$ and $\log^{(i+1)} n = \log(\log^{(i)} n)$. (Observe that, as one would expect, this “superpolynomial” becomes less and less super as the depth d increases.) Using the quoted result and an appropriate notion of reducibility, they also showed superpolynomial lower bounds on the bounded-depth circuit complexity of the *majority* function (which has value 1 if and only if more than half of the inputs are 1), and of such operations as multiplication and transitive closure. Chandra, Fortune, and Lipton extended these results, and provided an elegant group-theoretic characterization of precisely

those semigroup products for which superpolynomial lower bounds hold [15].

The results for parity, impressive though they were, left a substantial gap, since the best *upper* bound known for $c_d(\text{parity}_n)$ was $O(n2^{n^{1/(d-1)}})$. Adding much interest to this gap was a second major result from the Furst, Saxe, and Sipser paper, showing a surprising connection to the ‘‘Polynomial Hierarchy’’ introduced by Meyer and Stockmeyer [33].

Recall that this is a hierarchy of complexity classes Σ_i^P and Π_i^P , $i > 0$, based on quantifier alternation. For instance, Σ_1^P is the class of all languages that can be expressed as

$$\Sigma_1^P = \{y: \exists x_1 \forall x_2 \cdots \mathbf{Q}x_i R[y, x_1, x_2, \dots, x_i]\},$$

where R is a polynomial time computable function, and each quantification is over strings of length $|y|^k$ for some fixed k . (Π_i^P is defined analogously, with the first quantifier being a \forall . For more details, see Chapter 7 of [G&J].) If PH is defined to be the union of all the classes in the polynomial hierarchy, then we know that $\text{PH} \subseteq \text{PSPACE}$ but do not know whether the inclusion is proper.

As with many other such unresolved containment questions, one might expect the $\text{PH} =? \text{PSPACE}$ question to relativize both ways, i.e., that there would be oracle sets A and B such that $\text{PH}^A \neq \text{PSPACE}^A$ and $\text{PH}^B = \text{PSPACE}^B$. However, although it is easy to provide oracles that equate PH and PSPACE, no one had been able to find oracles that separated the two classes. Furst, Saxe, and Sipser showed that if $c_d(\text{parity}_n) \geq \Omega(n^{\log^k n})$ for every positive k and d , then there must exist an oracle set A such that $\text{PH}^A \neq \text{PSPACE}^A$. (They proved this by a diagonalization argument that considered the relativized language $L^A = \{1^n \mid \text{the number of strings of length } n \text{ in } A \text{ is odd}\}$. Note that membership of 1^n in L^A can be viewed as a Boolean function of 2^n variables, each variable asserting $z \in A$ for some string z of length n . The crux of their argument lay in showing how to construct a ‘‘small’’ depth $i + 1$ circuit for this Boolean function, given a $\Sigma_i^{P,A}$ (or $\Pi_i^{P,A}$) representation for L^A .)

Thus there is an intimate connection between relativized versions of the Polynomial Hierarchy and the bounded-depth circuit complexity for the parity function. Unfortunately, to make use of this connection, one would have to prove significantly better lower bounds on $c_d(\text{parity}_n)$ than Furst, Saxe, and Sipser had themselves obtained. The first improvement (actually an independent result) was due to Ajtai [1] and yielded lower bounds of the form $n^{c_d \log n}$. Unfortunately, this still fell far short of what was needed. We were still left with the problem of increasing the exponent from linear in $\log n$ to superpolynomial in $\log n$, the same order of gap we have been facing all along with P versus NP.

We don’t face this gap any longer, however, for Yao [53] has pushed the lower bound all the way up to a ‘‘true’’ exponential. Using a much more sophisticated application of the ‘‘probabilistic’’ restrictions of [21], he proved that $c_d(\text{parity}_n) = \Omega(2^{n^{1/4d}})$, and hence that there exists an oracle A such that $\text{PH}^A \neq \text{PSPACE}^A$. He also proved the existence of a second, even more sought-after

oracle, but in order to explain the second result, we will need some more background.

A standard question one asks when considering a proposed infinite hierarchy of complexity classes $C_1 \subseteq C_2 \subseteq \dots$, is whether that hierarchy is truly infinite. Might it not collapse into one of its early sets? With the Polynomial Hierarchy, the question is whether PH might not be totally contained in Σ_i^P for some i . This is, of course, an open problem, and is closely related to the P versus NP question: It is easy to show that for any i , $\Sigma_i^P = \Sigma_{i+1}^P$ implies $\text{PH} = \Sigma_i^P$. Thus $\text{P} (= \Sigma_0^P) = \text{NP} (= \Sigma_1^P)$ implies $\text{PH} = \text{P}$. The converse need not be true, however, since the hierarchy might collapse to a level higher than P, for instance $\text{PH} = \text{NP} \neq \text{P}$.

When we look at relativized versions of such questions, we once again find missing oracles. Although there are oracles A such that $\text{P}^A = \text{PH}^A$, $\text{P}^A \neq \text{NP}^A = \text{PH}^A$, and even such that $\text{P}^A \neq \text{NP}^A \neq \Sigma_2^{P,A} = \text{PH}^A$ (a recent result due to Heller [23]), no oracle A was known that yielded $\Sigma_i^{P,A} \neq \Sigma_{i+1}^{P,A}$ for all $i > 0$. (The most powerful result in this direction, due Baker and Selman [5], yielded only $\Sigma_2^{P,A} \neq \Sigma_3^{P,A}$.) Does an oracle exist that separates all levels of the hierarchy? This has long been considered *the* major open problem in relativized computational complexity, and it too has now been resolved in the positive by Yao. Again the result was obtained by proving a new lower bound on bounded-depth circuit complexity.

The connection with circuit complexity was established in 1983 by Sipser [44]. Sipser was looking at the analogous question of whether the classes of problems solvable by polynomial-size circuits of depth d form a true hierarchy. Here again the hierarchy would exist only if there were, for each d , a problem that could be solved by polynomial-size circuits of depth d , but not by polynomial-size circuits of depth $d - 1$. In searching for a natural candidate, Sipser came up with a sequence of functions $f_{d,1}, f_{d,2}, \dots$, that seemed to use depth d and polynomial size to their full advantage. If n is of the form m^d , the function $f_{d,n}$ is based on a depth- d tree with fan-out m at each interior vertex (and hence $n = m^d$ leaves). This tree can be turned into a monotone circuit of depth d by identifying each leaf with a distinct input and viewing each internal vertex as an \wedge or \vee of its m children, with the root being an \wedge and with \wedge 's and \vee 's alternating between levels from there on down. The function computed by this circuit is $f_{d,n}$. (For completeness, $f_{d,n}$ is assumed to be identically zero if n is *not* of the form m^d .)

Sipser's results for these functions were analogous to those of Furst, Saxe, and Sipser for parity. First, using the "probabilistic restriction" technique he was able to prove superpolynomial lower bounds on $c_{d-1}(f_{d,n})$, thus deriving the desired corollary about the non-collapse of the bounded-depth, polynomial-size circuit complexity hierarchy. Second, he showed that if the lower bounds on $c_{d-1}(f_{d,n})$ could be improved so that the exponent was superpolynomial in $\log n$, then an interesting oracle would exist, in this case the long-sought A such that $\Sigma_i^{P,A} \neq \Sigma_{i+1}^{P,A}$, for all $i \geq 0$. A partial step in this direction was made in [26], which

showed that true exponential lower bounds could be obtained if one restricted attention to *monotone* circuits of depth $d - 1$ or less. Yao leap-frogged over this result, however, and showed that true exponential lower bounds hold for *general* depth $d - 1$ circuits, thus establishing that the desired oracle exists. (The bounded-depth monotone circuit model, which had also yielded impressive results by Valiant [49] and Boppana [10], has unfortunately been reduced to an historical curiosity by the combination of Razborov and Yao, both of whom ignored it in deriving their much stronger results.)

Yao's precise bounds for $c_{d-1}(f_{d,n})$ have not yet been published as of this writing. The only write-up currently available is the preliminary one that appeared in the 1985 FOCS Proceedings [53], and that paper only provides details of the parity lower bound. Details of an *improved* lower bound on the gap between depth- d and depth- $(d - 1)$ circuits are already available, however. For a sequence of functions differing from the $f_{d,n}$ only in how the fan-out is distributed between successive levels of \wedge 's and \vee 's, Hastad [22] proves bounds exponential in $\Omega(cn^{2/(d^2+d-2)})$. Hastad [22] has also improved on Yao's lower bounds for $c_d(\text{parity}_n)$, bringing them up from $\Omega(2^{n^{1/(4d)}})$ to $\Omega(2^{cn^{1/(d-1)}})$, close to the best possible. (As often happens in such cases, the improved results also have simpler proofs.)

There has also been an improvement on the oracular front. Not only do there exist oracles A for which $\text{PH}^A \neq \text{PSPACE}^A$, but this is true for almost *all* oracle sets A . This result was proved by Cai [14] using a modification of Yao's proof. It does not appear that Yao's other result can be so directly extended, however, and so the probability that a random oracle separates all the classes of the Polynomial Hierarchy remains unknown. (For more on random oracles, see the [Sept. 1982] column.)

4. BOUNDED WIDTH BRANCHING PROGRAMS

The results discussed above for bounded depth circuits are intriguing, but one may understandably be unhappy with a machine model in which the computation of a problem as trivial as parity requires exponential resources. Thus researchers have looked for more powerful models where the same sorts of results could be proved. Many thought they had found such a model in the *bounded width branching program*. Branching programs were invented in 1959 by Lee [29], who called them "binary decision programs." Interest in them was rekindled 16 years later by Masek [31], who renamed them "decision graphs" and raised questions about lower bounds. The current terminology was popularized in 1983, when Borodin, Dolev, Fich, and Paul [12] first emphasized the restriction to constant width.

A branching program is a labelled, rooted acyclic directed graph G in which each vertex has outdegree two or less, and in which the labels obey the following constraints: (a) every sink (vertex of outdegree 0) is labelled either 0 or 1,

(b) every vertex with outdegree 2 is labelled by an input variable, and (c) the two edges leaving a vertex of outdegree 2 are labelled 0 and 1 respectively. One uses such a program to compute a Boolean function as follows. Start at the root. In general, if your current location is vertex v , do the following: If v has outdegree 0, output the label of v and halt. If v has outdegree 1, go to the vertex to which the one edge leaving v points. If v has outdegree 2 and the variable labelling v has value 1, go to the vertex pointed to by the edge leaving v that is labelled 1; otherwise go to the vertex pointed to by the edge labelled 0. Since G is acyclic, this procedure must eventually halt, outputting either a 1 or a 0.

The *size* of a branching program is the number of vertices it contains. The class BP of problems solvable by polynomial-size branching programs can be shown to consist precisely of those problems solvable by polynomial-size, $O(\log n)$ -width, bounded fan-in circuits. (The width of a circuit is defined as follows: The *thickness at level d* of a circuit is the number of gates with depth d or less that provide inputs to gates with depth exceeding d . The *width* of the circuit is the thickness of its thickest level.)

Thus BP would appear to be properly contained in the class of problems solvable by arbitrary polynomial-size circuits. BP in turn contains another famous class (again presumably properly). This is the class “non-uniform NC^1 ” of those problems that can be solved by polynomial-size, $O(\log n)$ -depth, bounded fan-in circuits. (Note that for such circuits the qualifier “polynomial-size” is redundant. The qualifier “non-uniform” is not redundant, however; it is needed to distinguish this class from the more common “ $ATIME(\log n)$ -uniform NC^1 .” Unless otherwise stated, all references to NC^1 that follow refer to the non-uniform class.) As shown by Spira [45], NC^1 coincides with the class of problems expressible by polynomial-size Boolean *formulas*. A Boolean formula is a linear expression involving variables, parentheses, and binary/unary Boolean connectives like \wedge , \vee , and \neg , that represents a Boolean function in the standard way. Alternatively (and more appropriately for this column), it can be viewed as a Boolean circuit in which no circuit element other than an input can have fan-out exceeding 1. It is not difficult to show that any Boolean formula can be simulated by a branching program with no more than a polynomial blow-up in size, and hence we have $NC^1 \subseteq BP$.

Now even though NC^1 would seem to be a very restricted class, we as yet know of no way to show that NC^1 does not contain all of NP. Indeed, the best lower bound known for the formula size of a problem in NP is still only $\Omega(n^2)$ [28] (or $\Omega(n^2/\log n)$ if arbitrary Boolean connectives are allowed [34]; note that the choice of connectives can only make a polynomial difference so long as a complete basis is chosen [35]). Branching programs, which are likely to be more powerful than formulas, are thus not weak enough to offer much current hope of proving superpolynomial size bounds for problems in NP. Thus we again look for restrictions on the model.

Our first restriction, to *levelled* programs, does not limit the power of the

programs, but puts them into a normal form that will make our more serious restrictions simpler to define and impose. In a levelled branching program the vertices can be partitioned into *levels* L_0, L_1, \dots , such that, for all i , an edge leaving a vertex at level L_i enters a vertex at level L_{i+1} . It is an easy exercise to show that any branching program of size n can be converted to a levelled program of size n^2 that computes the same function [13]. From now on, all branching programs will be assumed to be levelled.

The *length* of a program is the number of levels it contains. Its *width* is the maximum size of a level. One might at first think of restricting levelled programs by limiting their length. (Length is the natural analogue of running time for branching programs, just as depth was for Boolean circuits.) However, if one imposes a finite length bound, then for most interesting problems X , X_n become uncomputable for all sufficiently large n (because the programs can only examine a fixed number of variables during any computation). On the other hand, *any* Boolean function can be computed by a width-2 branching program. Moreover, it turns out that width is far more closely related to the depth of Boolean circuits than one might have expected. For any $d \geq 2$, the class of problems solvable by polynomial-size depth- d unbounded fan-in circuits is contained in the class solvable by polynomial-size width- d branching programs. (The former class is identical to the class of problems expressible by polynomial-size formulas with fewer than d quantifier alternations, and these can be simulated by branching programs of width d , so long as $d \geq 2$.) Width d is more powerful than depth d , however. Parity, the problem that Yao [53] showed had exponential circuit complexity for any fixed depth bound, can be solved by *linear* size width-2 branching programs.

Thus bounded-width branching programs provide a potentially interesting step up in power from bounded depth circuits. They also provide a sufficient step down from general branching programs that superpolynomial lower bounds may be possible: Let $b_w(f)$ be the size of the smallest branching program of width w or less that computes f . If we let BWBP_w denote the class of problems X for which $b_w(X_n)$ is bounded by a polynomial in n , then $\text{BWBP}_w \subseteq \text{NC}^1$ for all $w > 0$. We thus might hope to prove the same sorts of results for bounded width as we did for bounded depth, despite the increased power of bounded width. (Note that for bounded width, size and length grow at roughly the same rate, so here size can be viewed as measuring running time.) A first question to ask is whether bounded-width branching programs are still sufficiently weak that relatively simple problems X can be excluded from BWBP_2 . The answer is yes. Parity is no longer a candidate, but Shearer [43] has proved exponential lower bounds for width-2 programs that determine whether the number of non-zero inputs is divisible by 3, and Yao has proved superpolynomial lower bounds on width-2 programs for the majority function (which asks if more than half the inputs are non-zero) [52].

A more significant question is whether we can obtain superpolynomial lower

bounds that hold for all fixed widths, just as superpolynomial lower bounds hold on $c_d(\text{parity}_n)$ for all depths d . In other words, if $\text{BWBP} = \cup_{w \geq 2} \text{BWBP}_w$, are there any problems in $\text{NP} - \text{BWBP}$? The mod-3 problem above won't do, since it can be solved by polynomial-size programs as soon as width-3 is allowed. Borodin et al. [12] proposed the majority function as a candidate, conjecturing that $b_w(\text{majority}_n)$ was superpolynomial in n for all fixed w . The best that could be proved at the time, however, was an only slightly *superlinear* lower bound, due to Chandra, Furst, and Lipton [16]. As of this writing, the situation is only slightly better, the current best bounds, due to Pudlák [36], being of the form $\Omega(n \log \log n / \log \log \log n)$.

Another desideratum for bounded width branching programs would be a hierarchy result, i.e., a proof that BWBP_{w-1} is strictly contained in BWBP_w for all $w > 2$. (This would be in analogy with the superpolynomial gap between $c_d(f_{d,n})$ and $c_{d-1}(f_{d,n})$ for all $d > 2$.) However, here too the best bounds currently known are only superlinear: one can test whether at least w of the inputs are non-zero with width- w programs of size $O(n)$, but size $\Omega(n \log \log n / \log \log \log n)$ is required if only width $w - 1$ is available [36].

We thus remain far short of what is desired in both cases. Usually in such situations researchers shrug and mutter something about needing new proof techniques, but in this case a surprising new result of Barrington [6] gets us off the hook much more satisfactorily: The classes BWBP_w do *not* form an infinite hierarchy, and the majority function cannot be excluded from BWBP ! Both claims follow as corollaries from Barrington's main result, which states that $\text{BWBP}_5 = \text{NC}^1$. (In fact, any problem in NC^1 is solvable by a very restricted form of polynomial-size, width-5 branching program, called by Barrington a "permutation branching program.") Since we already know that $\text{BWBP} \subseteq \text{NC}^1$, this means that $\text{BWBP}_5 = \text{BWBP} = \text{NC}^1$. Thus the hierarchy of classes BWBP_w collapses into BWBP_5 and, since the majority function can be computed by polynomial-size, $O(\log n)$ -depth circuits (indeed, by polynomial-size monotone Boolean formulas! [50]), the complexity $b_w(\text{majority}_n)$ is polynomial for all $w \geq 5$.

Barrington's result also has corollaries back in the realm of circuit complexity. It is not difficult to show that BWBP equals the class of problems solvable by polynomial-size, bounded-width circuits [24]. Thus, since a width-5 permutation branching program can be simulated with only a linear blow-up in size by a width-5 circuit [7], we conclude that the bounded-width hierarchy for polynomial-size circuits also collapses to its 5th level (and equals NC^1). For polynomial-size, bounded fan-in circuits, width-5 is thus equivalent to $O(\log n)$ -depth (and, recalling Spira's result [45], both are equivalent to fan-out-1).

It is not clear whether Barrington's result and its corollaries (which also hold for the $\text{ATIME}(\log n)$ -uniform versions of the classes) make the bounded-width restriction less or more attractive. To the extent that we thought non-membership might be easier to prove for BWBP than for NC^1 , we have lost opportunities. On the other hand, we are now just three steps away from

proving that NC^1 does not contain NP. We know that $BWBP_2$ does not contain NP; this result need only be extended in turn to $BWBP_3$, $BWBP_4$, and $BWBP_5$, and we will be done. Just in case we get stuck, however, researchers have come up with another restriction on branching programs to investigate. This is a restriction on the number of times that a variable may be read during a computation (the number of times a given vertex label may occur in a path). Exponential lower bounds have already been proved for problems in NP and “read once only” programs [2,20,54], and although these have yet to be extended even to “read twice” programs, a “read k only” hierarchy with exponential gaps has already been conjectured [31].

5. WHAT NEXT?

Even though the main results I have discussed offer major breakthroughs in lower bound technology, is it unlikely that any of the specific techniques developed for taking advantage of monotonicity and/or constant depth and width will be applicable to the general case of polynomial-size circuits. Nevertheless, in talking to other researchers I begin to detect a new optimism about the possibility of an early proof that $P \neq NP$. The results of Razborov and Yao jumped over gaps that were as big as those we face with P versus NP (linear versus super-polynomial) and that seemed nearly as unjumpable. The proofs depended only on combinatorial arguments (albeit deep and original ones). This suggests an encouraging paradigm, and who knows what a little encouragement might do?

REFERENCES

1. M. AJTAI, Σ_1^1 -formulae on finite structures, *Annals of Pure and Applied Logic* **24** (1983), 1-48.
2. M. AJTAI, L. BABAI, P. HAJNAL, J. KOMLÓS, E. SZEMERÉDI, AND G. TURAN, Two lower bounds for branching programs, in “Proceedings 18th Ann. ACM Symp. on Theory of Computing,” Association for Computing Machinery, New York, 1986.
3. N. ALON AND R. B. BOPANA, The monotone circuit complexity of Boolean functions, *Combinatorica*, to appear.
4. A. E. ANDREEV, On a method for obtaining lower bounds for the complexity of individual monotone functions, *Dokl. Ak. Nauk. SSSR* **282** (1985), 1033-1037 (in Russian). English translation in *Sov. Math. Dokl.* **31** (1985), 530-534.
5. T. P. BAKER AND A. L. SELMAN, A second step toward the polynomial hierarchy, *Theor. Comput. Sci.* **8** (1979), 177-187.
6. D. A. BARRINGTON, Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1 , in “Proceedings 18th Ann. ACM Symp. on Theory of Computing,” Association for Computing Machinery, New York, 1986.
7. D. A. BARRINGTON, private communication (1986).
8. S. J. BERKOWITZ, private communication cited in [49] (1982).
9. N. BLUM, A Boolean function requiring $3n$ network size, *Theor. Comput. Sci.* **28** (1984), 337-345.

10. R. B. BOPPANA, Threshold functions and bounded depth monotone circuits, in "Proceedings 16th Ann. ACM Symp. on Theory of Computing," pp. 475-479, Association for Computing Machinery, New York, 1984.
11. R. B. BOPPANA, private communication (1986).
12. A. BORODIN, D. DOLEV, F. E. FICH, AND W. PAUL, Bounds for width two branching programs, in "Proceedings 15th Ann. ACM Symp. on Theory of Computing," pp. 87-93, Association for Computing Machinery, New York, 1983.
13. A. BORODIN, M. J. FISCHER, D. G. KIRKPATRICK, N. A. LYNCH, AND M. TOMPA, A time-space tradeoff for sorting on non-oblivious machines, *J. Comput. System Sci.* **22** (1981), 351-364.
14. J.-Y. CAI, With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy, in "Proceedings 18th Ann. ACM Symp. on Theory of Computing," Association for Computing Machinery, New York, 1986.
15. A. K. CHANDRA, S. FORTUNE, AND R. LIPTON, Unbounded fan-in circuits and associative functions, *J. Comput. System Sci.* **30** (1985), 222-234.
16. A. K. CHANDRA, M. L. FURST, AND R. J. LIPTON, Multi-party protocols, in "Proceedings 15th Ann. ACM Symp. on Theory of Computing," pp. 94-99, Association for Computing Machinery, New York, 1983.
17. A. K. CHANDRA, L. J. STOCKMEYER, AND U. VISHKIN, A complexity theory for unbounded fan-in parallelism, in "Proceedings 23rd Ann. Symp. on Foundations of Computer Science," pp. 1-13, IEEE Computer Society, Los Angeles, 1982.
18. D. CRYPTON, The limits of mathematical knowledge, *Science Digest* **94:3** (March 1986), 72-75.
19. A. EHRENFUCHT, "Practical decidability," Report No. CU-CS-008-72, Dept. of Computer Science, University of Colorado, Boulder, 1972. (Originally written in 1967.)
20. S. FORTUNE, J. HOPCROFT, AND E. M. SCHMIDT, The complexity of equivalence and containment for free single variable program schemes, in "Automata, Languages, and Programming," pp. 227-240, Lecture Notes in Computer Science, Vol. 62, Springer, Berlin, 1978.
21. M. FURST, J. SAXE, AND M. SIPSER, Parity, circuits, and the polynomial time hierarchy, in "Proceedings 22nd Ann. Symp. on Foundations of Computer Science," pp. 260-270, IEEE Computer Society, Los Angeles, 1981. Journal version appeared in *Math. Systems Theory* **17** (1984), 13-27.
22. J. HASTAD, Improved lower bounds for small depth circuits, in "Proceedings 18th Ann. ACM Symp. on Theory of Computing," Association for Computing Machinery, New York, 1986.
23. H. HELLER, Relativized polynomial hierarchies extending two levels, *Math. Systems Theory* **17** (1984), 71-84.
24. H. J. HOOVER, Characterizing bounded width, manuscript (1983).
25. M. JERRUM AND M. SNIR, Some exact complexity results for straight-line computations over semirings, *J. Assoc. Comput. Mach.* **29** (1982), 874-897.
26. M. KLAWE, W. PAUL, N. PIPPENGER, AND M. YANNAKAKIS, On monotone Boolean formula with restricted depth, in "Proceedings 16th Ann. ACM Symp. on Theory of Computing," pp. 480-487, Association for Computing Machinery, New York, 1984.
27. G. KOLATA, Must 'hard problems' be hard?, *Science* **228** (26 April 1985), 479-481.
28. V. M. KHRAPCHENKO, On the complexity of the realization of the linear function in the class of π -circuits, *Mat. Zametki* **9** (1971), 35-40 (in Russian). English translation in *Math. Notes of the Academy of Sciences of the USSR* **9** (1971), 21-23.
29. C. Y. LEE, Representation of switching functions by binary decision programs, *Bell Syst. Tech. J.* **38** (1959), 985-999.
30. O. LUFANOV, Implementing the algebra of logic functions in terms of constant depth formulas in the basis $\&$, \vee , \neg , *Dokl. Ak. Nauk. SSSR* **136** (1961), 1041-1042 (in Russian). English translation in *Sov. Phys. Dokl* **6** (1961), 107-108.
31. W. MASEK, "A fast algorithm for the string editing problem and decision graph complexity," M. S. Thesis, Dept. of Electrical Engineering, M.I.T., Cambridge, Mass., 1976.

32. A. R. MEYER, unpublished lecture notes, 1974.
33. A. R. MEYER AND L. J. STOCKMEYER, The equivalence problem for regular expressions with squaring requires exponential time, in "Proceedings 13th Ann. Symp. on Switching and Automata Theory," pp. 125-129, IEEE Computer Society, Los Angeles, 1972.
34. E. I. NECIPORUK, A Boolean function, *Dokl. Ak. Nauk. SSSR* **169** (1966), 765-766 (in Russian). English translation in *Sov. Math. Dokl.* **7** (1966), 999-1000.
35. V. R. PRATT, The effect of basis on size of Boolean expressions, in "Proceedings 16th Ann. Symp. on Foundations of Computer Science," pp. 119-121, IEEE Computer Society, Los Angeles, 1975.
36. P. PUDLÁK, A lower bound on complexity of branching programs, in "Mathematical Foundations of Computer Science," pp. 480-489, Lecture Notes in Computer Science, Vol. 176, Springer, Berlin, 1984.
37. A. A. RAZBOROV, Lower bounds on the monotone complexity of some Boolean functions, *Dokl. Ak. Nauk. SSSR* **281** (1985), 798-801 (in Russian). English translation in *Sov. Math. Dokl.* **31** (1985), 354-357.
38. A. A. RAZBOROV, A lower bound on the monotone network complexity of the logical permanent, *Mat. Zametki* **37** (1985), 887-900 (in Russian). English translation in *Math. Notes of the Academy of Sciences of the USSR* **37** (1985), 485-493.
39. J. SAVAGE, "The Complexity of Computing," John Wiley & Sons, New York, 1976.
40. C. P. SCHNORR, "An approach to $P \neq NP$," talk given at the Project MAC Workshop Conference on Concrete Computational Complexity, Dedham, Mass., August, 1973.
41. C. P. SCHNORR, A lower bound on the number of additions in monotone computations, *Theor. Comput. Sci.* **2** (1976), 305-315.
42. C. E. SHANNON, The synthesis of two-terminal switching networks, *Bell Syst. Tech. J.* **28** (1949), 59-98.
43. J. SHEARER, private communication cited in [52] (1983).
44. M. SIPSER, Borel sets and circuit complexity, in "Proceedings 15th Ann. ACM Symp. on Theory of Computing," pp. 61-69, Association for Computing Machinery, New York, 1983.
45. P. M. SPIRA, On time-hardware complexity tradeoffs for Boolean functions, in "Proc. 4th Hawaii Symp. on System Sciences," pp. 525-527, Western Periodicals Co., North Hollywood, Calif., 1971.
46. L. J. STOCKMEYER, "The Complexity of Decision Problems in Automata Theory and Logic," Doctoral Thesis, Dept. of Electrical Engineering, M.I.T., Cambridge, Mass., 1974.
47. J. TIEKENHEINRICH, A $4n$ lower bound on the monotone network complexity of a one-output Boolean function, *Inform. Process. Lett.* **18** (1984), 201-202.
48. L. G. VALIANT, Negation can be exponentially powerful, in "Proceedings 11th Ann. ACM Symp. on Theory of Computing," pp. 189-196, Association for Computing Machinery, New York, 1979.
49. L. G. VALIANT, Exponential lower bounds for restricted monotone formulae, in "Proceedings 15th Ann. ACM Symp. on Theory of Computing," pp. 110-117, Association for Computing Machinery, New York, 1983.
50. L. G. VALIANT, Short monotone formulae for the majority function, *J. Algorithms* **5** (1984), 363-366.
51. I. WEGENER, On the complexity of slice functions, in "Mathematical Foundations of Computer Science," pp. 553-561, Lecture Notes in Computer Science, Vol. 176, Springer, Berlin, 1984.
52. A. C. YAO, Lower bounds by probabilistic arguments, in "Proceedings 24th Ann. Symp. on Foundations of Computer Science," pp. 420-428, IEEE Computer Society, Los Angeles, 1983.
53. A. C.-C. YAO, Separating the polynomial-time hierarchy by oracles, in "Proceedings 26th Ann. Symp. on Foundations of Computer Science," pp. 1-10, IEEE Computer Society, Los Angeles, 1985.

54. S. ZAKS, An exponential lower bound for one-time-only branching programs, *in* "Mathematical Foundations of Computer Science," pp. 562-566, Lecture Notes in Computer Science, Vol. 176, Springer, Berlin, 1984.