Statistical Science
1993, Vol. 8, No. 1, 20-25

Probabilistic Algorithms for Speedup

Joan Feigenbaum and Jeffrey C. Lagarias

Abstract. This article surveys situations in which probabilistic algorithms offer speedup over what is possible using deterministic algorithms, either in practice or in principle.

Key words and phrases: Communication complexity, integer factorization, primality testing, probabilistic complexity classes.

1. INTRODUCTION

One of the most compelling reasons to use randomized algorithms is that they permit certain problems to be solved faster than is possible by deterministic methods.

One pays a price for such speedup, namely the possibility of occasional very long computations or of occasional errors in the computation. The amount of possible speedup may depend on whether the algorithm is required to give either a correct answer or no answer or it is permitted to give incorrect answers.

Section 2 describes a formal version of such questions from the viewpoint of computational complexity theory, in terms of probabilistic complexity classes. The question of whether probabilistic algorithms can give a superpolynomial speedup over deterministic algorithms is restated there as the unsolved problem, does P $\neq$ BPP? Thus it is unresolved whether or not speedup exists, and settling this question is likely to be as hard as the notorious P $\neq$ NP? problem of complexity theory.

There is a weaker sense in which probabilistic algorithms currently offer advantages over deterministic ones. There exist problems for which the fastest known algorithm is probabilistic and problems for which at present the fastest fully analyzable algorithm is probabilistic. Here the situation reflects the current limitations on our knowledge. Section 3 describes examples of such problems from number theory, such as primality testing and factorization. In both cases, at present, the best fully analyzed algorithms are probabilistic ones.

Probabilistic algorithms give a demonstrable speedup in the exchange of information among several people, each having partial information to begin with. This is the subject matter of communication complexity. It plays an important role in distributed computing, in which exchange of information among the processors is an important part of the computational task. Communication complexity results are described in Section 4.

2. PROBABILISTIC COMPLEXITY CLASSES

The theoretical possibility of speedup has been formalized in computational complexity theory with the notion of *probabilistic complexity classes*; these classify computational problems according to the running times and types of error allowed by probabilistic algorithms. Here we define several of these complexity classes.

The model of computation used is a probabilistic Turing machine. This is a Turing machine that has an extra "random" tape containing a random string of zeros and ones that can be read as necessary. A *probabilistic polynomial-time Turing machine* (*PPTM*) is such a machine equipped with a clock that, when given an input of n bits, always halts after $p(n)$ steps, where p is a fixed polynomial. The performance of such machines is averaged over the uniform distribution of all random bits read by the machine. (At most $p(n)$ random bits are read during the computation on an n-bit input.) The computational problems considered are those of recognizing a language $\mathcal{L} \subseteq \{0,1\}^*$, where $\{0,1\}^*$ denotes the set of all finite strings of zeros and ones. For example, $\mathcal{L}$ might consist of all prime numbers, expressed using their binary representations.

We say that $\mathcal{L}$ is in the complexity class *BPP* (*bounded-error-probability polynomial time*) if there is a PPTM that, given $x \in \mathcal{L}$, outputs "$x \in \mathcal{L}$" with probability at least 3/4, where probability is computed over all random tapes for the fixed input x, and given $x \notin \mathcal{L}$ outputs "$x \notin \mathcal{L}$" with probability at least 3/4 similarly. In this complexity class, two-sided classification errors are allowed: The machine may output "$x \in \mathcal{L}$" when in fact $x \notin \mathcal{L}$ and vice versa. Note that any problem in

Joan Feigenbaum is a Member of the Technical Staff, Computing Principles Research Department. Jeffrey C. Lagarias is a Distinguished Member of Technical Staff, Mathematical Sciences Research Center, AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974.

BPP is solvable by a PPTM for which the error probability is exponentially small. The new PPTM repeats the test for $x \in \{0, 1\}^n$ on the old PPTM n times using independent random bits each time and then takes a majority vote on the n outputs to decide whether $x \in \mathcal{L}$; the error probability for the new PPTM is less than $(3/4)^{n/2}$.

We say that $\mathcal{L}$ is in the complexity class *RP* (*random polynomial time*) if there is a PPTM that, when given an input $x \in \mathcal{L}$, outputs "$x \in \mathcal{L}$" at least 3/4 of the time and, when given an input $x \notin \mathcal{L}$, always outputs "$x \notin \mathcal{L}$." That is, only one-sided error is allowed. We say that $\mathcal{L}$ is in the complementary complexity class *co-RP* (*co-random polynomial time*) if there is a PPTM that, when given an input $x \notin \mathcal{L}$, outputs "$x \notin \mathcal{L}$" at least 3/4 of the time and, when given an input $x \in \mathcal{L}$, always outputs "$x \in \mathcal{L}$."

Finally, we say $\mathcal{L}$ is in *ZPP* (*zero-error-probability polynomial time*) if $\mathcal{L}$ is both RP and co-RP. In this case, when the PPTM outputs either "$x \in \mathcal{L}$" or "$x \notin \mathcal{L}$," it is correct, but it is allowed to give no output for a small fraction of the random tapes.

The complexity class BPP is closed under complementation, and we have the following inclusions shown in Figure 1, where P denotes the set of (deterministic) polynomial-time recognizable languages.

In this framework, it is unknown whether probabilistic algorithms ever give a superpolynomial speedup, that is, whether there are any languages in BPP that are not in P. Yao (1982) announced a converse result suggesting that at most a subexponential speedup is possible.

THEOREM 1 (Yao, 1982). *If there exists a uniform secure polynomial pseudorandom bit generator, then*

$$BPP \subseteq \bigcap_{\varepsilon > 0} \text{DTIME}(2^{n^\varepsilon}).$$

For elaboration of the notion of uniform secure polynomial pseudorandom bit generator, one can consult the article by Lagarias that appears in this issue, and for a proof of Yao's theorem, one can consult Boppana and Hirschfeld (1989). Several researchers, most recently Babai et al. (1991), have since improved the result by showing that the same conclusion follows from weaker hypotheses.

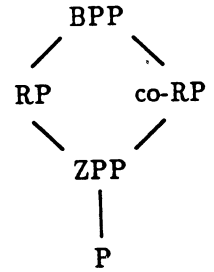

FIG. 1. *Relationships between complexity classes.*

A detailed discussion of probabilistic complexity classes is given in Johnson (1990) and Zachos (1988).

3. PROBABILISTIC ALGORITHMS IN NUMBER THEORY: FACTORING AND PRIMALITY TESTING

The *primality testing problem* is that of determining whether an integer N is prime or composite, and the *factoring problem* is that of finding all the prime factors of N. These are two of the most basic computational problems in number theory.

The primality testing and factoring problems have the added practical significance of playing complementary roles in the RSA cryptosystem, which is the current best candidate for public key cryptography and for digital signatures. The practicality of constructing RSA ciphers depends on primality testing's being easy to do, whereas the security of the cipher depends on the apparent difficulty of factoring large numbers.

At present, the fastest known fully analyzed algorithms for both primality testing and factoring are probabilistic.

Theoretical and practical progress in primality testing has been rapid since 1977. Analysis of various primality testing algorithms led to study of the complexity classes described in Section 2.

Most methods of primality testing are based on variants of Fermat's little theorem, which asserts that if N is prime, then

$$a^{N-1} \equiv 1 \pmod{N} \tag{1}$$

if N does not divide a. Most composite numbers have many residues $a \pmod N$ for which $a^{N-1} \not\equiv 1 \pmod N$; hence a test for probable primality is to test whether $a^{N-1} \equiv 1 \pmod N$ for many random a. Unfortunately, there exist composite numbers N for which $a^{N-1} \equiv 1 \pmod N$ if $(a, N) = 1$; these are called *pseudoprimes*. Refinements of the condition (1) can be used to rule out pseudoprimes, either probabilistically or deterministically.

The primality testing problem is believed by many to be solvable in polynomial time. In fact, Miller (1976) proposed a deterministic algorithm believed to have this property:

Miller's Primality Test

Given $N \equiv 1 \pmod 2$, find the largest 2^k exactly dividing $N - 1$.
  For $1 \le a \le 10 (\log N)^2$ do:
    Is $a^{N-1} \equiv 1 \pmod N$? If not, output "N composite."
    For $1 \le j \le k$, do
      Is $a^{N-1/2^j} \not\equiv \pm 1 \pmod N$? If so, output "$N$ composite."
      Is $a^{N-1/2^j} \equiv -1 \pmod N$? If so, exit loop.
    continue
  continue.
Output "N prime."

Note that $a^{N-1} \pmod N$ may be computed in polynomial time by expanding N in binary and computing $a^{2^j} \pmod N$ by repeated squaring and reduction (mod N), so this is a polynomial-time algorithm. Miller proved that this algorithm is always correct when it outputs "N composite" and that, assuming the truth of the extended Riemann hypothesis (ERH), it is correct when it outputs "N prime." However, because the ERH is unproven, it is currently not actually known to be a primality test.

Probabilistic analogs of this algorithm were developed by Solovay and Strassen (1977) and Rabin (1980):

Solovay-Strassen Primality Test

Given $N \equiv 1 \pmod 2$, find the largest 2^k exactly dividing $N - 1$.
  Draw a at random from $[1, N-1]$.
    Is $a^{N-1} \equiv 1 \pmod N$? If not, output "N composite."
    For $1 \le j \le k$, do
      Is $a^{N-1/2^j} \not\equiv \pm 1 \pmod N$? If so, output "$N$ composite."
      Is $a^{N-1/2^j} \equiv -1 \pmod N$? If so, exit loop.
    continue.
Output "N prime."

They proved that, if N is composite, this test outputs "N is composite" with probability at least 3/4 and, if N is prime, it always outputs "N is prime." This shows that $Primes \equiv \mathcal{L} \in$ co-RP. By repeating the test $(\log N)^2$ times, the probability of error can be made exponentially small.

The drawback of the Solovay–Strassen test is that it does not guarantee that N is prime when it outputs "N prime"; that is, it provides no proof of primality.

This situation was partially remedied by Goldwasser and Kilian (1986). They gave a probabilistic algorithm that counts points on "random" elliptic curves (mod N), and they proved that for almost all primes p it finds in probabilistic polynomial time a proof that p is prime. However, they could not rule out the possibility of the existence of a small set of "exceptional" p for which there was no proof of the kind they searched for. Recently, Adleman and Huang (1987) announced a proof that $Primes \in$ RP. Their result is technically difficult and proceeds by counting points on "random" Abelian varieties (mod p) of certain types. Taken together with the Solovay–Strassen result, it shows that $Primes \in$ ZPP.

In contrast, the fastest fully analyzed deterministic primality testing algorithm is due to Adleman, Pomerance and Rumely (1983). It tests N for primality in time $O((\log N)^{c_0 \log\log\log N})$ for a certain positive constant c_0.

For the factoring problem, all known algorithms, whether probabilistic or deterministic, require superpolynomial time. The fastest fully analyzed deterministic algorithm for factoring N, due to Pollard (1974), has the exponential running-time bound $O(N^{1/4+\varepsilon})$.

The algorithms used in practice for factoring large numbers are probabilistic in nature. The basic approach of most of them is to find solutions to $X^2 \equiv Y^2 \pmod N$ and hope that $(X + Y, N)$ is a nontrivial factor of N. This is done by generating probabilistically many pairs (X_i, a_i) with $X_i^2 \equiv a_i \pmod N$, where the a_i's are not squares, and then finding a product of the a_i's that is a square by an auxiliary method. The basic approach is to consider only a_i's having all prime factors less than a bound y; such numbers are called *y-smooth*. Those a_i's that do not have this property are discarded. Then the y-smooth a_i's are factored

$$a_i = \prod_{p_j < y} p_j^{e_{ij}}.$$

Now if sufficiently many such a_i's are known, the matrix $[e_{ij}]$ considered over $GF(2)$ must have a linear dependency. The product of the a_i's corresponding to this linear dependency is the desired square Y^2, where X^2 is the product of the corresponding X_i^2. This approach was first described in the Morrison and Brillhart (1975) continued fraction method. Dixon (1981) gave a fully analyzable version of the method. Let $L = \exp[(\log n \log\log n)^{1/2}]$. The Dixon algorithm uses y-smooth numbers with $y = L^\beta$, for a small constant β.

Dixon's Random Squares Factoring Algorithm

1. Test N for primality. If N is not prime, continue.
2. Randomly pick X_i and compute the least positive residue a_i of $X_i^2 \pmod N$.
3. Trial divide a_i by all primes $p_j < L^\beta$. If a_i completely factors as
$$a_i = \prod_{p_j < L^\beta} p_j^{e_{ij}},$$
add (X_i, a_i, e_{ij}) to a list. Continue until the list has L^β elements.
4. The matrix $[e_{ij}]$ is now linearly dependent over $GF(2)$. Compute a linear dependency, that is, a set S of rows with
$$\sum_{i \in S} e_{ij} \equiv 0 \pmod 2 \quad \text{for all } j.$$
5. Set $X = \prod_{i \in S} X_i$ and $Y^2 = \prod_{i \in S} a_i$. Then $X^2 \equiv Y^2 \pmod N$. Test whether $(X + Y, N)$ is a nontrivial factor of N. If so, halt. If not, go to Step 2 and repeat.

Here a dependency in the matrix $[e_{ij}]$ is found using Gaussian elimination over $GF(2)$. Dixon (1981) showed that the probability of finding a factor of a composite N in one pass through this algorithm is at least 1/2.

The key to analysis of the algorithm is that the probability that a random integer in $[1, N]$ is L^β-smooth is about $L^{-1/2\beta}$. Dixon showed that a similar probability

estimate holds for the a_i's that are random quadratic residues X_i^2. Hence the expected time needed to generate the matrix $[e_{ij}]$ is $L^{\beta+(1/2\beta)}$, and the time needed to Gaussian reduce the matrix $[e_{ij}]$ is about $L^{3\beta}$. This is minimized for $\beta = 1/2$, and the algorithm takes time $O(L^{3/2+\varepsilon})$ and space $O(L^{1+\varepsilon})$ bits. This is Dixon's result.

Note that this expected running time bound,

$$O\left(\exp\left[\left(\frac{3}{2}+\varepsilon\right)(\log N \log\log N)^{1/2}\right]\right),$$

is subexponential in N; that is, it represents a superpolynomial speedup of the currently best fully analyzed deterministic factoring algorithm.

Pomerance (1987) gives a refined variant of this probabilistic algorithm that is rigorously proven to run in expected time $O(L^{\sqrt{2}+\varepsilon})$. There are several other factoring algorithms of a similar nature that are believed to run in expected time $O(L^{1+\varepsilon})$ (Lenstra and Lenstra, 1990; Pomerance, 1982), and one such algorithm for which the bound $O(L^{1+\varepsilon})$ can be proven rigorously (Lenstra and Pomerance, 1992).

More recently, a new factoring algorithm, the number field sieve, has been found that incorporates probabilistic ideas and uses arithmetic in an auxiliary algebraic number field to generate $X^2 \equiv Y^2 \pmod N$. It appears to have running time $O\{\exp[(2+\varepsilon)(\log N(\log\log N)^2)^{1/3}]\}$ but is likely not to be fully analyzable. It is particularly efficient for factoring numbers of special form, such as $x^n + 2$, and has already been used in practice to factor a 158-digit number; a preliminary description appears in Lenstra et al. (1990).

4. COMMUNICATION COMPLEXITY

Communication complexity studies variants of the following problem.

A Boolean function f is defined on a finite set $X \times Y$. Two persons, P_X knowing $x \in X$ and P_Y knowing $y \in Y$, both want to know $f(x, y)$. How many bits of information do they need to exchange to do this?

This problem was proposed by Yao (1979). It is a basic problem in distributed computation, where the "persons" are processors, each possessing partial information about the current state of the computation.

The *worst-case (deterministic) communication complexity* $\hat{C}_D(f)$ is defined as follows. Let φ be a deterministic protocol describing a sequence of bit exchanges between P_X and P_Y that always computes f correctly. Let $l_\varphi(x, y)$ denote the number of bits exchanged using φ when P_X knows x and P_Y knows y. The worst-case communication complexity for the protocol φ is

$$\hat{C}_D(f)_\varphi := \max_{(x,y)\in X\times Y} [l_\varphi(x, y)],$$

and the worst-case deterministic communication complexity for f is

$$\hat{C}_D(f) := \inf_\varphi [\hat{C}_D(f)_\varphi].$$

Randomized algorithms offer provable speedups for some problems in communication complexity. Here one allows protocols φ in which P_X and P_Y each have access to different sources of independent random bits. Let φ be a protocol computing $f(x, y)$ with probability of error at most ε for all (x, y). Let $\bar{l}_\varphi(x, y)$ denote the expected number of bits needed by φ when P_X knows x and P_Y knows y, and define

$$\hat{C}_R(f)_\varphi := \max_{(x,y)\in X\times Y} [\bar{l}_\varphi(x, y)].$$

The *ε-error randomized communication complexity* is

$$\hat{C}_R(f, \varepsilon) := \inf[\hat{C}_R(f)_\varphi],$$

where the infimum is computed over all protocols with error probability at most ε. We single out the *zero-error randomized communication complexity* $\hat{C}_R(f, 0)$.

We describe two specific functions for which probabilistic computations give speedups. The first is the *equality function equ*. Here $X = Y = \{1, 2, \ldots, n\}$, and

$$equ(x, y) = \begin{cases} 1, & \text{if } x = y, \\ 0, & \text{otherwise.} \end{cases}$$

Yao (1979) showed that $\hat{C}_D(equ) \geq [\log(n)]$. In fact, $\hat{C}_D(equ)$ is exactly $[\log(n)] + 1$. A protocol attaining this bound is as follows: P_X transmits x to P_Y, and P_Y transmits back the value $f(x, y)$. Yao (1979) also gave a randomized protocol φ with error probability at most ε and showed that

$$\hat{C}_R(equ)_\varphi \leq c_0\left(\log\log n + \log\left(\frac{1}{\varepsilon}\right)\right),$$

thus demonstrating a speedup. To describe this protocol, let $\sigma = \sqrt{2}\,\dfrac{\log n}{\varepsilon}$.

Randomized Prime Protocol

1. P_Y picks a random prime p in the interval $\sigma \leq p \leq 2\sigma$. He transmits p and $y \pmod p$.
2. P_X transmits 1 if $x \equiv y \pmod p$ and 0 otherwise.
3. P_X and P_Y both take the bit transmitted to be the value $equ(x, y)$.

If this algorithm outputs $equ(x, y) = 0$, this answer is correct, but when it outputs $equ(x, y) = 1$, this answer may be incorrect, with probability at most ε. For the function *equ*, no speedup exists using a zero-error randomized algorithm (see Orlitsky and El Gamal, 1990).

Next we consider the function *componentwise-equal* (*c.e.*), which has $X = Y = \{0, 1\}^{n^2}$. Write $\mathbf{x} = (x_1, \ldots, x_n)$ with each $x_i \in \{0, 1\}^n$ and similarly for y. Then

$$c.e.(\mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if for some } i,\ x_i = y_i, \\ 0, & \text{else.} \end{cases}$$

Mehlhorn and Schmidt (1982) showed that $\hat{C}_D(c.e.) \geq n^2$. They exhibited a zero-error randomized protocol φ such that

$$\hat{C}_R(c.e.)_\varphi \leq 5n \log n + c_0 .$$

The protocol is as follows.

Repeated Equality Protocol

1. For $1 \leq i \leq n$,
   (a) P_X and P_Y determine $equ(x_i, y_i)$ using the randomized prime protocol with $\varepsilon = (\log n)/n$.
   (b) If they conclude that $x_i \neq y_i$, they move to Step (a) and next i. If they conclude $x_i = y_i$, they exchange x_i and y_i and check whether they are actually equal. If so, they output 1 and halt. If not, they move back to Step 1.
2. Output 0 if all i are scanned and no $x_i = y_i$ are exchanged.

This protocol is zero-error because it checks directly whether $x_i = y_i$ on indices i, where the randomized prime protocol might have made a mistake. Fürer (1987) further improved the zero-error bound for this function to

$$\hat{C}_R(c.e., 0) \leq c_0 n,$$

for a constant c_0.

Aho, Ullman and Yannakakis (1983) demonstrated that at most a quadratic speedup is possible for zero-error randomized communication complexity relative to worst-case communication complexity; namely,

$$\hat{C}_R(f, 0)^2 \geq \hat{C}_D(f).$$

The second example above shows that this bound is essentially sharp.

Finally, we mention a recent result of Ahlswede and Dueck (1989) that any object among $N = 2^{2^{nR}}$ objects can be identified in blocklength n using a discrete memoryless channel of rate R with arbitrarily small error probability via an encoding procedure that uses randomization.

Orlitsky and El Gamal (1988) give a general survey of communication complexity, and Halstenberg and Reischuk (1990) state the best currently known results about relationships among communication complexity classes.

REFERENCES

Adleman, L. and Huang, M. D. (1987). Recognizing primes in random polynomial time. In *Proceedings of the 19th Annual Symposium on Theory of Computing* 462-469. ACM Press, New York.

Adleman, L., Pomerance, C. and Rumely, R. (1983). On distinguishing prime numbers from composite numbers. *Ann. of Math.* **117** 173-206.

Ahlswede R. and Dueck, G. (1989). Identification via channels. *IEEE Trans. Inform. Theory* **35** 15-29.

Aho, A., Ullman, J. D. and Yannakakis, M. (1983). On notions of information transfer in VLSI circuits. In *Proceedings of the 15th Annual Symposium on Theory of Computing* 133-139. ACM Press, New York.

Babai, L., Fortnow, L. Nisan, N. and Wigderson, A. (1991). BPP has subexponential-time simulations unless EXP has publishable proofs. In *Proceedings of the 6th Annual Structure in Complexity Theory Conference* 213-219. IEEE Computer Society Press, Los Alamitos, CA.

Boppana, R. and Hirschfeld, R. (1989). Pseudorandom generators and complexity classes. In *Randomness and Computation* (S. Micali, ed.) 1-26. Advances in Computing Research, Vol. 5. JAI Press, Greenwich, CT.

Dixon, J. (1981). Asymptotically fast factorization of integers. *Math. Comp.* **36** 255-260.

Fürer, M. (1987). The power of randomness for communication complexity. In *Proceedings of the 19th Annual Symposium on Theory of Computing* 178-181. ACM Press, New York.

Goldwasser, S. and Kilian, J. (1986). Almost all primes can be quickly certified. In *Proceedings of the 18th Annual Symposium on Theory of Computing* 316-329. ACM Press, New York.

Halstenberg, B. and Reischuk, R. (1990). Relationships between communication complexity classes. *J. Comput. System Sci.* **41** 402-429.

Johnson, D. S. (1990). A catalog of complexity classes. In *Handbook of Theoretical Computer Science* A (J. van Leeuwen, ed.) 67-161. North-Holland, Amsterdam.

Lagarias, J. C. Pseudorandom numbers. *Statist. Sci.* **8** 31-39.

Lenstra, A. K. and Lenstra, H. W. Jr. (1990). Algorithms in number theory. In *Handbook of Theoretical Computer Science* A (J. van Leeuwen, ed.) 673-715. North-Holland, Amsterdam.

Lenstra, A. K., Lenstra, H. W., Jr., Manasse, M. S. and Pollard, J. (1990). The number field sieve. In *Proceedings of the 22nd Annual Symposium on Theory of Computing* 564-572. ACM Press, New York.

Lenstra, H. W. and Pomerance, C. (1992). A rigorous time bound for factoring integers. *J. Amer. Math. Soc.* **5** 483-516.

Mehlhorn, K. and Schmidt, E. M. (1982). Las Vegas is better than determinism in VLSI and distributed computing (extended abstract). In *Proceedings of the 14th Annual Symposium on Theory of Computing* 330-337. ACM Press, New York.

Miller, G. (1976). Riemann's hypothesis and tests for primality. *J. Comput. System Sci.* **13** 300-317.

Morrison, M. and Brillhart, J. (1975). A method of factoring and the factorization of F_7. *Math. Comp.* **29** 183-205.

Orlitsky, A. and El Gamal, A. (1988). Communication complexity. In *Complexity in Information Theory* (Y. S. Abu-Mostafa, ed.) 16-61. Springer, New York.

Orlitsky, A. and El Gamal, A. (1990). Average and randomized communication complexity. *IEEE Trans. Inform. Theory* **36** 3-16.

Pollard, J. M. (1974). Theorems on factorization and primality

testing. *Proceedings of the Cambridge Philosophical Society* **76** 521–528.

POMERANCE, C. (1982). Analysis and comparison of some integer factoring algorithms. In *Computational Methods in Number Theory* (H. W. Lenstra, Jr. and R. Tijdeman, eds.) 89–139. *Math. Centre* Tract **144**. Math. Centrum, Amsterdam.

POMERANCE, C. (1987). Fast rigorous factorization and discrete logarithm algorithms. In *Discrete Algorithms and Complexity* (D. S. Johnson, T. Nishizeki, A. Nozaki and H. Wilf, eds.) 119–144. Academic, New York.

RABIN, M. O. (1980). Probabilistic algorithms for testing primality. *J. Number Theory* **12** 128–138.

SOLOVAY, R. and STRASSEN, V. (1977). A fast Monte-Carlo test for primality. *SIAM J. Comput.* **6** 84–85 (erratum **7** 118).

YAO, A. (1979). Some complexity questions related to distributed computing (extended abstract). In *Proceedings of the 11th Annual Symposium on Theory of Computing* 209–213. ACM Press, New York.

YAO, A. (1982). Theory and applications of trapdoor functions (extended abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science* 80–91. IEEE Computer Society Press, Los Alamitos, CA.

ZACHOS, S. (1988). Probabilistic quantifiers and games. *J. Comput. System Sci.* **36** 433–451.