REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
MARCH 2023	TECHNICAL PAPER	JANUARY 1996	APRIL 1997

4. TITLE AND SUBTITLE

Design and Analysis of Parallel Search Strategies to Find a First Solution

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
IN-HOUSE	N/A	0602702F
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
2338	01	7B

6. AUTHOR(S)

Warren H. Debany Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8.PERFORMING ORGANIZATION REPORT NUMBER
Air Force Research Laboratory/RIG 525 Brooks Road Rome NY 13441-4505	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10.SPONSOR/MONITOR'S ACRONYM(S)	11.SPONSOR/MONITOR'SREPORT NUMBER(S)
Air Force Research Laboratory/RIG 525 Brooks Road Rome NY 13441-4505	AFRL/RI	AFRL-RI-RS-TP-2023-002

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA Case Number: RL/PA #97-209
DATE CLEARED: 10 April 1997

13. **SUPPLEMENTARY NOTES**

Published in the proceedings of the 1997 Summer Computer Simulation Conference (SCSC), Washington D.C. 13-17 July 1997.
This technical paper was written under the in-house work unit "Testability Measurement and Design-For-Testability" at the Rome Laboratory, now Air Force Research Laboratory.

14. ABSTRACT

Many problems involve the exploration of a large state space to find a solution - examples of such problems include automatic test vector generation for digital logic circuits, the traveling salesman problem, associative search, and cryptography. In most cases it is sufficient to stop the search after the first acceptable solution is found. This paper discusses some new results concerning the negative hypergeometric distribution and applies this distribution to the analysis of "sampling without replacement" search strategies. To shorten long run times, a search can be parallelized by assigning partitions of the search space to different processors. It is shown in this paper that linear average speedup of a search for a first solution is impossible even in the absence of such considerations as communication, contention, or load balancing, and even under the most favorable partitioning of solutions in the search space.

15. SUBJECT TERMS

Parallel processing, speedup, sampling with replacement, sampling without replacement, negative hypergeometric distribution

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	7

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
WARREN H. DEBANY, JR.	**N/A**

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

DESIGN AND ANALYSIS OF PARALLEL RANDOM SEARCH STRATEGIES TO FIND A FIRST SOLUTION

Warren H. Debany Jr., Ph.D., P.E.
Rome Laboratory/ERDA
525 Brooks Rd.
Rome NY 13441-4505

e-mail: debanyw@rl.af.mil

KEY WORDS

Parallel processing, speedup, sampling with replacement, sampling without replacement, negative hypergeometric distribution.

ABSTRACT

Many problems involve the exploration of a large state space to find a solution — examples of such problems include automatic test vector generation for digital logic circuits, the traveling salesman problem, associative search, and cryptography. In most cases it is sufficient to stop the search after the first acceptable solution is found. This paper discusses some new results concerning the negative hypergeometric distribution, and applies this distribution to the analysis of "sampling without replacement" search strategies. To shorten long run times, a search can be parallelized by assigning partitions of the search space to different processors. It is shown in this paper that linear average speedup of a search for a first solution is impossible even in the absence of such considerations as communication, contention, or load balancing, and even under the most favorable partitioning of solutions in the search space.

INTRODUCTION

A commonly performed operation is that of searching a *state space (SS)* for a *solution*. States are selected and *tested* until an acceptable solution is found. Examples of such problems include automatic test vector generation (ATVG) for digital logic circuits, the traveling salesman problem (TSP), associative search, and cryptography.

This work was supported by the Rome Laboratory Chief Scientist as a 6.1 Basic Research Entrepreneurial Research Project (ERP).

Depending on the type of problem, the SS may contain exactly one solution, many solutions (although the exact number may be unknown), or no solutions. In most practical cases it is not necessary to find *all* solutions that may exist in the SS. Instead, the search is terminated when the *first* solution is found.

Random Search

For many problems, algorithms or heuristics for guiding the search process may not be available, or may be so weak that the SS effectively is explored at random. This may occur by the fundamental nature of the problem or as a consequence of the particular data set. For example, in an instance of TSP the cost matrix may be such that a depth-first search guided by branch-and-bound must explore all or most branches of the search tree. Systematic search techniques are nonexistent for certain *key search* problems, where the most efficient strategy is simply to search the SS at random (Diffie and Hellman 1977).

ATVG algorithms based on PODEM (Goel 1981) use *implicit enumeration* of values at the primary inputs of a circuit-under-test to find required conditions for fault activation and error propagation. The fraction of time PODEM spends in the enumeration phase varies with the effectiveness of the heuristics used and the specific circuit-under-test, but one study has observed this fraction to be about 95% of the total test vector generation time (Wu and Hartmann 1997).

Parallel Processing

A *uniprocessing* (or *scalar*) implementation of a search technique uses a single computing node to solve a problem. Because of the long runtimes for some SS search problems, it is natural to consider *parallel processing (PP)* as a means of shortening the time to

find a solution. Special-purpose architectures exist to support efficient PP, such as hypercubes and the Intel Paragon. More commonly, however, researchers turn to the low-cost alternative of a network of workstations, where resources are accessed as needed through a package such as Parallel Virtual Machine (PVM) (PHENIX 1994).

There exists a large body of literature concerning parallel implementations of SS search problems, for example (Kalé and Saletore 1990) and (Janakiram, Agrawal, and Mehrotra 1988). Many deal specifically with parallelization of the ATVG problem, such as (Chandra and Patel 1988), (Arvindam, *et al.* 1991), and (Patil and Banerjee 1990).

A goal of any PP implementation is to achieve *linear speedup*. This is where a search problem runs k times faster on k processors than it does on a single processor. It is well known that many PP barriers frustrate attempts to achieve linear speedup, including:

- interprocess communication
- process creation
- memory management
- contention for shared resources
- load balancing
- additional computation

See, for example, (JáJá 1992) and (Lewis and El-Rewini 1992).

Most authors state only that sublinear speedup is *caused* by these PP barriers, but the impression is clear that, if they could be eliminated, then linear speedup would result. In fact, in (Lewis and El-Rewini 1992) the statement is made concerning hypercube architectures that, "A trivially parallel application is one that requires no interprocess communication. Such an application achieves linear speedup."

When a fixed amount of computation must be performed, as in many numerical applications such as signal processing, it may be true that elimination of these PP barriers may result in linear speedup. If only a first solution is required, however, then the remainder of the SS and any other solutions that may exist can be disregarded. It is demonstrated here that *linear average speedup* of a random SS search to find a first solution is not possible, even in the absence of these PP barriers, and with the most fortunate partitioning of solutions in the SS.

WAITING TIMES: SAMPLING WITH AND WITHOUT REPLACEMENT

Consider a SS that contains N distinct states, where exactly M of these states are solution states. In this paper it is assumed that $M \geq 1$, $N \gg M$, and state selection is random and equiprobable.

In a *sampling with replacement (SWR)* strategy, a state may be selected again even if it has already been tested. A *sampling without replacement (SWOR)* strategy selects only distinct states.

The *geometric distribution*, which is derived from the *negative binomial* (Feller 1957), is the probability mass function (pmf) for the waiting time to find a first solution using SWR:

$$\begin{aligned}&\Pr\{\text{first solution is found on step } n, \text{ for SWR}\}\\ &= pq^{n-1}\end{aligned} \tag{1}$$

where $p = M/N$ and $q = 1 - p$. Define a random variable (rv) G that is the number of the SWR step on which the first solution is found. The expected value of G is

$$EG = \frac{1}{p} = \frac{N}{M} \tag{2}$$

The probability distributions that apply to SWOR-based search SWOR are far less well known than those for the case of SWR. A distribution that corresponds to the negative binomial, but for SWOR, is known as the *negative hypergeometric*, *inverse hypergeometric*, or *hypergeometric waiting-time*. (Matuszewski 1962), (Guenther 1983), and (Johnson, Kotz, and Kemp 1992). There are many alternative forms for the negative hypergeometric distribution — the simplest form is given by (Matuszewski 1962):

$$\begin{aligned}&\Pr\{r^{th} \text{ solution is found on step } n, \text{ for SWOR}\}\\ &= \frac{\binom{n-1}{r-1}\binom{N-n}{M-r}}{\binom{N}{M}} \quad \text{for } r \leq n \leq N - M + r\end{aligned} \tag{3}$$

Setting $r = 1$ in Eqn (3) yields a pmf that corresponds to the geometric distribution, but for SWOR:

$$\begin{aligned}&\Pr\{\text{first solution is found on step } n, \text{ for SWOR}\}\\ &= \frac{\binom{N-n}{M-1}}{\binom{N}{M}} \text{ for } 1 \leq n \leq N - M + 1\\ &= h(n, N, M)\end{aligned} \tag{4}$$

(Being a pmf, $h(n, N, M) = 0$ for $n < 1$ or $n > N - M + 1$.)

We will refer to the pmf given by Eqn (4) as the *Negative Hypergeometric with $r = 1$ (NH1)* distribution. Note that for $M = 1$ the NH1 pmf degenerates to a *uniform distribution* (Feller 1957) and (Janakiram, Agrawal, and Mehrotra 1988).

The following two equations relating to the NH1 distribution are believed to be new results. Eqn (4) can be written in recursive form as

$$h(n,N,M) = \begin{cases} \frac{N-M-n+2}{N-n+1} \times h(n-1,N,M) \\ \qquad \text{for } 2 \le n \le N-M+1 \\ \frac{M}{N} \quad \text{for } n = 1 \end{cases} \tag{5}$$

Eqn (5) provides a great savings in computation and programming convenience in our analyses.

The cumulative distribution function (cdf) for the NH1 distribution is given by

$$H(n,N,M) = 1 - \frac{\binom{N-n}{M}}{\binom{N}{M}} \text{ for } 1 \le n \le N-M+1 \tag{6}$$

(Being a cdf, $H(n,N,M) = 0$ for $n < 1$, and is 1 for $n \ge N-M+1$.)

Define a rv H that is the number of the SWOR step on which the first solution is found. The expected value of H is

$$EH = \frac{N+1}{M+1}. \tag{7}$$

(This result is well known for $M = 1$.) The variance of H is

$$VH = \frac{M(N-M)(N+1)}{(M+1)^2(M+2)} \tag{8}$$

Eqns (7) and (8) are special cases of the moments of the negative hypergeometric distribution as given in (Matuszewski 1962).

Comparison of SWR and SWOR

It is obvious that, on the average, a SS search strategy that uses SWOR will find a first solution faster than a SWR-based search. However, the relevant literature seems to deal exclusively with analyses based on SWR. This may be justified in terms of simplicity of presentation as in (Janakiram, Agrawal, and Mehrotra 1988), or for fault tolerance as in (Quisquater and Desmedt 1991).

It is noted in the latter reference that a SWR-based search takes an average of only twice as long as a SWOR-based search. Let us quantify this ratio more precisely. For given values of N and M, the ratio of EG to EH is obtained from Eqns (2) and (3):

$$\text{Average search speedup of SWOR over SWR} = \frac{EG}{EH} = \frac{N}{M} \times \frac{M+1}{N+1} = \frac{NM+N}{NM+M} \tag{9}$$

For $M \ll N$ the speedup using SWOR is approximately $\frac{M+1}{M}$. For $M = 1$ the speedup approaches 2 as noted elsewhere. A speedup of at least 10% is achieved for $M \le 10$.

Design of SS Search Strategies

Clearly, if solutions are plentiful, then it does not matter whether SWOR or SWR is used. An additional design consideration is the overhead needed for a SWOR search. In some cases memory requirements may be excessive, or the computation required to sample only distinct states may be even greater than that required to actually test the states. It is assumed in this paper, however, that M is relatively small, and that the SS is explored randomly using SWOR.

SPEEDUP USING PARALLEL PROCESSING

Again, consider a SS that consists of N states and contains M solutions. Assume that k processors in parallel are to be used to find a first solution. The SS is partitioned *at random* into k non-overlapping sets, each of which is explored *at random* by SWOR by one processor. We consider only cases where k evenly divides N, so each partition contains exactly N/k states. Each processor explores its partition of the SS at the same, constant rate. Significantly, we disregard *all* of the PP barriers mentioned in the introduction, and show that this optimistic scenario fails to achieve linear average speedup even under the *most favorable* distribution of solutions.

Following standard conventions in the PP literature, such as (JáJá 1992), the comparison metric is the mean time to find a first solution. Define the *average speedup* achieved by a search that uses k processors as

$$S(k,N,M) = \frac{\text{Average time using 1 processor}}{\text{Average time using } k \text{ processors}} \tag{10}$$

where linear speedup would have $S(k,N,M) = k$. Define the *average efficiency* as

$$E(k,N,M) = \frac{S(k,N,M)}{k} \tag{11}$$

where $E(k,N,M) = 1$ for linear speedup.

We consider three cases:

- **Case 1:** All M solutions are in a single partition out of k
- **Case 2:** All possible distributions of M solutions to k partitions
- **Case 3:** Random distribution of M solutions to k partitions

Case 1

The most fortunate (albeit unlikely) case for PP speedup is where all M of the solutions in the SS are

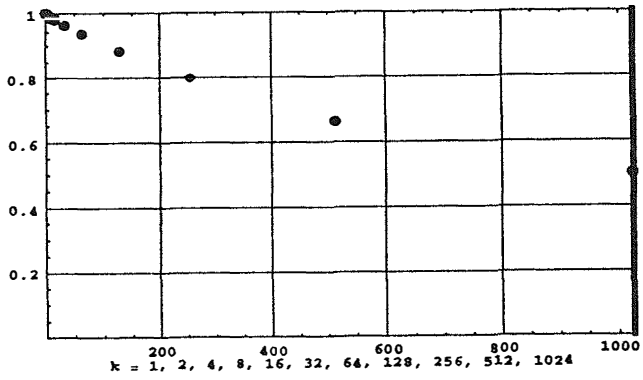

Figure 1. Efficiency for $N = 1{,}024$ and $M = 1$.

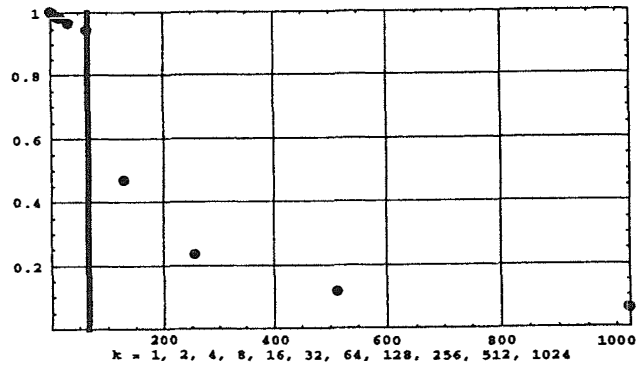

Figure 2. Efficiency when all solutions are in single partition (Case 1) for $N = 1{,}024$ and $M = 16$.

contained in the same partition, perhaps even to the point of saturation. The analysis is based on Eqn (7), which is the mean of the NH1 distribution.

Consider when $N/k \geq M$. From Eqn (10) we have

$$S(k,N,M) = \frac{\frac{N+1}{M+1}}{\frac{N/k+1}{M+1}} = \frac{kN+k}{N+k}$$

Note that this speedup does not depend on the value of M. From Eqn (11) we have

$$E(k,N,M) = \frac{N+1}{N+k} \qquad (12)$$

$E(k,N,M)$ is nearly 1 for small k, but, for any $k > 1$, the average efficiency is *strictly less* than 1.

An important special case is when $M = 1$. The behavior of Eqn (12) is shown in Figure 1 for $N = 1{,}024$. Note that, in the extreme case of massive parallelism where $N = k$ (i.e., every processor needs to test only a single state), we find that $E(N,N,1)$ approaches 1/2!

Now, consider when $N/k \leq M$. A single partition is saturated with solutions, so the PP search always terminates in one step. Then

$$S(k,N,M) = \frac{\frac{N+1}{M+1}}{1} = \frac{N+1}{M+1}$$

which is constant with respect to k. At the point where $N/k = M$ (a single partition contains all M solutions), from Eqn (12) we have

$$S(\tfrac{N}{M},N,M) = \frac{N+1}{N+N/M} \approx \frac{M}{M+1}$$

Then, as k increases, the efficiency continues to drop below $\frac{M}{M+1}$ because the speedup is constant, yet more processors are being used.

Figure 2 shows how the efficiency function behaves for $N = 1{,}024$ and $M = 16$, with the vertical bar at the saturation point $k = 64$, where $N/k = M$.

Thus, it is seen that, even when *all* M solutions end up in a single partition, the average efficiency is *still* always less than 1 for $k > 1$.

Case 2

The first case represented the best situation for PP speedup. We now generalize the discussion to determine how the efficiency function behaves for all possible distributions of M solutions to k partitions.

Consider the case where $k = 4$ and $M = 6$. Three possible distributions of solutions are $\{2,1,0,3\}$, $\{1,3,0,2\}$, and $\{3,2,0,1\}$; we group these cases as a single canonicalized *partition set* $\{3,2,1,0\}$ and report an overall occurrence frequency for it.

For a value of N, we wish to determine the average speedup and efficiency for each possible partition set for given values of M and k. Denote a partition set as $P = \{M_1, M_2, ...M_k\}$ where $M_1 + M_2 + ... + M_k = M$. We find the expected number of steps for *any one* of the processors to find a first solution by using the individual processor's NH1 cdfs to calculate an overall cdf, converting that cdf to an overall pmf, and summing terms to find the expected value. (Conceptually the individual cdfs are given by Eqn (6), but computationally it is more convenient to use the recursive form given in Eqn (5).) For a given solution partition P we obtain the following overall pmf:

$$\begin{aligned}
&\Pr\{\text{any processor finds a first solution on step } n\} \\
&= \prod_{i=1}^{k}\left[1 - H(n-1, \frac{N}{k}, M_i)\right] \\
&\qquad - \prod_{i=1}^{k}\left[1 - H(n, \frac{N}{k}, M_i)\right] \\
&= h'(n,N,P)
\end{aligned}$$

Then the expected number of steps for *any* processor

Table 1. All possible partitions of solutions for $N = 500$, $M = 5$, and $k = 5$.

Partition Set	Freq.	Speedup	Efficiency
$\{2,1,1,1,0\}$	0.384	4.872	0.974
$\{2,2,1,0,0\}$	0.288	4.882	0.976
$\{3,1,1,0,0\}$	0.192	4.892	0.978
$\{3,2,0,0,0\}$	0.064	4.901	0.980
$\{1,1,1,1,1\}$	0.038	4.863	0.973
$\{4,1,0,0,0\}$	0.032	4.921	0.984
$\{5,0,0,0,0\}$	0.002	4.960	0.992

to find a first solution is given by

$$EH_P = \sum_{n=1}^{N/k} nh'(n, N, P) \qquad (13)$$

Analytically, Eqn (13) is messy unless all $M_i = 1$, so the series is summed directly.

Table 1 shows the results for $N = 500$, $M = 5$, and $k = 5$, where there are seven possible canonical partitions of five solutions to five processors. The table shows the partition sets, frequencies (assuming uniformly random distribution of solutions), and the expected speedup and efficiency based on the EH_P given by Eqn (13) compared to the EH of the unpartitioned case given by Eqn (7).

The greatest efficiency is observed in the least likely case, i.e., when all solutions are in a single partition (our Case 1). The smallest efficiency is observed when the solutions are evenly partitioned among the k processors. The average efficiency could be obtained analytically by weighting the individual efficiencies by their probabilities of occurrence, but in the next subsection we give empirical results based on simulation.

Case 3

Now we find the efficiency that is expected when M solutions are randomly distributed among k processors. These data were obtained from simulations that model the random search process. (The value $N = 180$ was chosen for this example because it is evenly divisible by many values of k.) Figure 3 shows how the efficiency decreases as a function of k.

This case is the most realistic as a predictor of the efficiency of a single instance of a SS search. It should be noted that results vary widely from experiment to experiment. For the unpartitioned search in this example, with a mean value of 30.2 steps, the variance of the NH1 distribution is given by Eqn (8) as 628.5; thus, the standard deviation of the number of steps to find the first solution is approximately 83% of the mean. The PP results vary similarly; for $k = 5$ the observed standard deviation was 78% of the mean. Thus, wide variability in actual speedup is guaranteed.

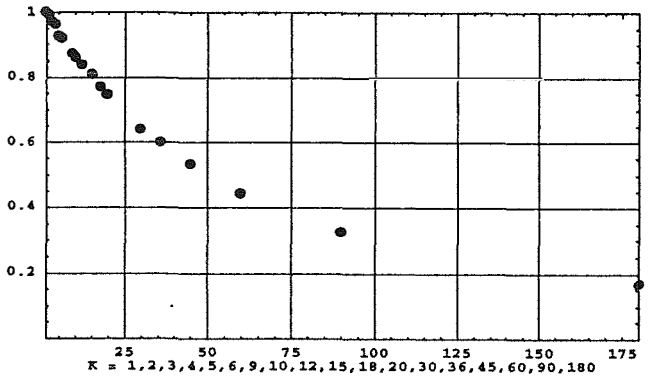

Figure 3. Efficiency for random distributions of solutions to k partitions (Case 3) for $N = 180$ and $M = 5$, based on 10,000 experiments.

Design of SS Search Strategy

These results indicate that a parallel SS search has near-linear speedup only when k is small compared to N. If the SS is very large, as in a random test of 56-bit keys (Diffie and Hellman, 1977), then efficiency will be close to 1 regardless of the number of processors used. For problems with smaller state spaces, however, massively parallel implementations consisting of hundreds or thousands of processors may have efficiencies much less than 1.

In the latter case, the algorithm designer must decide whether decreasing *latency* or increasing *throughput* is more critical. For a single instance of a search problem, PP will find a solution (on the average) faster than a uniprocessor will. However, when many individual search problems are to be solved, better throughput would result from partitioning complete searches to the available processors, rather than partitioning the search spaces. In ATVG, for example, it has been noted that *fault list* parallelism provides better speedup than other forms of parallelism; see, for example, (Chandra and Patel 1988).

CONCLUSIONS

This paper has presented a framework for the design and analysis of random state space search strategies. In contrast to the well-known theory of sampling with replacement, the corresponding probability distributions that apply to sampling without replace-

ment are relatively unknown to algorithm designers. This paper discusses properties of the negative hypergeometric distribution, which is suitable for analyzing the performance of searches based on sampling without replacement.

We have established that sampling without replacement is clearly superior to sampling with replacement when there are relatively few solution states. If there are more than about 10 solutions in a large state space, then the improvement in performance will be less than 10% by using sampling without replacement.

For a sampling without replacement strategy, we have derived average efficiencies gained by using k parallel processors for several scenarios for the distribution of solutions. Even in the most optimistic case, where all solutions are in the search space searched by a single processor, it is shown that the average efficiency is always less than 1. For a problem with a single solution the average efficiency of a massively parallel implementation may be as low as 1/2.

It is demonstrated here that, even in the absence of the well-known parallel processing barriers, *linear average speedup* of a random search space search to find a first solution is not possible. This is shown to be a consequence of the laws of probability, rather than a deficiency in the implementation. This result is not necessarily negative — it means that many of the published results that achieved at best slightly sublinear speedup may, in fact, have achieved the theoretical maximum limit.

ACKNOWLEDGMENT

The author wishes to thank Prof. Kishan G. Mehrotra (Syracuse University) for his help in providing information about the negative hypergeometric distribution function.

REFERENCES

Arvindam, S.; V. Kumar; V.N. Rao; and V. Singh. 1991. "Automatic Test Pattern Generation on Parallel Processors." *Parallel Computing* (Elsevier Science Pub.), vol. 17: 1323-1342.

Chandra, S. and J.H. Patel. 1988. "Test Generation in a Parallel Processing Environment." In *Proc. IEEE Int. Conf. on Computer Design (ICCD)*: 11-14.

Diffie, W. and M.E. Hellman. 1977. "Exhaustive Cryptoanalysis of the NBS Encryption Standard." *IEEE Computer*. (Jun.): 74-84.

Feller, W. 1957. *An Introduction to Probability Theory and Its Applications (Vol. 1)*. J. Wiley & Sons, NY.

Goel, P. 1981. "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits." *IEEE Trans. on Computers* (Mar.): 215-222.

Guenther, W.C. 1983. "Hypergeometric Distributions." In *Encyclopedia of Statistical Sciences, Vol 3* (Editors: S. Kotz and N.L. Johnson). J. Wiley & Sons, NY.

JáJá, J. 1992. *An Introduction to Parallel Algorithms.* Addison-Wesley, Reading, MA.

Janakiram, V.K.; D.P. Agrawal; and R. Mehrotra. 1988. "A Randomized Parallel Backtracking Algorithm." *IEEE Trans. on Computers* (Dec.): 1665-1676.

Johnson, N.L.; S. Kotz; and A.W. Kemp. 1992. *Univariate Discrete Distributions.* J. Wiley & Sons, NY.

Kalé, L.V. and V.A. Saletore. 1990. "Parallel State-Space Search for a First Solution with Consistent Linear Speedups." *Int. J. of Parallel Programming* (Plenum Pub. Corp.), vol. 19, no. 4: 251-293.

Lewis, T.G. and H. El-Rewini. 1992. *Introduction to Parallel Computing*. Prentice Hall, Englewood Cliffs, NJ.

Matuszewski, T.I. 1962. "Some Properties of Pascal Distribution for Finite Populations." *J. of the Amer. Statistical Assoc.*, vol. 57: 172-174. (See also correction, p. 919.)

Patil, S. and P. Banerjee. 1990. "A Parallel Branch and Bound Algorithm for Test Generation." In *Proc. ACM/IEEE Design Automation Conf. (DAC)*: 339-344.

PHENIX. 1994. "Interprocess Communication Using PVM." http://uther1.phy.ornl.gov/offline/code_develop/pvm/pvm.html.

Quisquater, J-J. and Y.G. Desmedt. "Chinese Lotto as an Exhaustive Code-Breaking Machine." *IEEE Computer* (Nov.): 14-22.

Wu, C-H. and C.R.P. Hartmann. 1997. "Delay Fault and Stuck-At Fault Test Generation Using Multiprocessing." Rome Laboratory Tech. Report (in publication).