REFERENCES

[1] K. E. Batcher, "Sorting networks and their applications," in *Proc. AFIPS Spring Joint Comput. Conf.*, vol. 32, pp. 307–314, 1968.
[2] J. W. Cooley and J. W. Tuckey, "An algorithm for the machine computation of complex Fourier series," *Math. Computation*, vol. 19, pp. 297–301, 1965.
[3] C. M. Fiduccia, "Fast matrix multiplication," in *Proc. 3rd ACM Annu. Symp. Theory of Computing*, pp. 45–49, 1971.
[4] D. E. Knuth, *Fundamental Algorithms—The Art of Computer Programming*, vol. 1. Reading, MA: Addison-Wesley, 1968, pp. 399–405.
[5] E. A. Lamagna, "The complexity of monotone functions," Ph.D. dissertation, Brown Univ., Providence, RI, 1975.
[6] E. A. Lamagna and J. E. Savage, "Combinational complexity of some monotone functions," in *Proc. 15th Annu. IEEE Symp. on Switching and Automata Theory*, pp. 140–144, 1974.
[7] O. B. Lupanov, "A method of circuit synthesis," *Izvestia v.u.z. Radiofizika*, vol. 1, pp. 120–140, 1958.
[8] K. Mehlhorn and A. Galil, "Monotone switching circuits and Boolean matrix product," *Computing*, vol. 16, pp. 99–111, 1976.
[9] D. E. Muller, "Complexity in electronic switching circuits," *IRE Trans. Electron. Comput.*, vol. EC-5, pp. 15–19, 1956.
[10] D. E. Muller and F. P. Preparata, "Bounds to complexities of networks for sorting and switching," *J. Assoc. Comput. Mach.*, vol. 22, pp. 195–201, 1975.
[11] E. I. Neciporuk, "On a Boolean matrix," *Syst. Theory Res.*, vol. 21, pp. 236–239, 1971.
[12] M. S. Paterson, "Complexity of monotone networks for Boolean matrix product," *Theor. Comput. Sci.*, vol. 1, pp. 13–20, 1975.
[13] N. Pippenger and L. G. Valiant, "Shifting graphs and their applications," *J. Assoc. Comput. Mach.*, vol. 23, pp. 423–432, 1976.
[14] V. R. Pratt, "The power of negative thinking in multiplying Boolean matrices," *SIAM J. Comput.*, vol. 4, pp. 326–329, 1975.
[15] W. V. Quine, "Two theorems about truth functions," *Bol. Soc. Math. Mexicana*, vol. 10, pp. 64–70, 1953.
[16] J. E. Savage, *The Complexity of Computing*. New York: Wiley-Interscience, 1976.
[17] A. Schönhage and V. Strassen, "Fast multiplication of large numbers," *Computing*, vol. 7, pp. 281–292, 1971.
[18] C. E. Shannon, "The synthesis of two terminal switching circuits," *Bell Syst. Tech. J.*, vol. 28, pp. 59–98, 1949.
[19] D. C. Van Voorhis, "An improved lower bound for sorting networks," *IEEE Trans. Comput.*, vol. C-21, pp. 612–613, 1972.
[20] A. C.-C. Yao and F. F. Yao, "Lower bounds on merging networks," *J. Assoc. Comput. Mach.*, vol. 23, pp. 566–571, 1976.

Edmund A. Lamagna (S'69–M'76) received the combined A.B.-Sc.B. degrees in 1970, the Sc.M. degree in electrical engineering in 1971, and the Ph.D. degree in applied mathematics in 1975, all from Brown University, Providence, RI.

He is currently Assistant Professor of Computer Science at the University of Rhode Island, Kingston, where he has served on the faculty since 1976. From 1975 to 1976, he was a member of the Technical Staff at the Mitre Corporation, Bedford, MA. His present research interests are in the areas of computational complexity and analysis of algorithms, theory of programming, and microcomputer systems.

Dr. Lamagna is a member of the Association for Computing Machinery, the American Mathematical Society, Sigma Xi, and Tau Beta Pi.

A Note on Open Shop Preemptive Schedules

TEOFILO GONZALEZ

***Abstract*—The problem of preemptively scheduling a set of n independent jobs on an m processor open shop is discussed. An algorithm to construct preemptive schedules with minimum-maximum finishing time is presented. The worst case time complexity is $0(r + \min\{m^4, n^4, r^2\})$, where r is the number of nonzero tasks. The maximum number of preemptions introduced is $0(\min\{rn, rm, n^3, m^3\})$. The algorithm is best possible for open shops with a fixed number of processors or jobs. In this case the maximum number of preemptions introduced is bounded by some fixed constant.**

***Index Terms*—Finish time, open shops, polynomial complexity, preemptive scheduling.**

Manuscript received October 10, 1978; revised April 11, 1979. This work was supported in part by NSF Grant MCS77-21092.

The author was with the Department of Computer Science, Pennsylvania State University, University Park, PA 16802. He is now with the Department of Computer Science, Institute of Technology, Monterrey, Mexico.

I. INTRODUCTION

AN OPEN shop consists of $m \geq 1$ processors (or machines). Each of these performs a different operation. There are $n \geq 1$ independent jobs. Each job consists of m tasks. The processing time for task j of job i is $t_{j,i} \geq 0$. The jth task of all jobs is to be executed by processor j. Let r be the total number of nonzero tasks.

Let A_k represent an *assignment* of tasks to machines. Each assignment A_k is required to satisfy the following restrictions:

1) at most one jth task is to be assigned to machine j, and
2) for each job, at most one task is to be assigned.

The processing of tasks given by assignment A_k is initiated at time l_{k-1} and terminates at time l_k. l_0 is zero. A *schedule* S

0018-9340/79/1000-0782$00.75 © 1979 IEEE

for an open shop is a sequence of s 2-tuples of the form (A_k, l_k), where A_k is an assignment and l_k is an instance of time. In addition, each nonzero task (j, i) must be assigned for a total of $t_{j,i}$ units of time in S. Note that this definition is not identical to the one given in [1]; however, both can be shown to be equivalent.

The *finish time* (f_i) *for job* i is the latest time job i is scheduled. The *finish time for schedule* S $(ft(S))$ is max $\{f_i\}$. An *optimal finishing time* (OFT) schedule is one with the least finishing time amongst all schedules. A *preemptive schedule* is one in which tasks are not required to execute continuously.

Open shop problems differ from flow shop and job shop problems (see [2], for example) in that there is no restriction placed on the order in which tasks from each job are to be processed. The problem of obtaining OFT preemptive schedules for open shops arises as a subproblem of some preemptive scheduling problems [5]. Open shops also model the following situation (as well as others). Consider the problem of scheduling meetings between professors and students. Let $t_{j,i}$ be the total time student i must meet with professor j. An OFT preemptive schedule for this problem will minimize the latest time at which there is a meeting. Of course, one would like to minimize the number of preemptions so as to avoid interruptions. The problem of constructing an OFT preemptive schedule for $m > 2$ with the minimum number of preemptions is NP-hard [1]. On the other hand, if $p(n)$ preemptions are introduced by an algorithm, then the time complexity for the algorithm is at least $\Omega(p(n))$, when the output is a schedule of the form described above. Therefore, we restrict ourselves to the problem of constructing OFT preemptive schedules with a small (not the minimum) number of preemptions.

In this paper we present an algorithm to obtain OFT preemptive schedules for open shops. Our algorithm is based on a combine-and-conquer approach. This should be contrasted with divide and conquer. The latter approach divides a problem into smaller size subproblems. The solution to these subproblems are combined to obtain the solution to the original problem. The former approach combines inputs (in this case jobs and machines) for the original problem to obtain a problem of smaller size. The solution to this problem is then divided to obtain the solution to the original problem. Note that it is *not* recursive. A similar approach was used in [5] for a restricted open shop problem.

The worst case time complexity for our algorithm is $0(r + \min\{m^4, n^4, r^2\})$. The maximum number of preemptions introduced is $0(\min\{rn, rm, n^3, m^3\})$. Previous algorithms for this problem have worst case time complexity $0(r^2)$ [1, pp. 670–673], $0(r \min\{r, m^2\} + rm \log n)$ [1, pp. 673–675], and $0(n)$ for $m = 2$ [1, pp. 666–669]. We denote these algorithms by R, P, and U, respectively. The maximum number of preemptions introduced by these algorithms is $0(rm)$, $0(rm)$, and 0. For $n = 2$ or $m = 2$, algorithm U is best possible as it introduces no preemptions and takes time $0(r)$. For n fixed,[1] algorithms R and P take time $0(m^2)$. Our algorithm takes time $0(m)$. For m fixed, algorithm R takes time $0(n^2)$ and algorithm P takes time $0(n \log n)$. Algorithm P introduces $0(n)$ preemptions. Our algorithm is of time complexity $0(n)$ and introduces only a fixed number of preemptions. Clearly, when there is a fixed number of jobs or machines our algorithm is best possible (linear time). In addition, the worst case time complexity for our algorithm as well as the maximum number of preemptions introduced are within a multiplicative constant of the ones for previous algorithms (R and P). The maximum number of preemptions introduced by our algorithm is $0(rm)$, which is of the same order as the ones for algorithms P and R. The worst case time complexity for our algorithm is $0(r^2)$, which compares to the one for algorithm R. For $r \leq m^2$, algorithm P and our algorithm are $0(r^2)$. When $r > m^2$, P is $0(rm^2)$ and our algorithm is $0(m^4)$.

[1] n is not a parameter but m is a parameter.

We use the following NP-hard problem [3] as a subproblem in our algorithm.

Bin Packing: Given a sequence of "bins" $B_1, B_2, \cdots$ and a list $L = (a_1, a_2, \cdots, a_n)$ of positive integers in the range $(0; \beta]$, find an assignment of pieces a_i into the bins so that no bin has contents totaling more than β, and the number of nonempty bins is minimized.

In our algorithm we make use of the following packing rule. This rule produces suboptimal packings for the bin packing problem.

Rule: Initially, the first piece is placed in bin B_1. j is set to 1. In step i, the ith piece is added to bin B_j if it does not exceed the capacity of the bin. Otherwise the ith piece is placed in bin B_{j+1}; bins B_j and B_{j+1} are interchanged if the sum of the lengths of the pieces in bin B_j exceed the length of the ith piece; j is incremented by 1. □

It is simple to show that the time complexity of an algorithm which uses this rule is $0(n)$, where n is the number of pieces. Clearly, all bins used except possibly for the last are at least half full.

II. OFT Preemptive Schedules

In this section we present a combine-and-conquer algorithm to construct OFT preemptive schedules for open shops. The algorithm is simple, so only an informal description will be given.

Let E represent an open shop problem. We now construct an OFT preemptive schedule S for E.

Our algorithm is divided into three phases.

Phase 1: Initially, combine jobs and machines from open shop E. This transforms E to open shop problem T, with a smaller number of machines and jobs. T is to be constructed in such a way that if given any OFT preemptive schedule S' for T, then:

a) $ft(S') = ft(S)$ where S is an OFT preemptive schedule for E;

b) S can be constructed from S'.

Phase 2: Algorithm R [1] is used to construct S' for T.

Phase 3: S is generated from S'. Assignments in S' are divided to the original tasks and machines from E. □

The first phase involves the solution of two bin packing problems. The bin packing problem is a hard combinatorial problem (NP-complete). There is no known efficient algorithm to obtain the minimum number of bins for this problem. However, there are fast algorithms which guarantee solutions to be within a fixed percentage from the optimal solution value [3], [4].

Given a set of n jobs with task times $t_{j,i}$, $1 \le i \le n$ and $1 \le j \le m$ for an m processor open shop E, we define the following quantities.

$T_j = \Sigma_{1 \le i \le n} t_{j,i} \cdots$ total time processor j is active, $1 \le j \le m$.
$L_i = \Sigma_{1 \le j \le m} t_{j,i} \cdots$ total length of job, i, $1 \le i \le n$.
$W = \Sigma_{1 \le j \le m} T_j = \Sigma_{1 \le i \le n} L_i$.
$r =$ number of nonzero tasks (assume $r \ge n$ and $r \ge m$ as otherwise some machines or jobs could be eliminated).
$\alpha = \max_{i,j}\{T_j, L_i\}$.

In [1] it is shown that the finish time of an OFT preemptive schedule for open shop E is α.

We now outline the algorithm which will be referred to as P'.

Algorithm P'

Phase 1: Open shop E is transformed to T. Jobs $1, 2, \cdots, n$ with execution times $L_1, L_2, \cdots, L_n$ are packed in bins of size α. Processors are packed in a different set of bins of size α. The packing is carried out by the rule given in Section I. Let n' and m' be the number of bins required to pack jobs and machines, respectively. Let J'_i, $1 \le i \le n'$, be sets whose elements are the indices of jobs packed in bin i. Let P'_j, $1 \le j \le m'$, be sets whose elements are the indices of processors packed in bin j. Then

$$\bigcup_{1 \le i \le n'} J'_i = \{1, 2, \cdots, n\}, \quad \bigcup_{1 \le j \le m'} P'_j = \{1, 2, \cdots, m\},$$

$$\sum_{i \in J'_j} L_i \le \alpha \quad \text{for} \quad 1 \le j \le n',$$

and

$$\sum_{j \in P'_i} T_j \le \alpha \quad \text{for} \quad 1 \le i \le m'.$$

Open shop T consists of n' jobs (J'_i) and m' processors (P'_j). The execution time for each task (j, i) is

$$t'_{j,i} = \sum_{j' \in P'_j} \sum_{i' \in J'_i} t_{j',i'} \quad \text{for} \quad 1 \le j \le m'$$

and

$$1 \le i \le n'.$$

Phase 2: Algorithm R [1] constructs an OFT preemptive schedule S' for open shop T.

Phase 3: An OFT preemptive schedule S for open shop E is constructed from S'[2]. Assume S' consists of a sequence of q 4-tuples. Each 4-tuple (i, j, s, f), indicates that task (j, i) is to be processed by machine j from time $s < f$ to time f. The q 4-tuples can be obtained in $0(q)$ time from the output given by algorithm R.

TABLE I
$t_{j,i}$, L_i, AND T_j FOR OPEN SHOP PROBLEM E

$t_{j,i}$	J_1	J_2	J_3	J_4	J_5	J_6	T_j
P_1	10	5	--	--	--	5	20
P_2	5	10	--	--	--	15	30
P_3	--	--	--	--	30	--	30
P_4	10	5	--	--	--	40	55
P_5	--	--	10	5	--	--	15
P_6	--	--	10	15	--	--	25
L_i	25	20	20	20	30	60	

TABLE II
$t'_{j,i}$ FOR OPEN SHOP PROBLEM T CONSTRUCTED IN PHASE 1

$t'_{j,i}$	J'_1	J'_2	J'_3	J'_4
P'_1	30	--	--	20
P'_2	--	--	30	--
P'_3	15	--	--	40
P'_4	--	40	--	--

S is constructed as follows: for each of the q 4-tuples (i', j', s, f) in S', we generate tuples for S. Let $k \ge 1$ and (j_1, i_1), (j_2, i_2), $\cdots$, (j_k, i_k) represent nonzero tasks from open shop E, such that $j_l \in P'_{j'}$, $i_l \in J'_{i'}$ for $1 \le l \le k$, and

$$f - s + t_{j_k,i_k} > \sum_{l=1}^{k} t_{j_l,i_l} \ge f - s.$$

The k 4-tuples for S are generated as follows:

$$\left(j_l, i_l, s + \sum_{w=1}^{l-1} t_{j_w,i_w}, \min\left\{f, s + \sum_{w=1}^{l} t_{j_w,i_w}\right\}\right)$$

for $1 \le l \le k$.

The first $k-1$ tasks for open shop E are eliminated from further consideration. t_{j_k,i_k} is set to $\sum_{l=1}^{k} t_{j_l,i_l} - f + s$, in case $t_{j_k,i_k} = 0$, task (j_k, i_k) is also eliminated from further consideration.

End of P'

The output of algorithm P' is S. Note that S is not in the form one would expect from the definition in Section I. However, it will be shown (Theorem 1) that S is a schedule.

Example 1: Let E be an open shop problem with $m = 6$ processors and $n = 6$ jobs. The execution times are given in Table I. T_j and L_i are as defined above. In Phase 1, $J'_1 = \{1,$

[2] Note that the schedule S to be constructed is not of the form one would expect from the definition in Section I. However, it will be shown (Theorem 1) that S is a schedule.

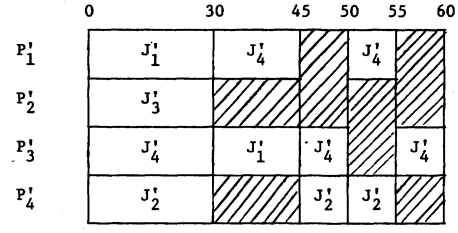


Fig. 1. Schedule S for open shop problem T.

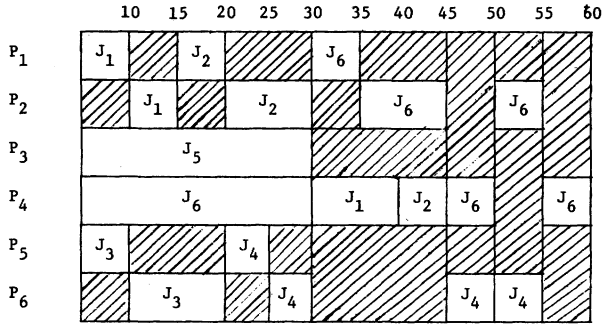


Fig. 2. Schedule S' for open shop problem E.

2}, $J'_2 = \{3, 4\}$, $J'_3 = \{5\}$, $J'_4 = \{6\}$, and $P'_1 = \{1, 2\}$, $P'_2 = \{3\}$, $P'_3 = \{4\}$, $P'_4 = \{5, 6\}$. Open shop problem T has $m' = 4$ and $n' = 4$. The execution times of each task is given in Table II. One possible schedule constructed in Phase 2 by algorithm R [1] is given in Fig. 1. Phase 3 converts the schedule S in Fig. 1 to the schedule S' for open shop problem E. Schedule S' is shown in Fig. 2.

The following theorems show the correctness and analysis of algorithm P'.

Theorem 1: Algorithm P' constructs OFT preemptive schedules for open shop problems.

Proof: Let E represent any open shop. T, S, S', and α are as defined in algorithm P'. Since the bins used (Phase 1) are of size α, then T has the property that all jobs have length $\leq \alpha$ and all machines are busy for at most α time units. In Phase 2, algorithm R [1] is used to construct an OFT preemptive schedule for open shop T. Since all jobs in T have length $\leq \alpha$ and all machines are busy for at most α time units, it then follows from [1] that algorithm R constructs a schedule with finish time α.

In order to complete the proof for this theorem, it is required to show:

1) S is a schedule for open shop E, and
2) $ft(S) = \alpha$.

In order to show S has property 1) we make the following observations from Phase 1:

a) tasks from E are included in only one task in T;
b) jobs from E are included in only one job in T; and
c) processors from E are included in only one processor in T:

Since S' is a schedule for T, it must be that no two tasks from any given job are scheduled at the same time. This together with Phase 3 and observations a)–b) imply that in S no two tasks from any given job in E are assigned at the same time. A similar argument can be used to show that at any given time no more than one task is assigned to a given processor from E in S. In order to complete the proof of 1), we need to show that all nonzero tasks (j, i) in E are assigned for $t_{j,i}$ time units. In Phase 3, only those tasks from open shop E included in task (j', i') are assigned to all the time intervals in which task (j', i') was assigned. Also, task (j, i) is assigned for no more than $t_{j,i}$ time units. Since task (j', i') was scheduled for $t'_{j',i'}$ time units and the sum of all tasks (j, i) included in (j', i') is $t'_{j',i'}$, it then follows that all tasks (j, i) are included for $t_{j,i}$ time units. This completes the proof that S has property 1).

It remains to be shown that $ft(S) = \alpha$. By construction (Phase 3), all tuples in S have finish time at most as large as any tuple in S'. All tuples in S' have finish time $\leq \alpha$, as α is the finish time for S'. Hence, S has finish time α.

Since an OFT preemptive schedule for open shop E has finish time α, then S is an OFT preemptive schedule for E. Therefore, algorithm P' constructs OFT preemptive schedules for open shops. □

Theorem 2: The time complexity for algorithm P' is $0(r + \min\{m^4, n^4, r^2\})$.

Proof: Phase 1 requires $0(r + n + m)$ time. This follows from the restriction imposed on the approximation algorithm for the packing problem. A simple $0(r + n + m)$ ordering algorithm can be used to construct open shop T if we use some form of linked allocation for all nonzero tasks.

Phase 2 is $0((r')^2)$. Let $u = \lceil W/\alpha \rceil$, then $(n')^* = (m')^* = u$, where $(n')^*$ and $(m')^*$ denotes the least number of jobs and machine T could have (note that these lower bounds for $(n')^*$ and $(m')^*$ might not be achievable). The rule described in Section I guarantees that all the bins used except possibly for one are at least half full. This implies that

$$n' \leq \min\{n, 2u - 1\}$$

and

$$m' \leq \min\{m, 2u - 1\}.$$

As $u \leq n$ and $u \leq m$ then

$$n' \leq \min\{n, 2m - 1\} \quad \text{and} \quad m' \leq \min\{m, 2n - 1\}.$$

By construction $r' \leq \min\{r, n'm'\}$

$$\leq \min\{r, nm, 2m^2 - m, 2n^2 - n\}$$

so Phase 2 is $0(\min\{r^2, n^4, m^4\})$.

In Phase 3, each nonzero task (j, i) in T has a link to all nonzero tasks in E used to build task (j, i). Thus, the substitution from S' to S can be carried out in $0(r + w)$ time, where w is the number of tuples in schedule S'. Note that w is bounded by the maximum number of preemptions, i.e., w is $0(\min\{rn, rm, n^3, m^3\})$ (see Theorem 3).

Hence, the total computing time for P' is $0(r + \min\{r^2, m^4, n^4\})$. □

Theorem 3: The maximum number of preemptions introduced by algorithm P' is $0(\min\{rn, rm, n^3, m^3\})$.

Proof: Algorithm R introduces $0(r'm')$ preemptions. In Phase 3, the maximum number of preemptions introduced in schedule S in the number of preemptions in S'. Hence, S has $0(r'm')$ preemptions. The result follows after replacing the bounds obtained for r' and m' in Theorem 2. □

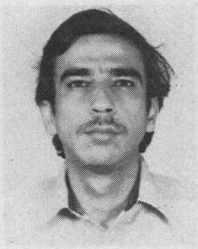
For m or n fixed, algorithm P' has worst case time complexity $0(n)$ or $0(m)$. The number of preemptions introduced is fixed. The time complexity for previous algorithms is $0(n \log n)$ and $0(m^2)$. The number of preemptions introduced is $0(n)$ and $0(m^2)$. For $m = 2$ algorithm U is optimal (linear time and introduces no preemptions).

We should note that we could se better approximation algorithms for the packing problem. This would guarantee solutions closer to the true optimal solution value [3], [4]. However, the bounds for n' and m' cannot be improved and the worst case time complexity for the algorithm would have nonlinear terms even when there are a fixed number of jobs or machines. In general, the use of better approximation algorithms for the bin packing problem and the use of algorithm P [1] would improve on the average time taken by the algorithm. It appears that even for small size problems, the actual time taken by our algorithm is smaller than the one taken by previous algorithms.

References

[1] T. Gonzalez and S. Sahni, "Open shop scheduling to minimize finish time," *JACM*, vol. 23, pp. 665–679, Oct. 1976.
[2] ——, "Flowshop and jobshop schedules: Complexity and approximation," *Oper. Res.*, vol. 26, pp. 36–52, Jan.–Feb. 1978.
[3] D. S. Johnson, "Fast allocation algorithms," in *Proc. 13th Ann. IEEE Symp. on Switching and Automata Theory*, 1972, pp. 144–154.
[4] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," *SIAM J. Comput.*, vol. 3, pp. 299–320, 1974.
[5] E. L. Lawler and J. Labetoulle, "On preemptive scheduling of unrelated parallel processors by linear programming," *JACM*, vol. 25, pp. 612–619, Oct. 1978.
[6] H.-G. Steiner, "Comparative analysis of various open shop preemptive scheduling algorithms," M.S. thesis, Pennsylvania State Univ., Aug. 1979.

✠



Teofilo Gonzalez was born in Monterrey, Mexico, on January 26, 1948. He received the B.S. degree in computer science from the Instituto Tecnologico de Monterrey in 1972, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, in 1975.

In 1975, he was an Assistant Professor in the Department of Computer Science, University of Oklahoma, Norman. From 1976 to 1979 he was an Assistant Professor in the Department of Computer Science, Pennsylvania State University, University Park. He is now a Visiting Professor in the Department of Computer Sciences, Institute of Technology, Monterrey, Mexico. His major research interests are in the design and analysis of algorithms.

Dr. Gonzalez is a member of the Association for Computing Machinery and the Operations Research Society of America.

Correspondence

On Analytical Modeling of Intermittent Faults in Digital Systems

PRAMOD K. VARSHNEY

Abstract—**This correspondence discusses three analytical models for intermittent faults in digital systems. These models attempt to represent the stochastic behavior of intermittent faults accurately. The models find applications in predicting the performance of fault detection algorithms. A fault detection procedure is described and its performance is examined based on the analytical models. A numerical example is presented which illustrates the performance prediction of the fault detection algorithm.**

Index Terms—**Analytical models, digital systems, fault tolerant computing, fault detection algorithms, intermittent faults.**

I. Introduction

Most of the effort in fault-tolerant computing has been directed towards the study of permanent faults, and a variety of techniques for the diagnosis of such faults have been considered [1], [2]. However, experience with many practical digital systems indicates that most of the digital system failures can be attributed to the faults of an intermittent nature [3]. The need for a systematic study of intermittent faults has strongly been advocated [3], [4] and it is only recently that intermittent faults have received some attention [5]–[12].

One important problem in this area is to accurately characterize the stochastic behavior of intermittent faults. A probabilistic framework is required due to the transient nature of intermittent faults. Analytical models can be employed to compute various statistics of intermittent faults. They can be used to examine the performance of intermittent fault detection algorithms. The models also provide an insight into the clustering properties of intermittent faults.

Analytical models for intermittent faults have been treated in the literature. For example, Breuer [5] has considered a two-state Markov model. Kamal and Page [6] have developed a finite-state model for intermittent faults. This correspondence considers three alternative approaches for the characterization of the fault process. Different degrees of memory are incorporated in these models. First, we formally define the intermittent fault process and some of the related probabilistic parameters. After discussing the three analytical models, a memoryless fault detection procedure is described. This is similar to the detection algorithms considered in the literature. The performance of the fault detection algorithm

Manuscript received November 15, 1977; revised May 19, 1979.
The author is with the Department of Electrical and Computer Engineering, Syracuse University, Syracuse, NY 13210.

0018-9340/79/1000-0786$00.75 © 1979 IEEE