Efficient Algorithms for k-out-of-n & Consecutive-Weighted-k-out-of-n:F System

Jer-Shyan Wu
Chung-Hua Polytechnic Institute, Hsinchu
Rong-Jaye Chen, Member IEEE
National Chiao Tung University, Hsinchu

Key Words — k-out-of-n system, consecutive-weighted-k-out-of-n:F system, system reliability, algorithm, parallel computation.

Reader Aids —
General purpose: Report new algorithms
Special math needed for explanations: Probability theory
Special math needed to use results: Same
Result useful to: Reliability analysts & theoreticians

Abstract — **A new reliability model, consecutive-weighted-k-out-of-n:F system, is proposed and an $O(n)$ algorithm is provided to evaluate its reliability. An $O(n \cdot \min(n,k))$ algorithm is also presented for the circular case of this model. We design an $O(n)$ parallel algorithm using k processors to compute the reliability of k-out-of-n systems, that achieves linear speedup.**

1. INTRODUCTION

A consecutive-k-out-of-n:F system consists of a sequence of n ordered components such that the system fails iff at least k consecutive components fail. The reliability of this system was first studied by Chiang & Niu [3], and later extensively studied in [1, 4-7, 10, 15, 17-18].

Sections 2 & 3 state a more general consecutive-weighted-k-out-of-n:F system, and designs an $O(n)$ algorithm to evaluate its reliability. Because n components need to be checked in any algorithm, this $O(n)$ algorithm is optimal. In addition, for the circular consecutive-weighted-k-out-of-n:F system, an $O(n \cdot \min(n,k))$ algorithm is proposed to compute the system reliability. If each component has weight 1, the original consecutive-k-out-of-n:F system is a special case of this new model.

The k-out-of-n systems were studied in [2, 8-9, 11-14, 19], where the system was [good, failed] iff the total number of [good, failed] components was at least k. The reliability of the k-out-of-n:G system is the complement of the probability of failure of the $(n-k+1)$-*out-of-n*:F system. Without loss of generality, we discuss k-out-of-n:G systems only.

A sequential algorithm is defined as using one processor, while the parallel algorithm uses more than one processor [16]. The speedup of a parallel algorithm is the ratio [computing time of the best sequential algorithm] ÷ [computing time of the parallel algorithm]. Given P processors, we would like our parallel algorithm to run P times as fast as the best sequential algorithm. When the speedup of a parallel algorithm is P, the parallel algorithm achieves linear speedup. It is often hard to propose a parallel algorithm with linear speedup.

So far, the best sequential algorithm for computing the reliability of the k-out-of-n:G systems needs $O(n \cdot k)$ time [2, 8-9, 11-14, 19] and it is hard to improve on this time complexity. This paper designs a parallel algorithm using k processors to compute the reliability in $O(n)$ computing time, and thus achieves linear speedup (the speedup is k).

For k-out-of-n systems, we achieved 2 important results:

- an $O(n)$ algorithm for computing system reliability;
- a parallel algorithm with linear speedup.

Section 2 describes the assumptions & notation. Section 3 shows an O(n) algorithm for consecutive-weighted-k-out-of-n:F systems. Section 4 shows an $O(n \cdot \min(n,k))$ algorithm for circular consecutive-weighted-k-out-of-n:F systems. Section 5 proposes an $O(n)$ parallel algorithm to compute the reliability of the k-out-of-n systems. All proofs are in the appendix.

2. MODEL

Notation (general)

- n — number of components in a system
- p_i, q_i — probability that component i [functions, fails]; $p_i + q_i = 1$
- $\mathcal{I}(\cdot)$ — $\mathcal{I}(\text{True})=1$, $\mathcal{I}(\text{False}) = 0$: Indicator function

Notation (weighted system)

- k — minimum total weight of failed consecutive components which causes system failure
- w_i — weight of component i
- S_i — minimum event which causes system failure
- m — total number of all possible S_i
- $\text{Beg}(i)$, $\text{End}(i)$ — [first, last] component of S_i
- $\text{Wet}(i)$ — total weight of S_i
- $Q(i)$ — $\prod_{j=\text{Beg}(i)}^{\text{End}(i)} q_j$
- $R_L(i,j)$, $R_C(i,j)$ — reliability of [linear, circular] system consisting of components $i, i+1, \dots, j$
- $F_\Omega(i,j)$ — $1 - R_\Omega(i,j)$, for $\Omega = L, C$

Notation (k-out-of-n:G system)

- k — minimum number of all good components which make the system good
- $R(i,j)$, $F(i,j)$ — [reliability, unreliability] of a j-out-of-i:G system.

Other, standard notation is given in "Information for Readers & Authors" at the rear of each issue.

Assumptions

1. Each component and the system either functions or fails.
2. All n component states are mutually s-independent.
3. For weighted systems: a) each component has its own positive integer weight; b) the system fails iff the total weight of the failed consecutive components is at least k.
4. For k-out-of-n:G systems: a) each processor is unique; b) the system is good iff the number of good components is at least k. ◄

3. CONSECUTIVE-WEIGHTED-k-out-OF-n:F SYSTEMS

We present our algorithm to compute the reliability of a consecutive-weighted-k-out-of-n:F system. Before deriving the algorithm, we need lemmas 1 & 2.

Lemma 1. A consecutive-weighted-k-out-of-n:F system needs only $O(n)$ computing time to derive each S_i, for $i = 1, \ldots, m$. ◄

Algorithm A

```
begin
  /* Set initial value for variables. */
  m = 0; event = 1;
  Wet(event) = 0; Q(event) = 1; Beg(event) = 1;
  /* Scan components from the first one. */
  for component = 1 to n
  begin
    Wet(event) = Wet(event) + W_component;
    Q(event) = Q(event)·q_component;
    if Wet(event) ≥ k then
    /* New event appears */
    begin
      m = m + 1;
      End(event) = component;
      /* Check whether the event is minimum */
      while (Wet(event) - W_Beg(event) ≥ k) then
        begin
          Wet(event) = Wet(event) - W_Beg(event);
          Q(event) = Q(event)/q_Beg(event);
          Beg(event) = Beg(event) + 1;
        end
      endwhile
      Beg(event+1) = Beg(event) + 1;
      Wet(event+1) = Wet(event) - W_Beg(event);
      Q(event+1) = Q(event)/q_Beg(event);
      event = event + 1;
    end
  end
endfor
end
/* End of Algorithm */
```

Example 1. Illustration of Algorithm A

Given a consecutive-weighted-4-out-of-7:F system, the weights of components 1 - 7 are: 1, 1, 1, 2, 3, 2, 1. Use algorithm A to obtain the results in figure 3.1. There are 3 minimum events (S_1: 2-4, S_2: 4-5, S_3: 5-6) which cause system failure. Figure 3.1 shows these minimum events as dark consecutive components. The computing time of the **for**-loop is $7=n$, and the **while**-loop is $5=\text{Beg}(m)$. ◄

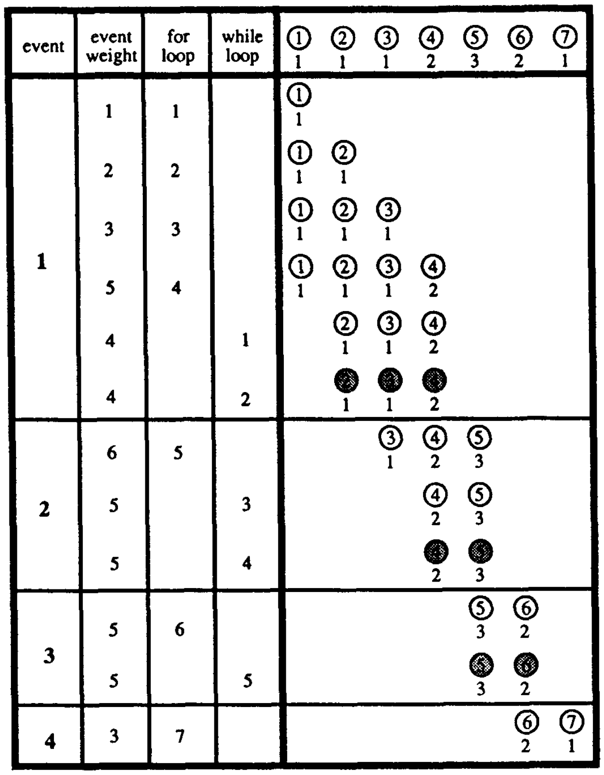

Figure 3.1. An Example of a Consecutive-Weighted-4-out-of-7:F system

We need to derive $F_L(1,i)$, for $i = 1,\ldots,n$.

Lemma 2. For a consecutive-weighted-k-out-of-n:F system, the $F_L(1,i)$, for $i = 1,\ldots,n$, is:

$$F_L(1,i) = 0, \text{ for } i = 0,1,\ldots,\text{End}(1) - 1. \tag{3-1}$$

$$F_L(1,i) = \Pr\{S_1 \cup S_2 \cup \ldots \cup S_j\} = F_L(1,\text{End}(j)),$$

$$\text{for } i = \text{End}(j), \text{End}(j)+1, \ldots, \text{End}(j+1)-1,$$

$$\text{and } j = 1,2,\ldots,m-1. \tag{3-2}$$

$$F_L(1,i) = \Pr\{S_1 \cup S_2 \cup \ldots \cup S_m\} = F_L(1,\text{End}(m)),$$

$$\text{for } i = \text{End}(m), \text{End}(m)+1, \ldots, n. \tag{3-3}$$ ◄

Apply lemmas 1 & 2; only O(n) computing time is needed to obtain $R_L(1,n)$ — as stated formally in theorem 1.

Theorem 1. For a consecutive-weighted-k-out-of-n:F system, the $F_L(1,\text{End}(j))$, for $j = 2, 3, \ldots, m$, is:

$$F_L(1,\text{End}(j)) = F_L(1,\text{End}(j-1)) + \sum_{i=0}^{\text{Beg}(j)-\text{Beg}(j-1)-1} R_L(1,\text{Beg}(j-1)+i-1)\cdot p_{\text{Beg}(j-1)+i} \cdot\left[\prod_{l=\text{Beg}(j-1)+i+1}^{\text{Beg}(j)-1} q_l\right]\cdot Q(j), \tag{3-4}$$

and the time to obtain $R_L(1,n)$ is $O(n)$. ◄

Example 2. Compute $R_L(1,n)$ in $O(n)$

For a consecutive-weighted-4-out-of-7:F system, the weight for components 1 - 7 is 1, 1, 1, 2, 3, 2, 1. From example 1, there are 3 minimum events (E_1: 2-4, E_2: 4-5, E_3: 5-6) which cause system failure.

Initially, by algorithm A:

$\text{Beg}(1) = 2$, $\text{End}(1) = 4$, $Q(1) = q_2\cdot q_3\cdot q_4$;

$\text{Beg}(2) = 4$, $\text{End}(2) = 5$, $Q(2) = q_4\cdot q_5$;

$\text{Beg}(3) = 5$, $\text{End}(3) = 6$, $Q(3) = q_5\cdot q_6$;

in $O(n)$ computing time. Then, by lemma 2:

$F_L(1,1) = F_L(1,2) = F_L(1,3) = 0$;

$F_L(1,4) = F_L(1,\text{End}(1))$;

$F_L(1,5) = F_L(1,\text{End}(2))$;

$F_L(1,6) = F_L(1,7) = F_L(1,\text{End}(3))$;

in $O(n)$ computing time. Furthermore, by theorem 1:

$F_L(1,\text{End}(1)) = Q(1)$;

$F_L(1,\text{End}(2)) = F_L(1,\text{End}(1)) + R(2)\cdot p_3\cdot Q(2) + R(1)\cdot p_2\cdot q_3\cdot Q(2)$;

$F_L(1,\text{End}(3)) = F_L(1,\text{End}(2)) + R(3)\cdot p_4\cdot Q(3)$;

in $O(n)$ computing time. So, it takes $O(n)$ computing time to derive $F_L(1,7)$. Finally,

$R_L(1,7) = 1 - F_L(1,7)$.

4. CIRCULAR CONSECUTIVE-WEIGHTED-k-out-OF-n:F SYSTEM

Section 3 gives an $O(n)$ algorithm for consecutive-weighted-k-out-of-n:F systems, and [18] proposed an $O(n\cdot k)$ algorithm for circular consecutive-k-out-of-n:F systems. Combining these two algorithms, we propose an $O(n\cdot\min(n,k))$ algorithm to compute the reliability of circular consecutive-weighted-k-out-of-n:F systems. The formula for circular consecutive-weighted-k-out-of-n:F systems is:

$$R_C(1,n) = \sum_A \delta(s,l), \tag{4-1}$$

$$A_{k,n;s,l} \equiv \sum_{i=1}^{s-1} w_i + \sum_{j=l+1}^{n} w_j < k$$

$$\delta(s,l) \equiv \left[\prod_{i=1}^{s-1} q_i\right]\cdot p_s\cdot R_L(s+1,\ l-1)\cdot p_l \cdot\left[\prod_{j=l+1}^{n} q_j\right]\cdot \mathcal{I}(A_{k,n;s,l}).$$

Consider two cases: 1) $n \ge k$, and 2) $n < k$.

4.1 $n \ge k$

Because each component has at least weight 1, (4-1) can be presented as:

$$R_C(1,n) = \sum_{s-1+n-l<k} \delta(s,l) = \sum_{s=1}^{k}\ \sum_{l=n-k+s}^{n} \delta(s,l). \tag{4-2}$$

It takes $O(k^2)$ computing time to check condition A since number of terms in (4-2) is $½k\cdot(k+1)$. To obtain $\delta(s,l)$, we need to calculate:

$$\{R_L(2,j)\}_{j=n-k}^{n-1},$$
$$\{R_L(3,j)\}_{j=n-k+1}^{n-1},$$
$$\vdots$$
$$\{R_L(k,j)\}_{j=n-1}^{n-1}; \tag{4-3}$$

$$\left\{\prod_{i=1}^{j} q_i\right\}_{j=1}^{k-1},\ \left\{\prod_{i=j}^{n} q_i\right\}_{j=n-k+2}^{n}. \tag{4-4}$$

In section 3, while computing $R_L(1,n)$, we also obtain $\{R_L(1,j)\}_1^{n-1}$ in $O(n)$ time. By this property, we can compute (4-3) in $O(n\cdot k)$ time. It needs only $O(k)$ time to obtain (4-4). Finally, (4-2) contains $½k\cdot(k+1)$ terms, so the time complexity for computing $R_C(1,n)$ is:

$$O(k^2) + O(n\cdot k) + O(k) + O(k^2) = O(n\cdot k).$$

4.2 n < k

Eq (4-1) can be presented as:

$$R_C(1,n) = \sum_{s=1}^{n-2} \sum_{l=s+2}^{n} \delta(s,l) \tag{4-5}$$

where $\delta(s,l$ is)the same as in section 4.1; and it is necessary to get:

$$\{R_L(2,j)\}_{j=2}^{n-1},$$

$$\{R_L(3,j)\}_{j=3}^{n-1},$$

$$\vdots$$

$$\{R_L(n-1,j)\}_{j=n-1}^{n-1}; \tag{4-6}$$

$$\left\{\prod_{i=1}^{j} q_i\right\}_{j=1}^{n-2}, \left\{\prod_{i=j}^{n} q_i\right\}_{j=3}^{n}. \tag{4-7}$$

The time complexity for computing $R_C(1,n)$ is $O(n^2)$ + $O(n^2)$ + $O(n)$ + $O(n^2)$ = $O(n^2)$. Therefore the time complexity for computing $R_C(1,n)$ is $O(n \cdot \min(n,k))$.

5. k-out-OF-n:G SYSTEM

This section presents an $O(n)$ parallel algorithm using k processors to compute the reliability of the k-out-of-n:G system.

$$R(n,k) = 0, \text{ if } n < k. \tag{5-1}$$

We start with an $O(n \cdot k)$ sequential algorithm.

Let $n \geq k$, for i = 1, ..., n, and j = 1, 2, ..., k. We derive a recurrence relation:

$$R(i,j) = p_i \cdot R(i-1,j-1) + q_i \cdot R(i-1,j). \tag{5-2}$$

In order to derive $R(n,k)$, by (5-2), it is necessary to obtain $R(i,j)$, for i = 0, 1, ..., n, and j = 0, 1, ..., k, during the recursive processing. Put all $R(i,j)$ in a table with $(n+1)$ rows and $(k+1)$ columns as in figure 5.1. We have initial $R(i,j)$ in row 1 and column 1:

$$R(0,j) = 0, \text{ for } j = 1, 2, \ldots, k; \tag{5-3}$$

$$R(i,0) = 1, \text{ for } i = 0, 1, \ldots, n. \tag{5-4}$$

The following details the method for computing $R(n,k)$.

By (5-1), if $n < k$ then $R(n,k)$ = 0. Otherwise, by (5-3) & (5-4), we construct column 1 and row 1 in the $R(i,j)$ table. Then, by (5-2), we construct row 2, row 3, ..., row $n+1$ — in that order. $R(n,k)$ is eventually derived. Because the size of the $R(i,j)$ table is $(n+1) \cdot (k+1)$, the sequential algorithm needs $O(n \cdot k)$ running time.

R(i,j)	0	1	2	-----	k-1	k
0	1	0	0	-----	0	0
1	1	R(1,1)	0	-----	0	0
2	1	R(2,1)	R(2,2)	-----	0	0
⋮	⋮	⋮	⋮		⋮	⋮
k-1	1	R(k-1,1)	R(k-1,2)	-----	R(k-1,k-1)	0
k	1	R(k,1)	R(k,2)	-----	R(k,k-1)	R(k,k)
⋮	⋮	⋮	⋮		⋮	⋮
n-1	1	R(n-1,1)	R(n-1,2)	-----	R(n-1,k-1)	R(n-1,k)
n	1	R(n,1)	R(n,2)	-----	R(n,k-1)	R(n,k)

Figure 5.1 The $R(i,j)$ Table with $(n+1)$ Rows and $(k+1)$ Columns.

We describe an $O(n)$ parallel algorithm to compute $R(n,k)$. Each of our designed processors contains, as shown in figure 5.2:

- 2 inputs (I, J),
- 1 output (O),
- 1 buffer (B).

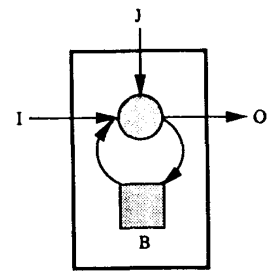

Figure 5.2 Our Designed Processor. There are two impusts (I,J), One Output (O) and One Buffer (B). The Circular Object Represents the Logic Cirguit for Computing Operation: & $= j \cdot I + (1-J) \cdot B$.

The computing operation is:

$$O = J \cdot I + (1 - J) \cdot B. \tag{5-5}$$

The processor stores (O) in (B): $B = O$. (5-6)

We use k processors and route them as figure 5.3 shows. Initially for each processor, the buffer stores 0, and input port J receives $p_1, p_2, \ldots, p_n$; and input port I receives the output of its left processor. Input port I of the most-left processor always receives 1.

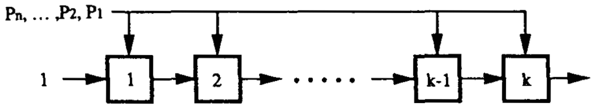

Figure 5.3 The Routing for k Processors. Each Processor's Buffer Initially Stores Zero. In Each Step, Each Processor's J Input Port Receives $P_1, P_2, \ldots, P_n$, Orderly; and I Input Port Recieves Its Left Processor's Output. The I Input of Processor 1 is Always 1.

Label the processors from left to right: 1, 2, ..., k.

- By (5-3), the buffer of processor j initially stores $R(0,j)$, for $j = 1, 2, \ldots, k$.
- By (5-4), the input port I of process 1 receives $R(i,0)$ (value $= 0$), in step i.
- By (5-2), (5-5), (5-6), the buffer of processor j stores $R(i,j)$ (value $= 1$), in step i, for $i = 1, \ldots, n$, and $j = 1, 2, \ldots, k$.
- After n steps, $R(n,k)$ can be obtained in the buffer of the processor k.

This is an $O(n)$ parallel algorithm using k processors for computing $R(n,k)$.

APPENDIX

A.1 Proof of Lemma 1

Initially, scan from the first component and compute the total weight until the total-weight $\geq k$, then check whether this event is minimum. Try to remove components from the beginning of this event until the total-weight $< k$. Hence obtain S_1. Let the beginning component of the second event be Beg(1) + 1, and continue to scan from the component End(1) + 1. By the this method, we obtain S_i, for $i = 2, \ldots, m$. The details of this method are given in algorithm A.

In algorithm A, the computing time of the **For** loop is $O(n)$, and the worst computing time of the **while** loop is:

$$\mathrm{Beg}(1) + (\mathrm{Beg}(2) - \mathrm{Beg}(1)) + (\mathrm{Beg}(3) - \mathrm{Beg}(2)) + \ldots$$

$$+ (\mathrm{Beg}(m) - \mathrm{Beg}(m-1)) = \mathrm{Beg}(m) = O(n).$$

So it needs only $O(n)$ computing time to derive each S_i, and obtain $\mathrm{Beg}(i)$, $\mathrm{End}(i)$, $Q(i)$, for $i = 1, 2, \ldots, m$. *Q.E.D.*

A.2 Proof of Lemma 2

$$F_L(1,i) = 0, \text{ for } i = 0, 1, \ldots, \mathrm{End}(1) - 1.$$

$F_L(1,i) = \Pr\{S_1 \cup S_2 \cup \ldots \cup S_j\}$, for $i = \mathrm{End}(j)$, and $j = 1, 2, \ldots, m$.

Consider a consecutive-weighted-k-out-of-i:F system, for $i = \mathrm{End}(j), \mathrm{End}(j) + 1, \ldots, \mathrm{End}(j+1) - 1$, and $j = 1, 2, \ldots, m - 1$; all the minimum events which cause the system failure are still $S_1, S_2, \ldots, S_j$. So (3-2) is proved.

Eq. (3-3) is obtained similarly to (3-2), if we consider only components from End(m) to n. *Q.E.D.*

A.3 Proof of Theorem 1

Initially, by lemma 2, $F_L(1,0) \equiv 0$; and $R_L(1,0) = 1$. By the sum-of-disjoint-products method, for $j = 2, 3, \ldots, m$ —

$$F_L(1,\mathrm{End}(j)) = \Pr\{S_1 \cup S_2 \cup \ldots \cup S_j\}$$

$$= \Pr\{S_1 \cup S_2 \cup \ldots \cup S_{j-1}\} + \Pr\{\overline{S_1 \cup S_2 \cup \ldots \cup S_{j-1}} \cap S_j\}$$

$$= F_L(1,\mathrm{End}(j-1)) + R_L(1,\mathrm{Beg}(j)-2) \cdot p_{\mathrm{Beg}(j)-1} \cdot Q(j)$$

$$+ R_L(1,\mathrm{Beg}(j)-3) \cdot p_{\mathrm{Beg}(j)-2} \cdot q_{\mathrm{Beg}(j)-1} \cdot Q(j) + \ldots$$

$$+ R_L(1,\mathrm{Beg}(j-1)-1) \cdot p_{\mathrm{Beg}(j-1)} \cdot q_{\mathrm{Beg}(j-1)+1} \cdot \ldots \cdot q_{\mathrm{Beg}(j)-1} \cdot Q(j)$$

$$= F_L(1,End(j-1))$$

$$+ \sum_{i=0}^{\mathrm{Beg}(j)-\mathrm{Beg}(j-1)-1} R_L(1,\mathrm{Beg}(j-1)+i-1) \cdot p_{\mathrm{Beg}(j-1)+i}$$

$$\cdot \left[\prod_{l=\mathrm{Beg}(j-1)+i+1}^{\mathrm{Beg}(j)-1} q_l \right] \cdot Q(j). \tag{A-1}$$

Apply lemma 1; we need only $O(n)$ computing time to derive each S_i, and obtain $\mathrm{Beg}(i)$, $\mathrm{End}(i)$, $Q(i)$, for $i = 1, 2, \ldots, m$. By lemma 2, each $F_L(1,i)$, for $i = 0, 1, \ldots, n$, is zero or any one of the $F_L(1,\mathrm{End}(j))$, for $j = 1, 2, \ldots, m$. By (A-1), the worst computing time for deriving $F_L(1,\mathrm{End}(j))$, for $j = 1, 2, \ldots, m$, is:

$$\mathrm{Beg}(1) + (\mathrm{Beg}(2) - \mathrm{Beg}(1)) + (\mathrm{Beg}(3) - \mathrm{Beg}(2)) + \ldots$$

$$+ (\mathrm{Beg}(m) - \mathrm{Beg}(m-1)) = \mathrm{Beg}(m) = O(n).$$

Because $R_L(1,n) = 1 - F_L(1,n)$, it takes $O(n)$ computing time to derive $R(n)$. *Q.E.D.*

REFERENCES

[1] I. Antonopoulou, S. Papastavridis, "Fast recursive algorithm to evaluate the reliability of a circular consecutive-k-out-of-n:F system", *IEEE Trans. Reliability*, vol R-36, 1987 Apr, pp 83-84.
[2] R.E. Barlow, K.D. Heidtmann, "Computing k-out-of-n system reliability", *IEEE Trans. Reliability*, vol R-33, 1984 Oct, pp 322-323.
[3] D.T. Chiang, S.C. Niu, "Reliability of consecutive-k-out-of-n:F system", *IEEE Trans. Reliability*, vol R-30, 1981 Apr, pp 87-89.
[4] C. Derman, G. Lieberman, S. Ross, "On the consecutive-k-out-of-n:F system", *IEEE Trans. Reliability*, vol R-31, 1982 Apr, pp 57-63.
[5] J.C. Fu, "Reliability of consecutive-k-out-of-n:F systems with (k-1)-step Markov dependence", *IEEE Trans. Reliability*, vol R-35, 1986 Dec, pp 602-606.

[6] G. Ge, L. Wang, "Exact reliability formula for consecutive-k-out-of-n:F system with homogeneous Markov dependence", *IEEE Trans. Reliability*, vol 39, 1990 Dec, pp 600-602.

[7] F.K. Hwang, "Fast solutions for consecutive-k-out-of-n:F system", *IEEE Trans. Reliability*, vol R-31, 1982 Dec, pp 447-448.

[8] S.P. Jain, K. Gopal, "Recursive algorithm for reliability evaluation of k-out-of-n:G system", *IEEE Trans. Reliability*, vol R-34, 1985 Jun, pp 144-146.

[9] P.W. McGrady, "The availability of a k-out-of-n:G network", *IEEE Trans. Reliability*, vol R-34, 1985 Dec, pp 451-452.

[10] S. Papastavridis, M. Lambiris, "Reliability of consecutive-k-out-of-n:F system for Markov-dependent components", *IEEE Trans. Reliability*, vol R-36, 1987 Apr, pp 78-79.

[11] H. Pham, S.J. Upadhyaya, "The efficiency of computing the reliability of k-out-of-n systems", *IEEE Trans. Reliability*, vol 37, 1988 Dec, pp 521-523.

[12] S. Rai, A.K. Sarje, E.V. Prasad, A. Kumar, "Two recursive algorithm for computing the reliability of k-out-of-n systems", *IEEE Trans. Reliability*, vol R-36, 1987 Jun, pp 261-265.

[13] T. Risse, "On the evaluation of the reliability of k-out-of-n systems", *IEEE Trans. Reliability*, vol R-36, 1987 Oct, pp 433-435.

[14] A.K. Sarje, E.V. Prasad, "An efficient non-recursive algorithm for computing the reliability of k-out-of-n systems", *IEEE Trans. Reliability*, vol 38, 1989 Jun, pp 234-235.

[15] J.G. Shanthikumar, "Recursive algorithm to evaluate the reliability of a consecutive-k-out-of-n:F system", *IEEE Trans. Reliability*, vol R-31, 1982 Dec, pp 442-443.

[16] F.T. Leighton, "Introduction to parallel algorithms and architectures: arrays trees hypercubes", 1992; Morgan Kaufmann Publishers.

[17] J.S. Wu, R.J. Chen, "An $O(k \cdot n)$ algorithm for a circular consecutive-k-out-of-n:F system", *IEEE Trans. Reliability*, vol 41, 1992 Jun, pp 303-305.

[18] J.S. Wu, R.J. Chen, "Efficient algorithm for reliability of a circular consecutive-k-out-of-n:F system", *IEEE Trans. Reliability*, vol 42, 1993 Ma, pp 163-164.

[19] J.S. Wu, R.J. Chen, "An algorithm for computing the reliability of weighted-k-out-of-n systems", *IEEE Trans. Reliability*, vol 43, 1994 Jun, pp 327-328.

AUTHORS

Jer-Shyan Wu; Dept. of Computer Science; Chung-Hua Polytechnic Inst; 30 Tung Shiang; Hsinchu 30067 TAIWAN - R.O.C.

Jer-Shyan Wu was born in Taipei, Taiwan in 1967. He received his BS (1989) in Computer Science from National Taiwan University, and MS (1991) & PhD (1994) in Computer Science from National Chiao-Tung University. Dr. Wu is Ass't Professor in the Dept. of Computer Science at Chung-Hua Polytechnic Institute. His research interests include reliability analysis, queueing theory, parallel computing, interconnected networks, and algorithms.

Dr. Rong-Jaye Chen; Dept. of Computer Science and Information Engineering; National Chiao-Tung University; 1001 Ta Hsueh Road; Hsinchu 30050 TAIWAN - R.O.C.

Rong-Jaye Chen (M'90) was born in Taiwan in 1952. He received his BS (1977) in Mathematics from National Tsing-Hua University, and PhD (1987) in Computer Science from University of Wisconsin-Madison. Dr. Chen is now Professor in the Department of Computer Science and Information Engineering at National Chiao-Tung University, and is a member of IEEE. His research interests include reliability theory, algorithms, mathematical programming, and computer networking.

Manuscript received 1993 September 30.

IEEE Log Number 92-15598

◄TR►

MANUSCRIPTS RECEIVED *MANUSCRIPTS RECEIVED* *MANUSCRIPTS RECEIVED* *MANUSCRIPTS RECEIVED*

"Safety & reliability of comma-free digital coding for railway track circuits" (R. Hill, J. Bouillevaux), Dr. R. John Hill • School of Electrical Engineering • University of Bath • Claverton Down • Bath BA2 7AY • GREAT BRITAIN. (TR94-115)

"Optimal release policy for hyper-geometric distribution software-reliability growth model" (R. Hou, S. Kuo, Y. Chang), Sy-Ken Kuo • Dept. of Electrical Eng'g • National Taiwan Univ. • Taipei • TAIWAN - R.O.C.. (TR94-116)

"Optimal design of parallel-series systems with both open & short failure modes" (V. Kirilyuk), Dr. V. Kirilyuk • Prosp. Nauki 35/4, fl.61 • 252028 Kiev • UKRAINE. (TR94-117)

"A reliability approach to transmission-expansion planning using minimal cut theory" (P. Bhattacharjee, et al.), P. K. Bhattacharjee • Electrical Eng'g Dept. • Jadavpur Univ. • Calcutta - 700 032 • INDIA. (TR94-118)

"Reliability-modeling techniques for fault-tolerant systems" (G. Yang, D. Yu), Guu-Chang Yang • Dept. of Electrical Engineering • National Chung-Hsing University • Taichung • TAIWAN - R.O.C.. (TR94-119)

"Estimating component-defect probability from masked system success/failure" (B. Reiser, B. Flehinger, A. Conn), Dr. Betty J. Flehinger • T.J. Watson Research Ctr • POBox 218 • Yorktown Heights, New York 10598 • USA. (TR94-120)

"A model for accelerated life testing" (O. Tyoskin, et al.), Dr. Oleg I. Tyoskin • c/o Sergej Terskin • 491 Sunburst Ct • Gaithersburg, Maryland 20877 • USA. (TR94-121)

"Contribution to an analytic theory of availability" (M. Nallino), Michel Nallino • 16 rue Louis Garneray • F06300 Nice • FRANCE. (TR94-122)

"Characterization of bivariate mean residual life function" (H. Kulkarni, et al.), H. V. Kulkarni • Dept. of Statistics • Shivaji Univ. • Kolhapur - 416 004 • INDIA. (TR94-123)

"Efficient algorithm for calculating minimal-path sets of a system" (N. Ebrahimi, et al.), Dr. Nader B. Ebrahimi • Division of Statistics • Dept. of Mathematical Sciences • Northern Illinois University • DeKalb, Illinois 60115-2888 • USA. (TR94-125)

"An automatic VLSI-tester correlation method" (B. Svrcek), Ben C. Svrcek • Digital Signal Processor Div • Motorola Inc. • 6501 William Cannon Dr; MD OE314 • Austin, Texas 78735-8598 • USA. (TR94-126)

"Accelerated life tests for products of unequal size" (D. Bai, H. Yu), Dr. Do Sun Bai, Professor • Dept. of Industrial Engineering • Korea Adv. Inst. of Science & Technology • 373-1 Gusung-dong, Yusung-gu • Taejon 305-701 • Rep. of KOREA. (TR94-127)

"Reliability modeling of large circular consecutive-(n-2)-out-of-n:G systems" (W. Schneeweiss), Dr. W. G. Schneeweiss, Professor • FernUniversitaet • Postfach 940 • D-58084 Hagen 1 • Fed. Rep. GERMANY. (TR94-128)

"Using 3-stage sampling for inferring fault-coverage probabilities" (C. Constantinescu), Dr. Cristian Constantinescu • Electrical Engineering Dept. • CBox 90291 • Duke University • Durham, North Carolina 27708-0291 • USA. (TR94-129)

"Developing ASIC testability requirements for high reliability multi-ASIC systems" (W. Willing, A. Helland), Walter E. Willing • 4 Mill Pool Court • Catonsville, Maryland 21228-2450 • USA. (TR94-130)