An Algorithm for Computing the Reliability of Connected (1,2) or (2,1) out of (m,.n) : F Lattice System.

S. M. khamis and N. Mokhlis.

Department of Mathematics
Faculty Of Science
Ain Shams University.

Abstract.

This paper presents a recursive algorithm for calculating the reliability functions of linear and circular connected (1.2) or (2,1) out of (m,n) : F Lattice systems. This algorithm depends on the one-to-one correspondence relation between the representation of the systems and the class of 0-1 matrices having no two consecutive l's at any row or any column.

Key words.

Linear. Circular connected (1.2) or (2.1) out of (m,n) : F lattice system, algorithm, computing, and the reliability.

CONGRESSUS NUMERANTIUM **127** (1997), pp. 143-154

§1 Introduction.

A linear (m.n)-lattice system consists of mn components arranged in m rows and n columns. i.e. the components are arranged like the elements of an (m,n) matrix. A circular (m.n)-lattice system consists of m circles centered at the same center and having n rays. The components are placed at the intersections of the rays and circles, i.e. each circle contains n elements. and each ray contains m elements. A linear or circular connected (1,2) or (2,l) out of (m,n) : F lattice system fails if and only if at least two connected components fail. So in the linear case. the system fails whenever at least two connected components fail in any row or any column. while in the circular case. the system fails whenever at least two connected components fail in any circle or any ray.

As a practical example of the linear connected (1.2) or (2.1) out of (m.n) : F lattice system is a supervision system as given by Boehme et. al. [1]. In this system if two neighboring cameras not connected by a line fail the system does not fail. Whereas. see [l]. a reactor as a cylindrical object covered by a system of feelers for measuring temperature may be represented by m circles including n feelers. This measure system fails whenever at least (l.2) or (2,1) matrix of failed components occur. Such system is a circular connected (1,2) or (2,1) out of (m,n) : F Lattice system.

Boehme et. al., [1], obtained the reliability formulas of simple systems, using the results of consecutive k-out-of-n F systems. They obtained the reliability for m and n taking only the values 2 or 3. In (2). N. Mokhlis et. al. derived recursive formulas for the reliability of more general models. linear and circular connected (1.2) or (2.1) out of (m.3) : F lattice systems for any m.

Up to now, in this field, a few studies has been found (e.g [1,1992] and [2,1997]). Most studies depended on using mathematical treatments to solve some special cases of the problem. However. in a general form, the problem is sophisticate and still open. So, we attempt to construct an algorithmic method to solve the general form suggested problem.

The purpose of this paper is to introduce an algorithm to generate a computer enumeration of the reliability function of connected (1,2) or (2,1) out of (m,n) : F lattice systems for any m and n. The algorithm depends on the relation between the structure of the linear lattice systems and (m,n) matrices with 0-1 entries. We have shown that, the reliability function depends on the number of 0-1 matrices having no two consecutive l's at any row or any column. The given algorithm is developed to count these matrices. which are recursively created in the colexicographic order w.r.t. its rows. Fortunately a slight modification is taken into account for demonstrating algorithm to cover the calculation of the circular case.

This paper splits into four sections. § 1 gives an introduction on the formalization of the required problem, §2 introduces the suitable assumptions and required notation. The recursive algorithm is demonstrated in §3. The treatment of circular case is developed in §4. Finally the paper contains some calculated results of two cases via using a Pascal code of the suggested algorithm.

§2. Assumptions and Notation.

§§ 2.1 Assumptions.

The following assumptions are used :-

1) For the linear system, we have n components in m rows.
2) For the circular system, we have m circles, each having n components.
3) Each component and the system are either operating or failed.
4) The components are s-independent and identical.
5) The system (linear or circular) fails if and only if at least one subsystem of connected (1,2) or (2,1) failed components occurs.
6) We assign a "O" for an operating component and a "1" for a failed component. This leads to represent a system (either linear or circular) by (m,n) 0-1 matrix. say **M**, whichis defined as:

$$\mathbf{M} = [m_{ij}] = \begin{cases} 1 & \text{if the } \{(i-1)n+j\}^{th} \text{component is failed} \\ 0 & \text{otherwise.} \end{cases}$$

7) The matrix **M** that simulates any system is called an *accepted* matrix if and only if no two or more consecutive l's appear

in any row or any column. Otherwise it is not an accepted matrix and is called a *redundant* one.

§§2.2 Notation.

Here we introduce the relevant notation that are used in the rest part of the paper.

m	: the number of rows. or number of circular:
n	: the number of components in each row or in each circular. or number of columns in a matrix:
α_i	: the number of operating states of the linear system[5] with i failed components:
β_i	: the number operating states of the circular system, with i failed components;
P,q	: reliability and unreliability of a component, p+q=l:
$R_L(m.n)$	:the reliability function of linear connected (1,2) or (2,1) out of (m,n) : F lattice system:
$R_c(m,n)$	: the reliability function of circular connected (1,2) or (2,1) out of (m,n) : F lattice system.

§3. Linear System.

We can see that the reliability function of linear connected (1,2) or (2,1) out of (m,n) : F lattice system. is given by

$$R_l(m,n) = \sum_{i=0}^{[\frac{mn}{2}]} \alpha_i \; q^i \; p^{mn-i}$$

where α_i is given in the notation (§§2.2). We demonstrate now how to compute the coefficient α_i, $0 \leq i \leq \lceil mn/2 \rceil$.

As mentioned in §§2.1 the linear system can be represented by an (m.n) matrix of binary digits . But according to assumption (5), if the component number ((i-l)n+j) is failed then the four components that are in positions (i-1,j), (i,j-1). (i,j+l) and (i+l,j) must be operating. Since otherwise the system is failed. and its matrix representation is not accepted.

Also, elements are called *forbidden elements* .More formally, $\forall\, i,j$ *if* $m_{ij} = 1$ then all four elements $m_{(i-1)j}$, $m_{l(j-1)}, m_{i(j+1)}$ *and* $m_{(i+1)j}$ must be equal to "0".

Obviously, there is one-to-one correspondence relation between the linear and circular systems and the class of matrices defined above. In fact this relation plays an important role to simplify a solution for the required problem and make it possible by using an algorithmic technique to obtain α_i 's. Since α_i can be interpreted as the exact number of (m,n) 0-1 matrices having i ones provided that no two or more consecutive ones appear in any row or any column (accepted matrices).

The algorithmic method is used to design a successive enumeration of accepted (m,n) 0-1 matrices. The suggested method can be thought as a bottom-up approach, since it begins with the matrix having mn O's and ends with the matrix having $\lceil mn/2 \rceil$ l's in which no two or more l's appear consecutively at any row or any column.

Now. we explain the steps of the algorithm. used to create all accepted (m,n) matrices. The matrices will be created in colexicographic order w.r.t. its rows. To take into account the forbidden places during creating matrices, we redefine the entries m_{ij} of the representation matrix M of a linear connected (1,2) or (2,1).out of (m,n) : F lattice system as follows:

a) If an operating component is in position (i,j) in the system. then put m_{ij} = "O";

b) If a component in position (i,j) is failed then put m_{ij} "l";

c) If m_{ij+1}=1. l≤ j <n-1;then replace m_{ij} = "0" by any marker or artificial number. say "2".

Note that the condition (c) is necessary to prevent creation of redundant matrices having two or more consecutive ones; so position (i,j) is the forbidden place.

For producing accepted matrices, we consider each row of a matrix as a tuple of n binary digits. Thus when any element of any tuple varies. a matrix will be varied. To avoid repetition of some matrices during creation, we construct tuples in the colexicographic order. Since the possibility of two or more ones is not occurred, the total number, L_n, of distinct accepted n-tuples is given by the following recurrence relation:

$$\left.\begin{aligned} L_n &= L_{n-1} + L_{n-2} \quad ; \; n \geq 2 \\ L_0 &= 1 \text{ and } L_1 = 2. \end{aligned}\right\} \qquad (2)$$

This relation is easily proved. Since we have two possibilities, if "O" appears in nth place. then the remaining (n-1) places will have L_{n-1} distinct accepted tuples. While if "l" appears in the nth place, the (n-l)st place is forbidden and does not take a 1, so the total number of filling (n-2) places under the same restriction is L_{n-2}

The illustrated observation helps us to present a recursive algorithm. The following algorithm deals with a matrix M of order (m,n) as a one dimensional array of size m. Each component of it is an n-tuple. Let k be an indicator to the position of a current n-tuple that must be modified. The algorithm works as follows. For certain k. step (1) initializes the tuple M[k], namely put all n digits equal zero. Step (2) is the forward step that is used to increase the value of k and go forward to the deepest level of M. We reach to step (3) either when km from step (2) or step (9). or k≠ m from step (10). Then steps from (3) to (7) are used together to create the next accepted tuple of the current tuple M(k] in colex. list. If such a tuple exists. then all other related results are modified at the same time. see step 8. Via step 9. the algorithm repeats this procedure until no such tuple exists. i.e. M[k] will be the last n-tuple in the colex. order. 1.e. M[k] = (2,1,2, ...,2,1). In such a case if k=l the work will be halt. Otherwise the backward step executes, k decreases then go back toward the top of M. When exists a new largest (k <m) at which M[k] is not (2,1, ...,2,1), we construct the next successive M(k] and go back to step (1) to initialize all tuples from M[k+l) to M[m); then all processing will be repeated again. The task of the algorithm halts when k=l and every row reach to the last form.

The description of the algorithm is mentioned in the next steps. Here, we introduce some relevant identifiers and data structures that will be used:

M :array of two dimensional;

(m,n) :the order of a matrix M; bounded by ability of computer device;

k :the position of the current tuple; l≤k≤m;

M[k): the kth tuple of the matrix M;

α[i]: α_i.

α : one-dimensional array whose components are α[i]; where varies from 0 to $\lceil mn/2 \rceil$;

Z[i] :the exact number of appeared l's at the ith tuple; and Z[i] is bounded by $\lceil n/2 \rceil$;

Z: one-dimensional array whose components are Z[i]; 1 im;

Last :one-dimensional array whose components are Last[i];l≤i≤m.

Last[i] is taken true if M[k] is the last member of the colex. list and false otherwise.

Step 0 : **"initial Step"**
Enter the values of m and n:
Put α[i] ←0 for all i : and
Put k ←1;
step 1 : Create the first member in the colex. list of M[k];
this done by putting for O in , M[k,i] ←0;
Z[K] ← 0; and last[kJ ← false;
step 2 : **"Forward Step"**
If k < m then $k \leftarrow k + 1$ and go to *Step 1 ;*
Step 3 : **"Get Next Matrix"**
Put j←1;
Step 4 : Search from M[k,j] to M[k.n] by increasing j to the place of the first "0":
If no such place then put Last[k] ← true: go to *Step 10 ;*
Step 5 : Replace this "0" by "1" if M[k-1,j] = 0;
and put Z[k] ← Z[k] + l ; Otherwise go to *Step 4;*
Step 6 : Remove all ones exist in places 1 to j-1;
simultaneously decreasing Z[k] ;
Step 7 : Remove all 2's exist in places 1 to j-2;
and Put M[k,j-1$_k$] ← 2;
Step 8 : Count ← $\sum_{i=1} z[i]; \alpha[Count] \leftarrow \alpha[Count] + 1;$
Step 9 : If (Last[k] = false) then
if (k=m) then go to *Step 3;* else go to *Step 2 ;*
Step 10 : **"Backward Step"**
If k > 1 then $k \leftarrow k - 1$; go to *Step 3;*
Step 11 : For 0≤i≤[mn/2] output α[i];
Step 12 :Stop.

The execution time of the above algorithm depends on the exact number of accepted M[k] in the colex. list. The number of accepted members of M[k]-list is given by the recurrence relation (2). Since, for each row of (m,n) matrix, we have at most L_n accepted tuples. Then, the running time, RT, depends on m, n and is bounded by $(L_n)^m$. Unfortunately, the number L_n followed the Fabinacci's sequence. At which when n increases, the value of L_n exceeds more rapidly and so RT. So, we must take reasonable values for both n and m that suit to the ability of a computer device and of the mantissa of computer's memory. Some results of implementing algorithm are given in Tables (1)-(6). In fact, the computing time of all α_i's in those tables took a little time, not exceeding few minutes.

§4. Circular System.

Obviously, the reliability function of circular connected

(1,2) or (2,1) out of (m,n) : F lattice system. is given by

$$R_C(m,n) = \sum_{i=0}^{m[\frac{n-1}{2}]} \beta_i \, q^i \, p^{mn-i}, \qquad (3)$$

Where β_i is given in the notation (§§2.2)

The circular system may be treated as a linear case, by making a cut between the first and nth ray, and unfold the m circles. obtaining an (m,n)-matrix. Clearly, in a such matrix. all components existing in the first and last column are in fact consecutive. Therefore, both m_{i1} and m_{in} cannot fail at the same time for an operating state. This diggers from the linear case at which these components are not consecutive.

Now we explain how to compute $\beta_i, 0 \leq i \leq m[\frac{n-1}{2}]$.. According to the 1-1 corresponding relation between the circular (m.n) systems and the class of (m,n) 0-1 matrices, β_i is viewed as the number of

(m,n) matrices having i ones provided that no two or more consecutive ones appear in any row or any column given that for each row if a one appears in the first column. it does not appear in the nth column and vice versa.

The above observation leads to a simple updating to the algorithm give in the linear case. This modification is done whenever a "1" appears. at the first time, in the last position in any n-tuple. Since the ordering of getting all accepted tuples is the colex. The advantage of such ordering is fixing a digit "1" when occurred in the nth place. for the remaining n-tuples that still not created. When arriving to this point, "1" must be forbidden to appear in the first place beside the (n-l)st one. This is the only main difference between the treatments of the two method for the two systems.

The slight modification of the previous algorithm relevant to the circular case is illustrated in the following steps. Note that all missing identifiers, data structures. and steps are the same as that mentioned in §3. It is sufficient now to present only the new variables and updating steps.

$\beta[i]$: β_i;

β :one-dimensional array whose components are β[i]; $0 \leq i \leq m\,[(n-1)/2]$

Step 0: **"initial Step"**
Enter the values of m and n;

Put $\beta[i] \leftarrow 0$ for all i;
and Put k ← 1;

Step 7 : Remove all twos exist in places 1 to j-2; and
If j = n then Put M[k,l] ← 2 and M[k,j-1) ← 2
else Put M[k,j-1) ← 2;

Step 8 :Count $\leftarrow \sum_{i=1}^{k} z$ [i];β [Count) $\leftarrow \beta$ [Count]+l;

step 11 :For $0 \le i \le m\left[\frac{n-1}{2}\right]$output$\beta$ [i];

From the above explanation, we can easily deduce the total number C_n of distinct accepted n-tuples in a circular case by using the following recurrence relation:

$$C_n = L_{n-l} + L_{n-3}; \quad n \geq 3 \qquad (4)$$

$$C_1 = 1 \text{ and } C_2 = 2$$

Consequently, the circular algorithm is guaranteed to execute in O(C_n^m). Of course the running time RTC in this case is less than RTL. Although the problem of increasing RTC still exists but with less complexity. So, here also, we deal with reasonable bonded m cycles and n rays.

Tables (1)-(6) illustrate some exact counts of numbers β_i; $0 \le i \le m\left[\frac{n-1}{2}\right]$ for same m and n. Finally, let us remark that the counts of α_i and β_i in case of (m.3) that are given in Tables (1) and f2) coincide with the results appeared in [2. 1997}.

At this point we conclude that:

1) From relations (2) and (4), $\forall i\ \alpha_i \geq \beta_i$ whenever the dimension (m.n) of two systems are the same.

2) The calculations of a linear system of (m,n) dimension are equal to that of (n,m). While in every circular case the counts Of β_i in (m.n) and (n,m) matrices are varied: *e.g.* see β_i's that are given in the 1st and 2nd columns of tables (2)-(6),

References

(l) T. K. Boehme, A. Kossow, and W. Preuss, "A Generalization of Consecutive-k-out-of-n : F Systems", IEEE Trans. Reliability, vol. 41, No. 3, (1992) Sep, pp. 451-457.

[2] N. Mokhlis, E. M. EL-Sayed and G. Youssef, "Reliability of a connected (1,2) or (2,1) out of (n,3): F lattice system", to be presented in International Conference on Combinatorial Methods and Applications to Probability and Staistics. Mc Master University, Hamilton, Ontario, Canada, 25th - 28th June 1997.

Results.

Table (1)

m,n / i	3,3	4,4	5,5	6.6	3.3	4.4	5.5	6,6
	β_i				α_i			
0	1	1	1	1	1	1	1	1
1	9	16	25	36	9	16	25	36
2	21	92	255	564	24	96	260	570
3	12	240	1385	5076	22	276	1474	5248
4		302	4400	29208	6	405	5024	31320
5		192	8500	113316	1	304	10741	127960
6		72	10125	305138		114	14650	368868
7		16	7415	579780		20	12798	763144
8		2	3245	784980		2	7157	1143638
9			780	763036			2578	1247116
10			80	537852			618	991750
11				280176			106	576052
12				111570			14	245030
13				35460			1	76716
14				9222				17834
15				1940				3120
16				318				416
17				36				40
18				2				2

Table (2)

m,n / i	3x4	4x3	3x4	3x5	5x3	3x5	3x6	6x3	3x6
	β_i		α_i	β_i		α_i	β_i		α_i
0	1	1	1	1	1	1	1	1	1
1	12	12	12	15	15	15	18	18	18
2	46	45	49	80	78	83	123	120	126
3	68	60	84	190	171	215	408	372	442
4	40	24	61	210	156	276	705	558	840
5	12		18	105	48	174	642	384	880
6	2		2	20		53	308	96	504
7						9	84		158
8						1	18		28
9							2		2

Table (3)

m.n	4×5	5×4	4×5	4×6	6×4	4×6	4×7	7×4	4×7
i	β_i		α_i	β_i		α_i	β_i		α_i
0	1	1	1	1	1	1	1	1	1
1	20	20	20	24	24	24	28	28	28
2	155	154	159	234	232	238	329	326	333
3	600	588	652	1208	1176	1276	2128	3068	2212
4	1255	1208	1502	3621	3430	4072	8372	7896	9091
5	1450	1384	1998	6540	6000	8052	20958	19016	24238
6	920	926	1537	7218	6472	10010	34083	29666	42864
7	300	396	678	4908	4480	7830	36330	30696	50726
8	40	112	170	2112	2110	3846	25480	21768	40235
9		20	24	624	696	1176	11760	11016	21356
10		2	2	144	160	226	3528	4078	7578
11				24	24	28	644	1108	1808
12				2	2	2	56	216	294
13								28	32
14								2	2

Table (4)

m.n	4×8	8×4	4×8	4×9	9×4	4×9
i	β_i		α_i	β_i		α_i
0	1	1	1	1	1	1
1	32	32	32	36	36	36
2	440	436	444	567	562	571
3	3424	3328	3524	5160	5020	5276
4	16740	15790	17791	30213	28552	31660
5	54064	48992	60168	120078	109096	130318
6	118352	102276	140050	333540	288710	379247
7	177920	146544	227456	658134	539720	793690
8	184870	147090	259289	930753	724344	1205457
9	133312	106208	207792	948084	710348	1334414
10	67136	56824	117030	697932	520146	1078279
11	24032	22992	46332	372708	290888	636276
12	6432	7086	12950	145140	126360	274450
13	1408	1648	2584	41328	42984	86864
14	256	280	370	8424	11422	20334
15	32	32	36	1116	2332	3544
16	2	2	2	72	352	454
17					36	40
18					2	2

Table (5)

m,n	5×6	6×5	5×6	5×7	7×5	5×7
i	β_i		α_i	β_i		α_i
0	1	1	1	1	1	1
1	30	30	30	35	35	35
2	381	380	386	532	530	537
3	2692	2670	2806	4634	4580	4773
4	11718	11520	12792	25732	25115	27381
5	32958	32000	38438	96005	92075	107004
6	61325	58550	78052	247590	232075	293409
7	76392	71230	108354	448308	408200	573797
8	64242	57560	103274	574847	504420	807161
9	37010	30430	67664	524552	437825	820006
10	15138	10110	30550	341642	264580	602827
11	4686	1920	9574	158830	108845	321496
12	1156	160	2104	52220	29080	125145
13	222		324	11739	4560	36053
14	30		34	1652	320	7895
15	2		2	112		1363
16						188
17						19
18						1

Table (6)

m,n	5×8	8×5	5×8	5×9	9×5	5×9
i	β_i		α_i	β_i		α_i
0	1	1	1	1	1	1
1	40	40	40	45	45	45
2	708	705	713	909	905	914
3	7336	7240	7500	10923	10775	11112
4	49642	48310	51991	87273	84855	90447
5	231936	221350	251354	491166	468075	522528
6	772156	718450	875407	2013729	1869800	2217382
7	1866200	1682320	2239218	6140727	5520715	7060833
8	3311104	2871750	4255370	14111991	12202430	17101603
9	4341976	3591580	6047712	24645762	20345765	31775340
10	4227448	3291645	6450890	32888133	25689520	45529930
11	3069208	2198530	5175586	33661260	24573185	50470829
12	1672870	1055520	3130171	26510817	17742910	43375231
13	694008	354520	1432508	16123041	9583745	28958479
14	225396	79120	499425	7602885	3806965	15062845
15	60016	10560	133914	2789319	1079000	6134235
16	13680	640	27841	794754	206640	1971567
17	2640		4466	172989	24000	506207
18	396		534	27441	1280	105535
19	40		44	2844		18162
20	2		2	144		2588
21						295
22						24
23						1
24						