CONGRESSUS NUMERANTIUM

WINNIPEG, CANADA

An Exact Enumeration of the Reliability of Some Linear & Circular Connected X-out-of-(n,m) :F Lattice Systems.

N. Mokhlis and S. M. Khamis.
Department of Mathematics,
Faculty of Science,
Ain Shams University,
Cairo, Egypt.

Abstract.

An exact enumeration of the reliability of some linear and circular connected X-out-of-(n,m) : F lattice systems is introduced. The reliability of (r,s): F lattice systems is presented via using a recursive algorithmic method. Then the reliability of (r,s)-or-(s,r)-out-of-(n,m) F : lattice systems is obtained by some development of the algorithm used in the case of (r,s)-out-of-(n,m) :F lattice systems. The algorithm depends on the relation between the representation of the specified (n,m) : F lattice systems and the class 'of (n,m) matrices having 0-1 entries. The suggested procedure counts all possible failure states of the System.

Key words.

Linear, circular connected X-out-of-(n,m) :F Lattice Systems.

§Introduction.

The definition of linear and circular connected X -out-of-(n,m) : F lattice systems is introduced by Boehme et al [1]. A linear (circular) connected X-out-of-(n,m) : F lattice system fails whenever at least one set of connected X failed components occurs. In linear systems, the components are arranged in n rows and m columns. For circular systems, the components are arranged in n circles (centered that at the same point) with m rays, Each ray contains m components .

In this paper, we are interested in linear (circular) (r,s)-out-of-(n,m), as well as (r,s)-or-(s,r)-out-of-(n,m) : F lattice systems. Exact reliability formulas for connected X -out-of- (n,m) : F lattice systems are given for some special cases in [1] and [4] . A recursive algorithm for computing the reliability of (1,2)-or-(2,1)-out-of-(n,m) is introduced in [2]. Bounds on the reliability of connected (r,s)-out-of-(n,m) : F lattice systems with identical and nonidentical components are independently introduced in [3] and [5]. Approximation formulas and bounds for (r,s)-or-(s,r)-out-ot:.(n,m) : F lattice systems with identical and nonidentical components are given in [6]. However, it is difficult to obtain an exact formula for the reliability of such systems for any r, s, n and m.

In the present paper, we give a recursive algorithm for computing the exact reliability of linear (circular) connected (r,s)-out-of-(n,m) : F lattice systems with identical components. A simple development of this algorithm produces a recursive algorithm for computing the reliability of linear (circular) (r,s)-or-(s,r)-out-of-(n,m) : F lattice systems. The relation between the representation of the specified {n,m) : F lattice systems and the class of {n,m) matrices having 0-1 entries, is used in designing the algorithm .

§ 2. Assumptions and Notation.

§§ 2.1 Assumptions.

1) For the linear system, we have m components in n rows.

2) For the circular system, we have n circles, each having m components.

3) Each component and the system are either operating or failed.

4) The components are s-independent and identical.

5) We assign a "0Type equation here." for an operating component and a "1" for a failed component.

This leads to represent the system by (n,m) 0-1 matrix, say A, which is

defined as:

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

6) A is called a useless matrix if and only if it represents a failed state of the system. This means that A contains a submatrix of order X in which all entries are 1's, where

$$X \equiv \begin{cases} (r,s) \text{ for the } (r,s)-out-of-(n,m){:}F \text{ lattice systems} \\ \text{either } (r,s) \text{ or } (s,r) \text{ for the } (r,s)-or-(s,r)-out-of-(n,m){:}F \\ \text{lattice systems.} \end{cases}$$

Otherwise A is called a useful one.

§§ 2.2 Notation.

X_1 connected (r,s)-out-of-(n,m).

X_2 connected (r,s)-or-(s,r)-out-of -(n,m).

$\propto_i\ \gamma_i$: the number of operating and failed states of the linear X_1 : F lattice system with i failed components, respectively.

$\beta_i \delta_i$: the number of operating and failed states of the circular X_1 : F lattice system, with i failed components, respectively.

$\propto'_i\ \gamma'_i$: the number of operating and failed states of the linear x_2 : F lattice system with i failed components, respectively.

$\beta'_i \delta'_i$: the number of operating and failed states of the circular x_2 : F lattice system with i failed components, respectively.

p, q : reliability and unreliability of a component, p+q=l.

λ : implies L (linear) or C (circular).

R_λ ((r,s);(n,m);p) : the reliability function of an λX_1 : F lattice systems, with component reliability p.

R_λ ((r,s)-(s,r);(n,m);p) : the reliability function of an λX_2 : F lattice systems, with component reliability p.

§3 The Reliability of x_1 : F Lattice Systems.

The reliability is calculated by means of a recursive algorithm. The algorithm enumerates the failed states of the system, i.e., the number of useless matrices. So, the useless matrices are considered the accepted ones to the specified algorithm.

§§3.1. The Linear System.

There is a well-known 1-1 correspondence relation between (n,m)-matrices of 'O' or '1' entries and the linear systems of nm components ordered in n rows and m columns, see e.g. [1]. [2] and [7]. Depending on this relation and computing numbers of failed states of the system, we design an efficient recursive algorithm.

The algorithm is based on the following two combinatorial objects:

(1) Create all compositions of z integer into n parts in a colexicographic (colex.) ordering list.

(2) Create all m-tuples of 0-1 entries in a colex. ordering list.

The required 0-1 (n,m)-matrices that include at least (r,s)-submatrix of l's entries, can be created by using (1) and (2) according to the following stages.

First stage, create a composition of z integer into n parts by using line 40) or 190) of the following algorithm. Not all compositions are needed but only those having no parts exceed than m, the width of a system. This is clear because no entry of a matrix exceeds than one. Also, all compositions with less than r consecutive parts of s values are not required . The redundant compositions are excluded by executing tests which are explained in lines 70) and 80). The accepted (unredandant) compositions that are not excluded will be used in the next stage to produce required (n,m)-matrices of 0-1 entries.

Second stage, how to produce 0-1 (n,m)-matrices that are related to a current accepted composition, COMP. Let the ith part of COMP be the number of l's that appear in ith row of a matrix A, ($l \leq i \leq n$). Each row can be considered as an m-tuple of 0-1 entries and so A is also viewed as a collection of n m-tuples. Thus, the number of I-entries of each tuple doesn't vary until COMP is varied. Hence creation of all related matrices to COMP depends only on changing the distributions of 1-entries of each m-tuple according to colex. ordering.

At the beginning, the algorithm creates the first matrix that is related to COMP. This is easily done via getting the first colex. n m-tuples which occupy the rows of a matrix A Consider k_1 to be the 1st part of COMP. Then to obtain the first colex. m-tuple of the 1st row of A, we put 1 for the first k_1 leftmost columns of 1st row of A. The same task is repeated to all rows of A This operation is explained in details at lines 100)-120) below. ·

Next, assume that we have an accepted (useless) matrix A which represents a failure state of the system. To get a next consecutive matrix to A, the algorithm takes the following major steps:

i) Search for the largest row index of A, say i ($i \leq n$) at which an m-tuple is not the last one of the colex. list of 0-1 m-tuples with a fixed number of l's, (see details at lines 170) and 180)).

ii) If there is no such i, all related matrices to the current COMP are obtained. And hence if it is possible, we repeat all processes by getting a new next composition according to line 150) of the suggested algorithm. The processes halt if there is no new composition.

iii) If there exists such i, the algorithm works to obtain the next colex. distribution of 0-1 entries of m-tuple of ith row of A. The details of this task are given at lines 150) and 170).

Unfortunately, not all constructed matrices represent the required failure states of the system. So, we don't take some of these matrices into our account. Necessarily, check whether or not a current matrix, A, contains at least one (r,s)- submatrix of l's entries. This test can't be used unless the consecutive property of a current composition, COMP is determined.

Executing step 90) leads to decided whether COMP contains at least r consecutive s parts or not. In case of COMP have such consecutive property, the algorithm tests whether or not A has an (r,s)-submatrix of l's as follows.

Obviously, the parts of COMP are viewed as the sum of A's rows entries. Also the sums of A's columns entries are viewed as parts of another composition say COMP 1. If COMP 1 does not contain at least s consecutive r- parts, we don't take A into account. Since it is a useful matrix. Otherwise, A is still not necessarily a useless one. So, the matrix-test can be interpreted as searching about a first (r,s)-submatrix of l's entries. To do so, we determine the following two boundaries for beginning an optimal search.

* The top boundary, Tb, is the smallest index of A's row which is the first position at which begins r consecutive s-parts of COMP.

* The left boundary, Lb, is the smallest index of A's column which is the first position at which begins s consecutive r-parts of COMP l.

According to these boundaries the test can be accomplished in a more efficient way. We imagine, there exists a moving block of dimension r x s whose horizontal sides are i & i+r-1 while vertical sides are j & j+s-1. According to changing i, j where $Tb \leq i \leq n+1-r$ and $Lb \leq j \leq m+1-s$, the moving block assigns new (r,s)-submatrices. Taking first i=Tb and j=Lb, the algorithm checks whether or not all entries of assigned submatrix are l's. If this condition is satisfied the matrix- test halts without checking the rest part of A. A represents a useless matrix and the counter of failed states of the system is increased by one. Otherwise, take another value of i, j through any nature order and then repeat the previous task again until no possible submatrix of order (r,s) can be tested. In this case also the test procedure halts but A is not required.

The sequence for creating compositions, testing , checking consecutive property , creating recursively all related matrices to a current composition and finally, during creation of matrices, testing whether or not a current matrix is required, is given by the following algorithm. The following data-structures are used :-

```
A              : array[l..n] of
                   record  L          : array [1..m] of integer;
                           Last Row   : boolean
                   end;
SumOfRow,COMP    : array[l ..n] of integer;
SumOfCol         : array[l..m] of integer;
Failure-System   : array[1..MaxSize] of real;
```

The Algorithm for Creating Useless matrices and Counting (γ_i)

```
Begin

10)  Enter the values of r, s, m, and n.
20)  For z←rxs to mxn Do
30)   Begin
40)      Get the first Composition COMP of z into n parts;
         Put COMP[l]←Z, COMP[i]←0    $2\leq i \leq n$
         LastComp←false.

50)     While not (lastComp) Do
60)      Begin
70)         Test whether or not the current COMP is valid  w.r.t the width
             of a given system.
             This is done as :
             for i ⟵ 1 to n Do
             If COMP[i] > m  then go to 190).
80)       Check_Composition(r,s);
          Check if a current COMP having r consecutive parts, each of which is
           not less than s.
```

```
        If the condition is not satisfied then goto 190);
 90)     Put i ←l
100)   Initial Step;
       Put all entries of ith Row equal zero
110)     Get_First_Colex_Distribution_Of_Row_i;
        This is done as:
        For k←1 to COMP[i] Do Put A[i].L[k]←1 ;
                                   Calculate SumOfCol[k];
        Put A[i].Last_Row←false;
        Put SumOfR.ow[i] ←COMP[i];
120)  Forward Step;
      If (n ≠1) and (i < n) then i←i+l; go to 100)
130)   Repeat
140)     Test whether or not a matrix represents a failed state of linear $X_1$: F
         lattice system;
          This is done as :
          Test whether or not a current matrix including at least (r,s)-submatrix
           whose entries are ones.
           If the condition is true
           then Failure-System[z] ←Failure-System[z] + 1.
150)    Gct_Ncxt_CoLcx_Distribution_Of_Row_i;
        This step executes as follows:
        Search for the first zero's location, say j, behind it at least left one;
        If j does not exist
        Then Put A[i].Last_Row←true
         Else Put A[i].Lfj] ←l;
              Put P← (Sum of ones left to j) - 1
              For k←1   to P  Do  A[i].L[k] ←1
              Erase all l's from kth location to G-1)st location Update the values
             of SumOfR.ow and SumOfCol
160)     Until (A[i].Last_Row);
170)     if ith row has a next distribution
        Then Get_Next_CoLex_Distribution_Of_Row_i; go to 120)
180)  Backward Step,
           if i ≠1 then Put i←i-1; go to 170)
190)     Get the next Composition COMP of z into n parts;
          This is done as follows:
             Search for a first left location of non-zero entry; say j if there is no
             j≤m then Last-Comp←true;
             otherwise
                  Put COMP[j+1] ←COMP[j+ 1]+1;
```

```
                    Temp    ←COMP[j];
                    COMP[j] ←0;
                    COMP[l] ←Temp-1;
200)    End;
210) End.
End.
```

§§3.2.The Circular system.

The previous version of the algorithm is used for solving the problem in a linear case only. Fortunately, this version can easily be developed to solve the problem in the circular case by modifying the linear matrix-test.

A circular connected (n,m)-system, also can be represented by 0-1 (n,m) matrix, where the first and last columns are considered consecutive. This means that the matrix representation can be viewed as a cylinder. This differs from the linear case, where the first and last columns arc not consective.

Thus all useless matrices which represent failed states of the linear (n,m)-system are also useless matrices of the circular (n,m)-system. While the (n,m)-matrices which represent operating linear system are not necessarily operating circular ones. Therefore, we check again w.r.t. circular cases only those matrices that are rejected via applying the linear matrix-test. This additional test is done as follows. Clearly, We do not need examine a whole current matrix A. The only part of A that is needed to complete the circular matrix-test begins at column m+2-s and cyclically ends at column s-1. We imagine that there exists a moving stripe whose vertical boundaries are (m+l+i-s,i), l≤i≤s-1. For i=l, (m+2-s,1) is the first stripe. The algorithm checks whether or not this stripe of width s having r consecutive rows whose entries are l's. If this is found the algorithm halts and A is a useless matrix. Otherwise the algorithm moves a stripe to the next i and repeat again the search until there is no other stripe. Then hence the algorithm is finished and the matrix is not taken into account. Via adding this development between lines 140) and 150) of the mentioned algorithm in §§ 3.1 we obtain a complete recursive algorithm for solving the problem of finding reliability of either linear or circular (r,s)-out-of-(n,m) for any integers m, n, r and s.

§§ 3.3. Calculations Of $\propto_i$'s and β_i's

To find the reliability of linear or circular X_1 : F lattice system for some integers n, m, r and s, we implement the algorithm by any Language having the recursive facility such as Pascal or C. Then execute a program to compute $\gamma_i's$ and $\delta_i\,'s$ for some values of n, m, r and s. Through these results, one can easily calculate $\propto_i$'s and β_i's by using the following equations:

$$\alpha_i = \binom{mxn}{i} - \gamma_i; \qquad 0 \le i \le mn \quad (1)$$

$$\beta_i = \binom{mxn}{i} - \delta_i\ ; \qquad 0 \le i \le mn \quad (2)$$

Note that : the values of γ_i and δ_i for 0 ≤i ≤rs-1 equal zeros. The calculated numbers $\propto_i$'s are the coefficients of the reliability function of a linear system, which is given by:

$$R_L\big((r,s);(n,m);p\big) = \sum_{i=0}^{m\,n} \propto_i\ p^{m\,n-i} q^i \qquad (3)$$

Also, the numbers β_i's are the coefficients of the reliability function of a circular system. It is give by :

$$R_c\big((r,s);(n,m);p\big) = \sum_{i=0}^{m\,n} \beta_i \; p^{m\,n-i} q^i \qquad (4)$$

As an illustration, we introduce the numerical example of output results of the program in case of connected (2,2)-out-of-(4,3): F lattice system in both cases. The first and second columns of the following Table(1) are the program's results while columns (3) and (4) are obtained via equations (1) and (2) respectively.

Table (l) The Coefficients of R_λ ((2,2),(4,3),p)

i	Υ_1	δ_1	α_1	β_1
0	0	0	1	1
1	0	0	12	12
2	0	0	66	66
3	0	0	660	660
4	6	9	489	486
5	48	72	744	720
6	161	240	763	684
7	290	420	502	372
8	301	408	194	87
9	182	220	478	440
10	63	66	3	
11	12	12		
12	1	1		

Substituting p = 0.7 and the above values of $\alpha_i' s$ and $\beta_i' s$ respectively, in equations (3) and (4), we have

R_L((2,2);(4,3);0.7) = 0.956956 and R_C ((2,2);(4,3);0.7)= 0.944647

Some results of executing this algorithm are given in the Appendix (tables 4 and 5). The results in case of a linear and circular connected (2,2)-out-of-(4,4) and (2,2)-out-of-(4,3) : F-lattice system, that are presented in table 4, concide with those in (1996, [3]).

§4 The Reliability of Linear (circular) x_2: F Lattice Systems.

Computing the reliability of a linear or a circular X_2 : F lattice system is a more complicated problem. Generally, for any r, s, n and m neither excat formulae nor recursive algorithms exist for computing the reliability of such systems.

Fortunately, the above algorithm can be extended to solve the problem of X_2 case. The extension depends on two major modifications. The first one is concerned with determination of whether the current composition has at least r consecutive s-parts, or at least s consecutive r-parts. While the second one depends on testing whether a matrix A contains at least (r,s)- or (s,r)-submatrix of 1's entries.

§§4.1 The Linear System

As mentioned before, not all compositions that are constructed by the given algorithm in §§ 3.1 are required to create useless matrices. Here we use only those having at least k consecutive L parts where (k =r , L=s or k=s, L=r). From the second stage (§§ 3.1), we create all related matrices to current accepted composition. Not all those matrices represent the failed states of the system. So, we must check whether or not a current matrix, A, contains at least one (r,s)- or (s,r)- submatrix of l's entries. This test can't be used unless the consecutive property of a current composition is determined. The two required types of a consecutive property are :

P1 : A composition has a subcomposition having at least r consecutive s-parts.

P2 : A composition has a subcomposition having at least s consecutive r-parts.

- The algorithm classifies compositions according to Pl and P2 as demonstrate in the following updated steps.

```
75)     PropertyNumber $\leftarrow$ 0;
80)     Check_Composition(r,s);
          Check if a current COMP having r consecutive parts, each of which is
          not less than s.
          If the condition is satisfied then put PropertyNumber $\leftarrow$ P1;
85)     Check_Composition(s,r);
        Check if a current COMP having s consecutive parts, each of which is
        not less than r.
        If the condition is not satisfied then goto 190)
        else if (PropertyNo = 0) then put PropertyNumer $\leftarrow$ P2;
                                    else put PropertyNumber $\leftarrow$ P3;
```

Executing these steps leads to determine whether a current composition, COMP having either property P1, P2 or both. If COMP has only P1, the algorithm tests whether or not A has an (r,s)-submatrix of 1's. But, in the case of COMP has only P2, the test executes for searching whether or not A has an (s,r)-submatrix of 1's. While if COMP has P1 and P2, we must first search for finding (r,s)-submatrix. If it does not exist, we repeat again searching for a submatrix of order (s,r). Thus, step 140) of the above algorithm in § 3.1 is modified to be:

```
140)    Test whether or not a matrix represents a failed linear $X_2$:F lattice system;
        This is done as:
         Case PropertyNumber Of
         'P1': Test whether or not a current matrix including at least (r,s)-submatrix
                 whose entries are ones.
                 If the condition is true
                 then Failure-System[z] $\leftarrow$ Failure-System[z] + 1.
         'P2': Test whether or not current matrix including at least (s,r)-submatrix
                 whose entries are ones.
                 If the condition is true
                 then Failure-System[z] $\leftarrow$ Failure-System[z] + 1.
         'P3': Test whether or not a current matrix including at least (r,s)- or (s,r)-
                 submatrix of 1-entries.
                 If either one of the two conditions is true
        then  Failure-System[z]←Failure-System[z] + 1.
    End
```

• Entering the suggested modifications into the algorithm in §§ 3.1. We can obtain the number of useless (n,m)-matrices with i l 's entries rs≤i≤nm . The numbers of these useless matrices are equivalent to γ_i; i number of failed components. Clearly, it is easy to obtain the coefficients of R_L ((r,s);(n,m);p) which is given by

$$R_L\big((r,s)-(s,r);(n,m);p\big)=\sum_{i=0}^{m\,n}\left(\binom{mxn}{i}-\gamma_i\right)p^{m\,n-i}q^i. \qquad (5)$$

§§4.2 The circular System

Similar to X_1 case, we can extend the algorithm of a linear X_2 case (§§4.1) to calculate the coefficients of R_c ((r,s)-(s,r);(n,m);p). This is done by determining the number of useless matrices that represent the failed states of circular systems. This is achivied via checking again if the rejected matrices w.r.t. the linear system are also rejected w.r.t. circular case. To execute the circular test, the algorithm determines as demonstrated in lines 75), 80) and 85), whether a current composition that is used to construct A, have both consecutive properties P 1 and P2 or have exactly one of them. The following step explains how to decide whether a current matrix A, have either (r,s)- or (s,r)- submatrix of l's .

```
145) If the system is a circular
 -    Then Begin
              Case PropertyNumber Of
         'Pl': Test whether or not a current matrix includes at least one submatrix of
               order at least (r,s) with entries ones. This test is done at the region of
               A beginning at the column m+2-s and cyclically ending at column s-1.
```

```
        If the condition is true
        Then Failure-System[z]←Failure-System[z] + 1.
  'P2': Test whether or not a current matrix including at least one submatrix of
        order at least (s,r) with entries ones. This test is done at the region of A
        beginning at the column m+2-r and cyclically ending at column r-1.
        If the condition is true
        Then Failure-System[z]←Failure-System[z] + 1.
  'PJ': Test whether or not a current matrix A contains at least one (r,s)-
        submatrix of 1's entries as mentioned in the above step 'PI'. Then if A
        does contain such submatrix, test again whether or not A includes at
        least one (s,r)- submatrix of 1's entries as mentioned in the above step
        'P2'. If either one of the two conditions is true then
        Failure-System[z]←Failure-System[z] + 1.
   End
 End.
```

The results of implementing the circular version are the numbers of useless matrices which are equivalent to δ_i's, where $rs \le i \le nm$. These numbers are used to determine the coefficients of the reliability function of a circular X_2:F lattice system, which is given by

$$R_C\big((r,s)-(s,r);(n,m);p\big) = \sum_{i=0}^{m\,n}\left(\binom{m x n}{i} - \delta_i\right)p^{m\,n-i}q^i \qquad (6)$$

As an illustration, we introduce the numerical example of output results of the program in case of connected (1,2)-or-(2,1)-out-of (5,3) :F lattice system in both cases.

Substituting p = 0.9 and the following values of γ_i's and δ_i's respectively in equations (5) and (6), we have

R_L ((1,2)-(2,1);(5,3);0.9) = 0.830029 and R_C ((1,2)-(2, 1);(5,3);0.9)= 0.8006668

Table(2) The Coefficients of R_{λ}.((1,2)-(2,1);(5,3);p)

i	γ_i	δ_i	α_i	β_i
0	0	0	1	1
1	0	0	15	15
2	22	27	83	78
3	240	284	215	171
4	1089	1209	276	156
5	2829	2955	174	48
6	4952	50005	53	
7	6426	6435	9	
8	6434	6435	1	
9	5005	5005		
10	3003	3003		
11	1365	1365		
12	455	455		
13	105	105		
14	15	15		
15	1	1		

The values of α_i's and β_i's in Table (2) and all other results of linear/circular connected (1,2)-or-(2,1)-out-of-(n,m) : F lattices and some other given results in table 6 coincided with those appeared in [2].

Note :-

The technique used in the present algorithm differs from that in[2], for connected (1,2)-or-(2,1)-out-of-(n,m) : F lattice system. The later enumertes useful matrices without appling any tests. While the former enumerates the useless ones with applying tests for the consecutive property.

For the present algorithm we prefer counting the useless matrices in order to reduce the running time. This is because generally for any r and s, s.t. $r \neq 1$, $s \neq 2$, the number of useless matrices are much more less than that of useful ones. However, the case is different for r=l, s=2.This is clear from tables (1),(2) and (3).

Table (3) The Coefficients of R_{λ}.((2,3)-(3,2);(4,3);p)

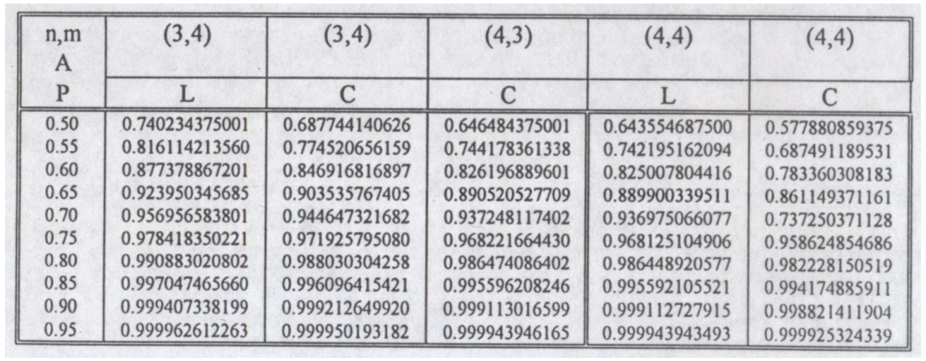
I	Υ_i	δ_i	α_i	β_i
0	0	0	1	1
1	0	0	12	12
2	0	0	66	66
3	0	0	660	660
4	0	0	495	495
5	0	0	792	792
6	7	12	817	812
7	42	72	750	720
8	95	158	400	337
9	102	156	558	504
10	53	64	13	2
11	12	12		
12	1	1		

For specified values of r, s, n and m some calculated reliabilities of linear/circular X_2: F lattice systems are given in the appendix at tables' (6)-(10).

Appendix

Table (4)

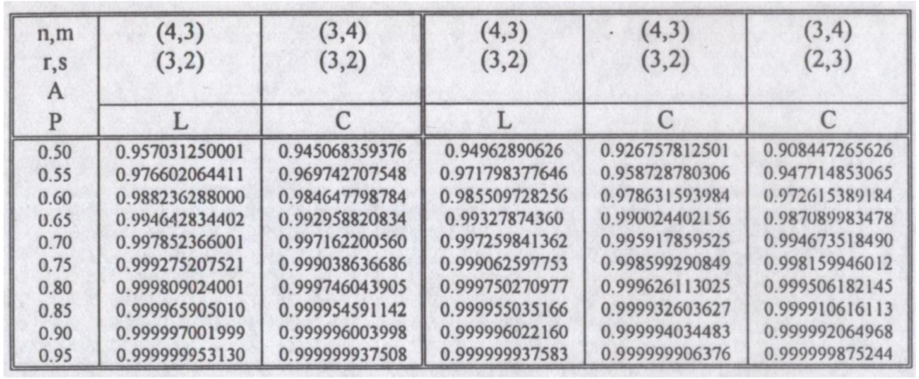
$R_\lambda((2,2),(n,m),p)$



n,m / A	(3,4)	(3,4)	(4,3)	(4,4)	(4,4)
P	L	C	C	L	C
0.50	0.740234375001	0.687744140626	0.646484375001	0.643554687500	0.577880859375
0.55	0.816114213560	0.774520656159	0.744178361338	0.742195162094	0.687491189531
0.60	0.877378867201	0.846916816897	0.826196889601	0.825007804416	0.783360308183
0.65	0.923950345685	0.903535767405	0.890520527709	0.889900339511	0.861149371161
0.70	0.956956583801	0.944647321682	0.937248117402	0.936975066077	0.737250371128
0.75	0.978418350221	0.971925795080	0.968221664430	0.968125104906	0.958624854686
0.80	0.990883020802	0.988030304258	0.986474086402	0.986448920577	0.982228150519
0.85	0.997047465660	0.996096415421	0.995596208246	0.995592105521	0.994174885911
0.90	0.999407338199	0.999212649920	0.999113016599	0.999112727915	0.998821411904
0.95	0.999962612263	0.999950193182	0.999943946165	0.999943943493	0.999925324339

Table (5)

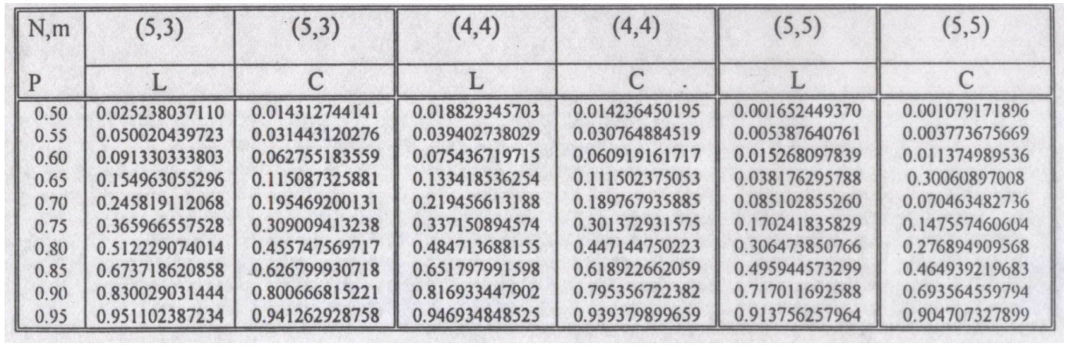
$R_\lambda((r,s),(n,m),p)$



n,m r,s A	(4,3) (3,2)	(3,4) (3,2)	(4,3) (3,2)	(4,3) (3,2)	(3,4) (2,3)
P	L	C	L	C	C
0.50	0.957031250001	0.945068359376	0.94962890626	0.926757812501	0.908447265626
0.55	0.976602064411	0.969742707548	0.971798377646	0.958728780306	0.947714853065
0.60	0.988236288000	0.984647798784	0.985509728256	0.978631593984	0.972615389184
0.65	0.994642834402	0.992958820834	0.99327874360	0.990024402156	0.987089983478
0.70	0.997852366001	0.997162200560	0.997259841362	0.995917859525	0.994673518490
0.75	0.999275207521	0.999038636686	0.999062597753	0.998599290849	0.998159946012
0.80	0.999809024001	0.999746043905	0.999750270977	0.999626113025	0.999506182145
0.85	0.999965905010	0.999954591142	0.999955035166	0.999932603627	0.999910616113
0.90	0.999997001999	0.999996003998	0.999996022160	0.999994034483	0.999992064968
0.95	0.999999953130	0.999999937508	0.999999937583	0.999999906376	0.999999875244

Table (6)

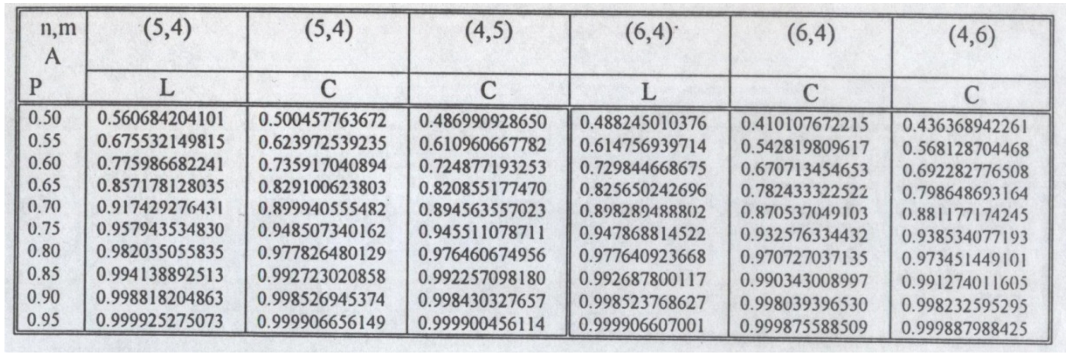
R_λ ((1,2)-(2,1),(n,m)p)



N,m	(5,3)	(5,3)	(4,4)	(4,4)	(5,5)	(5,5)
P	L	C	L	C	L	C
0.50	0.025238037110	0.014312744141	0.018829345703	0.014236450195	0.001652449370	0.001079171896
0.55	0.050020439723	0.031443120276	0.039402738029	0.030764884519	0.005387640761	0.003773675669
0.60	0.091330333803	0.062755183559	0.075436719715	0.060919161717	0.015268097839	0.011374989536
0.65	0.154963055296	0.115087325881	0.133418536254	0.111502375053	0.038176295788	0.30060897008
0.70	0.245819112068	0.195469200131	0.219456613188	0.189767935885	0.085102855260	0.070463482736
0.75	0.365966557528	0.309009413238	0.337150894574	0.301372931575	0.170241835829	0.147557460604
0.80	0.512229074014	0.455747569717	0.484713688155	0.447144750223	0.306473850766	0.276894909568
0.85	0.673718620858	0.626799930718	0.651797991598	0.618922662059	0.495944573299	0.464939219683
0.90	0.830029031444	0.800666815221	0.816933447902	0.795356722382	0.717011692588	0.693564559794
0.95	0.951102387234	0.941262928758	0.946934848525	0.939379899659	0.913756257964	0.904707327899

Table (7)

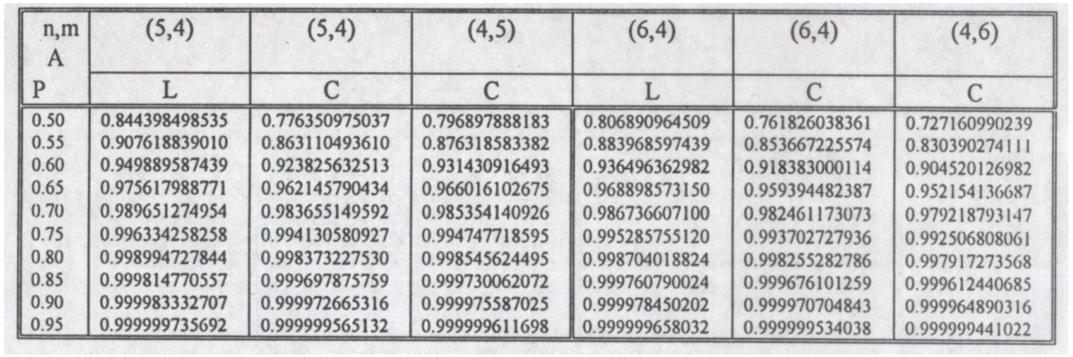
$R_\lambda((2,2),(n,m),p)$



n,m A	(5,4)	(5,4)	(4,5)	(6,4)	(6,4)	(4,6)
P	L	C	C	L	C	C
0.50	0.560684204101	0.500457763672	0.486990928650	0.488245010376	0.410107672215	0.436368942261
0.55	0.675532149815	0.623972539235	0.610960667782	0.614756939714	0.542819809617	0.568128704468
0.60	0.775986682241	0.735917040894	0.724877193253	0.729844668675	0.670713454653	0.692282776508
0.65	0.857178128035	0.829100623803	0.820855177470	0.825650242696	0.782433322522	0.798648693164
0.70	0.917429276431	0.899940555482	0.894563557023	0.898289488802	0.870537049103	0.881177174245
0.75	0.957943534830	0.948507340162	0.945511078711	0.947868814522	0.932576334432	0.938534077193
0.80	0.982035055835	0.977826480129	0.976460674956	0.977640923668	0.970727037135	0.973451449101
0.85	0.994138892513	0.992723020858	0.992257098180	0.992687800117	0.990343008997	0.991274011605
0.90	0.998818204863	0.998526945374	0.998430327657	0.998523768627	0.998039396530	0.998232595295
0.95	0.999925275073	0.999906656149	0.999900456114	0.999906607001	0.999875588509	0.999887988425

Table (8)

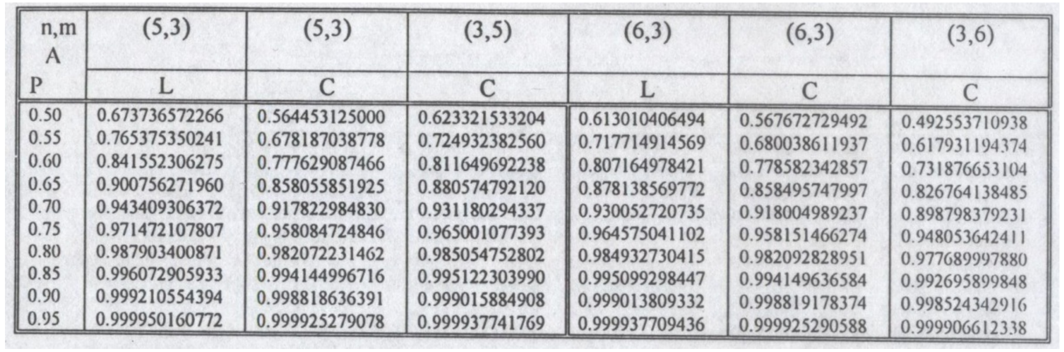
R_{λ} ((3,2)or(2,3),(n,m),p)



n,m A	(5,4)	(5,4)	(4,5)	(6,4)	(6,4)	(4,6)
P	L	C	C	L	C	C
0.50	0.844398498535	0.776350975037	0.796897888183	0.806890964509	0.761826038361	0.727160990239
0.55	0.907618839010	0.863110493610	0.876318583382	0.883968597439	0.853667225574	0.830390274111
0.60	0.949889587439	0.923825632513	0.931430916493	0.936496362982	0.918383000114	0.904520126982
0.65	0.975617988771	0.962145790434	0.966016102675	0.968898573150	0.959394482387	0.952154136687
0.70	0.989651274954	0.983655149592	0.985354140926	0.986736607100	0.982461173073	0.979218793147
0.75	0.996334258258	0.994130580927	0.994747718595	0.995285755120	0.993702727936	0.992506808061
0.80	0.998994727844	0.998373227530	0.998545624495	0.998704018824	0.998255282786	0.997917273568
0.85	0.999814770557	0.999697875759	0.999730062072	0.999760790024	0.999676101259	0.999612440685
0.90	0.999983332707	0.999972665316	0.999975587025	0.999978450202	0.999970704843	0.999964890316
0.95	0.999999735692	0.999999565132	0.999999611698	0.999999658032	0.999999534038	0.999999441022

Table (9)

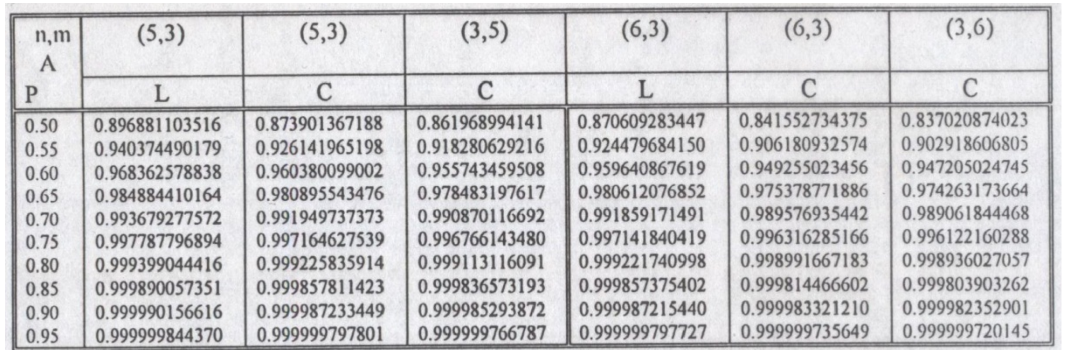
R_λ ((2,2),(n,m),p)



n,m A	(5,3)	(5,3)	(3,5)	(6,3)	(6,3)	(3,6)
P	L	C	C	L	C	C
0.50	0.673736572266	0.564453125000	0.623321533204	0.613010406494	0.567672729492	0.492553710938
0.55	0.765375350241	0.678187038778	0.724932382560	0.717714914569	0.680038611937	0.617931194374
0.60	0.841552306275	0.777629087466	0.811649692238	0.807164978421	0.778582342857	0.731876653104
0.65	0.900756271960	0.858055851925	0.880574792120	0.878138569772	0.858495747997	0.826764138485
0.70	0.943409306372	0.917822984830	0.931180294337	0.930052720735	0.918004989237	0.898798379231
0.75	0.971472107807	0.958084724846	0.965001077393	0.964575041102	0.958151466274	0.948053642411
0.80	0.987903400871	0.982072231462	0.985054752802	0.984932730415	0.982092828951	0.977689997880
0.85	0.996072905933	0.994144996716	0.995122303990	0.995099298447	0.994149636584	0.992695899848
0.90	0.999210554394	0.998818636391	0.999015884908	0.999013809332	0.998819178374	0.998524342916
0.95	0.999950160772	0.999925279078	0.999937741769	0.999937709436	0.999925290588	0.999906612338

Table (10)

$R_\lambda\ ((3,2)\text{or}(2,3),(n,m),p)$



n,m A	(5,3)	(5,3)	(3,5)	(6,3)	(6,3)	(3,6)
P	L	C	C	L	C	C
0.50	0.896881103516	0.873901367188	0.861968994141	0.870609283447	0.841552734375	0.837020874023
0.55	0.940374490179	0.926141965198	0.918280629216	0.924479684150	0.906180932574	0.902918606805
0.60	0.968362578838	0.960380099002	0.955743459508	0.959640867619	0.949255023456	0.947205024745
0.65	0.984884410164	0.980895543476	0.978483197617	0.980612076852	0.975378771886	0.974263173664
0.70	0.993679277572	0.991949737373	0.990870116692	0.991859171491	0.989576935442	0.989061844468
0.75	0.997787796894	0.997164627539	0.996766143480	0.997141840419	0.996316285166	0.996122160288
0.80	0.999399044416	0.999225835914	0.999113116091	0.999221740998	0.998991667183	0.998936027057
0.85	0.999890057351	0.999857811423	0.999836573193	0.999857375402	0.999814466602	0.999803903262
0.90	0.999990156616	0.999987233449	0.999985293872	0.999987215440	0.999983321210	0.999982352901
0.95	0.999999844370	0.999999797801	0.999999766787	0.999999797727	0.999999735649	0.999999720145

References

[l] T. K. Boehme, A. Kossow, W. Preuss, "A generalization of consecutive-k-out-of-n : F systems", IEEE Trans. Reliability, Vol 41, 1992 Sep, pp. 451-457.

[2] S. M. Khamis and N. Mokhlis," An Algorithm for computing the reliability of connected (1,2)-or-(2, 1)-out-of-(m,n) : f lattice system", Conressus Numeratium, 1997.

[3] Jacek Malinowski and Wolfgang Preuss " Lower & upper bounds for the Reliability of connected-(r,s)-out-of-(m,n) : F lattice systems", IEEE Trans. Reliability, Vol 45 No. l, 1996 march, pp 156,160.

[4] N. A. Mokhlis, E.M. El-Sayed, G. Youssef, " Reliability of Connected-(1,2)-or- (2,1) out-of-(n,m) :F Lattice Systems.", , The 32nd annual Conference on Statistics, Computer Sciences and Operation Research, I.S.S.R. Cairo University vol. 32, 1997, pp. 19-34.

[5] N. A. Mokhlis, E.M. El-Sayed, G. Youssef, "Bounds on Reliability of Connected-(r,s)-out-of-(n,m) :F Lattice Systems.", Journal of the Egyptian Mathematical Society, (1998) accepted.

[6] N. A. Mokhlis, G. Youssef, " Reliability of Connected-(r,s)-or-(s,r)-out-of-(n,m) : F Lattice Systems.", Journal of the Egyptian Mathematical Society, to appear.

[7] A. A. Salvia, A.C. Lasher, "2-dimensional consecutive-k- out-of-n : F models", IEEE Trans. Reliability, Vol 39, 1990 Aug, pp 382-385.