WORKING PAPER

USER'S GUIDE OF THE COMPUTER CODE FOR THE CALCULATION OF THE LOSS-OF-LOAD-PROBABILITY (LOLP)

T. Szántai

June 1988
WP-88-45

IIASA
International Institute for Applied Systems Analysis

USER'S GUIDE OF THE COMPUTER CODE FOR THE CALCULATION OF THE LOSS-OF-LOAD-PROBABILITY (LOLP)

T. Szántai

June 1988
WP-88-45

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

FOREWORD

The described computer code is one of the results of the IIASA contracted study "Modelling of interconnected power systems". Based on the latest results of A. Prékopa, it gives the possibility to compute the Loss-of-Load Probability of a given aggregated electric network on IBM/PC-XT or AT compatibles.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

CONTENTS

1	General Description	1
2	Mathematical Description	2
3	Description of the Input Data File	8
4	Description of the Output Data File	8
5	Solution of a Test Problem	9
6	Suggestions for Further Developments	11
References		11

USER'S GUIDE OF THE COMPUTER CODE FOR THE CALCULATION OF THE LOSS-OF-LOAD-PROBABILITY (LOLP)

T. Szántai

L. Eötvös University of Budapest

1. GENERAL DESCRIPTION

The algorithm for the computation of LOLP by A. Prékopa and E. Boros ([3]) has been implemented on IBM PC. The main steps of the algorithm are the following.

First we construct the necessary and sufficient conditions for the demand function to be feasible. These conditions consist of linear inequalities which can be determined by the Hoffmann-Gale theorem.

Then the redundant and trivially satisfied inequalities are eliminated. The algorithm of the elimination procedure was developed by A. Prékopa and E. Boros in the paper [3].

For the calculation of lower and upper bounds on the probability of a feasible flow (that is the probability of the remained inequalities to be satisfied) we have to solve two special linear programming problems (see A. Prékopa [2]). These bounds are usually close enough so their mean value can be regarded as the estimation of the probability. The input data of the LP problems are uniquely determined by the number of the remained inequalities except of the number of rows involved and of the right hand side values. The number of rows to be taken into account is an input data of the computer code (the more it is the closer the lower and upper bounds will be). The right hand side values of the LP problems consist of the first few binomial moments of the random variable giving the number of the not satisfied inequalities among the remained ones. For the calculation of the above binomial moments we apply a straightforward procedure which is based on the fact that the random demands have discrete probability distribution.

For the solution of the LP problems we use a special dual type algorithm proposed by A. Prékopa in the paper [2].

2. MATHEMATICAL DESCRIPTION

In this section the main results of the paper [3] by A. Prékopa and E. Boros are summarized. The purpose of this short summary is to make the calculations of the computer code clear and well defined.

(i) *Some basic notations and facts concerning networks*

A network $G = (N, A)$ is a finite collection of nodes N and a subset A of $N \times N$ which is the collection of arcs.

The arc capacity function is a real valued function $y(i, k)$, $(i, k) \in A$ on the set of arcs.

A flow is a real valued function $f(i, k)$, $(i,k) \in A$ which satisfies the following conditions

$$
\begin{aligned}
& f(i, k) + f(k, i) = 0 \\
& |f(i, k)| \le y(i, k) \quad \text{for} \quad (i, k) \in A \ .
\end{aligned}
\tag{1}
$$

The definition of y and f can be extended to the entire set $N \times N$, so we write $f(i, k) = y(i, k) = 0$ for $(i, k) \in N \times N$ and $(i, k) \in A$. We will use the notations

$$y(B, C) = \sum_{i \in B, k \in C} y(i, k)$$

$$f(B, C) = \sum_{i \in B, k \in C} f(i, k)$$

where B and C are subsets of N.

A demand function $d(i)$, $i \in N$ is a real valued function on the set of nodes. If $B \subseteq N$, then we assign a demand value $d(B)$ to B which is defined by

$$d(B) = \sum_{i \in B} d(i) \ .$$

A demand function is said to be *feasible* if there exists a flow f such that

$$f(N, i) \ge d(i) \quad \text{for every} \quad i \in N \ . \tag{2}$$

The relations (1) and (2) contain the variables $f(i, k)$, $y(i, k)$ and $d(i)$. It is an important problem to find the projection of the convex polyhedron defined by (1) and (2) onto the space of the variables $y(i, k)$ and $d(i)$, i.e. to give a necessary and sufficient condition in terms of these variables for the existence of a flow satisfying (1) and (2). This problem was solved by Hoffman and Gale in the following theorem:

Theorem (Hoffman and Gale) The demand function $d(i)$, $i \in N$ is feasible if and only if for every set $H \subseteq N$ we have the inequality

$$d(H) \leq y(\bar{H}, H) \ . \tag{3}$$

In power system engineering when considering interconnected power systems, one node of the network represents one power system and the whole network represents one power pool. To each node i a generating capacity x_i is assigned, moreover there exists a local demand corresponding to node i which is to be satisfied first by the use of the generating capacity x_i. The function

$$d(i) = \xi_i - x_i, \ i \in N$$

is a demand function corresponding to the network (network demand). If $\xi_i - x_i > 0$, then at node i we need an amount of power $\xi_i - x_i$ and if $\xi_i - x_i < 0$, then at node i there is a surplus generating capacity of $x_i - \xi_i$ which we may term supply. The variable ξ_i represents deficiency in the generation and excess local demand. If

$$\sum_{i \in N} x_i > \sum_{i \in N} \xi_i$$

then the total available power generating capacity is enough to supply the total demand. However, the transmission system may not be able to allow that the individual power systems assist each other to the extent it is necessary. The above theorem by Hoffman and Gale provides us with a necessary and sufficient condition for the possibility of the assistance, i.e. for the existence of a feasible flow.

If the ξ_i and/or $y(i,k)$ are random variables then (3) provides us with a system of linear inequalities which may not be fulfilled depending on the special values of the random variables. Our task is to find the probability

$$P(d(H) \leq y(\bar{H}, H) \quad \text{for every} \quad H \subseteq N) \ . \tag{4}$$

Subtracting the probability (4) from one we obtain the LOLP of the system.

(ii) *An algorithm for the elimination of redundant inequalities*

The following algorithm developed by A. Prékopa and E. Boros is applied to eliminate redundant inequalities out of the system of inequalities (3).

Let $b(H)$ and $e(H)$ be two binary variables depending on subsets H of N. The equality $b(H) = 1$ means that H derives an inequality of (3) which is not deleted. The other variable $e(H)$ is used only in the algorithm. $e(H) = 1$ means that the set $H \subseteq N$ was already tested. The subsequent steps of the algorithm are the following:

Step 0: Let $b(H) = 1$, $e(h) = 0$ for all $H \subseteq N$, $H \neq \emptyset$.

Step 1: Choose a non-empty subset $H \subseteq N$ such that $b(H) = 1$ and $e(H) = 0$. If there is no such subset H, then STOP.

Step 2: Let $T \subseteq N \backslash H$ be maximal with the property that there is no arc between T and H.

Step 3: Let $b(V) = 0$ for all $V \subseteq H \cup T$, $V \cap T \neq \emptyset \neq V \cap H$.

Step 4: Let $e(H) = 1$, and if the inequality derived by the subset H is trivially satisfied, then set $b(H) = 0$. GO TO Step 1.

(iii) *The calculation of the binomial moments*

Let $H_1, \ldots, H_n$ designate those subsets of N which derive those inequalities in (3) which are not eliminated by the algorithm described above. Assuming now the demand function to be random, we designate by A_i the event that $d(H_i) \leq y(\bar{H}_i, H_i)$ and by $\bar{A}_i$ the event that $d(H_i) > y(\bar{H}_i, H_i)$. We want to evaluate the probability

$$S_n = P(A_1 \cdots A_n) \tag{5}$$

which is one minus the LOLP.

As the number n of the events A_i may be quite large the direct evaluation of the probability (5) requires tremendous computation. This problem can be reduced to the evaluation of probabilities of smaller number of events. We can estimate the probability value (5) by the solution of two special LP problems to be described later. The right hand side values of these LP problems consist of the first few binomial moments of the random variable giving the number of the not satisfied inequalities among those which remained after the elimination of redundant ones. For the calculation of these binomial moments we apply a straightforward procedure which is based on the fact that the random demands have discrete probability distribution.

To keep the presentation relatively simple we make the following assumptions:

- the arc capacity function $y(i, k)$, $(i, k) \in A$ is non-random,

- the random variable corresponding to the nodes in the network, i.e. $d(i)$, $i \in N$ are independent of each other,
- the possible values of the random variable $d(i)$ is a finite set of integers D_i.

Now the sample space is the product space $\Omega = D_1 \times \cdots \times D_{|N|}$ and this consists of the set of $|N|$-tuples $\omega = (\omega_1, \ldots, \omega_{|N|})$, where $\omega_i \in D_i$, $i = 1, \ldots, |N|$. Introducing the notation

$$P(d(i) = j) = p_{ij},\ j \in D_i,\ i \in N \ ,$$

to the elementary event ω the probability

$$P(\omega) = \prod_{i \in N} p_{i\omega_i}$$

is assigned.

In order to compute the binomial moments

$$\bar{S}_k = \sum_{1 \leq l_1 < \cdots < l_k \leq n} P(\bar{A}_{l_1} \cdots \bar{A}_{l_k}),\ 1 \leq k \leq m$$

we have to compute the probabilities of the form $P(\bar{A}_{l_1} \cdots \bar{A}_{l_k})$, where $l_1, \ldots, l_k$ are distinct values. This equals

$$P(\bar{A}_{l_1} \cdots \bar{A}_{l_k}) = \sum_{\substack{\sum_{j \in H_{l_1}} \omega_j \ > \ y(\bar{H}_{l_1}, H_{l_1}) \\ \vdots \\ \sum_{j \in H_{l_k}} \omega_j \ > \ y(\bar{H}_{l_k}, H_{l_k})}} \prod_{i \in N} p_{i\omega_i}$$

(iv) *The estimation of the probability of a feasible flow*

Let $\bar{\mu}$ designate the number of those A_i which not occur i.e. the number of those $\bar{A}_i$ which occur. Then $\bar{\mu}$ is a random variable the possible values of which are among the numbers 0, 1, ..., n. Introducing the notation

$$P(\bar{\mu} = i) = v_i,\ i = 1, 2, \ldots, n$$

the binomial moment $\bar{S}_k$ of the random variable $\bar{\mu}$ can be expressed as

$$\bar{S}_k = E\left[\binom{\bar{\mu}}{k}\right] = \sum_{t=0}^{n} \binom{t}{k} v_t,\ k = 1, 2, \ldots, n \ . \qquad (6)$$

Relaxing the equation (6) by keeping the first m rows only but prescribing that $v_i \geq 0,\ i = 1, \ldots, n$ we can maximize resp. minimize the sum $v_1 + \cdots + v_n$ i.e. we can solve the linear programming problems

$$\begin{array}{llllllll}
\text{maximize (} & v_1 + v_2 + & \cdots & + & v_m + & \cdots & + v_n) & \\
\text{subject to} & v_1 + 2v_2 + & \cdots & + & m v_m + & \cdots & + n v_n & = \bar{S}_1 \\
 & v_2 + & \cdots & + & \binom{m}{2} v_m + & \cdots & + \binom{n}{2} v_n & = \bar{S}_2 \\
 & & \ddots & & \vdots & & \vdots & \vdots \\
 & & & & v_m + & \cdots & + \binom{n}{m} v_n & = \bar{S}_m \\
 & v_1 \geq 0,\ v_2 \geq 0, & \cdots, & & v_m \geq 0, & \cdots, & v_n \geq 0 &
\end{array} \tag{7}$$

resp.

$$\begin{array}{llllllll}
\text{minimize (} & v_1 + v_2 + & \cdots & + & v_m + & \cdots & + v_n) & \\
\text{subject to} & v_1 + 2v_2 + & \cdots & + & m v_m + & \cdots & + n v_n & = \bar{S}_1 \\
 & v_2 + & \cdots & + & \binom{m}{2} v_m + & \cdots & + \binom{n}{2} v_n & = \bar{S}_2 \\
 & & \ddots & & \vdots & & \vdots & \vdots \\
 & & & & v_m + & \cdots & + \binom{n}{m} v_n & = \bar{S}_m \\
 & v_1 \geq 0,\ v_2 \geq 0, & \cdots, & & v_m \geq 0, & \cdots, & v_n \geq 0 &
\end{array} \tag{8}$$

If $\bar{V}_{\max}$ and $\bar{V}_{\min}$ are the optimum values of problems (7) and (8), respectively then we have

$$\bar{V}_{\min} \leq P(\bar{\mu} > 1) \leq \bar{V}_{\max} \ .$$

As we have

$$P(A_1 \cdots A_n) = 1 - P(\bar{A}_1 + \cdots + \bar{A}_n) =$$

$$= 1 - P(\bar{\mu} > 1) \ ,$$

so we get the required lower and upper bound on the probability value (5)

$$V_{\min} = 1 - \bar{V}_{\max} \leq P(A_1 \cdots A_n) \leq 1 - \bar{V}_{\min} = V_{\max} \ .$$

If for a given m the lower and upper bound are not close enough, then one can increase m to get these bounds closer.

For constructing a fast solution algorithm of the linear programming problems (7) and (8) A. Prékopa proved the following theorems (see in [2]):

THEOREM A *A basis in Problem (7) is dual feasible if and only if it is of the form*

$$B = (a_1,\ a_i,\ a_{i+1},\ a_j,\ a_{j+1},\dots,a_k,\ a_{k+1},\ a_n)$$

for an even m, where

$$2 \le i,\ i+1 < j,\dots,k+1 \le n$$

and

$$B = (a_1,\ a_i,\ a_{i+1},\ a_j,\ a_{j+1},\dots,a_k,\ a_{k+1})$$

for an odd m, where

$$2 \le i,\ i+1 < j,\dots,k+1 \le n.$$

THEOREM B *A basis in Problem (8) is dual feasible if and only if it is of the form*

$$B = (a_i,\ a_{i+1},\ a_j,\ a_{j+1},\dots,a_k,\ a_{k+1})$$

for an even m, where

$$1 \le i,\ i+1 < j,\dots,k+1 \le n$$

and

$$B = (a_i,\ a_{i+1},\ a_j,\ a_{j+1},\dots,a_k,\ a_{k+1},\dots,a_n)$$

for an odd m, where

$$1 \le i,\ i+1 < j,\dots,k+1 \le n\ .$$

In the above theorems a_i denotes the column vector belonging to the variable v_i in the linear equality system of the linear programming problem.

Using Theorems A and B, unique algorithms developed by A. Prékopa ([2]) solve the problems (7) and (8). These can be summarized as follows. Starting by any dual feasible basis in either of the problems (7), (8) we check if $B^{-1}\bar{S} \ge 0$ or not. Here $\bar{S}$ is the vector of components $\bar{S}_1,\dots,\bar{S}_m$. If yes, then B is primal-dual feasible, hence optimal basis. If this is not the case then choose a p such that $(B^{-1}\bar{S})_r < 0$ and delete the p-th vector from B. Theorems A and B guarantee that there is one and only one way to restore the basis structure by including a vector (other than the one just deleted) into the basis. Having done this we analyze again the basic components corresponding to the new basis, etc. This algorithm is a special case of the lexicographic dual simplex algorithm, hence it is finite.

3. DESCRIPTION OF THE INPUT DATA FILE

The data input of the computer code consists of four type of records.

Record type 1

N – the number of nodes in the network.

Record type 2

The upper triangular part of the node to node incidence matrix. The matrix elements are given rowise (every row in different records). The number of type 2 data records equals to N-1. We remark that a nonzero incidence matrix element represents the corresponding arc capacity value.

Record type 3

In these records the discrete probability distributions of the demand function are given. The first record contains the number of discrete values the demand takes on at a given node. The second record contains the possible values of the demand and the third record contains the probability values according to the demand values. These three records are repeated N times for the different nodes.

Record type 4

M – the number of rows to be taken into account in the LP problems.

4. DESCRIPTION OF THE OUTPUT DATA FILE

In the first line of the output data file the name of the input data file appears.

The further content of the output data file is divided into four parts according to the different calculations. The elapsed time is measured for every part of the calculations. The output data file contains the starting and finishing times together with the elapsed time.

In the first part the conditions involved in the Hoffman – Gale theorem are generated. They do not appear in the output data file as it could take a lot of space also for relatively small problems.

The second part of the output data file consists of the zero-one coefficient matrix of the remained inequalities and of the calculated right hand side vector.

In the third part the binomial moments are listed.

Finally, in the fourth part the solutions of the linear programming problems are contained. This consists of the nonzero components of the solution vectors and the calculated lower resp. upper bounds on the estimated probability value.

5. SOLUTION OF A TEST PROBLEM

The four node example problem in [3] has been solved for $M = 2$.

The list of the input data file is the following:

```
4
1,1,1
0,0
0
5
-1,0,1,2,3
0.8145,0.1715,0.0135,0.00048,0.000006
7
-2,-1,0,1,2,3,4
0.7351,0.2321,0.0305,0.00214,0.000085,0.0000018,0.0000000156
5
-1,0,1,2,3
0.8145,0.1715,0.0135,0.00048,0.000006
5
-1,0,1,2,3
0.8145,0.1715,0.0135,0.00048,0.000006
2
```

The list of the output data file is the following:

The name of the input data file is : NODE4_2.DAT

Condition generation started at 14:29:14.33, finished at 14:29:14.33

Solution time=   .00 sec

Elimination started at 14:29:14.33

The linear inequality system (after the elimination procedure) :

0	1	0	0	1
0	0	1	0	1
0	0	0	1	1
1	1	0	0	2
1	0	1	0	2
1	0	0	1	2
1	1	1	0	1
1	1	0	1	1
1	0	1	1	1
1	1	1	1	0

Elimination started at 14:29:14.33, finished at 14:29:14.44

Solution time= .11 sec

Binomial moment calculation started at 14:29:14.50

The binomial moment values are :

S(1) = .00151413

S(2) = .00034421

Binomial moment calculation started at 14:29:14.50, finished at 14:29:54.44

Solution time= 39.94 sec

BINLP optimization started at 14:29:54.50

Problem name : BINLP (max)

The nonzero components of the solution :

v(1) = .001437636772451

v(10) = .000007649045799

The lower bound = .99855471

Problem name : BINLP (min)

The nonzero components of the solution :

v(1) = .000825713108568

v(2) = .000344207060934

The upper bound = .99883008

BINLP optimization started at 14:29:54.50, finished at 14:29:54.88

Solution time= .38 sec

6. SUGGESTIONS FOR FURTHER DEVELOPMENTS

The straightforward calculation of the binomial moments (right hand side values of the LP problems) is a time consuming job. This calculation procedure should be replaced by a faster algorithm based on the concept of generating functions (see A. Prékopa and E. Boros [3]).

For the solution of the LP problems one should improve the dual type solution technique for handling individual upper bounds (see A. Prékopa and E. Boros [3]). If this is not possible then one should try to use a general LP solver as the computer code MILP by I. Maros ([1], where it was still called MICROLP).

REFERENCES

[1] Maros, I., "A general Phase-I method in linear programming", European Journal of Operational Research 23 (1986) 64-77.

[2] Prékopa, A., "Boole-Bonferroni inequalities and linear programming", RUTCOR Research Report #4-86, April 1986, The State University of New Jersey, RUTGERS, New Brunswick, New Jersey.

[3] Prékopa, A. and E. Boros, "On the probability of the existence of a feasible flow in a transportation network", RUTCOR Research Report, August, 1986, The State University of New Jersey, RUTGERS, New Brunswick, New Jersey.