Network reliability evaluation by the Ahmad method

Héctor Cancela *María E. Urquhart* *Gerardo Rubino*

cancela@fing.edu.uy urquhart@fing.edu.uy rubino@irisa.fr

Investigación Operativa

Instituto de Computación, PeDeCiBa Informática

Facultad Ingeniería, Montevideo, Uruguay

Équipe MODEL

IRISA

Campus de Beaulieu, Rennes, France

January 20, 1996

Abstract — In the evaluation of the capacity of a communication network architecture to resist to the possible failures of some of its components, several reliability metrics are currently used. In this paper we consider the $\mathcal{K}$-*terminal reliability* measure $R_{\mathcal{K}}$. The exact evaluation of this parameter is in general a very costly task since it is, as an algorithmic problem, classed in the NP–hard family. Different methods exist for performing this task. In this paper, we consider an algorithm proposed by Ahmad for the special case R_{st} which has the advantage of using a small amount of memory. We present this algorithm in the general $R_{\mathcal{K}}$ case.

Key words — Network reliability evaluation.

Resumen — Una red de comunicaciones es un sistema compuesto por un conjunto de centros que reciben y envían información, y un conjunto de conexiones que transportan esa información. La capacidad de una red de comunicaciones para resistir a posibles fallos de algunos de sus componentes se evalúa a través de distintas métricas de confiabilidad.

La confiabilidad de la red es una medida definida como la probabilidad de comunicación exitosa entre diferentes pares o conjunto de centros, dadas las probabilidades de buen funcionamiento de los componentes de la red y la topología de las conexiones. Uno de los modelos más utilizados es el del cálculo de la medida $R_{\mathcal{K}}$ (confiabilidad entre el conjunto $\mathcal{K}$ de terminales) en el caso de una red con nodos perfectos y líneas falibles. La evaluación exacta de esta medida es, en general, muy costosa computacionalmente dado que pertenece a la familia de problemas NP-duros. Desde el punto de vista práctico esto quiere decir que el tiempo necesario para su evaluación es una función exponencial en el tamaño (cantidad de elementos) de la red.

Existen varios métodos para realizar este cálculo, aplicables para redes de pequeño y mediano tamaño (hasta 50 o 100 elementos). Uno de estos es el llamado "método de Ahmad", propuesto en [1] para el caso especial de la confiabilidad fuente-terminal (cálculo de R_{st}). En este trabajo presentamos dicho algoritmo en el caso general de la evaluación de $R_{\mathcal{K}}$.

Palabras clave — Confiabilidad de redes, métodos exactos.

1 Introduction

Consider a communication network $\mathcal{G}$ where nodes are perfect and links fail randomly and independently. We consider $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to be an undirected graph, connected and without loops; the node set $\mathcal{V}$ corresponds to the nodes of the network, and the edge set $\mathcal{E}$ corresponds to the links. When the link failure probabilities are known, the success of communication between nodes in some fixed subset $\mathcal{K}$ of the node-set is a random event. The probability $R_{\mathcal{K}}$ of this event is usually called the *$\mathcal{K}$-terminal reliability*, equivalently the parameter $Q_{\mathcal{K}} = 1 - R_{\mathcal{K}}$ is called the *$\mathcal{K}$-terminal unreliability*. The problem of its evaluation has received considerable attention from the research community (see [8] and [11] for many references). One of the reasons is that in the general case, this problem is *NP*-hard [3], [4], [9].

Depending on the choice of the set $\mathcal{K}$, we have different reliability metrics. The most used ones are source-terminal reliability R_{st} where s and t are two fixed nodes of $\mathcal{V}$, and all-terminal reliability $R_{\mathcal{V}}$.The Ahmad method for R_{st} evaluation was introduced in [1], and some improvements were presented in [2] and [10]. In this work, we describe this method for $R_{\mathcal{K}}$ evaluation, with general $\mathcal{K}$.

This report is organized as follows. Some notations and definitions are introduced in Section 2. In Section 3 we give the description of the Ahmad algorithm for the evaluation of $R_{\mathcal{K}}$. In Section 4 we present some computational results and some conclusions.

2 Global Notations and Definitions

Let us resume in this section the principal notations.

- $\mathcal{V} = \{1, \ldots, n\}$: the network node-set;
- $\mathcal{E} = \{e_1, \ldots, e_m\}$: the network link-set;
- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: the analyzed undirected network (graph) topology;
- m: the number of links (edges) of $\mathcal{G}$;
- n : the number of nodes of $\mathcal{G}$;
- $\mathcal{K} \subseteq \mathcal{V}$: the terminals set; that is the subset of nodes that must communicate with each other to consider the system (network) to be operational;
- x_e: the binary random variable "state of link e in $\mathcal{G}$", defined by

$$x_e = \begin{cases} 1 & \text{if link } e \text{ is up (operational),} \\ 0 & \text{if link } e \text{ is down (failed);} \end{cases}$$

- r_e: the elementary reliability of link e, that is, $r_e = \mathcal{P}(x_e = 1)$;
- $X = (x_1, \ldots, x_m)$: the random network state vector;
- Φ: the structure function associated with the *$\mathcal{K}$-terminal reliability* measure, that is, the function of $\{0,1\}^m$ in $\{0,1\}$ valued by 1 if the network deduced from $\mathcal{G}$ by deleting down links is $\mathcal{K}$-connected, 0 otherwise;
- $\Phi(X)$ the binary random variable "state of network $\mathcal{G}$", that is;

$$\Phi(X) = \begin{cases} 1 & \text{if the graph deduced from } \mathcal{G} \text{ by removing each failed link in } X \text{ is } \mathcal{K}\text{-connected,} \\ 0 & \text{otherwise;} \end{cases}$$

- $R_{\mathcal{K}} = \mathcal{P}\left(\Phi(X) = 1\right)$ the $\mathcal{K}$-*terminal reliability* parameter of network $\mathcal{G}$;
- A $\mathcal{K}$-tree (also called Steiner tree) is a minimal set of components (links) which connect all nodes in K;

3 Basic description of the Ahmad method

The basic idea behind the Ahmad method comes from the work [1]. In this report, we will follow the description given in [10], where an improved version was given, which does not need the generation of any intermediate list of events.

We want to compute the reliability $R_{\mathcal{K}}$ of the connection between the nodes of K under the assumptions of independence between the behavior of the different lines.

Let us denote by $C_{\mathcal{K}}$ the event "the graph is $\mathcal{K}$-connected"; we have then

$$R_{\mathcal{K}} = \mathcal{P}\left(C_{\mathcal{K}}\right) \tag{1}$$

If $\{\pi_1, \ldots, \pi_{MP}\}$ is the set of $\mathcal{K}$-trees and P_k denotes the event: "every link in the k^{th} $\mathcal{K}$-tree is working correctly", then

$$C_{\mathcal{K}} = \bigcup_{1 \leq k \leq MP} P_k. \tag{2}$$

Observe that the probability of events P_k is immediate from the independence hypothesis but that, in general, the events $\{P_1, P_2, \ldots\}$ are not disjoint, so the above formula is not very useful to compute the reliability $R_{\mathcal{K}}$.

Much of the considerable amount of work in the family of direct approaches to reliability computation has been inspired with the idea of constructing a partition of the event $C_{\mathcal{K}}$, that is finding a family of events $\{B_k\}$ such that:

$$C_{\mathcal{K}} = \bigcup_k B_k \tag{3}$$

with $B_i \cap B_j = \emptyset$, $\forall i \neq j$. With such a partition, we will be able to compute $R_{\mathcal{K}}$ using the following equation:

$$R_{\mathcal{K}} = \mathcal{P}\left(C_{\mathcal{K}}\right) = \sum_k \mathcal{P}\left(B_k\right) \tag{4}$$

The Ahmad method is based in this idea. It has the property of constructing the partition $\{B_k\}$ simultaneously with the exploration of the graph without calculating the list $\{P_1, P_2, \ldots\}$ and in this way, it needs a small amount of memory. In particular, from the degrees of the nodes, the total amount of space can be statistically calculated and then allocated.

We will construct a partition of $C_{\mathcal{K}}$ in terms of events which we shall call *branches*. The branches will be denoted by sequences of symbols taken from the alphabet $\mathcal{V} \cup \mathcal{GV} \cup \overline{\mathcal{V}}$ with $\overline{\mathcal{V}} \stackrel{\text{def}}{=} \{\overline{x}/x \in \mathcal{V}\}$ and $\mathcal{GV} \stackrel{\text{def}}{=} \{gx/x \in \mathcal{V}\}$. For instance, if $\mathcal{V} = \{1, 2, 3\}$ we have $\overline{\mathcal{V}} = \{\overline{1}, \overline{2}, \overline{3}\}$ and $\mathcal{GV} = \{g1, g2, g3\}$.

The algorithm consists of moving continuously a single branch, which is transformed from time to time into an element of the partition; let us call such an element a *finished branch*. At this point, the probability of this event is computed and accumulated, and the process continues. When the algorithm ends, the set of all the finished branches generated is a partition of $C_{\mathcal{K}}$.

Any branch, finished or not, will be represented by a sequence $b = (b_1, b_2, \ldots, b_H)$ with the following meaning. Let us denote by $x,\ y, \ldots$ the points of $\mathcal{V}$. The elements of $\overline{\mathcal{V}}$ will be denoted by $\overline{x},\ \overline{y}, \ldots$ and those of $\mathcal{GV}$ by $gx,\ gy, \ldots$ where $x,\ y, \ldots \in \mathcal{V}$. The first element is $b_1 = s$, where $s \in \mathcal{K}$ is chosen arbitrarily. The consecutive sequence (x, y) of $\mathcal{V} \times \mathcal{V}$ means that x and y are adjacent nodes and that in the event b the event "link (x, y) is working" is realized. The consecutive sequence (gx, y) has exactly the same meaning (we will see later the use of the prefix g). A sequence of consecutive symbols of the form $(X, \overline{y}_1, \overline{y}_2, \ldots, \overline{y}_L)$

where $X = x$ or $X = gx$ means that x is adjacent to $y_1, y_2, \ldots, y_L$ and that in the event b, link (x, y_l) does not work, for $l = 1, 2, \ldots, L$. If the subsequence is $(X, \overline{y}_1, \overline{y}_2, \ldots, \overline{y}_L, Z)$ where X means either x or gx then the meaning is as before with, in addition, $Z = z \in \mathcal{V}$ or $Z = gz \in \mathcal{GV}$ and in the first case, x and z are adjacent and link (x, z) works. A finished branch is a branch in which all nodes in $\mathcal{K}$ are present. As a consequence of the method, in every branch $b = (b_1, b_2, \ldots, b_H)$, if $h > 1$ then $b_h \neq s$ (but the case $b_h = gs$ is possible). Given a finished branch b, its probability is given by:

$$\mathcal{P}(b) = \prod_{e_i \in working(b)} r_{e_i} \ . \prod_{e_j \in failed(b)} (1 - r_{e_j}) \tag{5}$$

where $working(b)$ and $failed(b)$ are the sets of links defined respectively as working or down in b according to the rules defining the meaning of the representation.

The algorithm starts with the initial branch $b = (s)$, where s is an arbitrarily chosen node such that $s \in \mathcal{K}$. The main loop consists of looking at the last symbol of b and carrying out some actions depending on three possible cases. We will now describe in detail the algorithm. For any branch b, let us define the following notation.

- b_H: the last element of b (H is the size of b)
- $\forall h : \ b_h \notin \overline{\mathcal{V}}, \quad \mathcal{V}(b_h) \stackrel{\text{def}}{=} \begin{cases} x & \text{if } b_h = x \\ y & \text{if } b_h = gy \end{cases}$
- $\forall h < H : \quad next(h) \stackrel{\text{def}}{=} min\,\{i / h < i \leq H, b_i \notin \overline{\mathcal{V}}\}$
- $\forall h > 1 : \quad previous(h) \stackrel{\text{def}}{=} max\,\{i / 1 \leq i < h, b_i \notin \overline{\mathcal{V}}\}$
- $b_L \stackrel{\text{def}}{=}$ the last reached node in branch b; $L = \begin{cases} H & \text{if } b_H \in \mathcal{V} \cup \mathcal{GV} \\ previous(H) & \text{if } b_H \in \overline{\mathcal{V}} \end{cases}$
- $\forall h : b_h \notin \overline{\mathcal{V}}$, let $x \stackrel{\text{def}}{=} \mathcal{V}(b_h)$; then:

  $$E(h) \stackrel{\text{def}}{=} \{y \text{ adjacent to } x / \ \begin{array}{l} \text{there is no } k < h \text{ such that } b_k = y \text{ or } b_k = gy, \\ \text{there is no } k > h \text{ such that } k < next(h) \text{ and } b_k = \overline{y}\} \end{array}$$
- $\forall x \in \mathcal{V}, \quad first(x) \stackrel{\text{def}}{=} min\,\{i / 1 \leq i \leq H, b_i = x \text{ or } b_i = gx\}$
- $reached(b) = \{x \in \mathcal{V} / \exists i \leq H, b_i = x\}$

In Figure 1 we present here the pseudocode for main loop of the Ahmad method as modified in [10]. We use the function **Backtrack**, shown in Figure 2; this routine is used to start the search for a new branch when either there's a finished branch or there's no possibility to finish the branch under construction:

Let us make a few comments. Instead of looking at the evolution of the single current branch b, we can consider, as in [1], that the algorithm constructs a tree T with node set $\mathcal{V} \cup \overline{\mathcal{V}} \cup \mathcal{GV}$ and root s, in the following manner. There will always be a current branch of T in construction, identical to the current b. When b grows by the addition of a symbol, the corresponding branch of T does the same. When a "backtrack" happens, a branch $b = (b_1, \ldots, b_{k-1}, y, \ldots)$ is transformed into $(b_1, \ldots, b_{k-1}, \overline{y})$; in T, the current branch gives birth to a new one, having the first $k - 1$ nodes in common and ending in $\overline{y}$.

The algorithm backtracks when, in the current branch b, the communication between the nodes of $\mathcal{K}$ cannot happen. It looks for a node x with a non empty E set to continue the process, that is, to build a new branch in T. If such a node x exists, it is represented in b by the symbol gx (g as in growing). When this is no longer possible, the algorithm ends. If a new branch of the tree is built from a node x in position h of b, the new node y introduced in the tree is chosen from the elements of $E(h)$.

```
b = (s);
H := 1;
L := 1;
R := 0.0;
Partition_Completed := false;
repeat
    do cases
      case (b_H ∈ 𝒦) and (𝒦 ⊆ reached(b)) /* b is a finished branch */
            R := R + 𝒫(b);
            b := (b_1, ..., b_{H-1}, ¬b_H);
      case (b_H ∈ ¬𝒦) and (# of ¬b_H in b = degree(b_H))
                /* b can't give rise to a finished branch */
            Backtrack;
      otherwise
            do cases
              case E(L) = {y, ...} ≠ ∅ ⟶ b := ( b , y );
                                          H := H + 1;
                                          L := H;
              case E(L) = ∅ ⟶ i := previous(first(𝒱(b_L)));
                                 do cases
                                   case i ≠ 0 ⟶ z := 𝒱(b_i); b := ( b , gz );
                                                 H := H + 1;
                                                 L := H;
                                   case i = 0 ⟶ Backtrack;
                                 endcases
            endcases
    endcases
until Partition_Completed;
return R;
```

Figura 1: Pseudo-code for Ahmad method

```
j := max {i/1 ≤ i < L, b_i ∉ V̄, E(i) ≠ ∅};
do cases
  case j = 0 ⟶ Partition_Completed := true;
  case j ≠ 0 ⟶ k := next(j);
                y := V(b_k) ;
                b := (b_1, ..., b_{k-1}, ȳ)
                H := k;
                L := previous(H);
endcases
```

Figura 2: Pseudo-code for the **Backtrack** routine.

4 Numerical results and conclusions

We present here some numerical results obtained when testing the Ahmad algorithm in the particular case of the evaluation of $R_{\mathcal{V}}$, which is the case of interest for our work in the HEIDI tool. The test topologies we have taken are the same we have previously used in related work: the dodecahedron, a version of the Arpanet, and a version of Montevideo's optical fiber network, which we will call AntelMod (we don't reproduce here the topologies, which can be found in the following companion reports: [6], [7], [5]).

Topology	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	r_e	$R_{\mathcal{V}}$	Nb. branches	time (sec.)
AntelMod	14	21	0.99	0.998906	10238	12
Arpanet	21	26	0.99	0.997358	15667	20
Dodecahedron	20	30	0.99	0.999979	5184000	2700

Tabla 1: Results of three test topologies.

In Table 1 we present the results, obtained in a SPARCstation 5. The first columns identify the topologies, showing also the node set and edge set sizes. For the three topologies we will consider that all edges are equally reliable, with common reliability $r_e = 0.99$; the method is independent of this choice (but for values of r_e too close to 1, there may be numerical problems, related to the machine word size). We present then the computed reliability $R_{\mathcal{K}}$, and the number of branches that the method had to compute in each case. Finally, we show the running time in seconds. As it can be seen from this table, the efficiency of the method is strongly dependent on the size of the considered network.

Referencias

[1] S.Hasanuddin Ahmad. A simple technique for computing network reliability. *IEEE Tr. Reliability*, R-31(1), April 1982.

[2] S.Hasanuddin Ahmad and A.T.M. Jamil. A modified technique for computing network reliability. *IEEE Tr. Reliability*, R-36(5), December 1987.

[3] M.O. Ball. The complexity of network reliability computations. *Networks*, 10:153–165, 1980.

[4] M.O. Ball. Computational complexity of network reliability analysis: an overview. *IEEE Trans. Reliab.*, R-35(3), 1986.

[5] H. Cancela, G. Rubino, and M. E. Urquhart. Optimización en el diseño de redes de comunicación. In *Proceedings of the XIII CILAMCE*, Porto Alegre, Brasil, November 1992.

[6] H. Cancela and M. E. Urquhart. *Métodos Monte Carlo y confiabilidad de redes de comunicaciones - Estado del Arte.* Research Report 9405, PEDECIBA Informática, Facultad de Ingeniería, UROU, December 1994.

[7] H. Cancela and M. E. Urquhart. *Simulated Annealing for Communication Network Reliability Improvement.* Research Report, PEDECIBA Informática, Facultad de Ingeniería, UROU, 1995.

[8] M.O. Locks and A. Satyarayana editors. Network reliability – the state of the art. *IEEE Trans. Reliab.*, R-35(3), 1986.

[9] J.S. Provan. The complexity of reliability computations in planar and acyclic graphs. *SIAM J. Comput.*, 15(3), Aug. 1986.

[10] G. Rubino R. Marie. Direct approaches to the 2-terminal reliability problem. In E.Orhun E.Gelembe and E.Başar, editors, *The Third International Symposium on Computer and Information Sciences*, pages 740–747, Ege University, Çeşme, Izmir, Turkey, 1988.

[11] G. Rubino. Tutorial: efficient evaluation of network reliability. In , editor, *7th. Conference on Modeling Tools and Techniques*, page , Vienna, Austria, 1994.