Network reliability evaluation by the Ahmad method [1]

Héctor Cancela §† Gerardo Rubino §‡ María E. Urquhart †
cancela@fing.edu.uy rubino@irisa.fr urquhart@fing.edu.uy

§ *Équipe MODEL, IRISA/INRIA, Rennes, France*

† *Inv. Operativa, PeDeCiBa Informática, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay*

‡ *ENST Bretagne, Rennes, France*

Abstract

In the evaluation of the capacity of a communication network architecture to resist to the possible failures of its components, several reliability metrics are currently used. We consider the K-terminal reliability measure R_K, i.e. the probability of successful communication between nodes in some subset K of the network node-set.

The exact evaluation of this parameter is in general a very costly task since it belongs to the NP-hard complexity family. In this paper, we consider an algorithm proposed by Ahmad for the special case R_{st} (source-terminal reliability) which has the advantage of using a small amount of memory. The main idea is to construct a partition of the network working states set in terms of events which we shall call branches.

We develop an extension of this algorithm by adapting the definition of the partition to the general R_K case, and we present some computational results showing the interest of this evaluation method.

Key words — Network reliability evaluation.

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

[1]This work has been funded by BID/Conicyt Project Nb. 153 and by the ECOS French-Uruguayan scientific cooperation program, Action U93E03.

$R_{\mathcal{K}}$ of this event is usually called the *$\mathcal{K}$-terminal reliability*. The problem of its evaluation has received considerable attention from the research community (see [LS86] and [Rub94] for many references). One of the reasons is that in the general case, this problem is *NP*-hard [Bal86].

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 [Ahm82], and some improvements were presented in [AJ87] and [MR88]. In this work, we describe this method for $R_{\mathcal{K}}$ evaluation, with general $\mathcal{K}$.

This work 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.

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: the analyzed undirected network (graph) topology, with $\mathcal{V} = \{1, \ldots, n\}$, the network node-set, and $\mathcal{E} = \{e_1, \ldots, e_m\}$, the network link-set;
- $\mathcal{K} \subseteq \mathcal{V}$: the terminals set; that is the subset of nodes that must communicate with each other to consider the system (network) to be operational;
- $X = (x_1, \ldots, x_m)$: the random network state vector, where x_e is 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, $r_e = \mathcal{P}(x_e = 1)$;
- Φ: the structure function associated with the *$\mathcal{K}$-terminal reliability* measure;

- $\Phi(X)$ the binary random variable "state of network $\mathcal{G}$", that is;

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

- $R_{\mathcal{K}} = \mathcal{P}(\Phi(X) = 1)$ 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 [Ahm82]. In this report, we will follow the description given in [MR88], 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 $R_{\mathcal{K}} = \mathcal{P}(C_{\mathcal{K}})$ 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 $C_{\mathcal{K}} = \bigcup_{1 \leq k \leq MP} P_k$.

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: $C_{\mathcal{K}} = \bigcup_k B_k$ 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:

$$R_{\mathcal{K}} = \mathcal{P}(C_{\mathcal{K}}) = \sum_k \mathcal{P}(B_k) \tag{1}$$

The Ahmad method is based on 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, i.e. 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:

$$\mathcal{P}(b) = \prod_{e_i \in working(b)} r_{e_i} \cdot \prod_{e_j \in failed(b)} (1 - r_{e_j}) \tag{2}$$

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)
- $\forall h:\ b_h \notin \overline{\mathcal{V}}, \quad \mathcal{V}(b_h) \stackrel{\text{def}}{=} \begin{cases} x & \text{if } b_h = x \\ y & \text{if } b_h = gy \end{cases}$
- $\forall h < H: \quad next(h) \stackrel{\text{def}}{=} min\left\{i / h < i \leq H, b_i \notin \overline{\mathcal{V}}\right\}$
- $\forall h > 1: \quad previous(h) \stackrel{\text{def}}{=} max\left\{i / 1 \leq i < h, b_i \notin \overline{\mathcal{V}}\right\}$
- $b_L \stackrel{\text{def}}{=}$ 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:

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

In Figure 1 we present the pseudocode for main loop of the Ahmad method as modified in [MR88]. 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.

Instead of looking at the evolution of the single current branch b, we can consider, as in [Ahm82], 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 $\overline{y}$.

```
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}, b̄_H);
      case (b_H ∈ K̄) and (# of b̄_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;
```

Figure 1: Pseudo-code for Ahmad method

$j := max\left\{i/1 \leq i < L, b_i \notin \overline{V}, E(i) \neq \emptyset\right\};$
do cases
case $j = 0 \longrightarrow Partition\_Completed := true;$
case $j \neq 0 \longrightarrow k := next(j);$
$y := \mathcal{V}(b_k);$
$b := (b_1, \ldots, b_{k-1}, \overline{y})$
$H := k;$
$L := previous(H);$
endcases

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

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 $E(h)$.

4 Numerical results and conclusions

The algorithm thus developed was implemented and incorporated as a basic functionality of the HEIDI tool. HEIDI is the prototype of a tool for communication network reliability analysis and design [CRU95] [CRU92]. The tool includes reliability index evaluation by exact and Monte Carlo methods, driven by a graphic interface. The user can also use this tool to study the possibility of improving the network, by a simulated annealing search of alternative configurations

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. We consider test topologies which have been previously used in related work: a version of the well known "Arpanet" network [Col87] pictured in Figure 4; a model of a subset

of Montevideo's optical fiber telephonic network [CRU92], shown in Figure 3 (this topology is the one in operation in 1992, and was manually designed by the national telecommunications company); and the dodecahedron network [Har69] shown in Figure 5.

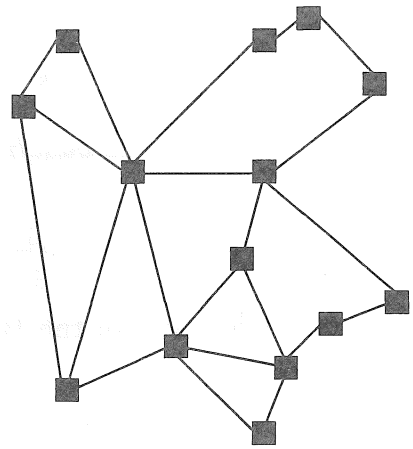

Figure 3: Montevideo's telephone optical fiber network core

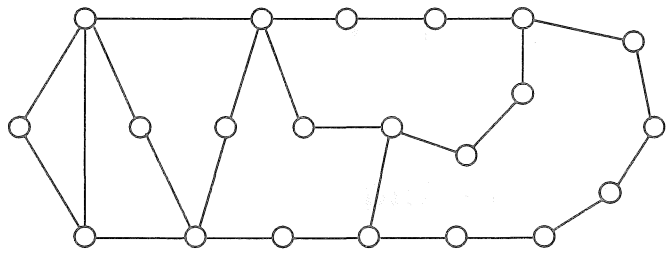

Figure 4: A version of the Arpanet

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

Table 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

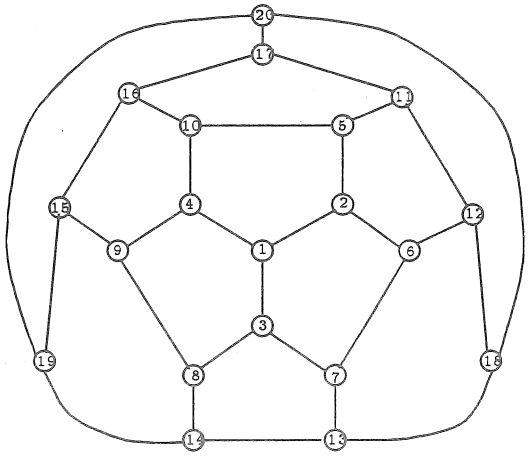

Figure 5: The Dodecahedron

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_K, and the number of branches that the method had to compute in each case. Finally, we show the running time in seconds. As expected, the efficiency of the method is strongly dependent on the size of the considered network. In the first two cases, the execution times are very small, showing the interest of using this exact evaluation method. The third case shows an increase of computation time, which may be significative if the evaluation is to be conducted in an interactive setting such as the HEIDI tool, leading to the use of other methods (such as Monte Carlo simulation) which can give an approximation of the true reliability in shorter times.

References

[Ahm82] S.H. Ahmad. A simple technique for computing network reliability. *IEEE Tr. Reliability*, R-31(1):41–44, April 1982.

[AJ87] S.H. Ahmad and A.T.M. Jamil. A modified technique for computing network reliability. *IEEE Tr. Reliability*, R-36(5):554–556, December 1987.

[Bal86] M.O. Ball. Computational complexity of network reliability analysis: An overview. *IEEE Trans. Reliab.*, R-35(3):230–239, August 1986.

[Col87] C.J. Colbourn. *The Combinatorics of Network Reliability.* Oxford University Press, New York, 1987.

[CRU92] H. Cancela, G. Rubino, and M. E. Urquhart. Optimization in communication network design (text in Spanish). In *Proceedings of the XIII CILAMCE (Iberian-Latin-American Congress of Computational Methods in Engineering)*, Porto Alegre, Brasil, November 1992.

[CRU95] H. Cancela, G. Rubino, and M.E. Urquhart. Evaluation and design of communication networks. In *Proceedings of the ICIL'95 (International Congress on Industrial Logistics*, Ouro Preto, Brazil, December 1995. University of Southampton, UK, and Naval Monterrey School, USA.

[Har69] F. Harary. *Graph Theory.* Addison-Wesley, 1969.

[LS86] M.O. Locks and A. Satyarayana (editors). Network reliability – the state of the art. *IEEE Trans. Reliab.*, R-35(3), 1986.

[MR88] R. Marie and G. Rubino. 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, Çeşme, Izmir, Turkey, 1988. Ege University.

[Rub94] G. Rubino. Tutorial: Efficient evaluation of network reliability. In *7th. Conference on Modeling Tools and Techniques*, Vienna, Austria, 1994.