An Algorithm for Computing All-terminal Reliability Bounds

Jaime Silva*†, Teresa Gomes‡,§ David Tipper ¶ Lúcia Martins‡,§ Velin Kounev ¶
*CISUC, Department of Informatics Engineering, University of Coimbra
Pólo 2, Pinhal de Marrocos 3030-290 Coimbra, Portugal
†CFC, Department of Physics, University of Coimbra
3004-516 Coimbra, Portugal
Email: silva.jaime@gmail.com
‡Department of Electrical and Computer Engineering, University of Coimbra
Pólo 2, Pinhal de Marrocos 3030-290 Coimbra, Portugal
Email: teresa@deec.uc.pt, lucia@deec.uc.pt
§INESC Coimbra, Rua Antero de Quental 199, 3000-033 Coimbra, Portugal
¶Graduate Telecommunications and Networking Program
University of Pittsburgh, Pittsburgh, PA 15260
Email: tipper@tele.pitt.edu, vkounev@pitt.edu

***Abstract*—The exact calculation of all-terminal reliability is not feasible in large networks. Hence estimation techniques and lower and upper bounds for all-terminal reliability have been utilized. We propose using an ordered subset of the mincuts and an ordered subset of minpaths to calculate an all-terminal reliability upper and lower bound, respectively. The advantage of the proposed approach results from the fact that it does not require the enumeration of all mincuts or all minpaths as required by other bounds. The performance of the algorithm is compared with the first two Bonferroni bounds, for networks where all mincuts could be calculated. The results show that the proposed approach is computationally feasible and reasonably accurate. Thus allowing one to obtain bounds when it not possible to enumerate all mincuts or all minpaths.**

***Index Terms*—All-terminal network reliability; Bonferroni bounds; Network availability; Network reduction**

I. Introduction

According to [1] reliability is the “probability that an item will perform a required function under stated conditions for a given time interval”. In communication networks, edges may fail, and are either in operational or failed state. The two terminal reliability is the probability that two specific nodes remain connected by at least one path [2]. The more general problem of calculating the probability that a set of K nodes remain connected is equally difficult; when K is equal to all the nodes in the graph, this is designated as the all-terminal reliability. The problem of determining the all-terminal reliability of a network is a well known NP-hard combinatorial problem [3], [4].

This work was motivated by the need to calculate the all-terminal availability for a potential smart grid communications network (designated netvkk in Table I) to be installed in parallel to the California Power Grid. The availability of a network (or system) is related to the fraction of time the network or system is considered to be in its up state. In this work the network will be considered in its up state if every node can communicate with every other node. Reliability is a measure of how long the network is continuously in its up state, performing its intended function (for example ensuring communication between all its nodes). Therefore a network with frequent but very short outages may have a low reliability (if the target time for flawless service is larger than the average network up time) while having a high availability (if the down time is very small compared with the up time). For historical reasons, we adopt the usual designations in the literature: two-terminal reliability, k-terminal reliability, all terminal reliability - which in fact coincide with the definition of two-terminal availability, k-terminal availability, all terminal availability, respectively.

The exact calculation of the all-terminal reliability is not feasible for large networks and so several approximations to the calculation of the all-terminal reliability have been proposed. One approach to this problem is the estimation of all-terminal reliability using Monte Carlo simulation which can be very precise, but at the cost of a high computational effort [5]. Furthermore, special care must be taken under rare event scenarios [6]. In [7] the authors use an artificial neural network for estimation of the all-terminal network reliability. They considered both networks with identical and variant link reliability, and conclude their approach could be used for network design at a reasonable computational cost. An artificial neural network was also used by Altiparmak et. al [8] to calculate the all-terminal reliability of a network with identical edges probabilities. The authors improved earlier approaches to the calculation of the all-terminal reliability of a network using the artificial neural network methodology. Three estimation techniques, crude Monte Carlo and the more sophisticated Permutation Monte Carlo and Merge Process, are considered in [9] and compared with their proposed Cross-Entropy method. The authors of [9] conclude that the Cross-Entropy method is the fastest of all three techniques.

The classical approach to calculating upper and lower bounds for network reliability is to consider minpaths, while for calculating upper and lower bounds for the network unreliability mincuts are used [10]. A cutset is defined as a minimal subset of components whose failure implies system failure [4]. A mincut is a cutset that does not contain another cutset. A pathset is defined as a minimal subset of components whose operation implies system operation [4]. A minpath is a pathset that does not contain another pathset.

In the case of two terminal reliability a minpath coincides with a minimum path, that is a path without cycles between the two specific nodes. In the case of all-terminal reliability a minpath is a spanning tree, and the number of spanning trees can be very large even for small networks.

In a network where nodes do not fail, and considering the system operational state is defined by all nodes being able to communicate, a cutset is a set of edges that when simultaneously in the failed state, ensures at least a pair of nodes can no longer communicate; similarly a pathset is a set of edges that when simultaneously in operational state, ensure all network nodes are able to communicate.

Assuming that all minpaths or mincuts can be enumerated, the all-terminal network reliability is difficult to calculate because it requires the calculation of a union of events. The calculation of an union of events can be solved by decomposing it into a union of disjoint events (disjoint products) whose probability can then be added. Since the first algorithm for the calculation of a sum of disjoint products was proposed by Abraham [11], several other algorithms have been proposed for solving this problem. Locks [12] and [13] proposed a new ordering of the minpaths seeking to reduce the number of resulting disjoint products. A note revising the explanation of the used Boolean inversion technique used in the previous algorithms was made in [14]. Additional forms of arranging the minpaths are proposed in [15], [16]. An efficient algorithm is proposed by Heidtmann [17] which considers the inversion of products of variables, instead of inverting single variables as done in previous works. An alternative method for calculating the probability of a union of events is proposed in [18], and compared with [12], [13], [17]. In [19] a new approach is proposed for pre-processing minpaths, but the authors also acknowledge that there are cases where the proposed pre-processing will not be able to reduce the number of disjoint products.

In [10] several approaches can be found for calculating network reliability. The most basic approach is based on state space enumeration. The pivotal decomposition formula, also known as link factoring, allows one to successively decompose the problem in two smaller problems, but can result in state space enumeration. However a good selection of the edges for factoring can significantly reduce the computational requirements of this approach [10]. It is worth also mentioning the well-known factoring theorem that leads to an exponential, in time, algorithm for the calculation of the network reliability proposed by Moskowitz [20], [2].

Two general approaches for calculating all-terminal reliability bounds are Bonferroni bounds [10], [21] and the Esary-Proschan bounds approximation [22]. The latter approaches are impractical for the evaluation of real large scale networks due to the fact that they need to calculate all the minpaths or mincuts which are in themselves NP-complete problems. A more recent approach proposed by Sebastio et al [23] computes the two-terminal reliability bounds using an binary decision diagram for representing the reliability graph. The latter authors also developed a new search heuristic that selects only the most important minpaths or mincuts for reducing the upper and lower reliability gap, in an iterative procedure. A procedure for reducing the upper and lower reliability gap iteratively has also been proposed by Won et. al [24] in the context of all-terminal reliability problem. In this work the authors proposed a greedy factoring process. The bounds are updated by a network factoring procedure, first selecting the edges in the network then enumerating all the states of the selected edges and finally evaluating the all-terminal reliability of the subnetworks associated with states. The greedy approach in the latter work is focused on how the selected branches, see [24], of the initial network are chosen, the latter authors proposed six greedy methodologies all of then using a minimum spanning tree or a mincut set, for starting of the factoring process that is recursively applied.

The main contribution of this work is an algorithm to calculate all-terminal reliability upper and lower bounds, which uses an ordered subset of the mincuts to calculate the reliability upper bound and an ordered subset of pathsets to calculate the reliability lower bound. Hence the proposed approach (as the algorithm in [23] for calculating two-terminal reliability bounds) does not need to enumerate all mincuts or minpaths and generally obtains bounds with the desired accuracy in a reasonable amount of time.

This article is organized as follows. In the next section a brief review on reliability bounds is presented. In section III an algorithm for the calculation of the all-terminal network availability bounds, is presented. The results of the algorithm are discussed in section IV followed by the conclusions.

II. Brief review on all-terminal reliability bounds

Consider an undirected network, represented by an undirected graph $G = (N, E)$, where $N = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes and $E = \{e_1, e_2, \ldots, e_m\}$ is the set of edges, where n is the number of nodes and m the number of edges. Each edge is an unordered pair of different elements belonging to N. An edge e has a probability of being operational given by p_e. We assume nodes do not fail and that edges fail independently. This represents random failure scenarios. The minpaths are designated by P_i, $i = 1, 2, \ldots, r$ where r is the number of spanning trees; the mincuts are designated by C_j, $j = 1, 2, \ldots, u$, where u is the number of mincuts.

The all-terminal reliability $R(G)$ and its complement the all-terminal unreliability, $U(G) = 1 - R(G)$, are given by

Eq. (1) and (2), respectively:

$$R(G) = \Pr(P_1 \cup P_2 \cup \cdots \cup P_r) \quad (1)$$

$$U(R) = \Pr(C_1 \cup C_2 \cup \cdots \cup C_u) \quad (2)$$

Equation (1), expanded using the principle of inclusion-exclusion, can be written as an alternating sum of terms. Let

$$S_w = \sum_{i_1 < i_2 < \cdots < i_w} \Pr(P_{i_1} P_{i_2} \cdots P_{i_w}) \quad (3)$$

we can write [21], [10]:

$$R(G) \leq S_1, R(G) \geq S_1 - S_2, R(G) \leq S_1 - S_2 + S_3, \ldots \quad (4)$$

Similarly, from equation (2), let

$$T_w = \sum_{i_1 < i_2 < \cdots < i_w} \Pr(C_{i_1} C_{i_2} \cdots C_{i_w}) \quad (5)$$

we can write [21], [10]:

$$U(G) \leq T_1, U(G) \geq T_1 - T_2, U(G) \leq T_1 - T_2 + T_3, \ldots \quad (6)$$

Equations (4) and (6) are termed the Bonferroni bounds [21], [10]. The number of elements in S_i is equal to $\binom{r}{i}$ and even for $i = 3$, if r is large, the number of elements can be too high to allow S_3 to be calculated in a reasonable amount of time (a similar observation can be made for T_i). From a practical point a view, bounds (4) are more effective when the p_e are small and (6) when the p_e are large.

In real networks, considering the edges have high probability of being operational, the first two Bonferroni Bounds in Eq. (6) will be designated by $R_U^B = 1 - T_1$ and $R_L^B = 1 - (T_1 - T_2)$, and will be shown in section IV to result in very close bounds.

The Esary-Proschan lower and upper bounds are given by [10]:

$$R_U^{EP} = 1 - \prod_{i=1}^{r} [1 - \Pr(P_i)] \quad (7)$$

$$R_L^{EP} = \prod_{i=1}^{u} [1 - \Pr(C_i)] \quad (8)$$

In this case to have both upper and lower bounds, one needs to enumerate all minpaths and all mincuts. Here we develop a new set of bounds based on a novel iterative procedure.

III. A NEW PROCEDURE FOR THE CALCULATION ALL-TERMINAL RELIABILITY BOUNDS

Note, that the calculation of a union of events, can be obtained as the sum of the probability of disjoint events:

$$\begin{aligned} R(G) &= \Pr(P_1 \cup P_2 \cup \cdots \cup P_r) && (9)\\ &= \Pr(P_1 \cup \bar{P}_1 P_2 \cup \cdots \cup \bar{P}_1 \bar{P}_2 \cdots, \bar{P}_{r-1} P_r) && (10)\\ &= \Pr(P_1) + \Pr(\bar{P}_1 P_2) + \cdots + && (11)\\ &\qquad \Pr(\bar{P}_1 \bar{P}_2 \cdots, \bar{P}_{r-1} P_r) && (12) \end{aligned}$$

where $\bar{P}_j$ represents the complement of event P_j. One can iteratively define a dynamic lower bound for the all-terminal reliability:

- $R_{L_1}(G) = \Pr(P_1)$,
- $R_{L_2}(G) = R_{L_1}(G) + \Pr(\bar{P}_1 P_2)$,
- $\cdots$
- $R_{L_i}(G) = R_{L_{i-1}}(G) + \Pr(\bar{P}_1 \bar{P}_2 \bar{P}_3 \cdots \bar{P}_{i-1} P_i)$.

and similarly a dynamic lower bound for the network unreliability using mincuts:

- $U_{L_1}(G) = \Pr(C_1)$,
- $U_{L_2}(G) = U_{L_1}(G) + \Pr(\bar{C}_1 C_2)$,
- $\cdots$
- $U_{L_i}(G) = U_{L_{i-1}}(G) + \Pr(\bar{C}_1 \bar{C}_2 \bar{C}_3 \cdots \bar{C}_{i-1} C_i)$.

resulting in a dynamic upper bound for the all-terminal with reliability: $R_{U_i}(G) = 1 - U_{L_i}(G)$.

In the bound calculations each new term of order i, $(\bar{P}_1 \bar{P}_2 \bar{P}_3 \cdots \bar{P}_{i-1} P_i)$ or $(\bar{C}_1 \bar{C}_2 \bar{C}_3 \cdots \bar{C}_{i-1} C_i)$ can be expressed as a sum of disjoint products, by an iteration (the i-th) of Abraham's algorithm [11] or any of its improved versions [12], [13], [14], [17]. Having obtained this sum of disjoint products the corresponding probability can easily be calculated.

Considering that we wish to obtain fast converging bounds, it seems a good strategy to include the minpaths by decreasing order of occurrence. Using the same reasoning, a similar approach is used regarding the calculation of the network unreliability, that is, mincuts should be generated in decreasing order of their probability.

Hence, the central ideas of the procedure are:

- Generating minpaths iteratively, by decreasing probability, to obtain an increasing lower bound for the all-terminal reliability.
- Generating mincuts iteratively, by decreasing probability, in order to obtain a decreasing upper bound for the all-terminal reliability.
- Leveraging the iterative nature of the algorithms for calculating the sum of disjoint products.

The pseudo-code for the proposed algorithm can be found in Algorithm 1. The algorithm starts with an undirected connected (otherwise $R(G) = 0$) graph and in order to reduce the size of the problem, the network is pruned of all the spurs (or pendants); a series reduction is also performed to remove all edges incident in nodes of degree two [25]. This last modification may also result in the need to make parallel reductions. The reduction procedure is repeated until all network nodes are at least of degree three. The edges probability of being operational are dully adjusted and a reliability correction factor (Ω in Algorithm 1) is also calculated as explained in [25]. Using this procedure, some networks can be completely reduced, like in the case of the abilene network [26] where $R(G) = \Omega$, while in other networks, like pioro40 [26], no reduction is possible and $\Omega = 1$.

In order to iteratively obtain the minpaths, by increasing probability, a k spanning trees enumeration algorithm is used. The probability of a spanning tree being operational, P_i, is given by $\prod_{e \in P_i} p_e$. Similarly the probability of all the edges in cutset C_i being simultaneously in failed state is given by $\prod_{e \in C_i} (1 - p_e)$. It is well known that these metrics can transformed into additive metrics, using logarithms. For pathset enumeration we define the cost of edge e, $c_e^p = -\log p_e$, and for cutset enumeration we define the cost $c_e^c = -\log(1-p_e)$. The cost of the minpath P_i and mincut C_i become $c^p(P_i) = \sum c_e^p$ and $c^c(C_i) = \sum c_e^c$, respectively. Therefore, generating spanning trees and mincuts by increasing cost corresponds to generating spanning trees in order of decreasing total probability (of all edges in the spanning tree being operational) and to generating cuts by decreasing probability (of every edge in the cut being in failed state), respectively.

In step 6 of Algorithm 1 the mincuts are generated iteratively by decreasing cost (that is by decreasing order of unreliability) using the algorithm proposed by Vazirani and Yannakakis [27]. The algorithm proposed by Kapoor and Ramesh [28] for the iterative generation of spanning trees by increasing cost, was used in step 11 of Algorithm 1.

For updating U'_L and R'_L algorithm KDH88, proposed in [17], was used in steps 7 and 12 of Algorithm 1, respectively. This algorithm was selected for two reasons: according to its author it is more efficient than the algorithms in [11], [12], [15], and in [18] it was also verified that this algorithm did perform better than [13]. Moreover, the performance KDH88 is not strongly dependent on the order of the pathset/cutset, hence avoiding the need of ordering the $i-1$ pathset/cutset before making them disjoint with the i-th pathset/cutset.

Therefore, Algorithm 1 iteratively reduces the width of the bounds using the generated mincuts and k-trees in conjugation with a min-sum of disjoint product algorithm.

Algorithm 1 has several stop conditions. The first stop condition holds when the difference between the upper and lower bound achieves the desired error, Δ – this should be the main stopping condition of the algorithm. The second stop condition is satisfied if the used CPU time exceeds $cpu_{\max}$ – in the experiments we considered $cpu_{\max} = 3600$ seconds. The third stop condition is of a topological nature. The first part of the third condition is related with the intersection of the current mincut with the spanning tree with higher probability of being operational, the first k-tree – see step 15 of Algorithm 1. If the this intersection results in a set with a size equal to the floor of the average degree, k, this implies the following mincuts will not contribute significantly to the diminishing of the upper bound due to the fact that the mincut reliability will be very small. In fact, it was observed that U'_L, in most cases, did not change significantly after considering the first few cuts. The second part of the third stop condition tests if the first mincut, the one the largest probability, is contained in the current generated k-tree – see step 18 of Algorithm 1. If the latter condition is met it implies that the following spanning trees (namely in networks with varying edges probability) will not contribute significantly to increase the lower bound due to the fact that their contribution to the reliability bound will be very small. In this case a large number of spanning trees would be required to effectively increase R'_L.

Algorithm 1 Algorithm for the determination of the reliability bounds.

```
Require: A connected undirected graph $G = (N, E)$, the edges reliability, the width of the interval between the bounds ($\Delta$), cpu time limit ($cpu_i \wedge cpu_{\max} \geq 1$).
Ensure: Returns the obtained reliability lower and upper bounds: $R_L$ and $R_U$.
 1: Make pendant, series and possibly parallel reductions of the network, creating a network $G'$, $G' = (N', E')$, $N' \subseteq N$, $E' \subseteq E$, with corrected edges probability of being operational, where the multiplicative conditioning factor is $\Omega : R(G) = \Omega R(G')$.
 2: $k \leftarrow \lfloor$average node degree of G'$\rfloor$
 3: $i \leftarrow 1$, $cut_{stop} \leftarrow 0$, $tree_{stop} \leftarrow 0$, $cpu_1 \leftarrow 0$
 4: while $((R'_U - R'_L) < \Delta/\Omega) \wedge (cpu_i < cpu_{\max}) \wedge (cut_{stop} = 0 \vee tree_{stop} = 0)$ do
 5:     if $cut_{stop} = 0$ then
 6:         generate the $i$-th cutset, $C_i$, using an iterative mincut enumeration algorithm
 7:         update $U'_L$, calculating iteratively the min-sum of disjoint products
 8:         update $R'_U : R'_U \leftarrow (1 - U'_L)$
 9:     end if
10:     if $tree_{stop} = 0$ then
11:         generate the $i$-th pathset, $P_i$, using a $k$-shortest spanning tree enumeration algorithm
12:         update $R'_L$, calculating iteratively the min-sum of disjoint products
13:     end if
14:     if $i \neq 1$ then
15:         if $C_i \cap P_1 = k$ then
16:             $cut_{stop} \leftarrow 1$
17:         end if
18:         if $C_1 \subset P_i$ then
19:             $tree_{stop} \leftarrow 1$
20:         end if
21:     end if
22:     $i \leftarrow i + 1$ and then gets $cpu_i$
23: end while
24: $R_L \leftarrow \Omega R'_L$
25: $R_U \leftarrow \Omega(1 - U'_L)$
```

IV. Results and discussion

The availability a_e of an edge e is the fraction of the time that edge e is operational, and has a value in [0.0,1.0]. The probability of an edge being operational is in fact the edge availability. So the bounds for all-terminal reliability described in the previous section, are in fact bounds for the all-terminal availability, that is the fraction of time that every node is able to communicate with every other node.

Here, the results of the algorithm presented in section III are compared with the Bonferroni bounds, calculated using the mincuts. The mincuts were generated using the algorithm proposed in [29], taking into account the notes in [30].

The networks used for testing the proposed algorithm are from the SNDlib [26] with the exception of netvkk (representing a possible communication network for the California power grid). The California power grid consists of over three thousand sub-stations. The minimum distance between two sub-stations is 1.2 miles, the maximum is 1074 miles, and on average is 310 miles [31]. Given the availability performance of fiber optic cable, calculated according to Eq. (13), and the distances between all sub-stations, it is unlikely that a communication network build in parallel to the existing power network (like netvkk for the California power grid) should meet the 99.999% availability recommended by the U.S. Department of Energy (DOE) [32].

All the results were obtained using a Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz processor laptop with 8G of RAM. The width of the interval between the bounds (Δ) was considered equal to 1E-4 and 1E-5, which ensures 4 and 5 digits in common, respectively, between the upper and lower bounds (unless one of the other stop conditions is attained). The edge availability a_e was calculated using the following equation[33]:

$$a_e = 0.99987^{d_e/(250\times 1.6093)} \tag{13}$$

where d_e is the distance between the two end nodes of edge e in the network (calculated assuming that the coordinates of the nodes correspond to their GPS locations). In Eq. (13) the value 1.6093 converts $miles$ to km. Structural properties of the networks studied (all from the SNDlib [26], with the exception of netvkk) are presented in Table I. In Tables I and II the total number of spanning trees of the original networks and of the reduced networks (where all nodes have at least degree 3) are presented, respectively. The number of spanning trees was calculated using the determinant of the Kirchhoff's matrix, after removing one column and row, using the LU decomposition [34], [35]. As can be seen in Tables I and II the number of spanning trees can be very large. In fact, Cayley [36] demonstrates that for a complete graph the number of distinct trees with N vertices is exactly N^{N-2}.

TABLE I
NETWORK TOPOLOGICAL INFORMATION OF THE ORIGINAL NETWORKS

Network	Nodes	Edges	Number of spanning trees
polska	12	18	5161
atlanta	15	22	2.0607E4
newyork	16	49	1.4538E10
nobel-germany	17	26	1.0995E5
geant	22	36	2.6454E7
france	25	45	1.2042E9
nobel-eu	28	41	1.6883E8
pioro40	40	89	5.0612E20
germany50	50	88	4.5872E19
netvkk	54	70	3.7803E9
ta2	65	108	1.6901E22

TABLE II
NETWORK TOPOLOGICAL INFORMATION FOR REDUCED NETWORKS

Network	Nodes	Edges	Number of spanning trees
polska	10	16	2501
atlanta	7	11	192
newyork	15	47	6.2391E5
nobel-germany	7	12	320
geant	10	21	3.8208E4
france	11	21	3.8909E4
nobel-eu	16	26	4.72554E5
pioro40	40	89	5.0612E20
germany50	39	73	9.0786E16
netvkk	14	26	4.8851E5
ta2	36	69	4.3905E15

Note, that a wide range of networks was studied from sparse networks like polska to denser networks like newyork. It is also possible to observe, in Table II, that the network reduction proposed by Shooman [25] reduces the number of spanning trees. Nevertheless, even with this reduction the number of spanning tress in the network can be very large. The calculated Bonferroni bounds for the networks are presented in Table III, without using the pendant and series reductions proposed by Shooman [25].

It is possible to observe in Table III that as the density of the network increases, for instance from polska to newyork, the time to calculate Bonferroni bounds also increases. One of the most important steps in the calculation of Bonferroni bounds is the determination of all the cutsets. As can be seen in Table III for larger and denser networks (pioro40, germany50 and ta2) it is not possible to calculate all the mincuts due to memory exhaustion. Using the procedure for network reduction in [25] the time for the calculation of Bonferroni bounds diminishes and the accuracy slightly improves in some cases, as can be seen in Table IV.

While, the time for computation of the Bonferroni bounds

TABLE III
BONFERRONI BOUNDS FOR SEVERAL NETWORKS WITH NO NETWORK REDUCTION

Network	$\mathbf{R_L^B}$	$\mathbf{R_U^B}$	Time(s)
polska	0.99999999	0.99999999	0.50
atlanta	0.99995341	0.99995341	1.11
newyork	0.99998801	0.99998801	1164.6
nobel-germany	0.99999999	0.99999999	2.12
geant	0.99999469	0.99999469	648.72
france	0.99992549	0.99992561	183.42
nobel-eu	0.99999953	0.99999953	485.20
pioro40	-	-	-
germany50	-	-	-
netvkk	0.99935935	0.99935954	1115.1
ta2	-	-	-

TABLE IV
BONFERRONI BOUNDS FOR SEVERAL NETWORKS WITH NETWORK REDUCTION

Network	$\mathbf{R_L^B}$	$\mathbf{R_U^B}$	Time(s)
polska	0.99999999	0.99999999	0.18
atlanta	0.99995341	0.99995341	0.03
newyork	0.99998801	0.99998801	517.85
nobel-germany	0.99999999	0.99999999	0.02
geant	0.99999469	0.99999469	0.48
france	0.99992557	0.99992557	0.58
nobel-eu	0.99999953	0.99999953	10.26
pioro40	-	-	-
germany50	-	-	-
netvkk	0.99935954	0.99935954	2.71
ta2	-	-	-

TABLE V
BOUNDS FOR SEVERAL NETWORKS WITH NETWORK REDUCTION USING ALGORITHM 1 WITH AN ERROR $\Delta = 1E-5$

Network	$\mathbf{R_L}$	$\mathbf{R_U}$	Time(s)
polska	0.99999994	0.99999999	0.28
atlanta	0.99995334	0.99995363	0.02
newyork	0.99998042	0.99998801	3210.66
nobel-germany	0.99999996	0.99999999	0.02
geant	0.99999363	0.99999469	0.04
france	0.99991582	0.99992559	0.09
nobel-eu	0.99999907	0.99999953	0.19
pioro40	0.98902399	0.99999999	3600
germany50	0.99992589	0.99999999	3600
netvkk	0.99935952	0.99935954	3.41
ta2	0.99312439	0.99860466	3600

TABLE VI
BOUNDS FOR SEVERAL NETWORKS WITH NETWORK REDUCTION USING ALGORITHM 1 WITH AN ERROR $\Delta = 1E-4$

Network	$\mathbf{R_L}$	$\mathbf{R_U}$	Time(s)
polska	0.99991850	0.99999999	0.25
atlanta	0.99990241	0.99995363	0.02
newyork	0.99991225	0.99998801	168.34
nobel-germany	0.99990433	0.99999999	0.01
geant	0.99993909	0.99999469	0.04
france	0.99988254	0.99992559	0.05
nobel-eu	0.99995338	0.99999953	0.07
pioro40	0.98902399	0.99999999	3600
germany50	0.99992589	0.99999999	3495.65
netvkk	0.99927854	0.99935954	0.04
ta2	0.99312439	0.99860466	3600

decreases dramatically using network reduction, for networks with a high number of edges the number of mincuts is so large that ($2^{N-1} - 1$, for complete graphs) the time and memory needed to retrieve all the mincuts becomes infeasible.

The results for the bounds calculated using Algorithm 1 presented in section III and using the network reduction can

TABLE VII
DIFFERENCE BETWEEN THE UPPER AND LOWER BOUND FOR BONFERRONI CASE AND FOR ALGORITHM 1.

Network	$\mathbf{\Delta^B}$	$\mathbf{\Delta^{B'}}$	$\mathbf{\Delta}$=1E-5	$\mathbf{\Delta}$=1E-4
polska	0.0	0.0	5.14E-8	8.15E-05
atlanta	3.32E-9	1.62E-11	2.93E-7	5.12E-05
newyork	1.08E-15	0.0	7.59E-6	7.58e-05
nobel-germany	1.61E-13	0.0	2.92E-8	9.57E-05
geant	6.91E-12	0.0	1.06E-06	5.56E-05
france	1.19E-7	3.90E-13	2.18E-7	4.30E-05
nobel-eu	1.09E-10	0.0	4.58E-7	4.61E-05
pioro40	-	-	1.10E-2	1.10E-2
germany50	-	-	7.41E-5	7.41E-05
netvkk	1.9E-7	0.0	2.14E-8	8.10E-05
ta2	-	-	5.48E-3	5.48E-3

be seen in Table V. Comparing Table V with Table IV it is possible to observe that the time of execution is in the same order of magnitude for the two methods. The exceptions are the newyork network, which is one order of magnitude higher for the proposed algorithm, and the geant and nobel-eu networks, which are lower one and two orders of magnitude, respectively, for the proposed algorithm. It can be seen in Table V that the proposed algorithm calculates the bounds for networks pioro40, germany50 and ta2, which can not be calculated using the method described by Nelson [21] to compute the Bonferroni bounds.

In Table VII data on the tightness of the bounds is presented specifically, the difference between the upper and lower bound, for Bonferroni method without network reduction, Δ^B, for Bonferroni method with network reduction, $\Delta^{B'}$, and the corresponding value for the proposed algorithm depending on Δ. Note that in the case of the germany50 network the difference between the upper and lower bound is apparently equal regardless of the Δ used – in fact the values differ less than 1E-13.

The tightness of the bounds calculated with Bonferroni method is generally lower than the obtained with the proposed algorithm (for the specified error limits of 1E-4 and 1E-5). Comparing the CPU time as registered in Tables V and VI we can verify that the CPU time decreases significantly when Δ = 1E-4, especially in the case of the newyork network; this CPU time is now less than the CPU time for Bonferroni bounds. It can also be observed a slight reduction of the CPU time of the germany50 network, but for pioro40, and ta2 the CPU time remains equal to the allowed value for $cpu_{\max}$. Note that although in the ta2 network the required error (1E-5 or 1E-4) was not achieved, the obtained bounds show that the network availability for this network has only two nines. The precision achieved by the proposed algorithm is adequate for the all-terminal availability problem with the exception of the pioro40 network. In the case of the germany50 network the results obtained ensured that an availability with four nines is guaranteed, but possibly more could be obtained. Analysing the results presented in Table V and Table VI it can be observed that the difference between the upper and lower bound is diminished by increasing the execution time of Algorithm 1. The data presented in Table VII also demonstrates that the proposed algorithm can calculate bounds where Bonferroni method was unfeasible (for pioro40, germany50 and ta2).

As expected, in the case of netvkk, the all-terminal availability (see Tables III-VI) even if a perfectly available communication network is assumed within every sub-station, only achieves three nines quite below the required 99.999% availability [32].

V. Conclusions

In this paper we proposed a new algorithm for computing all terminal reliability bounds for large networks. It was confirmed that is not possible to use the classic approaches to calculate availability bounds of large networks with methods that require all the mincuts or minpaths. We illustrated the advantages of using a procedure to reduce the network size to decrease reliability bounds computational time. Nevertheless, it was verified that the number of spanning trees continues to be very large making calculation of the reliability bounds using Bonferroni or the Esary-Proschan bounds infeasible for large networks.

The performance of the new algorithm was compared with Bonferroni bounds and it was shown that it can calculate bounds for networks where Bonferroni bounds are not computationally feasible. The results show the proposed approach is computationally feasible and reasonably accurate. Thus allowing one to obtain bounds when it not possible to enumerate all mincuts or all minpaths.

Aknowlegements

The work of Jaime Silva has been supported by ICIS Project CENTRO-07-0224-FEDER-002003. The work of Teresa Gomes and Lúcia Martins has been partially supported by the Portuguese Foundation for Science and Technology under project grant PEst-OE/ EEI/UI308/2014 and ICIS Project CENTRO-07-0224-FEDER-002003.

References

[1] "Recommendation E.800: Quality of service and dependability vocabulary," ITU-T, 1994.

[2] C. J. Colbourn, *The combinatorics of network reliability*, ser. The International Series of Monographs on Computer Science. Oxford University Press, 1987.

[3] J. Provan and M. Ball, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 777–788, 1983.

[4] M. Ball, "Computational complexity of network reliability analysis: An overview," *Reliability, IEEE Transactions on*, vol. 35, no. 3, pp. 230–239, Aug 1986.

[5] G. S. Fishman, "A monte carlo sampling plan for estimating network reliability," *Operations Research*, vol. 34, no. 4, pp. 581–594, 1986.

[6] G. Rubino and B. Tuffin, Eds., *Rare Event Simulation using Monte Carlo Methods*. Wiley, 2009.

[7] C. Srivareeratana, A. Konak, and A. E. Smith, "Estimation of all-terminal network reliability using an artificial neural network," *Computers & Operations Research*, vol. 29, no. 7, pp. 849 – 868, 2002.

[8] F. Altiparmak, B. Dengiz, and A. Smith, "A general neural network model for estimating telecommunications network reliability," *Reliability, IEEE Transactions on*, vol. 58, no. 1, pp. 2–9, March 2009.

[9] K.-P. Hui, N. Bean, M. Kraetzl, and D. Kroese, "The cross-entropy method for network reliability estimation," *Annals of Operations Research*, vol. 134, no. 1, pp. 101–118, 2005.

[10] D. R. Shier, *Network Reliability and Algebraic Structures*. Claredon Press - Oxford, 1991.

[11] J. A. Abraham, "An improved algorithm for network reliability," *IEEE Transactions on Reliability*, vol. R-38, no. 1, pp. 58–61, 1979.

[12] M. O. Locks, "A minimizing algorithm for sum of disjoint products," *IEEE Transactions on Reliability*, vol. R-36, no. 4, pp. 445–453, 1987.

[13] J. M. Wilson, "An improved minimizing algorithm for sum of disjoint products," *IEEE Transactions on Reliability*, vol. 39, no. 42-45, p. 1, April 1990.

[14] M. O. Locks and J. M. Wilson, "Note on disjoint algorithms," *IEEE transactions on Reliability*, vol. 41, no. 3, pp. 81–92, March 1992.

[15] F. Beichelt and L. Spross, "An improved Abraham-method for generating disjoint sums," *IEEE Transactions on Reliability*, vol. R-36, no. 1, pp. 70–74, April 1987.

[16] ——, "Comment on "an improved Abraham-method for generating disjoint sums"," *IEEE Transactions on Reliability*, vol. 38, no. 4, pp. 422–424, October 1989.

[17] K. D. Heidtmann, "Smaller sums of disjoint products by subproduct inversion," *IEEE Transactions on Reliability*, vol. 38, no. 3, pp. 305–311, 1989.

[18] T. Gomes and J. Craveirinha, "An alternative method for calculating the probability of an union of events," in *$\lambda\mu$13 - ESREL 2002, European Conference on System Dependability and Safety, Decision Making and Risk Management*, vol. 2, Lyon, France, 19-21 March 2002, pp. 426–430.

[19] A. Balan and L. Traldi, "Preprocessing minpaths for sum of disjoint products," *Reliability, IEEE Transactions on*, vol. 52, no. 3, pp. 289–295, Sept 2003.

[20] F. Moskowitz, "The analysis of redundancy networks," *American Institute of Electrical Engineers, Part I: Communication and Electronics, Transactions of the*, vol. 77, no. 5, pp. 627–632, Nov 1958.

[21] A. Nelson, J. R. Batts, and R. L. Beadles, "A computer program for approximating system reliability," *Reliability, IEEE Transactions on*, vol. R-19, no. 2, pp. 61–65, May 1970.

[22] J. D. Esary and F. Proschan, "A reliability bound for systems of maintained, interdependent components," *Journal of the American Statistical Association*, vol. 65, no. 329, pp. 329–338, 1970.

[23] S. Sebastio, K. S. Trivedi, D. Wang, and X. Yin, "Fast computation of bounds for two-terminal network reliability," *European Journal of Operational Research*, vol. 238, no. 3, pp. 810 – 823, 2014.

[24] J.-M. Won and F. Karray, "A greedy algorithm for faster feasibility evaluation of all-terminal-reliable networks," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 6, pp. 1600–1611, Dec 2011.

[25] A. Shooman, "Algorithms for network reliability and connection availability analysis," in *Electro/95 International. Professional Program Proceedings.*, Jun 1995, pp. 309–333.

[26] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007, http://sndlib.zib.de, extended version accepted in Networks, 2009.

[27] V. V. Vazirani and M. Yannakakis, "Suboptimal cuts: Their enumeration, weight and number," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, W. Kuich, Ed. Springer Berlin Heidelberg, 1992, vol. 623, pp. 366–377.

[28] S. Kapoor and H. Ramesh, "Algorithms for enumerating all spanning trees of undirected and weighted graphs," *SIAM Journal on Computing*, vol. 24, no. 7, pp. 247–265, 1995.

[29] W.-C. Yeh, "A simple algorithm to search for all MCs in networks," *European Journal of Operational Research*, vol. 175, no. 3, pp. 1694–1705, November 2006.

[30] T. Gomes and L. Fernandes, "A note on "a simple algorithm to search all MCs in networks"," INESC Coimbra, Tech. Rep., 2010. [Online]. Available: http://www.inescc.pt/documentos/11-2010.PDF

[31] California Energy Commission, http://www.energy.ca.gov, February 2014.

[32] Department of Energy, “Department of Energy Commission: Requirements for Smart Grid Technologies,” DoE, Oct. 2010.

[33] M. Mezhoudi and C.-H. K. Chu, “Integrating optical transport quality, availability, and cost through reliability-based optical network design,” *Bell Labs Technical Journal*, vol. 11, no. 3, p. 91104, 2006.

[34] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. New York, NY, USA: Cambridge University Press, 2007.

[35] N. Biggs, *Algebraic Graph Theory*, 2nd ed. Cambridge University Press, 1993.

[36] A. Cayley, “A theorem on trees,” *The Quarterly Journal of Pure and Applied Mathematics*, vol. 23, pp. 376–378, 1889.