A Comparison between Two All-Terminal Reliability Algorithms

Willem Pino, Teresa Gomes, and Robert Kooij

***Abstract*—Two algorithms computing the all-terminal reliability are compared. The first algorithm uses the pathsets and cutsets of the network to find lower and upper bounds respectively. The second algorithm uses decomposition and is able to give an exact outcome. The latter approach is significantly faster and able to calculate the exact value. However, for some large networks the decomposition algorithm does not give results while the path- and cutset algorithm still returns bounds. The low computation times of the proposed decomposition algorithm make it feasible to incorporate the repeated calculation of the all-terminal reliability in new types of network analysis. Some illustrations of these analyses are provided.**

***Index Terms*—All-terminal network reliability, decomposition, network availability, pathwidth.**

I. Introduction

The definition of reliability as given in [1] is the 'probability that an item will perform a required function under stated conditions for a given time interval'. Within the sphere of networks it is taken to be the probability that the network remains functioning when parts of the network are subject to failure. In this paper we consider the case of (independent) failures of links in the network. The most general version of this metric is the k-terminal reliability of the network, i.e. the probability that a set of nodes K is connected. The problem of finding this reliability is well known to be NP-hard [2]. Two common instances of the k-terminal reliability are the two-terminal and all-terminal reliability where $|K| = 2$ and $|K| = n$ respectively, where n denotes the number of nodes in the network. In this paper we will focus on the all-terminal reliability of networks with undirected links that can have different availabilities, which is also NP-hard [3].

Reliability is an important indicator of the resilience of networks and plays an important role in the design and maintenance of networks, such as computer, communication and power networks. Network communications are a critical infrastructure of today's society. Financial transactions require a reliable and secure communication infrastructure. With the introduction of the smart power grid, the correct operation of the energy distribution network also becomes dependent on the existence of a resilient communication network. The communications reliability requirements for smart power grids are discussed in [4]. The focus in that paper is on all-terminal reliability. First responders also depend on reliable communication services: they must be able to communicate with the control center and also among themselves. In [5] the complexity and challenges of providing reliable services in an evolving communications infrastructure are discussed. In [6] a set of principles for designing resilient networks is put forward, and techniques for network resilience analysis are detailed.

Techniques to find the reliability of a network give either exact answers or approximations. In the exact case the main methods used have traditionally been either path- and cutset enumeration, which uses the inclusion-exclusion principle to obtain a value for the reliability [7], or reductions followed by a factoring process [8]. These reductions are the same as those discussed in Section II of this paper. The factoring process chooses a link and splits the problem in two problems, one in which the link works and one in which it does not. Doing this creates a binary tree which can become prohibitively large. The approximation algorithms often use either Monte Carlo simulations [9] or artificial neural networks [10].

The first approach discussed in this paper is a modification of the path- and cutset enumeration to find bounds instead of an exact value. The second approach finds exact values for the all-terminal reliability using a decomposition of the problem.

The paper is organized as follows. In the next section we will talk about reductions of the network, which will be used by both approaches. In Section III we explain the method used in [11] and [12] that uses iterative path- and cutset enumeration. In Section IV the decomposition method proposed in [13] and [14] will be explained. In Section V the results of the two algorithms will be compared with each other and some extra analysis will be done on two illustrative networks. In the conclusion additional remarks about the algorithms will be made and another useful exact approach using *binary decision diagrams* (BDD) [15], [16] will be given some attention.

II. Network Reductions

Consider a network $G(V,E)$ where V is a set of nodes and E is a set of links $l = (v, w)$ with $v, w \in V$. We denote $|V| = n$ and $|E| = m$. For each $l \in E$, a_l is the link availability. We

Manuscript received July 9, 2015; revised November 29, 2015. The work of Teresa Gomes has been supported by the Fundação para a Ciência e a Tecnologia (FCT) under project grant UID/MULTI/00308/2013

Willem Pino is with the Netherlands Organization for Applied Scientific Research (TNO), the Hague, the Netherlands, he is also with the Department of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands (e-mail: willem.pino@tno.nl).

Teresa Gomes is with the Department of Electrical and Computer Engineering, University of Coimbra, Coimbra, Portugal, she is also with INESC, Coimbra, Portugal (e-mail: Teresa@deec.uc.pt).

Robert Kooij is with the Netherlands Organization for Applied Scientific Research (TNO), the Hague, the Netherlands, he is also with the Faculty of Electrical Engineering, Delft University of Technology, Delft, the Netherlands (e-mail: robert.kooij@tno.nl).

assume the link failures are independent events. Let $R(G)$ denote the reliability of network G, i.e. the probability that the network is connected.

Before an algorithm to calculate reliability is applied to the network it is possible to make some reductions [8], [17]. These reductions are transformations of the network that adjust its topology and probabilities. The resulting network retains all information crucial to calculating the reliability. Apart from adjusting the topology and probabilities they also introduce a global multiplicative factor Ω such that $R(G) = \Omega \times R(G')$ where G' is the network after the transformations. The reductions remove all spurs, or pendants, from the network. They also remove all edges adjacent to nodes of degree 2. The result is a network with minimum degree 3. Series-parallel networks fully collapse, i.e. reduce to a single edge, under these transformations [17]. This means that the reliability of series-parallel networks can be computed in linear time.

III. Iterative Pathset and Cutset Generation

A pathset is a minimal subset of components whose operation implies system operation and a cutset is a minimal subset of components whose failure implies system failure [3]. In [11] and [12] an algorithm is proposed for calculating lower and upper bounds for the all terminal reliability[1]. This algorithm uses an ordered subset of the cutsets to calculate the reliability upper bound and an ordered subset of the pathsets (spanning trees) to calculate the reliability lower bound. This means that, unlike in the exact method, it is not necessary to enumerate all cutsets or pathsets, which quickly becomes prohibitively slow. A more thorough description of this approach can be found in [11], [12].

A complete enumeration of the path- or cutsets could be used to compute the exact reliability of a network. If the probability that all links in a pathset i are operational is $\Pr(P_i)$ and the probability that all links in a cutset j have failed is $\Pr(C_j)$ then we have:

$$\begin{gathered} R(G) = \Pr(P_1 \cup P_2 \cup \dots \cup P_r), \\ 1 - R(G) = \Pr(C_1 \cup C_2 \cup \dots \cup C_u), \end{gathered} \tag{1}$$

where r is the number of pathsets and u is the number of cutsets. It is possible to obtain the union of events as the sum of the probability of disjoint events.

$$\begin{gathered} R(G) = \Pr(P_1 \cup P_2 \cup \dots \cup P_r), \\ R(G) = \Pr(P_1 \cup \bar{P}_1 P_2 \cup \dots \cup \bar{P}_1 \bar{P}_2 \dots \bar{P}_{r-1} P_r), \\ R(G) = \Pr(P_1) + \Pr(\bar{P}_1 P_2) + \dots + \Pr(\bar{P}_1 \bar{P}_2 \dots \bar{P}_{r-1} P_r), \end{gathered} \tag{2}$$

with $\bar{P}_i$ being the complement of P_i. Now it is possible to iteratively obtain lower bounds for the reliability:

$$\begin{gathered} R_{L_1}(G) = \Pr(P_1), \\ R_{L_2}(G) = R_{L_1}(G) + \Pr(\bar{P}_1 P_2), \\ \vdots \\ R_{L_r}(G) = R_{L_{r-1}}(G) + \Pr(\bar{P}_1 \bar{P}_2 \dots \bar{P}_{r-1} P_r). \end{gathered} \tag{3}$$

[1] These papers use the terms *minpath* and *mincut* to denote what we here call pathset and cutset respectively.

The same, with some slight adjustments, can be done with cutsets to obtain an upper bound. The objective is to find fast converging bounds. In order to achieve this, the algorithm generates pathsets iteratively, by decreasing probability, to obtain lower bounds. To obtain upper bounds it generates cutsets iteratively, also by decreasing probability. To find the pathsets and cutsets with the highest probability the algorithms in [18] and [19] can be used.

The algorithm keeps generating increasingly tight upper and lower bounds until the desired difference between the two is obtained, a time limit is reached or some topologically based conditions are verified (see [11], [12] for more details). In [11] the calculation of the probability of disjoint events was done using algorithm KDH88 [20]. Every sequentially selected cutset and pathset is considered in this algorithm. A more effective algorithm is presented in [12] where, similarly to the approach in [21], the pathsets and cutsets that do not contribute significantly to reducing the reliability gap are ignored. Also in [12] the maximally disjoint pathsets are first considered, followed by the pathsets generated by decreasing probability. Additionally a *binary decision diagram* (BDD) [22] was used for the calculation of the reliability lower bound instead of KDH88. This resulted in a lower execution time and an improved lower bound.

IV. Decomposition

In [13] and [14] the idea is proposed to use decomposition to obtain the reliability of a network. The main concept of this approach is that the network is divided into two parts separated by a boundary set, see Fig. 1.

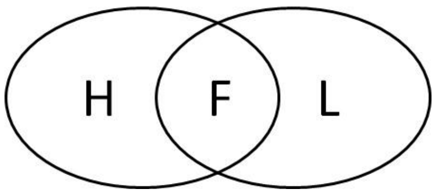

Fig. 1. Two parts of a graph G separated by a boundary set.

In Fig. 1, $H= (V', E')$ is any sub-network of G, and $L= (V'', E'')$ is the complement of H in G. The set of vertices $F= V' \cap V''$ is called the boundary set of H [23].

It is assumed that part H forms the processed part of the graph, implying that all relevant information from part H for computing the reliability of the network, can be saved in a state which pertains to the boundary set F. By adding and removing nodes from this boundary set and computing a new state from the previous state, the processed part can be expanded until the whole network is processed. In the beginning of the algorithm the whole network is unprocessed, i.e. $V = L$. When the algorithm is done the whole network is processed, i.e. $V = H$.

The state of the algorithm can be understood by looking at instances of H, situations where each link is either available or failed. There is a probability associated with every instance. In an instance of H there are several connected components. A connected component that is not connected to the boundary set can never be connected to the other components (this could only happen by means of a connection through L).

Instances that have such a connected component therefore are failed instances. These can all be discarded. Because all connected components in the relevant instances are connected to the boundary set, all instances can be categorized by giving the connected components associated to each node in the boundary set. This also means that all instances that have the same partition of the boundary set, i.e. that have a similar division of nodes of the boundary set among connected components, are equivalent. The state now consists of all possible partitions of the boundary set and an associated probability for each of these partitions. These probabilities are obtained by adding the probabilities of all instances with this partition of the boundary set. This state can be denoted by a set of pairs (π, P_π) where π is a partition on the boundary set and P_π is the associated probability. For examples concerning the state of the algorithm the reader is referred to Section IV in [13].

Each state of the algorithm can be computed from a previous state. In [23] this is done by processing either a node activation, an edge activation or a node deactivation. An activation of node v extends all partitions in the state by a singleton and does not change the probability:

$$\{(\pi, P_\pi)\} \rightarrow \{(\pi|\{v\}, P_\pi)\}. \qquad (4)$$

A deactivation of node v first looks if v is the last node in the boundary set. If this is the case then the probability of the single possible partition gives the all-terminal reliability, since the whole network is processed. If this is not the case then all partitions where v is a singleton are removed and the rest of the partitions are adjusted in the following manner:

$$\cup_{\pi\in M(\sigma,v)}\{(\pi, P_\pi)\} \rightarrow \{(\sigma, \textstyle\sum_{\pi\in M(\sigma,v)} P_\pi)\}. \qquad (5)$$

Here $M(\sigma, v)$ is the set of all partitions that can be obtained from the partition σ by inserting v in one of the subsets of the partition σ. When a link $l = \{v, w\}$ is activated equation (6) is used, where $\pi \vee l$ is the partition that results when the blocks with v and w are merged.

$$\{(\pi, P_\pi)\} \rightarrow \{(\pi, (1 - a_l) \times P_\pi), (\pi \vee l, a_l \times P_\pi + P_{\pi\vee l})\} (6)$$

The algorithm now uses these rules (equations (4) – (6)) to process the whole network, see the pseudo code in *Decomposition Algorithm*. The decomposition series referred to in the algorithm is a way of representing the path decomposition. It indicates what node has to be added, or removed, from the boundary set or which link has to be processed for each step in the algorithm.

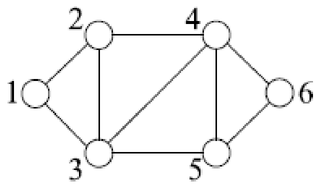

Fig. 2. An example network.

For added clarity a step in a small example will be shown. Consider the graph in Fig. 2. Assume that at some point part H of the graph consists of the nodes {1,2,3,4,5} and the links between nodes {1,2,3,4}. No links to node 5 have been processed since node 5 has just been activated. Part L is the complement of part H and therefore consists of the links of the network that are not in part H and the nodes at their endpoints, i.e. {3,4,5,6}. This means the boundary set at this point is {3,4,5}. Assume the corresponding state at this point is {(3/4/5, Pr1), (34/5, Pr2)}. The first part of this state, (3/4/5, Pr1), signifies that the probability that nodes 3, 4 and 5 are not in the same connected components, while all connected components are connected to the boundary set, in an instance of part H, is Pr1. The next step in the decomposition series is the activation of link l = (3,5). After this step, the boundary set remains the same but the state has changed according to equation (6) and now is: {(3/4/5, (1-a_l) × Pr1), (34/5, (1-a_l) × Pr2), (35/4, a_l × Pr1), (345, a_l × Pr2)}. In Section II of [23] this example network is completely processed step by step.

Decomposition Algorithm

<u>Input</u>: a connected network, edge availabilities, a decomposition series
<u>Output</u>: the all-terminal reliability (allTerminalReliability)

```
state = { ({∅},1) }
foreach step in decompositionSeries
  if step is node activation
    v = node that is activated
    use equation (4) to update the state
  else if step is node deactivation
    if step is final step
      /* Only one probability remains /*
      return final probability
    else
      v = node that is activated
      remove all partitions where v is a singleton
      use equation (5) to update the state
    end
  else if step is link activation
    (v,w) = link that is activated
    use equation (6) to update the state
  end
end
```

The total amount of steps from state to state in the algorithm is $2n + m$ and therefore linear in the input. The complexity of the algorithm depends most heavily on the size of the state, more specifically on the amount of partitions possible on the number of nodes in the boundary set, also known as the Bell number. This means that the algorithm is exponential in the maximum size of the boundary set. Therefore, it is important to find a decomposition of the graph that has a maximum boundary set which is small. A decomposition of a graph in the way needed for this algorithm is known as a path decomposition. More formally a path decomposition of a graph G is a sequence of subsets X_i of nodes of G, with two properties [24]. It must be the case that for each edge of G, there exists an i such that both endpoints of the edge belong to subset X_i and for every three indices $i \le j \le k$, $X_i \cap X_k \subseteq X_j$. The width of this decomposition is

$\max_i |X_i| - 1$. The minimum width over all path decompositions is defined as the pathwidth of a graph [24]. Because the algorithm is exponential in the maximum size of the boundary set and this is determined by the pathwidth, this algorithm will work well on networks with a small pathwidth. However, finding the pathwidth of a network (and a decomposition belonging to this pathwidth) is NP-hard [25]. Because of this, we use a heuristic to find an effective path decomposition. The heuristic used is suggested in [23]. It is a greedy heuristic that works as follows. Let the neighborhood $N(X)$ of a set X be all the nodes not in X that have a neighbor in X. Start with a vertex v and choose a vertex $w \in N(\{v\})$ that minimizes the boundary set between $\{v, w\}$ and the rest of the graph. Continue until all nodes are included. Repeat this with every node as start node and find the solution with the minimal pathwidth.

In the analysis in [23] it is shown that the algorithm runs in $O\big(p(n) \times f(pw)\big)$, where $p(n)$ is a polynomial function and $f(pw)$ is an exponential function depending only on the pathwidth of the network. The general thought of this analysis is given in the previous paragraph. The fact that the algorithm runs in $O\big(p(n) \times f(pw)\big)$ means that if the pathwidth is fixed, then the algorithm is linear in the size of the input. This is known as *fixed parameter tractable* (FPT) [26].

V. Results

A. Comparison of the Two Algorithms

In order to compare the two different algorithms we used the same networks as tested in [11], [12] and also four additional networks (*sun*, *giul39*, *zib54* and *brain*). These are networks from the SNDlib [27]. This is a library with several realistic telecommunication network design instances. Some information about the networks can be found in Table I. Note that in the case of *sun*, *giul39* and *brain* each pair of directed arcs was replaced by an undirected edge; also a duplicated edge in *zib54* was ignored. The last two columns in Table I give the amount of nodes and edges after the reductions of the network as described in Section II.

The link availability a_l of link l is given by the following equation [28]:

$$a_l = 0.99987^{d_l/250*1.6093} \tag{7}$$

where d_l is the length in kilometers of link l. This corresponds to an assumed availability of 99.987% for an end-to-end connection of 250 miles in an optical network. This comes down to planned and unplanned downtimes up to a total of around 68 minutes per year. The length of each edge in the considered networks was calculated as the distance between its end nodes GPS coordinates (given in the SNDlib).

The results for the iterative path- and cutset algorithm were obtained using an Intel(R) Core(TM) i7-3770M CPU @ 3.40GHz processor desktop with 16G of RAM. The results for the decomposition algorithm were obtained using an Intel(R) Core(TM) i5-4310M CPU @ 2.70GHz processor laptop with 8G of RAM. The desired gap between the upper and lower bound for the iterative path- and cutset algorithm used as one of the stopping criteria was 1E-6. This value was used because it allows to determine six digits for the network reliability, which is, in most practical cases, accurate enough. In Table II the numerical results for both algorithms can be found. The exact outcomes from the decomposition algorithm do indeed fall within the bounds provided by the iterative path- and cutset algorithm for each case. In Table II, it can also be seen that [12] does achieve a reliability gap less than 1E-6 for the majority of the considered networks. Furthermore, it is noteworthy that in almost every case the upper bound coincides with the exact value. This is in agreement with the fact that the cutset stop condition usually becomes true after a small number of iterations, while the algorithms continues to improve its lower bound generating spanning trees (the pathsets).

TABLE I: SNDLIB NETWORKS

Network	Nodes	Edges	Nodes Reduced	Edges Reduced
polska	12	18	10	16
atlanta	15	22	7	11
newyork	16	49	15	47
nobel-germany	17	26	7	12
geant	22	36	10	21
france	25	45	11	21
nobel-eu	28	41	16	26
pioro40	40	89	40	89
germany50	50	88	39	73
ta2	65	108	36	69
sun	27	51	25	49
india35	35	80	31	75
giul39	39	86	39	86
zib54	54	80	17	31
brain	161	166	6	9

TABLE II: RELIABILITY RESULTS

Network	Path- and cutset		Decomp.
	Lower Bound	Upper Bound	Exact Value
polska	0,99999997	0,99999999	0.99999999
atlanta	0,99995326	0,99995341	0.99995341
newyork	0,99998748	0,99998801	0.99998801
nobel-germany	0,99999992	0,99999999	0.99999999
geant	0,99999405	0,99999469	0.99999469
france	0,99992539	0,99992557	0.99992557
nobel-eu	0,99999886	0,99999953	0.99999953
pioro40	0,99900339	0,99999999	0.99999999
germany50	0,99999975	0,99999999	0.99999999
ta2	0,99835925	0,99860460	0.99860459
sun	0,99995719	0,99998650	0.99998649
india35	0,99999297	0,99999982	0.99999982
giul39	0,99882553	0,99999969	0.99999969
zib54	0,99832331	0,99832866	0.99832866
brain	0,99675410	0,99675413	0.99675413

In Table III, the computation times for both algorithms are compared. For the small networks the algorithms run in the same order of magnitude but it is evident that for the larger instances the decomposition algorithm outperforms the iterative path- and cutset algorithm, being three orders of magnitude faster. Besides this, the decomposition algorithm finds exact values as opposed to bounds, another reason to prefer this algorithm.

However, there are networks that are too large for the decomposition algorithm. For these networks the path- and cutset algorithm might still find bounds. This is the case, for example, with the *USAir97* [29] and the *c.elegans* [30] networks. The first consists of nodes representing airports and has edges between two nodes if there was a flight between those airports in 1997. The second is the neural network of the Caenorhabditis elegans worm. The *USAir97* network has 332 nodes and 2126 edges while the *c.elegans* network has 306 nodes and 2345 edges. They are both quite dense and have a pathwidth of over 20 which make a decomposition approach impossible. The path- and cutset algorithm still finds bounds, albeit not within the desired accuracy. For p = 0.99999, we chose this value for lack of accurate information on the reliability of links in these networks, the gap between the bounds was about 1E-5, which is often accurate enough to provide useful information.

TABLE III: COMPUTATION TIMES (S)

Network	Path & cutsets	Decomp.	Found pw	LB pw
polska	0.05	0.08	3	3
atlanta	0.04	0.06	4	3
newyork	36.34	0.22	6	6
nobel-germany	0.03	0.06	3	3
geant	0.05	0.08	4	4
france	0.15	0.10	5	3
nobel-eu	0.13	0.13	4	3
pioro40	3600	1.11	7	5
germany50	28.63	0.77	6	4
ta2	3600	0.62	6	4
sun	2264.34	0.49	5	4
india35	1918.34	0.45	6	4
giul39	3600	0.82	7	5
zib54	2.25	0.12	4	3
brain	0.07	0.05	3	3

In the last two columns of Table III the pathwidth found by our greedy heuristic and a lower bound on the pathwidth are given, respectively. The lower bound is actually a lower bound on the treewidth found using the Minor-Min-Width heuristic [31]. Since the treewidth of a network is always smaller than, or equal to, the pathwidth of a network, this also gives a lower bound on the pathwidth of the network. The lower bound shows our greedy heuristic produces results that are quite close to the actual pathwidth. Although a slight correlation can be seen between the width of the found path decompositions and the computation times, this correlation is not very obvious. If we consider complete networks, i.e. fully connected networks, the maximum size of the boundary set is always the number of nodes in the network. Running the algorithm on these networks makes the relation between computation time and pathwidth clearer, see Table IV, where network K_n denotes a complete network with n nodes.

TABLE IV: TIMES TO COMPUTE THE ALL-TERMINAL RELIABILITY OF FULLY CONNECTED GRAPHS

Network	Computation Times (s)
K_8	0.13
K_9	0.30
K_{10}	1.24
K_{11}	7.71
K_{12}	52.58
K_{13}	380.98

B. Illustration of New Analysis Possibilities

The reduced order of magnitude of the computation times makes it possible to conduct new types of reliability analysis. Next we will discuss two illustrations of this. We take the *ta2* and *pioro40* networks as examples in this section but this approach works for all networks in Table III.

First of all, we can revisit the formula that gives the link availabilities, (7). Imagine that a given network needs to have a certain reliability level. The availabilities of the links are known at the moment and assume this level is currently met by the network. However, in the future, as the network ages, each link might become less reliable. It would be valuable to know at which point the network no longer has a high enough reliability. In order to answer this question we assume that it is known how the formula that gives the link availabilities will change in the future. We take the original formula with an added variable α, that in some way depends on time, as an example:

$$a_l = ((1-\alpha) \times 0.99987)^{d_l/250\times 1.6093} \qquad (8)$$

Since the computation times are low we can compute the reliabilities for a range of α-values and plot a graphic that allows one to see the variation of the all-terminal reliability with α, see Fig. 3. With help of this plot it could easily be seen for which α, i.e. at which time in the future, the network no longer has the desired reliability level.

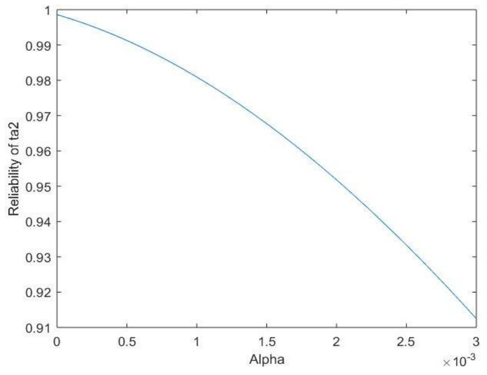

Fig. 3. The reliability of $ta2$ for different values of α.

Another issue could be that one wants to increase the reliability of a given network. Of course there is a platitude of ways to achieve this, but here we will examine one particular case. Assume that only one link from the network could be protected (or shielded). This would mean that this particular link would practically never fail, for simplicity we assume that the availability of this link would become 1. The problem is to find out which link should be protected in order to ensure the greatest increase of the reliability. Running the algorithm first on $ta2$ as the initial network and then on the adjusted $ta2$ network, with each of the link availabilities in turn set to 1, would give the plot depicted in Fig. 4. This figure can be used to determine which link in $ta2$ should be protected. Even though the links are sorted by ascending original availability this does not seem to be a strong indicator of the resulting rise in reliability. In Fig. 4, there is one link (link 83) that is obviously the most suitable candidate for protection. The underlying topology shows that this makes sense, link 83 is a pendant, if it fails the network is no longer connected. However, since this is also quite trivial, a network such as *pioro40*, that has no bridges or pendants, shows that the technique can also provide useful information in non-trivial cases, see Fig. 5. For this case, the link availabilities were multiplied by 0.7 because otherwise the differences in network reliability would be extremely small (with an initial availability of 0.99999999 there is little room for improvement).

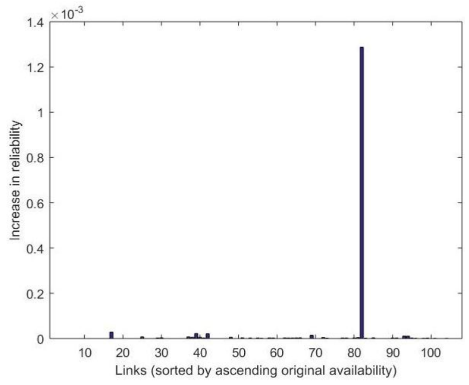

Fig. 4. The increase in reliability of *ta*2 when different links are protected.

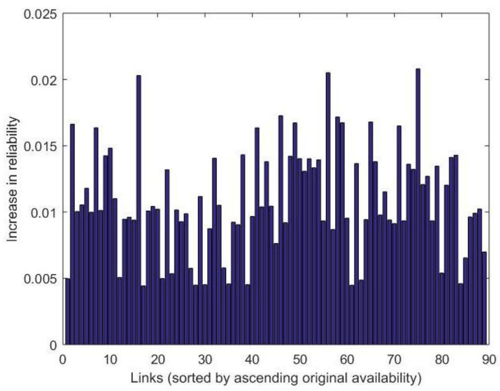

Fig. 5. The increase in reliability of *pioro40* when different links are protected. In order to make the difference between the resulting reliabilities substantial the initial link availabilities were multiplied by 0.7.

VI. Conclusion

In this paper it was shown that the decomposition method compared to the iterative path- and cutset method is not only exact, but also faster, particularly for the larger SNDlib networks. Therefore, we recommend using the decomposition algorithm instead of the iterative path- and cutset algorithm. Nevertheless, the decomposition method does not work for networks with a large pathwidth, in these cases the path- and cutset approach may be an alternative to obtain some information on network reliability, albeit possibly not as accurate as desired. Finally, we also mention a method using BDD, which also leads to an exact algorithm for obtaining the all-terminal reliability [15], [16]. In fact, the implementation in [16] is faster than our implementation, most likely because of the numbering of partitions they use to quickly find specific partitions in the state of the algorithm. Although the BDD may seem unrelated to the decomposition, we believe the underlying properties of the graph that allow the algorithms to work are actually closely related to the pathwidth of the graph. This means the two approaches could have comparable results on all networks. We do not known of any papers that compare these two approaches so this might be an interesting topic for future research. Finally, another advantage of both BDD and decomposition approaches is that they can easily be adjusted to compute the k-terminal reliability problem.

References

[1] ITU-T, "Terms and definitions related to quality of service and network performance including dependability," *Recommendation E 800*, 1994.

[2] M. O. Ball, "Complexity of network reliability computations," *Networks*, vol. 10, no. 2, pp. 153–165, 1980.

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

[4] V. Kounev, M. Lévesque, D. Tipper, and T. Gomes, "On smart grid communications reliability," in *Proc. 11th International Workshop on the Design of Reliable Communication Networks*, 2015.

[5] D. Tipper, "Resilient network design: challenges and future directions," *Telecommunication Systems*, vol. 56, no. 1, pp. 5–16, 2014.

[6] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245–1265, 2010.

[7] S. H. Ahmad, "Simple enumeration of minimal cutsets of acyclic directed graph," *IEEE Transactions on Reliability*, vol. 37, no. 5, pp. 484–487, 1988.

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

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

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

[11] J. Silva, T. Gomes, D. Tipper, L. Martins, and V. Kounev, "An algorithm for computing all-terminal reliability bounds," in *Proc. 6th International Workshop on Reliable Networks Design and Modeling*, 2014, pp. 76–83.

[12] J. Silva *et al*., "An effective algorithm for computing all-terminal reliability bounds," *Networks*, vol. 66, issue 4, pp. 282-295, 2015.

[13] J. Carlier and C. Lucet, "A decomposition algorithm for network reliability evaluation," *Discrete Applied Mathematics*, vol. 65, no. 1, pp. 141–156, 1996.

[14] A. Pönitz, "Über eine methode zur konstruktion von algorithmen für die berechnung von invarianten in endlichen ungerichteten hypergraphen," Ph.D. dissertation, Technical University Freiberg (Sachsen), 2003.

[15] F.-M. Yeh, S.-K. Lu, and S.-Y. Kuo, "OBDD-based evaluation of k-terminal network reliability," *IEEE Transactions on Reliability*, vol. 51, no. 4, pp. 443–451, 2002.

[16] G. Hardy, C. Lucet, and N. Limnios, "K-terminal network reliability measures with binary decision diagrams," *IEEE Transactions on Reliability*, vol. 56, no. 3, pp. 506–515, 2007.

[17] A. Satyanarayana and R. K. Wood, "A linear-time algorithm for computing k-terminal reliability in series-parallel networks," *SIAM Journal on Computing*, vol. 14, no. 4, pp. 818–832, 1985.

[18] S. Kapoor and H. Ramesh, "Algorithms for enumerating all spanning trees of undirected and weighted graphs," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 247–265, 1995.

[19] V. Vazirani and M. Yannakakis, "Suboptimal cuts: Their enumeration, weight and number," *Automata, Languages and Programming*, pp. 366–377, 1992.

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

[21] 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.

[22] S. B. Akers, "Binary decision diagrams," *IEEE Transactions on Computers*, vol. 100, no. 6, pp. 509–516, 1978.

[23] A. Pönitz and P. Tittmann. (2001). Computing network reliability in graphs of restricted pathwidth. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.11.7220&rep=rep1&type=pdf

[24] H. L. Bodlaender, “A partial k-arboretum of graphs with bounded treewidth,” *Theoretical Computer Science*, vol. 209, no. 1, pp. 1–45, 1998.
[25] T. Kashiwabara and T. Fujisawa, “NP-completeness of the problem of finding a minimum-clique-number interval graph containing a given graph as a subgraph,” in *Proc. Symposium of Circuits and Systems*, 1979.
[26] R. Niedermeier, “Invitation to fixed-parameter algorithms,” *Oxford Lecture Series in Mathematics and its Applications*, vol. 31, 2006.
[27] S. Orlowski, R. Wess äly, M. Pi óro, and A. Tomaszewski, “SNDlib 1.0 survivable network design library,” *Networks*, vol. 55, no. 3, pp. 276–286, 2010.
[28] 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, pp. 91–104, 2006.
[29] V. Batagelj and A. Mrvar. (2006). Pajek datasets. [Online]. Available: http://vlado.fmf.uni-lj.si/pub/networks/data/
[30] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’networks,” *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
[31] V. Gogate and R. Dechter, “A complete anytime algorithm for treewidth,” in *Proc. the 20th Conference on Uncertainty in Artificial Intelligence*, 2004, pp. 201–208.

Willem Pino was born in the Netherlands on May 23, 1991. He studied theoretical physics and philosophy at University College Utrecht before pursuing a M.Sc. degree in computing science at Utrecht University. His master research focuses on complexity of network algorithms. Now, he is doing an internship at Netherlands Organization for Applied Scientific Research (TNO) where he works on resilience metrics for distribution networks.

Teresa Gomes is an assistant professor in telecommunications at the Department of Electrical Engineering and Computers of the Faculty of Sciences and Technology, University of Coimbra, Portugal, since 1998, with tenure since 2003, and a researcher at the INESC Coimbra, which is a non-profit R&D institute. She obtained the M.Sc. degree in computer science, in 1989 and the Ph.D. degree in electrical engineering-telecommunications and electronics, in 1998, all from the University of Coimbra. Her main present interests are routing, protection and reliability analysis models and algorithms for communications networks.

Robert Kooij has a background in mathematics, he received his PhD degree in cum laude at Delft University of Technology, in 1993. From 1997 to 2003, he was employed at Royal Dutch Telecom (KPN) Research.

Since 2003, he is employed at Netherlands Organization for Applied Scientific Research (TNO), where he deals with quality aspects of ICT networks. In 2011, he became the principal scientist, conducting and managing research on Critical ICT Infrastructures. Since 2005, Robert is part-time affiliated with the Delft University of Technology, at the Faculty of Electrical Engineering, Mathematics and Computer Science. Since 2010, he is a part-time full professor with the chair "Robustness of Complex Networks".