The Reliability of a Gas Distribution Network: A Case Study

Willem Pino
TNO (Netherlands Organisation for Applied Scientific Research)
The Hague, The Netherlands
Department of Information & Computing Sciences
Utrecht University
Utrecht, The Netherlands
e-mail: wjapino@gmail.com

Daniël Worm, Ruud van der Linden
TNO (Netherlands Organisation for Applied Scientific Research)
The Hague, The Netherlands
e-mail: {daniel.worm, ruud.vanderlinden}@tno.nl

Robert Kooij
TNO (Netherlands Organisation for Applied Scientific Research)
The Hague, The Netherlands
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University
Delft, The Netherlands
e-mail: robert.kooij@tno.nl

***Abstract*—In this paper we take one of the cutting edge algorithms for computing the all-terminal reliability and the k-terminal reliability of a network and use it to compute the reliability of a real life gas distribution network in the Netherlands. To do this we estimate network properties using industry knowledge and combine several different techniques to make the problem computable. This is the first time known to us that these techniques have been applied to a large, in this case over 20000 nodes, real life network. Besides this, we show the versatility of this pathwidth-based dynamic programming algorithm by suggesting some powerful but simple modifications and argue that this network is representative for other distribution networks.**

Keywords-component; formatting; style; styling; insert

I. INTRODUCTION

There is a growing need to increase the resilience of critical infrastructures such as power and communication networks, because the impact of failures of these systems is becoming larger and because these systems are becoming more complex [1]. The introduction of hybrid systems and *smart grids*, grids that provide full visibility and control [2], results in added complexity. This is also true for gas distribution networks. Natural gas consumption forms about 25 percent of the total energy consumption in Europe [3]. As a consequence, disruptions in gas distribution networks can have major impacts. Furthermore, there is a shift from natural gas to renewable gas, which will increase the number of distribution points in the network [4]. This, in combination with increased interaction with electricity and heat grids, will cause greater uncertainty in the gas distribution networks.

Because of these developments there is a growing need to quantify the resilience of the network by means of resilience metrics. One of such metrics is the *reliability*, which is defined in the IEEE 90 standard [5] as *"the ability of a system or component to perform its required functions under stated conditions for a specific period of time"*. Because a simulation of the system is often infeasible or takes very long, faster methods to estimate the reliability are needed.

In the literature there are two state-of-the-art approaches for quantifying reliability of a network (or graph). One uses binary decision diagrams [6] [7] and the other uses the pathwidth of a graph [8]-[10]. These two approaches are based on the same structural aspects of a graph and although they differ in their practical implementation the underlying principles are related. We have chosen to work with the pathwidth approach since it is simple to implement and very agile.

Here, as is common in the literature [11], [12], we define the reliability of a network as the probability that the network, or part of it, is connected given the failure probabilities of its components. There are different forms of reliability, the most general being the k-terminal reliability. The k-terminal reliability is the probability that a specific subset of *k* nodes is connected. Two common cases of the k-terminal reliability are the all-terminal reliability and the two-terminal reliability. In the case of the all-terminal reliability *k* equals the total amount of nodes in the network. In other words, the all-terminal reliability is the probability that the network is fully connected. For the two-terminal reliability *k* = 2. This means finding the probability that there is a path between two specified nodes in the network. All these problems are well known to be NP-hard [12], [13].

In this paper we will compute both the all-terminal reliability and the k-terminal reliability of the gas distribution network in Texel, an island in the Netherlands, where we choose the subset of k nodes to be the union of the consumer nodes and the distribution node. So the k-terminal reliability is the probability that all consumers are connected to the distribution node.

The method used for computing the k-terminal reliability is an extension of the method used to compute the all-terminal reliability, of which the latter is easiest to explain. Therefore, in various sections we first discuss the approach for the all-terminal reliability and afterwards the extension needed for the k-terminal reliability. The reliability of an edge is defined as one minus the failure probability. We assume the network is undirected. It is also assumed that the nodes are completely reliable and the edges have independent failure probabilities. This assumption is defended in Section III.

The next section will introduce graph notation, the notion of pathwidth and the peculiarities of the gas distribution network. In Section III we will explain how we estimated the reliabilities of the edges. In section IV we describe which reductions we apply to the network. In Section V the heuristic for finding a path decomposition with low pathwidth will be explained. Section VI gives the details for several variants of the dynamic programming routine that can be used to compute the reliability using the path decomposition. The last three sections deal with the results, the discussion and the conclusion respectively.

II. Preliminaries

A. Graphs and Pathwidth

We consider a network, or graph, $G(V,E)$ where V is a set of nodes and E is a set of edges $e = (v,w)$ with $v,w \in V$. Furthermore, we denote $|V| = n$ and $|E| = m$. The neighborhood $N(X)$ of a set $X \subseteq V$ defined as all the nodes not in X that have a neighbor in X. For each edge $e \in E$ there is an edge reliability r_e. Let $R(G)$ denote the reliability of graph G. The subset of k nodes selected for the k-terminal reliability is denoted as K and a node $v \in K$ is called a *terminal* node. A *non-terminal* node is a node not in K.

We define an *instance* of the graph $G(V,E)$ as a situation in which every edge $e \in E$ is either operational or failed. Since each edge has a probability of being operational, a probability can also be associated with each instance.

Definition: *(Path decomposition)* A path decomposition of a graph $G(V,E)$ is a path P where every vertex $x \in P$ is associated with a set of nodes $B_x \subseteq V$ (a bag) for which it holds that $\cup_{x \in P} B_x = V$ and with the following properties:

- for any edge $(u,v) \in E$ there exists a vertex $x \in P$ such that $\{u,v\} \subseteq B_x$.
- if $v \in B_x$ and $v \in B_y$ then $v \in B_z$ for every vertex z on the path from vertex x to vertex y in P.

The width of a path decomposition is the size of the largest bag minus one. The pathwidth of a graph is the minimum width over all path decomposition of the graph. Different but equivalent definitions also exist [14].

When writing a dynamic program for a path decomposition it is very convenient if the path decomposition has a certain structure. This is why we introduce the notion of a nice path decomposition. The definition given here is slightly different than the original definition [15].

Definition: *(Nice path decomposition)* A nice path decomposition is a path decomposition with a root where each of the bags is of one of the following types:

- **End bag**: a vertex x of P with degree one and with $B_x = \emptyset$.
- **Introduce node bag**: an internal vertex x of P with one child vertex y for which $B_x = B_y \cup \{v\}$ for some node $v \notin B_y$. The bag is said to *introduce* vertex v.
- **Introduce edge bag**: an internal vertex x of P labeled with an edge $(u,v) \in E$ with one child vertex y for which $u,v \in B_x = B_y$. The bag is said to *introduce* edge (u,v).
- **Forget bag**: an internal vertex x of P with one child vertex y for which $B_x = B_y/\{v\}$ for some vertex $v \in B_y$. The bag is said to *forget* vertex v.

We additionally require that every edge is introduced exactly once.

In [16] it is shown that, given a path decomposition, a nice path decomposition of the same pathwidth pw can be found in $O(n\ pw^{O(1)})$ time. We choose one of the end bags to be the root.

B. Texel Gas Distribution Network

Texel is an island in the north of the Netherlands. It has a surface of about 170 km^2 and approximately 13000 inhabitants. The data about its gas distribution network was supplied by Alliander, a Dutch energy distributor. There is one central distribution point for the island where gas arrives from the mainland. The network can be divided into three separate parts: the part that can handle pressures of up to 8 bar, the part that can handle pressures of up to 3 bar and the part that can handle pressures of up to 0.1 bar, i.e. 100 mbar. The 8 and 3 bar parts combined form a connected subnetwork consisting of 1845 nodes and 1851 edges. The complete network consists of 20567 nodes and 20749 edges, this represents over 400 km of pipe. The maximum degree of the network is 3. These numbers indicate that the network is quite sparse. This is the case because the edges in the graphs are very closely related to the physical pipes in the actual network, i.e. the mapping of pipes to edges is almost bijective.

In the network we consider three different types of nodes: consumer nodes, connection nodes and distribution nodes. A consumer node is where the gas company delivers gas to a consumer, i.e. someone who pays for it. This does not mean that it is a single household, it can also be a factory or a camping for example. There are 7240 consumer nodes on the island and one distribution node, implying that there are 13326 connection nodes. The consumer nodes are almost always, namely in 7236 out of 7240 cases, at the end of a pendant in the network.

The network is a good instance for a case study since the complete gas network is segmented similarly throughout the Netherlands: a transportation grid that operates at high pressures and at specific points feeds the gas into distribution grids [4], such as the network under consideration in this paper.

III. Estimating Reliabilities

In order to assess the reliability of the network we need to have information about the edge reliabilities. In this paper some elementary calculations are made to produce a formula that assigns a reliability to each edge depending on the material of the edge and its length. Although industry experts indicate that these two variables are most indicative for edge reliability, more research should be done to make more accurate predictions.

We assume that nodes are completely reliable and the edges have independent failure probabilities. The assumption that edges may fail and nodes are reliable is realistic for this network since most component failures occur because of corrosion or digging accidents and therefore almost always affect pipes [17], rather than connecting points. The edge failure rates can be viewed as independent, because neither of the main causes is of a dependent nature.

The average disruption duration is known to be 1.81 hours [17]. Alliander provided data from which it is possible to deduce the probability that a pipe, or edge, of average length is disrupted during a year for different materials. This combined with the average disruption duration gives the fraction of time an edge of average length is failed. If this fraction is known then the fraction of time an edge is operational, i.e. the reliability, is also known. From this the reliability of an edge of one meter, $r_{unit,M}$, can be determined for each material *M*. For an edge with a length of *x* meter and material *M* the reliability is $(r_{unit,M})^x$.

IV. Reductions

The reductions introduced in this section reduce the size of the network while keeping all relevant information for computing the reliability. All reductions mentioned here can be found in [18] and [19]. By Ω we denote a global multiplicative factor such that

$$R(G) = \Omega \cdot R(G') \tag{1}$$

where *G'* is the network after the reductions.

A. All-Terminal Reductions

To reduce the network in the all-terminal reliability method, we start with *simple reductions*: There are three types of simple reductions: *pendant*, *series* and *parallel* reductions. Each reduction modifies the network structure and reliabilities and some also adjust Ω. Pendant reductions remove all *pendants*, i.e. nodes with degree 1, from the network. Series reductions remove all nodes with degree 2 from the network and introduce an edge between the two neighbors of the degree 2 node. Parallel reductions merge two parallel edges. By applying these reductions on a network until no more reductions are possible, all series-parallel networks will collapse, i.e. reduce to a single edge [18]. The reliability of this edge times Ω, the global multiplicative factor, gives the all-terminal reliability of the original network. This means that the all-terminal reliability of all series-parallel networks can be computed in linear time in the size of the input. Since the part of the network that can handle 8 and 3 bar is series-parallel, it collapses and the reliability of that part can be given in linear time.

When the network has not collapsed, another reduction, a *bridge split*, can be performed. A bridge is an edge that disconnects the graph if it is removed. In order for the graph to remain connected it is therefore always necessary that this edge is functional. If there is a graph *G* with a bridge *i* and two components *G'* and *G''* *t*hat are the result of removing the bridge, then

$$R(G)=r_i \cdot R(G') \cdot R(G'') \tag{2}$$

To find the reliability of the graph, remove the bridge, set $\Omega_{new} = \Omega_{old} \cdot r_i$, and recurse on the two resulting networks.

The mentioned reductions remove all nodes with degree 1 and 2 and do not increase the degree of nodes. Therefore, since we started with a graph with maximum degree 3, the resulting graph is now cubic, or 3-regular.

B. K-Terminal Reductions

The simple reductions of the all-terminal reliability method, with some minor adjustments, can still be used in the k-terminal reliability method. The main restriction is that a series reduction of a degree 2 terminal node is only possible without loss of essential information if the two neighbors of the terminal node are also terminal nodes. However, despite this restriction, using *polygon-chain* reductions it is still possible to collapse all series-parallel networks [18].

The bridge split can also be used for the k-terminal reliability method. However, some details of the method have to be changed. If one part of the network, after splitting, does not contain a terminal node then this part can be discarded. If both parts contain terminals then the endpoints of the removed bridge need to be set as terminals in the two resulting parts of the network, because after the split these points still have to be in the solution to ensure adding the bridge results in a connected solution.

V. Decomposition

Unfortunately the reductions do not always fully collapse a network, and the remaining network may still be too large for computation of the reliability in reasonable time. Therefore we use an algorithm that makes use of path decomposition. Different versions of the algorithm are proposed in the literature [8]-[10]. How well the decomposition works depends on the pathwidth of the graph.

A. Finding a Decomposition

To find a good path decomposition, i.e. one with a small pathwidth, within reasonable time, we use a heuristic [10]:

- First of all, remove all nodes with degree 2. After finding a decomposition, these can be easily inserted in the decomposition without increasing the width.
- Then, start with a node v and choose a node w ∈ N({v}) that minimizes |N(V \{v,w})|, i.e. the amount of nodes in the selected part of the graph that have a neighbor in the unselected part of the graph. Now

add a node u ∈ N({v,w}) that minimizes |N(V \ {v,w,u})|. Continue until all nodes are included.

- Repeat this with every node as start node.

The path decomposition can be deduced from the order of addition of the nodes. Select the run of the heuristic that has the corresponding path decomposition with the smallest pathwidth.

To form a nice path decomposition start with an end bag. Then take the order of nodes in the optimal run of the heuristic. This is the order of the introduce node bags. Now enter an introduce edge bag for every edge at the points in the decomposition where both nodes of the edge have been introduced. Then add forget bags as soon as all edges to a node have been introduced. Finish with an end bag.

As an illustration a possible outcome of the heuristic and the corresponding nice path decomposition for the graph in Fig. 1 are given.

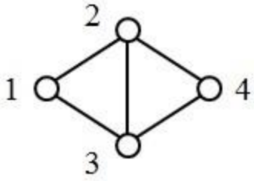

Figure 1. An example graph

The graph has a pathwidth of 2. The outcome of the heuristic could be: (1,2,3,4). The decomposition series would then be

$$(1,2,e_{(1,2)},3,e_{(1,3)},-1,e_{(2,3)},4,e_{(2,4)},-2,e_{(3,4)},-2,-4)$$

where v denotes the node activation of node v, $-v$ denotes the node deactivation of node v and $e_{(v,w)}$ is the edge activation of edge $e = (v,w)$. The resulting nice path decomposition is:

$$(\{\emptyset\},\{1\},\{1,2\},\{1,2\}_{12},\{1,2,3\},\{1,2,3\}_{13},\{2,3\}$$
$$\{2,3\}_{23},\{2,3,4\},\{2,3,4\}_{24},\{3,4\},\{3,4\}_{34},\{4\},\{\emptyset\})$$

where $\{u,v,w\}_{uv}$ denotes an introduce edge bag that introduces edge $(u,v) \in E$.

VI. Dynamic Program

The idea behind the dynamic program is that the graph is split into two parts: A processed subgraph $H = (V',E')$ and a non-processed subgraph $L = (V'',E'')$, where L is the complement of H, i.e.

$$V' \cup V'' = V, E' \cup E'' = E \text{ and } E' \cap E'' = \emptyset \qquad (3)$$

The boundary set is defined as $F = V' \cap V''$. In the algorithm all information about H needed to compute the all-terminal reliability is stored in the state of the algorithm. This state is closely related to the boundary set. At the start of the algorithm H is empty and L is the entire graph. At the end of the algorithm the situation is reversed and the whole graph is processed. To achieve this, the algorithm enlarges H and reduces L, it does so by modifying F. The algorithm does this by following the path decomposition. The bags associated with vertices in the path decomposition assume the role of F.

In order to describe how the algorithm processes the graph by means of the path decomposition, we must first introduce some notation. We denote the state by a set of pairs (π,P_π), where π is a partition of F and P_π is the probability corresponding to this partition. An element of a partition is also called a block. The partition $\pi = vw\backslash u$ is to be read as having v and w in the same block of π and u being in a separate block. Furthermore, $M(\pi,v)$ denotes the set of all partitions that can be obtained from the partition π by inserting v in one of the blocks of the partition π. Given an edge $e = (v,w)$, $\pi \vee e$ denotes the partition that results when the blocks of π containing v and w are merged. A partition π represents all instances where nodes in the same connected component are also in the same block in π.

Now we will explain why partitions on F and associated probabilities are the only details that need to be stored in the state. If F is a single node, the relevant information about H is the probability that it is connected. This is the sum of the probabilities of all instances in which H is a single connected component, i.e. the all-terminal reliability of H. The all-terminal reliability equals $R(H)\cdot R(L)$ in this case. This can be generalized for $|F| > 1$: in this case an instance of H does not have to be a single connected component, but the different connected components of the instance could become a single connected component by means of edges in L. However, if a connected component of H does not have a node in F, it can never form a connected solution. This means that all instances of H with connected components without a node in F can be discarded, because they can never result in an instance that is fully connected.

If we know how the different connected components of the instances of H are connected to F and an associated probability for all these instances, i.e. if we know (π,P_π), we can compute the probability that the whole graph will be connected. The crucial point here is that we do not need any information about the actual instances of H, only about their structures with regard to F. The number of ways the nodes in F can be connected to connected components is upper bounded by the number of partitions possible on F.

At each step in the algorithm the state is modified. How this is done depends on the type of bag. This is why we will now specify for each bag (except for the empty end bag) how it modifies the state. Afterwards some examples will be given to illustrate the operations.

- *Introduce node bag:* Extend all partitions in the state by a singleton consisting of the introduced node v and do not change the probability:

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

- *Introduce edge bag:* The introduced edge vw can either be operational or not operational:

$$\{(\pi,P_\pi)\} \rightarrow \{(\pi, (1-r_e)\cdot P_\pi), (\pi \vee e, r_e \cdot P_\pi + P_{\pi \vee e})\}$$

- *Forget bag:* When a node v is deactivated, the algorithm first determines if this is the last node in F, if so the network is fully processed, the algorithm terminates and the probability saved in the state is the probability that the whole graph is a single connected component, i.e. the all-terminal reliability. If this is not the case, all partitions where *v* is a singleton are removed, because the connected component that *v* was in could never reach the other connected components since it is no longer connected to the bag. The other partitions are adjusted as follows:

$$\bigcup_{\pi\in M(\sigma,v)} \{(\pi, P_\pi)\} \mapsto \{(\sigma, \sum_{\pi\in M(\sigma,v)} P_\pi)\}$$

An example of an introduce node bag introducing *v*, if the state consisted of nodes *w* and *u*:

$$\{(u\backslash w, P_{u\backslash w}), (uw, P_{uw})\} \mapsto \\ \{(u\backslash w\backslash v, P_{u\backslash w\backslash v}), (uw\backslash v, P_{uw\backslash v})\}$$

An example of an introduce edge bag introducing vw, with a state consisting of nodes v and w:

$$\{(v\backslash w, P_{v\backslash w}), (vw, P_{vw})\} \mapsto \\ \{(v\backslash w, (1-r_e)\cdot P_{v\backslash w}), (vw, r_e \cdot P_{v\backslash w} + P_{vw})\}$$

An example of a forget bag forgetting v, with a state consisting of nodes v, w and u (for simplicity only a part of the state is given):

$$\{(v\backslash w\backslash u, P_{v\backslash w\backslash u}), (vw\backslash u, P_{vw\backslash u}), (vu\backslash w, P_{vu\backslash w})\} \mapsto \\ (w\backslash u, P_{vw\backslash u} + P_{vu\backslash w})$$

The total amount of steps in the algorithm is 2 · *n* + *m* and therefore linear in the input. The time needed for each step is the pivotal aspect in the total running time. This time depends on the size of the state and therefore on the amount of partitions possible on *F*. This is exponentially dependent on the size of *F*. The running time of the algorithm is therefore exponential in the maximum size of *F*. The maximum size of *F* −1 is the width of the path decomposition. This means the running time of the algorithm is exponential in the width of the found path decomposition. Therefore, a path decomposition with a small width is essential to a fast algorithm.

The running time of the algorithm is linear in the input and exponential in the pathwidth, i.e. it runs in O(*p*(*n*) · *f*(*pw*)) where *p*(*n*) is a polynomial function of the input and *f*(*pw*) is an exponential function of the pathwidth [8]. When the pathwidth is fixed, the algorithm runs in polynomial time, which means the algorithm is *fixed parameter tractable* (FPT) [20].

A. Adjustments for K-Terminal Reliability

The all-terminal reliability algorithm can be modified to compute the k-terminal reliability. The main issue is that the state of the all-terminal reliability algorithm does not contain enough information. The information that needs to be added to the state is which blocks of the partitions correspond to a connected component with terminals in it. This can be done by maintaining a label for each block of each partition, 0 if there is no terminal in the corresponding connected component and 1 if there is. This information is needed because non terminal nodes do not have to be connected to other nodes. Instances with connected components without terminals need to be kept in the state since terminals in *L* could be connected through this connected component. However, if a node is deactivated, the partitions with that node as a singleton do not have to be removed from the state if the corresponding connected components do not contain terminal nodes. In this case it does not matter if these connected components can never be connected to the other connected components since they do not have to be. The instances corresponding to these partitions are still valid.

Consider a state with two nodes *v* and *w* in the boundary set. Two partitions are possible on the nodes of this boundary set, *vw* and *v**w*. However, adding labels brings the amount of possibilities to six, i.e. (*vw*,1), (*vw*,0), (*v**w*,1\1), (*v**w*,1\0), (*v**w*,0\1) and (*v**w*,0\0). As is visible from this small example, there is a risk of the state becoming much larger. However, there are some restrictions on this. For example, a subset of a partition that contains a node that is a terminal itself can never have the label 0. In the previous example, if *w* was a terminal (*vw*,0),(*v**w*,1\0) and (*v**w*,0\0) would not be possible partitions.

Another thing to take into consideration is that, if all terminals are in *H*, partitions with a singleton block with label 1 and with label 0 on all other blocks are valid instances for the k-terminal reliability problem. This is true because such partitions correspond to situations where all terminal nodes are in the same connected component. Therefore, if the node in this singleton is deactivated, the probability belonging to the partition, before discarding the partition, should be saved and added to the final reliability. This final reliability is, after all, the sum of the probabilities corresponding to all valid instances, i.e. instances where all terminals are in a single connected component.

VII. Results

A. All-Terminal Results

As mentioned in Section IV, the simple reductions collapse the 8 and 3 bar subnetwork. Unfortunately this does not happen for the full network. However, due to the tree-like structure of the graph, the reductions are substantial. After the network is split, the different sub-problems are even smaller. The resulting sizes after reductions and splitting can be found in Table I. Subnet 1 to 5 are the networks created after splitting the network. The next step is to find path decompositions. In total, for all subproblems, this can be done in 11 seconds. Running the heuristic with several random seeds often results in a path decomposition with a smaller width, meaning a faster run of the decomposition algorithm. This trade-off needs to be considered. We used 10 runs to find a good seed for the largest subnet. The smallest width found was 10. If a good path decomposition is found it can be used in consecutive runs of the algorithm if the topology remains identical. Finding a good pathwidth for the smaller subnets was not necessary since the first run gave a really small pathwidth.

TABLE I. THE NETWORK SIZE AFTER DIFFERENT REDUCTIONS

Problem	***Reductions***	***Nodes***	***Edges***	***Terminals***	***Non-terminals***
Full Net.	-	20567	20749	7241	13326
All-Term.	Simple	262	393	262	0
	Subnet 1	34	51	34	0
	Subnet 2	12	18	12	0
	Subnet 3	186	279	186	0
	Subnet 4	4	6	4	0
	Subnet 5	12	18	12	0
K-Term.	Simple	1111	1281	771	340
	Polygon	841	972	579	262
	Subnet 1	117	134	83	34
	Subnet 2	45	51	33	12
	Subnet 3	584	677	398	186
	Subnet 4	16	18	12	4
	Subnet 5	41	47	29	12

The time each step of the algorithm takes can be found in Table II. The times in brackets show the times needed if a run also has to find a good seed for the pathwidth heuristic. All results were obtained using an Intel(R) Core(TM) i5-4310M CPU @ 2.70GHz processor laptop with 8G of RAM. The all-terminal reliability found was 0.9919. This means that about 70 hours per year at least one node does not receive gas. It is difficult to assess how reasonable this is because exact statistics on this are not readily available. From industry experts it is known that a household in the Netherlands is disconnected from the network for about 1 minute per year on average. If we assume that the average amount of households that is affected by a disruption is five, a construction incident in a street for example, then the found reliability translates to households being disconnected for 3 minutes per year on average. Although this is a remarkable similarity, more accurate industry data would be preferable before drawing conclusions.

TABLE II. COMPUTATION TIMES FOR THE DIFFERENT STEPS

	All-Terminal	***K-Terminal***
Reductions	12	129
Pathwidth Heuristic	11 (x10)	11 (x10)
Decomposition	13	390
Total Method	36 (135)	531 (630)

Overall, the method works well, because the graph has a very low pathwidth compared to the graph size. This is mainly the case because the structure of the graph ensures that, after reductions, the graph is quite sparse. The low computation times allow for additional analysis of the network: We can vary the edge reliabilities and see how they relate to the overall reliability of the network, see Fig. 2.

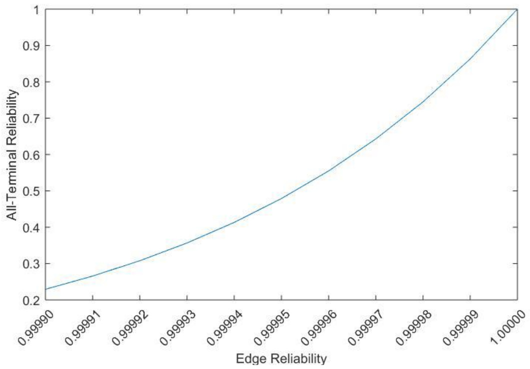

Figure 2. The all-terminal reliability of the texel gas distribution network for different uniform edge reliabilities

B. *K-Terminal Results*

The consumer nodes and the distribution node are the *k* nodes that are selected when computing the k-terminal reliability of the gas distribution network. This means that the k-terminal reliability is the probability that every consumer is connected to the distribution point. Both the all-terminal reliability and the k-terminal reliability are very dependent on the probabilities of bridges in the graph, most of which are in chains with a pendant as one end node. Every one of these must function, otherwise the network is disconnected.

Every pendant in the gas distribution network is a terminal, otherwise the edge towards it would be pointless. Because of this, both reliabilities being very dependent on the same edges, it is not expected that the all-terminal reliability and the k-terminal reliability are very different. This turns out to be a correct expectation, the value found for the k-terminal reliability is 0.9922. This is only 0.0003 more than the all-terminal reliability.

Although less reductions are possible for the k-terminal reliability and the network is therefore bigger, the consequences of this are not as large as expected. First of all, because of the similarities of overall structure (still present after reductions), the pathwidth is not much larger than it was for the all-terminal reliability. Secondly, the dispersion of terminal nodes throughout the network limits the amount of labelings per partition that are possible and thus prevents an explosion of the state.

The time needed for each step of the method and the total time needed to compute the k-terminal reliability are shown in Table II. Although for each step the times are higher, the difference is not very large because of the favorable properties the network and the locations of the terminals have.

VIII. DISCUSSION

A. *Improvements*

The whole method was implemented using MATLAB. A C or C++ implementation would most likely be faster. However, the method implemented here is already much faster than other techniques currently in use [21].

Furthermore, in various papers close attention is spent on quickly finding a number for each partition and finding specific partitions by means of that numbering [6], [8]. Our implementation, on the other hand, walks through the whole state in each step of the algorithm. This could probably be done faster.

The main restriction on the method presented here is memory. For certain networks the state becomes too large to perform some essential operations on it. A time-memory trade-off could be useful here. A hypothesis is that this could be achieved by using a tree decomposition instead of a path decomposition. The tree decomposition would have treewidth ≤ pathwidth. This means a state which uses less memory. However, merging the different boundary sets would probably be a time-expensive operation. We do not know of any paper in the literature that attempts this and are not sure about the feasibility, but we believe it is worth looking into. The most straightforward extension of the method is to compute bounds instead of an exact value. If the edge reliabilities are very high, which is almost always the case for real world networks, then partitions with many subsets become very unlikely. Eliminating these from the state would greatly reduce the size of the state whilst still giving a tight bound.

One of the major changes occurring in gas distribution networks is that more distribution points are added to the network [4]. When a network has one distribution point, the all-terminal reliability basically asks: 'what is the probability that every node is connected to a distribution point?'. However, when there are several distribution points, this is no longer the question being answered by the all-terminal reliability. If you would answer this question when there are several distribution points there could be several connected components in a valid instance, as long as each connected component contains a distribution point. A modification of the k-terminal reliability algorithm can be used to compute the probability that every node is connected to a distribution point. The label 1 on a subset of a partition now means that there is a distribution point in the corresponding connected component and the label 0 means there is not. Instead of keeping partitions where a singleton node with label 0 is deactivated, partitions where a singleton node with label 1 is deactivated should be kept. In these instances the respective connected component can never be connected to the other connected components. However, since there already is a distribution point in this connected component, it is not necessary that it is connected to the other connected components and these instances are still valid. It would even be possible to use a double labelling to find the probability that all k-terminals are connected to a distribution point, but of course the state could become very large very quickly.

B. Extensions

The most straightforward extension of the method is to compute bounds instead of an exact value. If the edge reliabilities are very high, which is almost always the case for real world networks, then partitions with many subsets become very unlikely. Eliminating these from the state would greatly reduce the size of the state whilst still giving a tight bound.

Throughout the whole paper we assumed an *undirected* network, while the physical gas distribution network is partially *directed*: in the transition nodes from high pressure to lower pressure (8 bar to 3 bar, or 3 bar to 100 mbar) the gas can only flow one way. These restrictions will result in a lower reliability than when assuming an undirected network. However, since the number of transition nodes is small, we assume the impact of these restrictions is minor. An extension to the method will be required in order to take partial directedness into account.

One of the major changes occurring in gas distribution networks is that more distribution points are added to the network [4]. When a network has one distribution point, the all-terminal reliability can be interpreted as the probability that every node is connected to a distribution point. However, when there are several distribution points, there could be several connected components in a valid instance, as long as each connected component contains a distribution point. A modification of the k-terminal reliability algorithm can be used to compute the probability that every node is connected to a distribution point. The label 1 on a subset of a partition now means that there is a distribution point in the corresponding connected component and the label 0 means there is not. Instead of keeping partitions where a singleton node with label 0 is deactivated, partitions where a singleton node with label 1 is deactivated should be kept. In these instances the respective connected component can never be connected to the other connected components. However, since there already is a distribution point in this connected component, it is not necessary that it is connected to the other connected components and these instances are still valid. It would even be possible to use a double labelling to find the probability that all k-terminals are connected to a distribution point, but of course the state could become very large very quickly.

IX. Conclusion

In this paper we have shown it is possible to use exact methods to find the all-terminal and k-terminal reliability of real life gas distribution networks of considerable size. For the Texel gas distribution network this can be done in 36 and 531 seconds respectively. The values obtained are around 0.9920. This seems to agree with values from the industry but both the assumptions made in estimating the edge reliabilities and the assumptions made in converting reliability to terms that are standard in the industry can be investigated further.

Since computing the all-terminal reliability can be done quite fast, it is possible to do some analysis of the network using this metric. For example, when a gas distribution company is considering upgrading the pipes in certain sections of the network, this metric can be used to compare the effects of different scenarios on the reliability of the network. The increase in pipe quality can be modelled by increasing the reliabilities of the edges corresponding to the pipes under consideration.

The all-terminal method works well on this network due to the tree-like structure and the low density of the initial network. The k-terminal method works well because limited reductions in this case have little effect on the pathwidth. Furthermore, since in the reduced networks the terminals are plentiful and quite evenly distributed, the state remains about the same size as it would when computing the all-terminal reliability.

We believe that a low density and many terminals throughout the network are general properties of distribution networks and that, because of this, this method would work well on other distribution networks as well. However, this is something that remains to be explored. The extensions and improvements mentioned are not complicated and can facilitate even more meaningful analysis if implemented.

Acknowledgment

The authors would like to thank Noëlle Fischer and Thomas Attema for valuable discussions and feedback.

References

[1] G. A. Pagani and M. Aiello, “The power grid as a complex network: a survey,” *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 11, pp. 2688–2700, 2013.

[2] H. Farhangi, “The path of the smart grid,” *Power and Energy Magazine, IEEE*, vol. 8, no. 1, pp. 18–28, 2010.

[3] Eurostat, “Simplified energy balances - annual data,” tech. rep., 2015.

[4] T. Weidenaar, S. Hoekstra, and M. Wolters, “Development options for the Dutch gas distribution grid in a changing gas market,” in *2011 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pp. 32–37, IEEE, 2011.

[5] A. Geraci, F. Katki, L. McMonegal, B. Meyer, J. Lane, P. Wilson, J. Radatz, M. Yee, H. Porteous, and F. Springsteel, *IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries*. IEEE Press, 1991.

[6] 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.

[7] 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.

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

[9] A. Ponitz,¨ *Uber eine Methode zur Konstruktion von Algorithmen fur die¨ Berechnung von Invarianten in endlichen ungerichteten Hypergraphen*. PhD thesis, Technical University Freiberg (Sachsen), 2003.

[10] A. Ponitz and P. Tittmann, “Computing network reliability in graphs of¨ restricted pathwidth,” *Unpublished manuscript*, 2001.

[11] C. J. Colbourn, *The Combinatorics of Network Reliability*, vol. 200. Oxford University Press New York, 1987.

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

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

[14] H. L. Bodlaender, “A partial k-arboretum of graphs with bounded treewidth,” *Theoretical computer science*, vol. 209, no. 1, pp. 1–45, 1998.

[15] T. Kloks, *Treewidth: Computations and Approximations*, vol. 842. Springer Science & Business Media, 1994.

[16] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof, “Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth,” in *Automata, Languages, and Programming*, pp. 196– 207, Springer, 2013.

[17] A. Hendriks, “Storingsrapportage gasdistributienetten,” Tech. rep., Netbeheer Nederland, 2012.

[18] 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.

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

[20] R. Niedermeier, “Invitation to fixed-parameter algorithms,” *Oxford Lecture Series in Mathematics and its Applications*, vol. 31, 2006.

[21] W. Pino, T. Gomes, and R. Kooij, “A comparison between two allterminal reliability algorithms,” *Journal of Advances in Computer Networks*, vol. 3, no. 3, pp. 284–290, 2015.