Calculating the Risk of Valve Failures when Maintaining Water Supply Networks

Jeffrey Bakker
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
j.bakker@student.utwente.nl

ABSTRACT

When performing maintenance and repairs on parts of a Water Supply Network, those parts need to be cut off in order to prevent spillage. It can occur, however, that the valves needed to cut off the target section cannot be closed, in which case other sections need to be cut off as well. Parties tasked with the maintenance of these networks need to know how many households will be affected by maintenance. This paper explains the fault in the current approach and researches what is needed to make a correct solution. One of the approaches makes use of Binary Decision Diagrams as a solution to the Inclusion-Exclusion Principle. Finally, one correct, but inefficient solution is formed and another solution is given, which still has some implementation issues.

Keywords

Water Supply Network, Risk Analysis, Formalization, Planar Graph, Binary Decision Diagrams

1. INTRODUCTION

Water Supply Networks (WSNs) describe the public infrastructure of water pipes that provide households with fresh and clean water. Parties tasked with maintenance and upkeep of this infrastructure have to compute the effective number of households (risk) that are affected by the downtime due to maintenance or repairs of failed sections of pipe.

A current solution provided by a local company[1] is computationally inefficient for larger networks and shows an error margin when compared to a Monte Carlo simulation [10] due to simplified calculations.

2. PROBLEM STATEMENT

Consider a network consisting of pipes and *valves*, then a *section* is a set of pipes bounded by valves. Let the *target* be the section that is up for maintenance; in order to prevent spillage of water, the target must be cut off from the network by closing valves. If a valve cannot be closed – due to mechanical failure or if it is not accessible – then a neighbouring section has to be closed off from the network as well. Then the goal is to calculate the effective number of affected households due to maintenance on some target section.

2.1 Design Requirements

An optimal solution to the stated problem must:

- be accurate;
- be computationally efficient; and
- be easy to implement.

2.2 Current Solution

The current solution relies on the following logic concerning Probability Theory:

$$\begin{aligned} P(A \cup B) &= P(A) + P(B) - P(A \cap B) \\ &= 1 - P(\overline{A}) * P(\overline{B}) \\ &= 1 - (1 - P(A)) * (1 - P(B)) \end{aligned}$$

The current solution then assumes that the same principle holds for a union of more than two terms:

$$P\left(\bigcup_{u \in U} u\right) = 1 - \left(\prod_{u \in U} (1 - P(u))\right)$$

This, however, does not work if the different events have some overlap (ie. they are not disjoint) leading to a miscalculation if this formula is used. Implementations using the given calculation generally yield a total probability that is a bit too high since the intersections are not properly subtracted from the result. This leads to an outcome that is higher than the correct result.

Cases like these, where the terms of the union are not disjoint, should use the Inclusion-Exclusion Principle [5] in order to account for the overlap in events.

2.3 Research Questions

The proposed research will try to answer the following research questions:

RQ1 How can the effective number of affected households be computed?

- **RQ1.1** Which sections can be affected by maintenance on the target section?
- **RQ1.2** What is the probability that a section is affected by maintenance on the target section?
- **RQ1.3** How can the total risk of maintenance on a target section be computed?

[1]The solution is based on internal expertise and has – to the best of our knowledge – no concrete scientific basis.

35th Twente Student Conference on IT July 2nd, 2021, Enschede, The Netherlands.

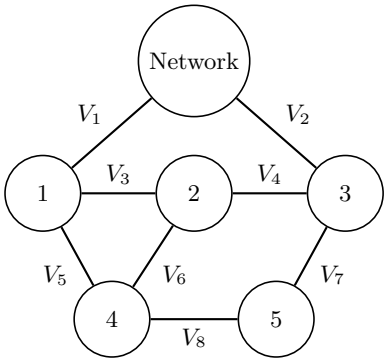


Figure 1. Example network G consisting of 6 sections and 8 valves.

RQ2 Can Fault Tree Analysis be used to compute the effective number of affected households?

RQ3 How can the performance of the computation of the risk of maintenance be improved?

3. RELATED WORK

Some preliminary investigation[2] into the problem has already been done. This preliminary investigation was not published, but it highlighted the error in the current solution and it gave an initial (not fully fledged) solution. This initial solution evaluates all different ways a given section can be affected, but this initial solution still suffers from the same problem that is mentioned in Section 2.2.

Professor John Andrews [2] describes a few techniques that lie very close to the method implemented by the aforementioned local company.

Finally, most work related to Water Supply Networks (or similar systems) describe the risk of certain components [6, 7] (ie. pipes and valves) of the network failing. Other works related to WSNs focus on which components should be given priority when performing maintenance [11]. These topics are not what this research aims for.

4. BACKGROUND

4.1 Water Supply Networks

According to Di Nardo et al [4], Water Supply Networks can be represented as planar graphs in "which pipes (and valves) correspond to links, and nodes or junctions correspond to graph nodes". This will be simplified a bit for the specific use case that is being researched:

DEFINITION 4.1. *A Water Supply Network is defined by a planar graph $G =< V, E, P(e), \gamma(v) >$ with:*

V as the set of vertices (sections);

E as the set of edges (valves);

$P(e)$ as the probability of a valve $e \in E$ failing when closing; and

$\gamma(v)$ as the amount of households in a section $v \in V$.

An example of such a network G can be seen in Figure 1.

[2]By Lisandro Jimenez, M.Sc. PDEng at the University of Twente

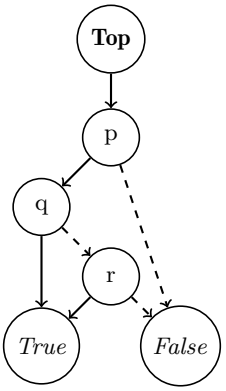


Figure 2. BDD representing $p * (q + r)$ with $high(n)$ as solid lines and $low(n)$ as dashed lines.

4.2 Risk of Maintenance

We define the *risk* (or the *effective number of affected households*) of maintenance on some target section based on the expected utility function [10]:

DEFINITION 4.2. *Let $P(u)$ denote the probability that section $u \in V$ is affected by maintenance on target section $v \in V$. Then the total risk of maintenance on target section v is:*

$$risk = \sum_{w \in V} P(w) * \gamma(w) \tag{1}$$

4.3 Binary Decision Diagrams

Binary Decision Diagrams [8] (BDDs) are a way to represent Boolean equations in a tree-like data-structure. BDDs are computationally expensive to construct from a given Boolean equation, but evaluating the probability of the top node occurring can be done relatively efficiently.

A BDD n in universe N is recursively defined with the following three variables:

- $var(n) \in N$,
- $high(n)$ is either $True$ or a BDD, and
- $low(n)$ is either $False$ or a BDD.

The probability of the top node of the BDD can be computed using the following recursive equations:

$$P(True) = 1.0$$

$$P(False) = 0.0$$

$$P(n) = P(var(n)) * P(high(n)) + (1 - P(var(n))) * P(low(n))$$

Take, for example, the independent events p, q and r, each with an arbitrary probability of occuring, then the BDD corresponding to the proposition $p * (q + r)$ can be found in Figure 2.

4.4 Fault Tree Analysis

According to E. Ruijters et al [9], "Fault Tree Analysis (FTA) is one of the most prominent techniques" in risk analysis. The technique provides a good way to visualize how a few errors can cause an entire system to fail.

For example, take – again – independent events p, q and r, each with an arbitrary probability of occurring, then the Fault Tree corresponding to the proposition $p * (q + r)$ can be found in Figure 3.

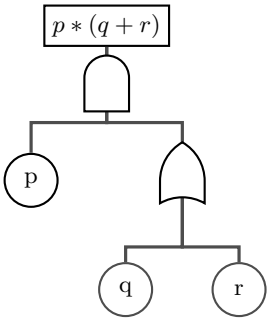


Figure 3. Fault Tree representing $p * (q + r)$.

5. METHOD

The correctness/validity of the results of the proposed research will be tested against manual calculations [10] on handcrafted networks.

5.1 Computing the Risk (RQ1)

Finding Relevant Sections (RQ1.1)

A pre-requirement of computing the effective number of affected households is finding which sections are at risk. In order to do this, a set of conditions will be formalized to define when a section is affected by maintenance on a target section.

Probability of a Section Being Affected (RQ1.2)

The next step is to compute the probability of a section being affected by the maintenance of a target section due to valve failures. This will be done by composing a set of paths from the section to the target section that could lead to the section being affected. The failure rates of the valves along the paths can then be used to compute the probability of the section being affected.

Computing the Total Risk (RQ1.3)

Given all the probabilities of relevant sections being affected, the total risk (or effective number of affected households) can be computed using Equation 1.

Finally, all of this will be composed into a single algorithm for computing the risk of maintenance on a target section.

5.2 Fault Tree Analysis (RQ2)

Next to the method for RQ1, another approach that can be considered for this computation is Fault Tree Analysis. It will be analyzed whether Fault Tree Analysis is a suitable approach and whether some modifications to traditional Fault Tree Analysis need to be implemented in order for this method to work.

5.3 Improving Performance (RQ3)

In order to improve the computational performance (measured with time complexity) of the algorithm developed in RQ1, the following options will be explored:

- Reducing the amount of nested loops;
- Saving intermediate calculations for usage in other nodes;
- Approximating the result by limiting the computational depth (by, for example, limiting the amount of edges that can fail at one time); and
- Minimizing logic using Boolean algebra to reduce the amount of computations needed.

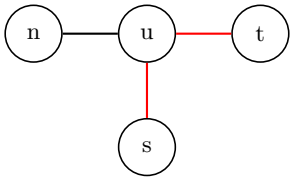


Figure 4. Example of path (from s to t) running through a dependency u with water source n, target t and current section s.

The performance of all algorithms will then be measured by taking the average runtime of 5 consecutive runs on a set of manually created networks of different sizes.

6. RESULTS

The implementations corresponding to the developed algorithms can be found in the GitHub repository jeffreybakker/wsn [3].

6.1 Computing the Risk (RQ1)

6.1.1 Finding Relevant Sections (RQ1.1)

Observing example graphs (see Appendix A) quickly leads to the insight that a section s is affected by maintenance on target section t if:

- There exists a path from s to t along which all valves fail; or
- There is some other section u that is affected by the maintenance on t; then s is affected as well if, in the graph obtained by removing all affected sections (and incident valves), there is no path from s to the water source anymore. In this scenario u is a *dependency* of s since the availability of water in s relies on the availability of water in u.

The first statement is rather obvious and should not require any further explanation. The second statement, however, might require some explanation:

EXAMPLE 6.1. *Consider a network G with target section t, network section n (the water source) and section p. Then we say p is affected by maintenance on t if there exists no uninterrupted path from p to n.*

Let q be a section such that all possible paths from p to n go through q. Then, if q is affected by maintenance on t, that must mean that p is automatically also affected.

It can also be argued that the statement in the second bullet point is an extension of the first since there exists a path from s to t through u. Therefore, the effective path that affects section s, is the path from u to t:

EXAMPLE 6.2. *Figure 4 shows an example where the path from section s to target section t goes through u. If u were to be affected by the maintenance on t, then - logically - s would also be cut off from the rest of the network. Therefore this case is an extension of the first statement.*

6.1.2 Probability of a Section being Affected (RQ1.2)

For computing the probabilities of the different sections in the network being affected, there are two clearly obvious approaches:

1. Computing all different paths from section s to target section t and taking the union of the probabilities that all valves along the given paths fail.
2. Enumerating all possibilities of failing valves in the network and adding the chance of one permutation happening to the failure probability of a section, if that section happens to be affected due to the given set of failing valves.

The first approach strongly resembles the current solution to the problem. So the probability of a section s being affected by the maintenance on target section t consists of a union of all paths (which are themselves intersections of edges) happening. If a path from s to t visits a dependency u of s, then only the path from u to t needs to be considered instead of the path from s to t.

This approach, however, is a bit more complex than is initially evident: If different paths from s to t overlap (have edges in common), then the Inclusion-Exclusion Principle [5] has to be applied. Implementing this for an unknown number of terms has proven to be more difficult than initially expected. Therefore the decision has been made to use Binary Decision Diagrams (more specifically the Python library *dd* [1]).

The second approach is less efficient, but is less likely to contain any errors as it relies on simpler techniques. Given a set of failed edges it can be computed whether the given section is still connected to the water source. If it is not, then the probability of the specific event (combination of failed edges) occurring can be added to the probability of the given section being affected.

6.1.3 Computing the Total Risk (RQ1.3)

After the total probabilities of all sections being affected by maintenance on target section t have been computed, the total risk can be computed by using the earlier defined risk calculation (see Equation 1).

Using the first approach described in Section 6.1.2 the following algorithm can be constructed:

Algorithm 1: Basic approach (with BDDs)

```
input : graph G =< V, E, P(e), f(v) >,
        target section t, water source n
output: Total risk
risk = 0.0;
for v ∈ V do
    probability = 0.0;
    if v == t then
        probability = 1.0;
    else if v == n then
        probability = 0.0;
    else
        expr = False;
        for each path from v to t do
            term = True;
            for each edge e in path do
                term = term ∧ e;
            expr = expr ∨ term;
        probability = evaluate(expr);
    risk += probability * f(v);
return risk
```

In Algorithm 1, a function *evaluate*(*expr*) is used, this refers to the recursive calculation of the probability of the top node defined in Section 4.3.

Using the second approach yields the following algorithm:

Algorithm 2: Naive approach

```
input : graph G =< V, E, P(e), f(v) >,
        target section t, water source n
output: Total risk
risk = 0.0;
for failed ∈ P(E) do
    probability = 0.0;
    for e ∈ E do
        if e in failed then
            probability += P(e);
        else
            probability += 1 - P(e);
    closed = all edges in failed reachable from t by
     just using the edges in failed;
    closed += all edges incident to the heads and tails
     of the edges in closed;
    unaffected = the set of vertices reachable from n
     without using any of the edges in closed;
    for v ∈ V \ unaffected do
        risk += f(v) * probability;
return risk
```

6.1.4 Measurements

The raw test results per algorithm per example network (defined in Appendix A) can be found in Appendix B.

The set of example networks contains a few base cases and a lot of edge cases in order to be able to determine whether an algorithm is correct or not.

When comparing the results from the Naive algorithm to the results from the manual computations (see Table 1), it becomes clear that the results from the Naive approach and the manual computations are the same.

Furthermore, the results from the basic algorithm differ in some cases from the manual computations and the results of the naive algorithm.

6.2 Fault Tree Analysis (RQ2)

Fault Trees can be used to describe and visualize the different ways (path, split up in edges) a given section can be affected by maintenance on some other section.

For example, using node C in the "Complex" example network (see Appendix A), the Fault Tree in Figure 5 can be constructed.

However, just as described earlier (in Section 6.1.2), when trying to compute the probability of the top node, one would - again - run into the problem concerning the Inclusion-Exclusion Principle [5]. A logical solution would be to use BDDs to compute the probability of the top node instead

Table 1. Results from the different algorithms on the different networks with the incorrect results highlighted.

Network	Manual	Naive	Basic with BDDs
Lisandro	706.40	706.40	706.40
Simple	365.00	365.00	365.00
Complex	312.50	312.50	312.50
Surrounded	2087.77	2087.77	2111.99
Middle	1650.00	1650.00	848.00
Triple	300.66	300.66	318.26
Combi	340.00	340.00	320.00

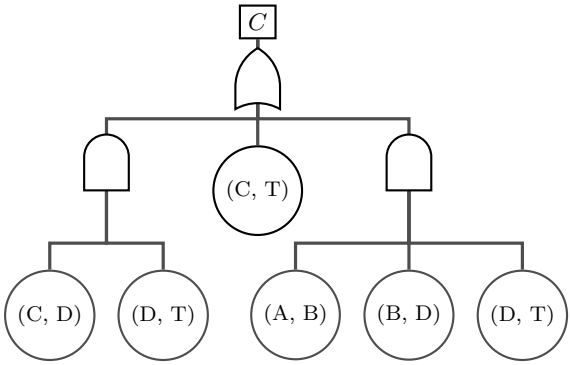


Figure 5. Fault Tree for node C in the "Complex" network.

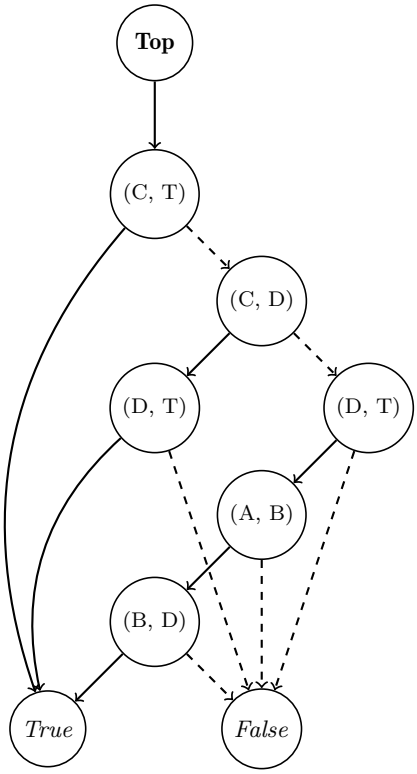


Figure 6. BDD for section C in the "Complex" network.

(see Figure 6), which is already the idea of the basic approach.

6.3 Improving Performance (RQ3)

6.3.1 Basic Approach

Even though the performance of the Basic algorithms look promising in comparison to the Naive approaches, some improvements can be made to increase the computational performance. The basic algorithm searches through the entire network for paths from some section s to target section t for every section in the network. However, if it were to go the other way around, then a lot of steps would only have to be done once. This could lead to a significant performance improvement:

Algorithm 3: Search-based approach

```
input : graph G =< V, E, P(e), f(v) >,
        target section t, water source n
output: Total risk
def construct(vertex v, list of nodes that already
 failed):
    bdds = map of vertices that fail if v also fails
     (alongside failed);
    failed += new vertices in bdds;
    for neighbour n of v do
        if n in failed then
            continue
        edge = edge from v to n;
        res = construct(n, failed);
        for each pair (w, bdd) in res do
            bdds[w] = bdds[w] ∨ (edge ∧ bdd);
    return bdds
risk = 0.0;
trees = construct(t, []);
for each pair (v, bdd) in trees do
    risk += f(v) * evaluate(bdd);
return risk
```

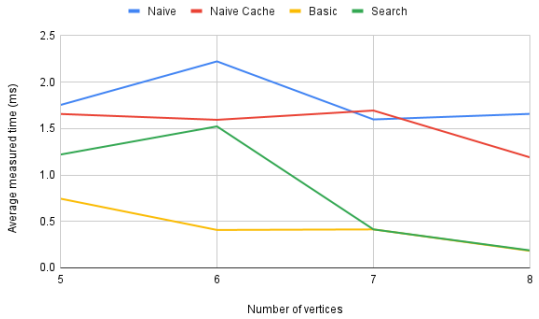


Figure 7. Average performance of algorithms for different amounts of vertices.

Algorithm 3 uses (just like Algorithm 1) the *evaluate*(*bdd*) function; the calculation of the probability of the top node of a BDD is explained in Section 4.3.

6.3.2 Naive approach

Furthermore, the naive approach iterates over all possible combinations of failing edges. It can occur, however, that different combinations of failing edges have the same impact, therefore it can be beneficial to cache some intermediate results such that some computations can be skipped.

6.3.3 Measurements

The graphs in Figures 7 and 8 show how the different algorithms perform as the input graphs get more complex. As is visible in these graphs and the overall performance results (in Appendix B), the search-based algorithm appears to perform worse than the basic algorithm with BDDs.

Next to that, as seen in Appendix B, the results from the search-based algorithm differ from the manual calculations.

Furthermore, the variant of the naive approach with caching only yields a minor speed improvement (8% to 10% in most cases) if it even improves the speed in that case.

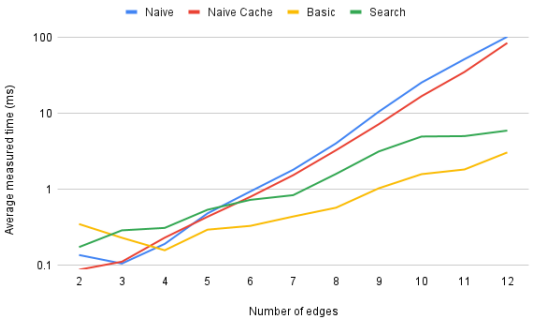


Figure 8. Average performance of algorithms for different amounts of edges.

7. CONCLUSION

The main problem is solved; one correct solution is given as well as some other approaches. However, these other approaches still have some implementation issues.

7.1 Computing the Risk (RQ1)

Both the basic and naive approaches to the given problem work quite well; the naive approach does not appear to have any errors – even in the edge cases – so it would suffice to say that this is a correct solution to the given problem; the basic solution, however, produces some small errors in some of the edge cases. The problem here lies in how the dependencies are computed and applied. If this is fixed, then the basic algorithm would also be correct.

This problem is purely an implementation issue that could not be solved during this research.

7.2 Fault Tree Analysis (RQ2)

Technically, Fault Tree Analysis can be used to compute the effective number of affected households by using FTA on each of the other sections in the network. However, since FTA is not that different from the basic approach and it does not offer any additional functionality over BDDs, it is not a viable solution to the given problem.

The only real benefit of FTA in this case would be that it might be handy to visualize how a given section may be implicated in the maintenance on another section, but BDDs are – subjectively – easier to read and understand.

7.3 Improving Performance(RQ3)

As is evident from the results, the search-based algorithm actually performed worse than the basic algorithm. Next to that, the search-based algorithm still has the same flaws concerning the computation and application of dependencies that the basic approach also has.

Furthermore, the speed improvement of the naive approach with caching is too small to really make a difference in its ability to compute the risk for larger networks.

Overall the computational performance of the computation of the risk of maintenance has not been improved (significantly).

8. DISCUSSION

8.1 Inclusion-Exclusion Principle

Due to the difficulties of trying to implement a solution that works with the Inclusion-Exclusion Principle, a choice was been made to go with Binary Decision Diagrams instead. These, however are not the most computationally efficient to create. This contrast can be seen in Table B.2 (in Appendix B), where also the performance of the Basic algorithm is shown without the creation of a BDD.

Even though the approach using BDDs has yielded correct results, where earlier iterations of the algorithm did not, this does pose a problem with regard to the third research question.

8.2 Implementation Issues

As explained, there are some issues with the implementation that deals with the dependencies of sections and their application in the risk calculation. Because of this implementation issue all algorithms (except for the naive implementations) yield wrong results in some edge cases.

8.3 Handcrafted Networks

The networks on which the different approaches have been tested are – as mentioned – handcrafted (or imaginary) networks that try to simulate real situations. However, a general conclusion for the problem cannot be drawn since these networks are not based on real-world data.

Furthermore, real networks are larger than the networks that are tested with in this research. Using larger networks will significantly increase the runtimes of the different algorithms as has become evident from Figure 8.

9. FUTURE WORK

9.1 Replacing BDDs

As acknowledged before (in Section 8.1), BDDs have not been the most computationally efficient solution to the problem. Future research could try to solve the problem regarding the Inclusion-Exclusion Principle (as described in Section 6.1.2). This might give a significant speed improvement if it is implemented well enough; a small insight into this speed improvement is given by Table B.2 (in Appendix B): This table shows how just iterating the paths and summing their probabilities is significantly faster than the version of the algorithm that also creates a BDD.

Another approach would be to find a way to include BDDs in such a way that their creation is not as demanding (time-wise).

9.2 Improving Accuracy

At the moment, there are a few edge cases that yield an incorrect result for some of the algorithms. This is, as explained (see Section 7.1), due to an error in the way that the earlier mentioned dependencies (see Section 6.1.1) are computed and applied. Future research could go into how to set up a correct data structure for these dependencies and how to compute and apply them properly.

9.3 Other Algorithms

Only a few approaches to solve the problem have been tried in this research. Future research could look into different approaches to solve the problem that might be less error-prone and more computationally efficient.

10. ACKNOWLEDGEMENTS

I would like to thank my supervisors Matthias Volk and Moritz Hahn for their supervision and for their help when I got stuck during my research.

Furthermore I would like to thank Lisandro Jimenez for his preliminary investigation into the problem.

Finally, I want to express my appreciation for my friends who have helped me to proof-read my submissions on multiple occasions.

11. REFERENCES

[1] tulip-control/dd, June 2021. https://github.com/tulip-control/dd.

[2] J. Andrews. Next Generation Prediction Methodologies and Tools for Engineering Risk Assessment. Talk at PrimaVera Colloquium, 2021. https://www.youtube.com/watch?v=uCn9s3IfC6Y.

[3] J. Bakker. jeffreybakker/wsn, June 2021. https://github.com/jeffreybakker/wsn.

[4] A. Di Nardo, M. Di Natale, C. Giudicianni, R. Greco, and G. F. Santonastaso. Water Supply Network Partitioning Based On Weighted Spectral Clustering. In H. Cherifi, S. Gaito, W. Quattrociocchi, and A. Sala, editors, *Complex Networks & Their Applications V*, volume 693, pages 797–807. Springer International Publishing, Cham, 2017.

[5] K.-C. Lin, I.-E. Liao, T.-P. Chang, and S.-F. Lin. A frequent itemset mining algorithm based on the Principle of Inclusion–Exclusion and transaction mapping. *Information Sciences*, 276:278–289, Aug. 2014.

[6] M. S. Marlim, G. Jeong, and D. Kang. Identification of Critical Pipes Using a Criticality Index in Water Distribution Networks. *Applied Sciences*, 9(19):4052, Sept. 2019.

[7] A. Matyash, I. Usenko, R. Myagkohlib, and S. Kostenko. Estimation of failure-free operation of metal water pipes. *Eastern-European Journal of Enterprise Technologies*, 3(1 (87)):35–41, June 2017.

[8] A. Rauzy. A brief introduction to binary decision diagrams. *Journal Europeen des Systemes Automatises*, 30(8):1033–1050, 1996.

[9] E. Ruijters and M. Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15-16:29–62, Feb. 2015.

[10] S. J. Russell, P. Norvig, and E. Davis. *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, 3rd ed edition, 2010.

[11] Y. Zhang, S. Li, Y. Zheng, and Y. Zou. Multi-model based pressure optimization for large-scale water distribution networks. *Control Engineering Practice*, 95:104232, Feb. 2020.

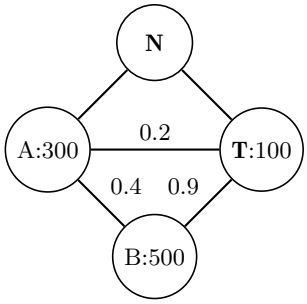
APPENDIX

A. EXAMPLE NETWORKS

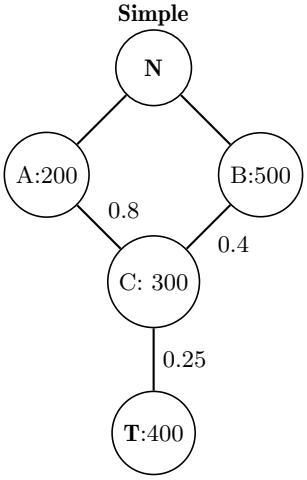
In the networks below, "**N**" represents the rest of the network (the water source) and "**T**" represents the target section that is up for maintenance.

The network "Lisandro" is named after Lisandro Jimenez, M.Sc. PDEng at the University of Twente since he created this network in his preliminary investigation (see Section 3).

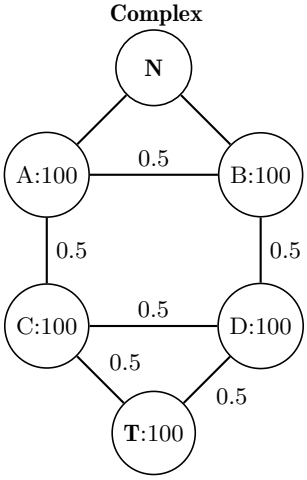
Lisandro



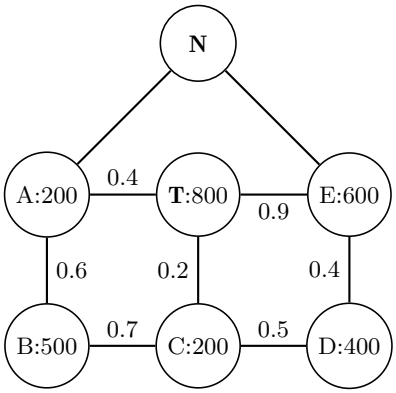
Simple



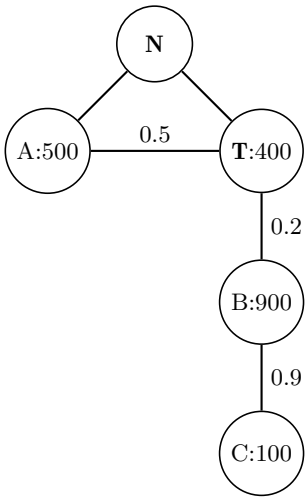
Complex



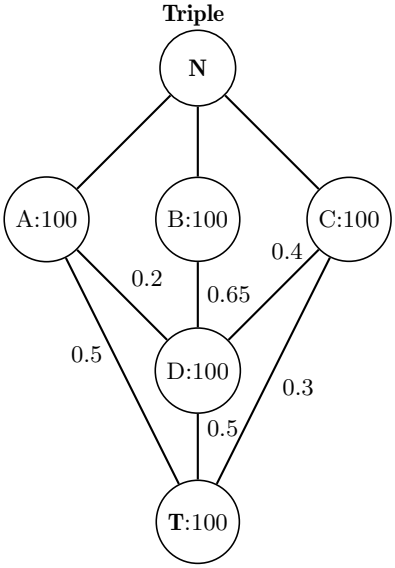
Surrounded



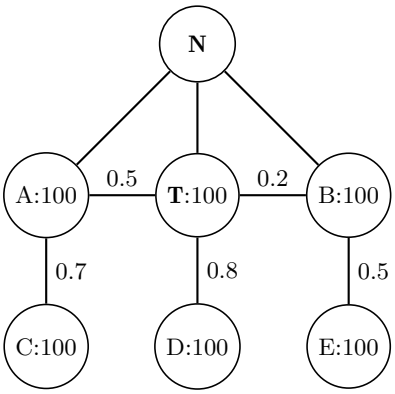
Middle



Triple



Combi



B. TEST RESULTS

B.1 Results

Network	Manual	Naive	Naive with Cache	Basic	Search
Lisandro	706.40	706.40	706.40	706.40	706.40
Simple	565.00	565.00	565.00	565.00	565.00
Complex	312.50	312.50	312.50	312.50	312.50
Surrounded	2087.77	2087.77	2087.77	2111.99	1930.50
Middle	1650.00	1650.00	1650.00	848.00	848.00
Triple	300.66	300.66	300.66	318.26	300.66
Combi	340.00	340.00	340.00	320.00	320.00

Table B.1: Results from the different networks defined in Appendix A with the used algorithms. The incorrect results are highlighted.

B.2 Performance measurements

$\|V\|$	$\|E\|$	$\|V\| * \|E\|$	Naive	Naive with Cache	Basic without BDDs	Basic with BDDs	Search
3	2	6	0.136	0.087	0.015	0.348	0.173
3	3	9	0.092	0.106	0.019	0.153	0.281
4	3	12	0.118	0.116	0.035	0.306	0.294
4	4	16	0.191	0.267	0.037	0.163	0.408
4	5	20	0.573	0.575	0.053	0.426	0.882
4	6	24	1.043	0.994	0.059	0.364	1.341
5	4	20	0.190	0.193	0.046	0.151	0.211
5	5	25	0.533	0.415	0.068	0.307	0.435
5	6	30	0.977	0.778	0.061	0.536	0.981
5	7	35	1.754	1.657	0.078	0.745	1.220
6	5	30	0.337	0.313	0.061	0.148	0.294
6	6	36	0.936	0.742	0.063	0.279	0.410
6	7	42	2.223	1.594	0.125	0.408	1.523
6	8	48	4.261	3.254	0.161	0.692	2.744
6	9	54	12.163	7.795	0.265	1.517	4.866
6	10	60	26.420	18.914	0.281	2.159	7.669
7	6	42	0.788	0.668	0.080	0.138	0.170
7	7	49	1.598	1.694	0.096	0.414	0.414
7	8	56	4.043	3.177	0.173	0.613	1.543
7	9	63	9.732	7.232	0.259	0.879	2.945
7	10	70	28.387	15.991	0.417	1.511	3.877
7	11	77	44.680	38.632	0.334	2.190	5.489
8	7	56	1.658	1.190	0.130	0.183	0.188
8	8	64	3.774	3.375	0.126	0.409	0.491
8	9	72	9.719	6.600	0.236	0.695	1.642
8	10	80	21.807	15.731	0.351	1.077	3.331
8	11	88	58.927	31.639	0.442	1.457	4.551
8	12	96	102.433	84.727	0.486	3.058	5.935

Table B.2: Runtimes of the different algorithms on networks of different sizes in milliseconds.