



ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Robust Placement of Water Quality Sensors in Water Distribution Networks

Securing Critical Infrastructures against Deadly Terrorist Attacks

MASTER'S THESIS IN OPERATIONS RESEARCH AND QUANTITATIVE LOGISTICS
MASTER ECONOMETRICS AND MANAGEMENT SCIENCE

Author:

C.J. DE WINTER, 385910

Supervisor EUR:

A.P.M. WAGELMANS

Supervisors TNO:

R.E. KOOLJ

D. WORM

Second Reader:

W. VAN DEN HEUVEL

Supervisor SUTD:

V.R. PALLETI

January 26, 2018

Abstract

Water Distribution Networks (WDNs) are used to supply drinking water to billions of people around the world. An upcoming threat to these networks are accidental and deliberate contaminations which can lead to poisoned water, many fatalities and large economic consequences. In order to protect against these intrusions or attacks, an efficient sensor network with a limited number of sensors should be placed in a WDN. It is assumed that attacks can only happen at certain vulnerable nodes in the network. In this thesis, we focus on improving the robustness of these sensor placements by introducing multiple greedy-based algorithms in which the imperfection of sensors, dynamic demand and multiple objectives can be taken into account. For this, the sensor placement formulation is converted into one or multiple weighted bipartite graph with changing weights. The algorithms were tested using several benchmark instances. It was shown that our algorithms are able to find sensor placements in large networks in reasonable time and that its solutions are provably close to optimal. Furthermore, relaxing the previously used assumption that sensors work perfectly results in different sensor placements than were found before. To show that this assumption is incorrect, experiments have been performed on the real-life WaDi testbed. It was concluded that sensor readings can be unreliable and unstable such that sensor imperfection should be taken into account when deciding on the sensor placement.

Keywords: Water Distribution Network, Sensor Placement, Contaminant Detection, Vulnerable Nodes, Source Identification, Sensor Imperfection, Bipartite Graph, Greedy Algorithm, Experiments, Water Testbed

Acknowledgments

This master's thesis is written in order to fulfill the requirements for the specialization Operations Research and Quantitative Logistics of the master program Econometrics and Management Science at the Erasmus University of Rotterdam. I combined this thesis project with an internship at the Cyber Security & Robustness (CSR) department of the Netherlands Organisation for applied scientific research (TNO) and the Singapore University of Technology and Design (SUTD). These internships lasted from April to November 2017.

Firstly, I would very much like to express my gratitude for the wonderful opportunities that were given to me by TNO, the SUTD and especially Rob Kooij who mainly arranged it all. Beforehand, I never expected to see myself doing research for my thesis abroad. I have learned and experienced so much during the two months in Singapore. The experiments I could do and the people I talked to who know a lot about water networks have helped me in my research. I really enjoyed my time in Singapore and I want to thank the people from the SUTD and the people from TNO Singapore for this.

Besides that, I would like to thank professors Albert Wagelmans and Wilco van den Heuvel from the Erasmus University, Robert Kooij and Daniel Worm from TNO and Venkata Palleti from the SUTD for their supervision. I want to thank them for their feedback, their new ideas and advises. I want to express my gratitude for the time they spent helping me.

Furthermore, there are some other people which should be named. I already mentioned the departments of the SUTD and TNO Singapore, but my time over there was not possible without the opportunity of the CSR department to do my internship at their department. I enjoyed my time at the TNO office in the Hague which is mainly due to the large number of interns at the CSR department and the nice working atmosphere. Finally, I want to thank family and friends for the support during this thesis project and for the little bit of effort which was necessary to convince me to take the opportunity to go to Singapore.

Casper de Winter, November 2017

Contents

| | Page |
|--|-----------|
| 1 Introduction | 4 |
| 2 Background | 6 |
| 2.1 Perfect-Sensor Placement Algorithms | 6 |
| 2.2 Imperfect Sensors | 7 |
| 2.3 Sensor Placement Objectives | 8 |
| 3 Problem Definition | 10 |
| 3.1 Assumptions | 11 |
| 4 Methodology | 12 |
| 4.1 Sensor Placement Formulation | 12 |
| 4.2 Objective Formulations | 12 |
| 4.2.1 Detection likelihood D | 13 |
| 4.2.2 Identification probability F | 14 |
| 4.2.3 Detection Time T | 17 |
| 4.2.4 Contamination Impact Z | 19 |
| 4.3 Exact Solution Methods | 20 |
| 4.4 Greedy Algorithm | 20 |
| 4.4.1 Weighted Set Covering Formulation | 20 |
| 4.4.2 Weight Formulations | 22 |
| 4.4.3 The Greedy Part | 25 |
| 4.5 Local Search Algorithm | 28 |
| 4.6 Non-Dominated Greedy Algorithm | 29 |
| 4.7 Dynamic Demand | 30 |
| 4.8 Sensor Degradation | 31 |
| 4.9 Extensions | 32 |
| 5 Benchmark Instances | 34 |
| 5.1 Bangalore Network | 34 |
| 5.2 Richmond Network | 35 |
| 5.3 BWSN-2 Network | 36 |
| 6 Results of Proposed Methodology | 37 |
| 6.1 Perfect-Sensor Placement vs Imperfect-Sensor Placement | 37 |
| 6.1.1 Perfect-Sensor Placement | 37 |
| 6.1.2 Imperfect-Sensor Placement | 38 |
| 6.1.3 Sensor Placements for each Objective Separately | 41 |
| 6.2 Performance of the Greedy Algorithm | 42 |
| 6.2.1 Comparison with other Presented Heuristics | 42 |
| 6.2.2 Comparison with Improved Random Solutions | 44 |
| 6.2.3 Comparison with Optimal Solutions and Upper-bounds | 45 |
| 6.3 Identification Probability | 47 |

| | | |
|----------|--|-----------|
| 6.4 | Dynamic Demand | 50 |
| 6.5 | Sensor Degradation | 52 |
| 7 | Experiments | 53 |
| 7.1 | The WaDi Testbed | 53 |
| 7.1.1 | Overview of the Testbed | 53 |
| 7.1.2 | Contaminations and Sensors | 54 |
| 7.2 | Experiments and Results | 57 |
| 7.2.1 | Long Runs | 57 |
| 7.2.2 | Standard Solutions | 59 |
| 7.2.3 | Adding Contaminations | 61 |
| 8 | Discussion and Further Research | 65 |
| 9 | Conclusion | 66 |
| | References | 67 |
| | Appendix | 70 |
| A | List of Symbols | 70 |
| B | Proof of Proposition 2 | 72 |
| C | Results | 73 |
| C.1 | All Results Comparing the Different Heuristics | 73 |
| D | Experiments | 75 |
| D.1 | The Testbed | 75 |
| D.2 | Long Runs | 79 |
| D.3 | Standard Solutions | 80 |
| D.4 | Adding Contaminations | 81 |

1 Introduction

Water Distribution Networks (WDNs) form a crucial part in our life by providing clean, safe drinking water to billions of people around the world. A WDN supplies fresh water from water sources to households, companies e.g. using a large hydraulic system. This system or network consists of many elements such as reservoirs, tanks, treatment facilities, pumps, pipes, and valves. These networks are diverse and can be very large, consisting of hundreds or thousands of kilometers of underground pipes. For example, there are more than 6000 kilometers of pipelines in the small country of Singapore [1]. An increasing number of people make use of water from such a system every day and rely on the safety and the quality of water in their lives or work. If a problem arises within the WDN, the impact on a society can be enormous.

Following the terrorist attacks of September 11, 2001, governments became more aware of possible threats to the critical infrastructure of a country including drinking water networks. One of the highest priorities was placed on the monitoring of these water systems. In the United States, the Environmental Protection Agency (EPA) has initiated massive actions to improve the security of drinking water facilities. There are several threats to a WDN which can be divided into physical disruptions and chemical disruptions. Physical disruptions, such as leaking pipelines, failing pumps or intentional attacks on the network itself, will have a big economical impact but are not considered as a serious risk to human beings. The biggest threat to a population comes from intentional or accidental chemical contaminations within the water network. For this, a WDN should incorporate an early warning system with a sensor network to monitor the quality of the drinking water effectively and efficiently. Together with EPA, Murray et al. (2009) [2] estimated that a contamination warning system could save half of the expected fatalities and over 19 billion dollars of associated economic impact on a water network of a large municipality.

A system to monitor drinking water quality is necessary because polluted water is the leading cause of diseases and deaths worldwide. As stated by Yan et al. (2016) [3], the majority of problems occur in countries with rapid urbanization as India, Bangladesh, and China. Due to the contamination of rivers and other water sources, approximately 90% of the urban drinking water in China is polluted and around 580 people in India die of the effects of water pollution every day [3]. Even in first world countries, incidents happen regularly with water distribution systems. In August 2016 the worst waterborne disease outbreak of New Zealand occurred when bacteria found in sheep feces found its way into the groundwater [4]. It resulted in around 5000 sickened people and at least three deaths could be linked to the outbreak. The bacteria was detected too late as routine tests were only done every few days. Other outbreaks in the last couple of years in Texas (2016) [5], West Virginia (2014) [6] and Scotland (2015) [7] were also due to waste or industrial problems but did not result in any known deaths.

The previously described incidents are all declared as accidental. However, a much bigger and more lethal threat to a society happens with the intentional poisoning of drinking water by criminals or terrorists. In every volume of the report *The World's Water*, Gleick and Heberger (2014) [8] enumerate all known conflicts and threats which involve water resources or water systems. Most known events which involve poisoning or other acts of deliberately contaminating water networks over the past 30 years happened in countries as Syria, Turkey, Somalia, and Afghanistan. A large recent act of terrorism in the United States was in 1984 when members of a religious cult contaminated a city water supply tank. More recently, the U.S. and other western countries have been threatened multiple times by organizations as Al-Qaeda or ISIS to poison drinking water facilities (see for example: [9], [10]). The most recent version of the list of conflicts can be found

on the website of The World’s Water.

These upcoming threats show the relevance of a good warning system. A sensor system should be able to quickly detect intrusions in the WDN and therefore reduce sickness, fatalities and the associated economic consequences. Due to cost and maintenance reasons, it is of course not possible to place an infinite amount of sensors in the network. A brand new sensor with the latest detection technologies can cost 5,000 up to even 50,000 euros [11]. So, a small number of sensors need to be placed efficiently to achieve an effective monitoring. Several objectives and different algorithms have been considered to achieve an optimal or suboptimal placement of sensors. In previous work, it has almost always been assumed that the sensors are 100 percent reliable and that they always detect contamination. This is most likely an unrealistic assumption and with only a limited amount of sensors to be placed in the WDN, the failure of one sensor could have severe consequences.

In this research, the focus will be on devising new algorithms for solving the sensor placement problem in which the sensors are not 100% reliable in order to make sensor placements more robust. We would like to quantify the effect of imperfect sensors on the sensor placement and to find real evidence for the imperfection of sensors. Furthermore, we would also like to use other, more realistic assumptions in our research such as dynamic demand and the fact that attacks can most likely only happen at some points in the network.

A basis of this research is a recent paper by Palleti et al. (2016) [12] in which they design a perfect-sensor network using a greedy heuristic based on the set covering problem (SCP) to be able to detect the contamination and identify the source of the contamination. While maximizing the probability of detection has always been one of the main objectives of a sensor network, the objective of identifying the source of the attack is quite new in literature. When the point of intrusion is known, it is possible to take action instantly and to get some parts of the water network back to operation sooner. Besides these objectives of maximizing detection and identification, we will also try to minimize the time to detection and the impact of a contamination in our research.

In our thesis, we can make use of several WDN benchmark instances and the Water Distribution testbed (WaDi testbed) located in Singapore. The benchmark instances can be processed using the simulation software EPANET (see: Rossman et al. (2000) [13]), which was developed by EPA to gain more understanding and to support research on drinking water distribution networks and its water quality. The WaDi testbed is a small real-life network with tanks, pipes, sensors, and dosing pumps to put contaminations in the network. WaDi will be used to perform experiments in order to learn more about sensor unreliability, sensor specifications, and the detection of contaminants.

The thesis is structured as follows. We will first discuss some important literature on sensor placement optimization and its objectives in Chapter 2. The problem and scope of this research along with all the necessary assumptions are formally defined in Chapter 3. In Chapter 4, we describe the sensor placement formulation, the different objectives we consider and the algorithms used to solve the problem. Furthermore, several extensions will be given. The networks used for our case studies will be described in Chapter 5 after which the results of the case studies will be presented in 6. The experiments on the WaDi testbed and its findings and insights are given in Chapter 7. Finally, the thesis ends with a discussion along with some points of future research and a conclusion in Chapters 8 and 9.

2 Background

The mathematical-based placement of sensors in water distribution networks to detect contaminations has been studied by several researchers. It was first mentioned by Lee et al. (1991, 1992) [14, 15] where they maximized the demand coverage by sensors. Later, more complex cases and algorithms were devised. In Section 2.1, some of the most important algorithms and ideas on our research field are explained. Afterwards, different implementations of imperfect sensors in literature and all different objectives considered in the sensor placement problem are described in Sections 2.2 and 2.3.

2.1 Perfect-Sensor Placement Algorithms

One of the main research works on sensor placements is the Battle of the Water Sensor Networks (BWSN) Ostfeld et al. (2008) [16] which was a multi-objective network design competition. In total, fifteen independent research teams participated in the competition. The problem was multi-objective: teams had to minimize the detection time, the population affected and the amount of contaminated water consumed and maximize the detection likelihood with a limited number of sensors. Due to the multiple objectives, a winner of the BWSN could not be declared, but the number of non-dominated solutions per team on the different networks can give a good indication of the performance of certain methods. It was concluded that engineering judgment and intuition are not sufficient and need to be supported by quantitative mathematical analysis. In the next two paragraphs, the four best solution methods based on the number of non-dominated solutions in the BWSN are explained [17, 18, 19, 20]. A major limitation of the BWSN formulation is the way non-detected events were handled. Events which could not be detected were ignored which could result in very promising results on impact reduction with a very small chance of detection.

One of the four solution methods was provided by the research team of Berry et al. (2006) [17], which performed a lot of research on sensor placement in WDNs. They designed a mixed-integer programming (MIP) formulation which was very similar to the p -median facility location problem. In that problem, p facilities should be placed and each customer should be assigned to one facility in order to minimize the distance between the facility and the customer. In the formulation of Berry et al., each contamination scenario should be detected by one 'witness' using some number of sensors in order to minimize the impact over all contamination scenarios. The 'witness' is defined such that it is the first sensor in the network to detect the contamination or it is a dummy location, which means it is a non-detection. Several optimal methods and heuristics are introduced by the research team. This MIP formulation and heuristic solution methods formed the basis of the most used sensor placement toolkit in practice, the TEVA-SPOT Toolkit (see: Hart et al. (2008) [21]).

Krause et al. (2006) [18] constructed a greedy algorithm which provides both performance and running time guarantees by using the diminishing returns effect of an extra sensor on the objective. Their algorithm will place a number of sensors in a network one by one in order to minimize the expected impact over all generated contamination scenarios. Dorini et al. (2006) [19] uses a modification of the Cross-Entropy method, an iterative search algorithm to solve combinatorial problems with large state spaces. Wu and Walski (2006) [20] implemented a genetic algorithm in which contamination events were randomly generated using a Monte Carlo scheme.

As said before, Palleti et al. (2016) [12] used a new and different approach to the sensor placement problem. The main difference to all previous work was the fact that a contamination can only happen at a subset of nodes, denoted as vulnerable nodes. The vulnerable nodes are parts of the

water network which are easily accessible such that large, deliberate attacks will probably happen on these nodes. They consider water reservoirs, tanks, pumping stations, water treatment plants and valves as vulnerable. The other parts of the water network, mainly all of the pipes, are usually buried underground which means that deliberate attacks or huge accidents will not likely happen here.

Palleti et al. were interested in the objectives of detecting the attack and identifying the source of the attack. This smaller subset of vulnerable nodes makes sure that the objective of source identification can be investigated from another graph-based point of view. Assuming static flow directions, each vulnerable node affects a subset of the water network. The WDN can now be represented as a bipartite graph with the vulnerable nodes on one side and all other possibly affected nodes on the other side. For detection, the problem is now converted to a set covering problem as each vulnerable node should be covered by a sensor on one of its affected nodes. Using some extra artificial nodes and links in the bipartite graph, Palleti et al. show that this set covering formulation can also be used to perfectly identify the source of the contamination. The formulation makes sure that the resulting set of sensors affected by each vulnerable node is unique. With perfect sensors and these unique response sets of sensors, it is known beforehand which sensors alarm if a contamination takes place on some node. In the same way, when a certain response is given in a network, it is known where the source of the attack was. The well-known greedy algorithm for SCPs of Chvatal (1979) [22] is used to solve the set covering problem.

Another formulation to solve the sensor placement problem which is worth mentioning is the widely used non-dominated sorting based evolutionary algorithm (NSGA-II) of Deb et al. (2002) [23] adapted to water networks by Preis and Ostfeld (2008) [24]. This algorithm has proven to be fast and works well for multi-objective formulations.

Besides that, research in which more realistic cases are considered, such as time-varying demand loadings and flow patterns, is also helpful in providing better, more robust sensor placements. Time-varying or dynamic demand is also known in literature as demand uncertainty. In real-life WDNs, each consumer or company will need different amounts of water at different parts of the day. Furthermore, some water sources could not be in use on some parts of the day. This can cause changing flow directions, volumes or velocities of the water which can make sure that the best sensor placement changes over different time periods. In most researches on dynamic demand each demand pattern was considered to be static, after which different methods were used to find a sensor placement over all the demand patterns (see: Berry et al. (2005) [25], Shastri and Diwekar (2006) [26] and Shen et al. (2014) [27]). Another more realistic subfield of research, which is the main subject of our thesis, are imperfect sensors.

2.2 Imperfect Sensors

Imperfect sensors in the sensor placement problem have only been mentioned in a very few papers. In many papers, it is mentioned in the section with directions for future research. The first researches including the implementation of imperfect sensors were done approximately a decade ago. Researchers started to relax the assumption that sensors are able to perfectly detect a contamination. In this section, the main ideas and findings of the papers related to imperfect sensors will be summarized.

As the data analysis inside a sensor works with some range or threshold, there is a fair chance not to detect every intrusion. For example when the contamination concentration has become too low at that point in the network. In researches as the ones by Ostfeld and Salomons (2004) [28]

or Weickgenannt et al. (2010) [29], these thresholds are considered in their formulation, but the sensors still work perfectly above some threshold. Other researchers just only considered large enough contamination events to overcome this problem.

In the previous section, the adjusted p-median facility location problem introduced by Berry et al. (2006) [17] was explained. In later work of Berry et al. (2008,2009)[30, 31], they extended their work by allowing the sensors to not detect every contamination. They assign false negative probabilities to each sensor location and change the impact of an intrusion into an expected impact. An intrusion at some point is now with some probability first detected by sensor 1, with some probability by sensor 2, and so on and finally with some probability not detected at all. In contrast to the standard BWSN formulation, they did add penalty costs to non-detections. In [30], the formulation is extended for multiple sensor types and thus multiple different probabilities of detecting false negatives across the network. In [31], Berry et al. experiment with a number of solution methods to deal with imperfect sensors and conclude that it is worth using optimization methods that are aware of the sensor imperfection as more robust solutions are found.

Krause et al. (2008) [32] describe their contribution to the BWSN of [18] along with several extensions. They show ways to handle multi-objective optimization by scaling all objectives and also how to handle sensor failures. Imperfect sensors can be implemented in the framework they described by using a random binary variable associated per location which indicates if a sensors works or not. In that way, the objective function is changed with an extra expectation over all possible failure scenarios as the average impact changes per different failure scenario. In practice, this implementation only worked with a low failure probability and at most one sensor failure per scenario, according to the paper, as the number of different failure scenarios can increase rapidly. Results of this implementation have not been given.

In a research by Xu et al. (2010) [33], a two-stage model is proposed to tackle the problem that sensors in a WDN may provide false positive and false negative signals. They combine a facility location model with Bayesian networks such that the probability that a contamination goes undetected and the false alarm rate are minimized. Preis and Ostfeld (2008) [24] and Shen et al. (2014) [27] coped with the unreliability by using detection redundancy as an objective. This means that most contaminations should be detected by more than one sensor which makes sure that when a sensor fails, other sensors can still detect the contamination.

2.3 Sensor Placement Objectives

Many different objectives for the sensor placement problem in WDNs have been considered in literature. The first researchers were only using a single objective, later the focus has moved quickly to multiple objective formulations. In this section, all of the possible objectives for the sensor placement will be named along with some optimization choices regarding these objectives.

In the first researches on sensor placement, coverage of the network was used as the objective such that possible contaminations could be detected by a minimal number of sensors. Researchers only focused on one directed graph from which sensor placements were devised. This was a good starting point for the objective probability of detection, which is widely used as objective in multi-objective formulations.

Two other commonly used objectives are the detection time and the impact of a contamination. The expected time until detection is even considered to be the most important objective for contaminations in WDNs, which ensures that the public can be warned as fast as possible and action can be taken. Impact minimization is equally important as parts of the network with more

demand or more consumers should be secured better. The impact of an attack can be shown with the amount of population exposed, the volume of contaminated water consumed or the total length of pipes in the network which are contaminated. In our research, the volume of contaminated water consumed will be used to quantify the impact of an attack as it is in our opinion the most effective way to measure impact. For calculating the amount of population exposed, many uncertain assumptions are necessary while the estimated demand for every node per hour is already given. The total length of contaminated pipes only shows something about the economical impact and does not have much to do with the impact on a society. An important optimization choice with these objectives is the given value for a non-detection. For detection time for example, a couple of days is mostly used as non-detection value.

Due to the cost of sensors, the number of sensors placed should understandably be minimized. However, in almost all research methods a small and fixed number of sensors is used in their research. With this fixed number, it is easier to compare methods with each other, but it is impossible to know beforehand how many sensors are sufficient for a network in reality. It is always possible to design sensor placements for several values of the number of sensors and decide afterwards. A somewhat opposite objective is sensor redundancy which is used to make sure that a contamination is detected by more than one sensor to be more certain or to make sure that when one sensor fails, the contaminant can still be detected.

Source identification, one of the objectives of our research, is used to make sure that the problem can be quickly found and fixed. By identifying the source, it is possible to close some valves or pipelines to make sure that the damage to other parts of the network is minimized and that other parts of the water network can get back to operation sooner. Therefore, this objective can also be seen as an important practical objective.

As it is not always easy to use multiple objectives in an optimization formulation in order to decide which sensor placement should be chosen, several tricks have been used. It is possible to use a single objective and put the other objectives in a target constraint, for example in order to make sure that the detection likelihood should always be higher than 90 percent. Other widely used methods are converting several objectives into a single objective with some weights or finding all non-dominated solutions. One can also choose what to minimize exactly. The most common statistic is minimizing the average total impact of all possible contamination, but it is also possible to choose more robust objectives such as minimizing the number of worst-case incidents. In Krause and Guestrin (2009) [34], minimizing the worst-case impact is compared with minimizing the average impact over all scenarios and they conclude that choosing a more robust optimization method does not affect the average impact by a lot while the number of high-consequence intrusions does decrease much more.

3 Problem Definition

In this chapter, the problem of our research will be formally defined. Besides that, the research questions and the main assumptions used in our research are given.

The problem we consider in this thesis is the uncertainty that can arise in a WDN and in particular in the sensors defending the network. By taking that uncertainty into account, we try to make the sensor networks more robust such that the consequence of a deliberate attack is minimized. Robustness is a broad concept with many interpretations in several disciplines. It can be defined in this research as the ability of a system to maintain a high level of service when failures or changes occur in the system. In the defense of water distribution networks, failures and changes can affect the ability of a sensor network to detect a contamination. This should therefore be taken into account within the models which decide where to place sensors in the network.

The main uncertainty of a sensor network considered in our thesis is the possible failure of sensors to detect a contamination in the network when it should in fact have detected it. This can cause large problems: outbreaks of sickness, a number of fatalities and the associated economic consequences. The failure of detection of a sensor can have several causes. The sensor can be faced with measurement errors, communication errors or fail completely. A huge problem with possible failures is the fact that sensors are difficult to maintain, as they are mostly located in the pipelines which are buried underground. Due to the location of sensors, in which large amounts of water flow rapidly all day long, sensors also suffer from aging because of the extreme conditions they are placed in. The currently most used sensors in water networks need to be replaced each year as a result of this aging effect. This implies that there is some sort of sensor degradation present. Sensor degradation means that the sensors get worse over time in detecting contaminations and do not achieve their standards after some time. Sensor drift is also a common sensor problem, meaning that the sensor response gradually changes over a long period of time when the conditions are not changing. Another big challenge these days is that of coping with cyberterrorism. A sensor could be hacked and disabled or altered by cyberterrorists such that a possible contamination goes undetected. As can be seen in Chapter 2, a few papers have addressed the problem of imperfect sensors in recent years. However, a lot still needs to be done and investigated.

Besides the imperfection of sensors, dynamic demand and changing flow patterns should also be taken into account. Changes in the demand patterns can make sure that a poisonous water stream goes to different directions in the network or provide more water to some parts of the network at different times of the day or week. It is therefore expected that different sensor networks perform better for different demand patterns and that taking this into account affects the resulting sensor placement. This extension has been studied before, but never in combination with imperfect sensors.

Making the sensor placements more robust, and thus relaxing some unrealistic assumptions as perfect sensors and static demand, will be done by presenting some new methods to place sensors in the network. Furthermore, multiple objectives are being studied. With a sensor placement, the probability of detection and the probability that the source of the attack can be identified should be maximized and the time to detection and impact of a contamination should be minimized. We are especially interested in the objective of source identification as it was previously only analyzed in the perfect setting. Finally, we will try to make significant progress in this field of research by gaining more insights into how imperfect sensors are. For this, experiments will be performed on the WaDi testbed located in Singapore. When water agencies understand the unreliability of sensors better, they can use this knowledge when new sensor networks are being deployed or extra sensor locations are being chosen.

In this thesis we would like to answer the following research questions:

1. Can we devise an efficient algorithm for the robust sensor placement in large water distribution networks, which can incorporate imperfect sensors, dynamic demand patterns and multiple objectives?
2. How can we best measure the probability of identification when sensors are imperfect? And is it useful to use source identification as objective when we take imperfection into account?
3. Can we find evidence for the imperfection of sensors on real sensors in the WaDi testbed? And is it necessary to take the imperfection into account when deciding on the sensor placement?

3.1 Assumptions

In order to start this research on contamination in water networks with imperfect sensors, a couple of assumptions are necessary. In this section, we list the most important assumptions we use in our research. Assumptions (i) and (ii) describe what we assume about a contamination in a WDN. Assumptions (iii), (iv) (v), (vi) and (vii) are about the sensors, whereas Assumption (viii) concerns demand in the water distribution network.

- (i) An attack or a contamination will only happen at one vulnerable node in the network. This can happen on each vulnerable node with equal probability. As long as the contamination is not detected, contamination will be added to the network at the intrusion point.
- (ii) The contamination will travel through the network with the same speed as the water does.
- (iii) Each sensor will have the same known probability p of detecting the contamination. We do not consider multiple types of sensors. It will be assumed that the contamination is so large that dilution has no effects on the probability of detection. When sensor degradation is considered, it is assumed that all sensors age with the same expected rate.
- (iv) The detection of an attack by a certain sensor is independent of the detection of the attack by other sensors.
- (v) False positives will not be considered. The effect of a false positive reading by a sensor is negligible with only some economic impact compared to the consequences of a real contamination which is not detected.
- (vi) A sensor which is able to detect the contamination will immediately raise an alarm. At this time of detection, it is assumed that several actions will be taken directly such that no more contaminated water will be consumed.
- (vii) It is assumed that the number of sensors to be placed in the network is given and that they can only be placed on nodes in the network.
- (viii) The system initially has a steady loading condition with a steady non-negative demand. When dynamic demand is considered, a different demand or flow pattern can occur within the network every time period. It is however assumed that the set of demand patterns is known beforehand and that each pattern holds steady for sufficiently long. When for example two demand patterns can occur during a day, two versions of the same network are being analyzed and transitions between two demand patterns are ignored.

4 Methodology

In this chapter, we will mathematically define the sensor placement problem and the objective formulations in Sections 4.1 and 4.2. Afterwards, several solution methods for this problem will be presented in Sections 4.3 to 4.6. The incorporation of dynamic demand, sensor degradation and some other extensions will be shown in Sections 4.7, 4.8 and 4.9.

4.1 Sensor Placement Formulation

In order to elaborate on the objectives and algorithms, we will first present the definition of a water distribution network and the sensor placement formulation as it is used in our research. We consider a WDN in which water flows from certain sources or tanks to the customers. This network can be represented as a directed graph $G = (V, E)$, where the vertices or nodes V represent sources, demand points, and junctions in the network. Pumping stations, treatment plants, valves and fire hydrants are also illustrated as nodes. E , the set of edges, represents all the pipes between the different nodes. The directions of these edges show how the water flows in the network. A WDN usually consists of hundreds to tens of thousands nodes and pipes. A small subset of the set vertices is considered to be vulnerable or accessible for an attack. In our research, the same classification for a vulnerable node is used as was done by Palleti et al. (2016) [12]. So, the vulnerable nodes are water reservoirs and tanks from which water can flow and pumping stations, treatment plants and valves. The set of m vulnerable nodes is denoted by V' .

Sensors can be placed on all nodes j in the set $\{V \setminus V'\}$. When there exists a directed path from vulnerable node v to node j , a sensor at node j is able to detect a contamination event which occurred at node v . The probability that the sensor actually detects the contamination is denoted with p , $0 \leq p \leq 1$. If $p = 1$, the sensor is considered to be perfect. A sensor placement X is a subset of the set of possible sensor locations. We consider the number of sensors to be placed in the network as input for the formulation. The number of sensors is denoted as B . So, from the set $\{V \setminus V'\}$, a subset X of size B should be picked. This should be done such that the objective formulation used in our research is maximized. The objective function f does not only depend on the sensor placement but also on the probability p that a sensor detects the contamination. More details on the objective formulation will be given in the next section. The sensor placement formulation can in general be written as follows.

$$\max \quad f(X|p) \tag{1}$$

$$s.t. \quad X \subseteq \{V \setminus V'\} \tag{2}$$

$$|X| = B \tag{3}$$

4.2 Objective Formulations

In this section, we will look at the four objectives considered in our research and how values for the objectives can be derived when sensor uncertainty plays a role. We will elaborate on the network probability of detection D , the probability of identification F , the time to detection T and the estimated impact of an attack Z in Sections 4.2.1, 4.2.2, 4.2.3 and 4.2.4 respectively. The estimated impact of an attack will be calculated using the amount of water consumed prior to detection. D, F, T and Z all depend on the sensor placement X and the detection probability of a

sensor p . D and F should be maximized and T and Z should be minimized. It is possible to weigh the four objectives differently, depending on which objective is thought to be more important. For simplicity and comparison reasons, we scale all four objectives between 0 and 1 and make sure that the sum of the four non-negative weights w_D , w_F , w_T and w_Z is equal to 1. As a start, all of these weights are equal and thus $\frac{1}{4}$. In general, the objective formulation of Equation (1) can also be written as follows.

$$\max f(X|p) = \max w_D D + w_F F + w_T(1 - T) + w_Z(1 - Z) \quad (4)$$

One general example will be used in the next sections to illustrate the whole methodology. We consider the simple water distribution network of Figure 2, which has two vulnerable nodes, v_1 and v_2 , and three demand nodes, j_1 , j_2 and j_3 . This network will be called the Ex-1 Network.

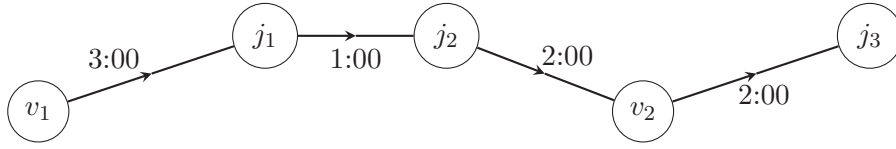


Figure 2: The Ex-1 Network with two vulnerable nodes and three demand nodes.

Water in this network flows from node v_1 to node j_3 via nodes j_1 , j_2 and v_2 . Some necessary data about the flow time in hours in this network can also be seen in Figure 2. It takes in total 8 hours for the water to travel from v_1 to j_3 . The demand at each of the three demand nodes is equal to one per hour. In this example, sensors can be placed on nodes j_1 , j_2 and j_3 and the probability of detection p of each sensor is 0.8.

4.2.1 Detection likelihood D

Given a sensor network where each sensor has a probability p , $0 \leq p \leq 1$, of detecting a contamination, it is possible to calculate the detection likelihood of the whole network. For each vulnerable node $v \in V'$ we can calculate the probability of detecting a contamination entering node v by knowing how many sensors are downstream of node v . For a given loading condition with static flow directions, the set of nodes downstream of node v is equal to A_v , $A_v \subseteq \{V \setminus V'\}$. Given a sensor placement X , the set of sensors which are able to detect a contamination on node v is the intersection between X and A_v . This is denoted by S_v :

$$S_v = X \cap A_v \quad (5)$$

The number of sensors which can detect a contamination on node v is the set cardinality of set S_v .

$$n_v = |S_v| \quad (6)$$

Using Assumption (iv), which stated that detection at distinct sensors occurs independently of each other, the probability of detection can now be calculated. The probability D_v of detecting an

attack on vulnerable node v is one minus the chance of not detecting the contamination by all n_v sensors, which is equal to

$$D_v = 1 - (1 - p)^{n_v} \quad (7)$$

When the sensors are perfect, so $p = 1$, this is reduced to $D_v = 1$ when there is at least one sensor and $D_v = 0$ when $n_v = 0$. As $(1 - p)$ is always between 0 and 1 and the same applies to its power, the probability of detection is also always between 0 and 1. Using the assumption that a contamination can happen on each vulnerable node with equal probability, the probability D that a sensor network detects a contamination is the average of all D_v 's.

$$D = \frac{1}{m} \sum_{v \in V'} D_v = \frac{1}{m} \sum_{v \in V'} (1 - (1 - p)^{n_v}) \quad (8)$$

If in the Ex-1 Network of Figure 2 a sensor is placed on nodes j_1 and j_3 , there are two sensors which could detect a contamination on node v_1 while only the sensor on j_3 can detect an attack on v_2 . So, $n_{v_1} = 2$ and $n_{v_2} = 1$. Using $p = 0.8$ and $m = 2$, it can be calculated that $D_{v_1} = 0.96$, $D_{v_2} = 0.8$ and $D = 0.88$.

4.2.2 Identification probability F

When sensors are perfectly detecting contaminations, identifying the source of the contamination is fairly straightforward. It is only necessary to make sure that the set of sensors which are triggered by attacks on a vulnerable node is unique for each vulnerable node v . However, with imperfect sensors, some problems can occur. This can be illustrated with the Ex-1 Network of Figure 2 in which it is again assumed that sensors are placed on nodes j_1 and j_3 . In Table 1, it is shown which sensor can detect an attack on each vulnerable node.

Table 1: An example to illustrate identification with imperfect sensors. Sensors are placed on nodes j_1 and j_3 in the Ex-1 Network. A one denotes that a sensor can detect an attack on v_i .

| Sensor: | j_1 | j_3 |
|-----------------------|-------|-------|
| Vulnerable node v_1 | 1 | 1 |
| Vulnerable node v_2 | | 1 |

In this example, an attack on node v_1 can be detected by sensors on j_1 and j_3 and an attack on v_2 only by the sensor on j_3 . The sets S_{v_1} and S_{v_2} are unique, so with perfect sensors, every possible attack can be identified. When the sensors are imperfect and $p < 1$, a detection of only the sensor of j_3 can, in theory, mean an attack on both vulnerable nodes, given that the sensor on j_1 could fail with probability $(1 - p)$. Only in cases where the sensor on j_1 gives an alarm, it is known for sure that the contamination happened at node v_1 . When j_1 gives an alarm, it does not matter if the other sensor works or not. So, a contamination on location v_1 can be identified if the sensor on j_1 works, which happens with probability $p = 0.8$, and certain identification is not possible at the other vulnerable node. So with this sensor network and $p = 0.8$, the total probability F of certainly identifying the source of the attack in this network can now be calculated. F is the average of the identification probabilities of both vulnerable nodes, which is $\frac{0.8+0}{2} = 0.4$.

To generalize this, certain new variables and parameters are introduced. From the set S_v , every subset or combination of working sensors $c \in \mathcal{P}(S_v)$ is considered, where $\mathcal{P}(S_v)$ is the power set of set S_v without the empty set. The probability that combination c is actually the set of sensors alarming when node v is contaminated depends on S_v : the sensors in c should work (with probability p) and the sensors in S_v which are not in c should fail (with probability $1 - p$). This probability that subset c alarms when the attack takes place on v is denoted as

$$P(c|v, p) = p^{|c|}(1 - p)^{|S_v \setminus c|} \quad (9)$$

Sensor combination c only adds to the identification of an attack on node v if combination c is unique and can not occur at any other vulnerable node. So, only if c is not present in the power set of S_u for any other vulnerable node u , the combination is unique and will be identified as an intrusion located at node v . For a node v and combination c , define the indicator function $I(v, c)$ as

$$I(v, c) = \begin{cases} 1, & c \notin \mathcal{P}(S_u) \forall u \in V' \setminus v \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

Using Equations (9) and (10), the probability F_v that an attack on node v can be identified, can now be calculated.

$$F_v = \sum_{c \in \mathcal{P}(S_v)} I(v, c)P(c|v, p) = \sum_{c \in \mathcal{P}(S_v)} I(v, c)p^{|c|}(1 - p)^{|S_v \setminus c|} \quad (11)$$

Again, as both $I(v, c)$ and $P(c|v, p)$ are 0, 1 or in between, F_v will always be between 0 and 1. The probability F that any attack is identified by the sensor network can be calculated as follows:

$$F = \frac{1}{m} \sum_{v \in V'} F_v = \frac{1}{m} \sum_{v \in V'} \sum_{c \in \mathcal{P}(S_v)} I(v, c)p^{|c|}(1 - p)^{|S_v \setminus c|} \quad (12)$$

There are also other ways to look at this. It could be seen in the previous example that for some vulnerable nodes, it is never possible to certainly identify the source of the contamination such that those F_v 's will always be zero even though it is almost certain where the source was, given a set of sensors c which alarmed. Consider again the Ex-1 Network of Figure 2, but now all three possible sensors are placed. This can be seen in Table 2.

Table 2: An example to illustrate identification with some percentage. All three sensors are placed in the Ex-1 Network.

| Sensor: | j_1 | j_2 | j_3 |
|-----------------------|-------|-------|-------|
| Vulnerable node v_1 | 1 | 1 | 1 |
| Vulnerable node v_2 | | | 1 |

When sensor j_3 detects something and the sensors are known to be imperfect, it is not possible to identify the source with 100% certainty. Consider again $p = 0.8$ and the sensor combination

$c = \{j_3\}$, $P(c|v_1, p)$ and $P(c|v_2, p)$ can be calculated. $P(c|v_1, p = 0.8) = 0.8(1 - 0.8)^2 = 0.032$ and $P(c|v_2, p = 0.8) = 0.8$. So, given combination c , we are $0.8/(0.8 + 0.032) = 0.962 = 96.2\%$ certain that the point of intrusion was node v_2 . This result is obtained using the Bayes Theorem. Given the set c of sensors detecting a contamination, the probability that the contamination happened at vulnerable node v can be denoted as $P(v|c)$. Using Bayes, we get

$$P(v|c) = \frac{P(c|v)P(v)}{\sum_{u \in V'} P(c|u)P(u)} \quad (13)$$

$P(v)$ is the probability that a contamination happens at vulnerable node v . Using Assumption (i), it is known that this probability is equal for all possible v such that $P(v) = P(u)$. Furthermore, $P(c|u)$ is only not equal to zero if subset c is present in set S_u . Therefore, we introduce the following subset V'_c of V' .

$$V'_c = \{u \in V' | c \in S_u\} \quad (14)$$

$P(v|c)$ can now be written as

$$P(v|c) = \frac{P(c|v)}{\sum_{u \in V'_c} P(c|u)} \quad (15)$$

which we used to calculate the 96.2% previously. This result that the source of the attack is highly likely at some point can be used to calculate the identification probability differently. A new parameter α , an identification likeliness threshold value, is introduced. If the probability $P(v|c)$ is above the threshold α , which is for example 0.95, the indicator function should be equal to 1. Therefore, the indicator function $I(v, c)$ is changed into $I^\alpha(v, c)$.

$$I^\alpha(v, c) = \begin{cases} 1, & P(v|c) = \frac{P(c|v)}{\sum_{u \in V'_c} P(c|u)} = \frac{p^{|c|}(1-p)^{|S_v \setminus c|}}{\sum_{u \in V'_c} p^{|c|}(1-p)^{|S_u \setminus c|}} \geq \alpha \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

Using $\alpha = 1$, $I^\alpha(v, c)$ changes back to $I(v, c)$ as c should again be unique. This new formulation results in the following formulas for F_v^α , the probability that the source of an attack on node v is identified with at least $\alpha\%$ certainty and F^α , the probability that the source is identified with at least $\alpha\%$ for any attack in the network:

$$F_v^\alpha = \sum_{c \in \mathcal{P}(S_v)} I^\alpha(v, c) p^{|c|} (1-p)^{|S_v \setminus c|} \quad (17)$$

$$F^\alpha = \frac{1}{m} \sum_{v \in V'} F_v^\alpha = \frac{1}{m} \sum_{v \in V'} \sum_{c \in \mathcal{P}(S_v)} I^\alpha(v, c) p^{|c|} (1-p)^{|S_v \setminus c|} \quad (18)$$

We have now shown two ways to calculate the probability of identifying the source of the attack when sensor uncertainty plays a role in Equations (12) and (18). F and F^α can be interpreted

differently. If $F = 0.6$, after any attack and some set of sensors which raised the alarm, the source of the attack can certainly be identified in 60% of these attacks. If $F^\alpha = 0.6$ and $\alpha = 0.95$, the source can be identified with at least 95% certainty in 60% of the attacks. In both examples, for the other 40% of the possible attacks and sensor readings, it is considered to be too uncertain to decide where the contamination was inserted into the network. Later, it will be tested how each of the formulations F and F^α influences the resulting sensor placement and what can be concluded from certain values of F and F^α .

4.2.3 Detection Time T

For calculating the other two objectives, detection time and contamination impact, a method is used which is similar to the one used in Berry et al. (2008) [30]. In the formulation, the sensors which are able to detect an intrusion are ordered based on when the sensor can detect the intrusion. For the detection time, we define the time it takes to detect an attack on node v with a sensor on location j as $q_{v,j}^t$. This time is the time it takes for the water to travel from node v to location j . The detection time can be computed using the flow speeds of the water, the length and volume of the pipes and Assumptions (ii) and (vi).

We also define the ordered list L_v^t for each vulnerable node v in which the set of sensors S_v which are able to detect a contamination on node v are sorted based on the detection time of each sensor. $L_v^t(i)$ is the i -th best sensor to detect the contamination based on the time it takes. The corresponding time is defined as $q_{v,L_v^t(i)}^t$. The intuition behind this is the fact that a contamination will first pass the first sensor which can detect the contamination, with the lowest detection time, and if this sensor does not detect, the contamination travels further through the network to the next sensor.

Given a chance p that a sensor detects a contamination, the first sensor $L_v^t(1)$ contributes $pq_{v,L_v^t(1)}^t$ to the average detection time of vulnerable node v , the second sensor $p(1-p)q_{v,L_v^t(2)}^t$, etcetera. With probability $(1-p)^{n_v}$, no sensor will give an alarm. This should still be taken into account and not neglected as was done in some previous work. Therefore, the parameter $q_{v,\infty}^t$ is introduced. $q_{v,\infty}^t$ is a predetermined time in which the contamination is not detected by sensors but in another way, for example by observing an outbreak of sickness. Mostly, 48 hours is used for this parameter. Large numbers are only necessary if the maximum value of all $q_{v,j}^t$'s is larger than 48 hours, which could theoretically happen in large networks with low flow speeds.

We can illustrate this with an example. Figure 3 shows the Ex-1 Network of Figure 2 with two sensors viewed from vulnerable node v_1 .

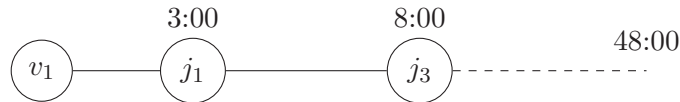


Figure 3: The Ex-1 Network of Figure 2 with two sensors on locations j_1 and j_3 and the detection times viewed from vulnerable node v_1 using the given flow time data.

The two sensors j_1 and j_3 can both detect a contamination with $p = 0.8$. The detection time of sensors j_1 and j_3 is 3 and 8 hours respectively, while $q_{v_1,\infty}^t = 48$ hours. The detection time for this vulnerable node v_1 can be calculated as follows. With 80% chance, j_1 detects and the detection time is 3 hours. If not, j_3 can still detect the contamination and raise an alarm. j_3 will be the one

to give an alarm with probability $(1-p)p = 0.2 * 0.8 = 16\%$ and its detection time will be 8 hours. With 4% chance, no sensor will detect which will result in an estimated detection time of 48 hours. The expected detection time for attacks on node v_1 is $0.8 * 3 + 0.16 * 8 + 0.04 * 48 = 5.6$ hours.

In previous sections, it was shown that D and F are always between 0 and 1. To compare the detection time with the objectives of detection and identification the detection times are normalized to a score between 0 and 1. For each vulnerable node v , $q_{v,\infty}^t$ is set to 1 and a detection time of 0 is set to 0. The detection times of the sensors in between are linearly scaled. Take for example the detection time of sensor j_1 in Figure 3. The score for the detection time of 3 hours node will become $3/48 = 1/16$. The normalized average detection time for a vulnerable node can now be defined as

$$T_v = \frac{1}{q_{v,\infty}^t} \left(\sum_{i=1}^{n_v} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^t \right) + (1-p)^{n_v} q_{v,\infty}^t \right) \quad (19)$$

The first term, one divided by $q_{v,\infty}^t$ is used to make sure that T_v is between 0 and 1. The normalized detection time of the whole network is the average over all vulnerable nodes.

$$T = \frac{1}{m} \sum_{v \in V'} T_v = \sum_{v \in V'} \frac{1}{mq_{v,\infty}^t} \left(\sum_{i=1}^{n_v} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^t \right) + (1-p)^{n_v} q_{v,\infty}^t \right) \quad (20)$$

4.2.4 Contamination Impact Z

For the contamination impact, the same method is used. We define the impact with the estimated volume of water consumed at the demand nodes. As long as a contamination is not detected, the water will be used at demand nodes for consumption or production. The estimated volume of contaminated water consumed grows over time, given the fact that negative demands are not possible. When the detection time increases, more contaminated water will be consumed which means that the sensors for a certain vulnerable node are ordered the same way as in the detection time. The same ordered list of sensors L_v^t can therefore be used.

$q_{v,j}^z$ is the estimated volume of water consumed when a contamination on node v is detected at node j and $q_{v,\infty}^z$ is the estimated volume of water consumed at maximum time $q_{v,\infty}^t$. In the Ex-1 Network, where all the demands are one per hour, we can calculate $q_{v_1,\infty}^z$ and $q_{v_2,\infty}^z$ as follows. v_1 affects j_1 after 3 hours, j_2 after 4 hours and j_3 after 8 hours. These demand nodes will keep getting affected until 48 hours, or $q_{v,\infty}^t$, have passed. This gives $q_{v_1,\infty}^z = (48 - 3) + (48 - 4) + (48 - 8) = 129$. In the same way, $q_{v_2,\infty}^z = 48 - 2 = 46$.

As each vulnerable node can affect a different part of the network with different impacts, the $q_{v,\infty}^t$'s are different. This should be taken into account by weighing the different impacts of each vulnerable node. By doing this, we place more importance on the parts of the network in which the most water flows. The resulting objective values are again normalized such that they are always between 0 and 1. The following formulas for Z_v and Z can be derived. Z_v is the estimated normalized amount of contaminated water consumed for an attack on node v and Z is the estimated normalized amount of contaminated water consumed for any attack in the network.

$$Z_v = \frac{1}{q_{v,\infty}^z} \left(\sum_{i=1}^{n_v} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^z \right) + (1-p)^{n_v} q_{v,\infty}^z \right) \quad (21)$$

$$Z = \frac{\sum_{v \in V'} Z_v q_{v,\infty}^z}{\sum_{u \in V'} q_{u,\infty}^z} = \sum_{v \in V'} \frac{1}{\sum_{u \in V'} q_{u,\infty}^z} \left(\sum_{i=1}^{n_v} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^z \right) + (1-p)^{n_v} q_{v,\infty}^z \right) \quad (22)$$

In total, we can now show the complete objective formulation which is considered in our research. In Equation (4), the formulas for D , F , T and Z can be entered such that the complete objective function is equal to the formula in Equation (24).

$$f(X|p) = w_D D + w_F F + w_T (1 - T) + w_Z (1 - Z) \quad (23)$$

$$\begin{aligned} &= \sum_{v \in V'} \left(\frac{w_D}{m} (1 - (1-p)^{n_v}) + \frac{w_F}{m} \sum_{c \in \mathcal{P}(S_v)} I(v, c) p^{|c|} (1-p)^{|S_v \setminus c|} + \right. \\ &\quad \left. + w_T - \frac{w_T}{m q_{v,\infty}^t} \left(\sum_{i=1}^{n_v} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^t \right) + (1-p)^{n_v} q_{v,\infty}^t \right) + \right. \\ &\quad \left. + w_Z - \frac{w_Z}{\sum_{u \in V'} q_{u,\infty}^z} \left(\sum_{i=1}^{n_v} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^z \right) + (1-p)^{n_v} q_{v,\infty}^z \right) \right) \end{aligned} \quad (24)$$

4.3 Exact Solution Methods

We have now seen the sensor placement formulation of Equations (1), (2) and (3) in Section 4.1 and the objective formulations in Section 4.2. Clearly, we are dealing with a multi-objective and non-linear problem. It is a conjecture in literature that the water quality sensor placement problem is in general NP-hard [32, 35]. In this section, an exact solution method is introduced and evaluated using some complexity results.

The considered exact solution method is a brute force method which is also known as the exhaustive search method. Even though this is the worst possible exact solution method, more efficient methods have never been devised. This is mainly due to the lack of structure which makes sure that other well-known methods, such as dynamic programming or branch and bound methods, are not suitable for the sensor placement problem. Besides that, in all papers in which heuristics are compared with exact methods, the exhaustive search method is used (See for example: [29, 36]). In the sensor placement problem, a total of B sensors should be chosen from the $|V \setminus V'| = M$ possible sensor positions. The enormous number of ways to choose a sensor placement can be visualized by the binomial coefficient which can be seen in Equation (25).

$$\text{Number of possible sensor placements} = \binom{M}{B} = \frac{M!}{B!(M-B)!} \quad (25)$$

The main problem here is that with this brute force method it is necessary to search over all these possible sensor placements before it is perfectly sure that a certain placement is optimal. It is not guaranteed by any means that the best solution of a part of the network or the best solution of $B - 1$ or fewer sensors states something about the overall best solution. As normal WDNs have hundreds to thousands of nodes, only placing a couple of sensors would require billions of possible sensor placements to review. The upper-bound complexity of generating all possible sensor placements is known to be $\mathcal{O}(M^{M/2})$ [37] which is exponential time complexity.

Besides that, reviewing a single sensor placement takes some time. Calculating the four objectives is non-linear and requires methods as sorting lists of maximum length B (with complexity $\mathcal{O}(\log(B)B)$ [37]) and generating every set of the power set of S_v for each node v and checking for each of these sets if it also occurs in another power set (with upper-bound complexity $\mathcal{O}(2^{2B}m^2)$, using results found in Aziz et al. (2012) [37]).

It is probably possible to invent some short-cuts to make this brute force method somewhat faster while still preserving optimality of the algorithm. However, due to the total complexity, it is assumed that it is not possible to find the optimal solution for practical problems in reasonable time and that it is necessary to resort to heuristics. Searching the whole state space by this brute force algorithm on smaller networks can however give us a good reference for the performance of the heuristic approaches. These heuristic methods are presented in the next sections.

4.4 Greedy Algorithm

4.4.1 Weighted Set Covering Formulation

The first heuristic approach we consider is a greedy algorithm. For this, we reformulate the problem formulation into a weighted set covering formulation. A static water distribution network with vulnerable nodes can be converted to a bipartite graph as was shown by Palleti et al. (2016) [12]. In the bipartite graph, the set of vulnerable nodes is on one side and all other, possible sensor

nodes on the other side. Each vulnerable node v can affect a subset A_v of the set of nodes $\{V \setminus V'\}$. Therefore, arcs are drawn between nodes v and each node j from their corresponding affected set A_v . In contrast to the work of Palleti et al. (2016) [12], weights are placed on each arc. In fact, four weights are added to each arc (v, j) and they describe what the placement of a sensor on node j adds to objectives D , F , $(1 - T)$ or $(1 - Z)$ for a contamination on vulnerable node v . Each weight is bounded between 0 and 1. The contribution of the placement of a certain sensor at node j on the whole network can be calculated by summing all the weights of the arcs entering that node j . However, after one sensor position has been chosen, the weights on the other arcs should be updated as they are not valid anymore.

A small example of this idea for the objective detection likelihood can be seen in Figure 4. Again, the Ex-1 Network is used here. The vulnerable nodes are shown on the left side and the affected nodes on the right side. The initial weights for detection likelihood are shown in Figure 4a. Each arc has weight $p/2$, because each arc (v, j) only contributes p to D_v , and 0 to the other vulnerable node in the network. As there are two vulnerable nodes in this example, the contribution of that arc to D is $p/2$. A sensor placed on node j_3 can detect attacks on both vulnerable nodes such that a sensor on that node contributes $p/2 + p/2 = p$ to the detection likelihood D of the whole network.

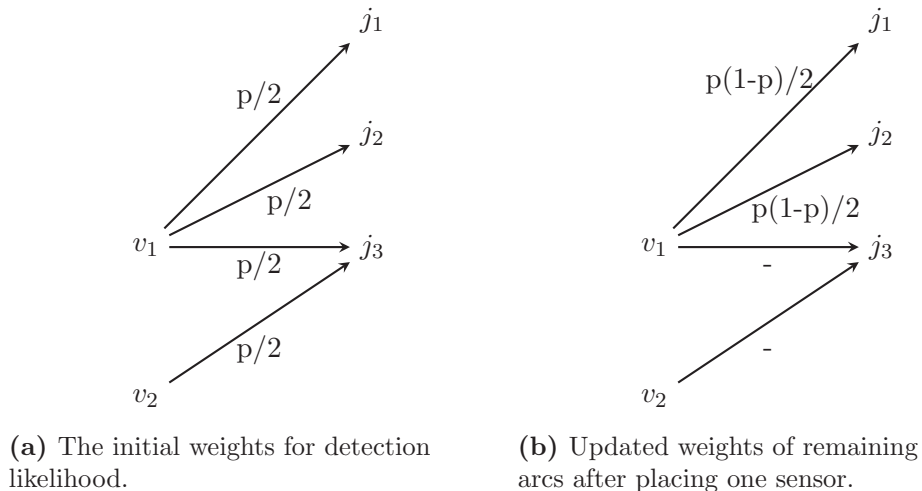


Figure 4: The Ex-1 Network with two vulnerable nodes and three possible sensor placements converted to a weighted bipartite graph for detection likelihood. After placing a sensor in j_3 , the other weights are updated in Figure 4b.

After placing a sensor on node j_3 , the weights of the other arcs need to change. This can be seen in Figure 4b. A sensor on node j_1 or j_2 does not contribute $p/2$ to the total detection likelihood D anymore. A sensor on j_1 only adds something to the objective D when the sensor on j_3 does not detect the contamination, which happens with chance $(1 - p)$. So, the weight for arcs (v_1, j_1) and (v_1, j_2) become $p(1 - p)/2$. Because of these changing weights, it is very important to note that you cannot use this method to place multiple sensors in one step. The idea of changing weights can be applied to all objectives, which will be explained in the next section.

4.4.2 Weight Formulations

In this section it will be formally shown how all of the weights in the weighted set covering formulation should be calculated and updated. These weights will be denoted with a Γ , $0 \leq \Gamma \leq 1$. The weight on a specific arc for a specific objective will be $\Gamma_{(v,j)}^\Delta$ in which (v, j) describe the arc between vulnerable node v and possible sensor node j and Δ is the objective for which the weight applies.

First, we will look at the procedure to update the weights for the probability of detection, which generalizes the example given in the previous section. The addition of one extra sensor for vulnerable node v will change n_v into $n_v + 1$. The added value of the arc for the total objective D will be the difference between the old and new value of D_v divided by m . In Equations (26) to (29), the derivation of the formula for $\Gamma_{(v,j)}^D$ is shown.

$$\Gamma_{(v,j)}^D = \frac{1}{m} \left((1 - (1 - p)^{n_v+1}) - (1 - (1 - p)^{n_v}) \right) \quad (26)$$

$$= \frac{1}{m} \left((1 - p)^{n_v} - (1 - p)^{n_v+1} \right) \quad (27)$$

$$= \frac{1}{m} \left((1 - p)^{n_v} (1 - (1 - p)) \right) \quad (28)$$

$$\Gamma_{(v,j)}^D = \frac{1}{m} (p(1 - p)^{n_v}) \quad (29)$$

With $n_v = 0$, this formula is reduced to p/m , as could be seen in the example of Figure 4a. The weights for the time to detection and the estimated impact of the contamination also depend on the place of the possible sensor j in the ordered list L_v^t . Let us return to Figure 3 in Section 4.2.3. Here, we had a sensor after 3 hours, a sensor after 8 hours and a non-detection time of 48 hours. The order determines how much each sensor contributes to the objective function as the first sensor is the first to raise an alarm. For example, the first sensor contributes p times its normalized detection time to the objective. When the detection time of the added sensor is less than 3 hours, this sensor will be the first sensor, along with its contribution p , and all other sensors contribute less to the objective than before. When the added sensor is placed at the end of the network, only the contribution of the non-detection time of 48 hours will be less than before and the sensor itself will also contribute a smaller fraction to the total objective.

Recall the formula for T_v of Equation (19).

$$T_v = \frac{1}{q_{v,\infty}^t} \left(\sum_{i=1}^{n_v} p(1 - p)^{i-1} q_{v,L_v^t(i)}^t + (1 - p)^{n_v} q_{v,\infty}^t \right) \quad ((19) \text{ revisited})$$

First, we will look at the easiest example, a sensor on node j placed further away than the already placed n_v sensors such that $q_{v,L_v^t(n_v)}^t < q_{v,j}^t < q_{v,\infty}^t$. Only the last term of T_v (which is $(1 - p)^{n_v} q_{v,\infty}^t$) will be split and changes. The sensor with location j will be the sensor to detect the contamination when all n_v sensors fail and this added sensor works. This happens with probability $p(1 - p)^{n_v}$. With probability $(1 - p)(1 - p)^{n_v}$, there is a non-detection. The new formula for T_v , denoted as T_v^* , can be seen in Equation (30).

$$T_v^* = \frac{1}{q_{v,\infty}^t} \left(\sum_{i=1}^{n_v} p(1 - p)^{i-1} q_{v,L_v^t(i)}^t + p(1 - p)^{n_v} q_{v,j}^t + (1 - p)(1 - p)^{n_v} q_{v,\infty}^t \right) \quad (30)$$

It should also be noted that the contribution of vulnerable node v to the total objective T is T_v divided by m and that our objective is to maximize $(1 - T)$. The difference between $\frac{T_v^*}{m}$ and $\frac{T_v}{m}$ is equal to the decrease of objective T via vulnerable node v if sensor j is added to the end of the ordered list. So, $\frac{T_v}{m} - \frac{T_v^*}{m}$ is equal to the weight of arc (v, j) as we consider $(1 - T)$ as an objective. This weight is denoted by $\Gamma_{(v,j)}^T(k)$ in which k is used to show the position of the possible new sensor j in the ordered list. The variable k represents the number of already placed sensors which need to change a position in the ordered list when sensor j is added. In this example, k is equal to zero as the added sensor is added to the end. $\Gamma_{(v,j)}^T(0)$ is calculated in Equations (31), (32) and (33).

$$\Gamma_{(v,j)}^T(k=0) = \frac{T_v}{m} - \frac{T_v^*}{m} = \frac{1}{mq_{v,\infty}^t} \left((1-p)^{n_v} q_{v,\infty}^t - (p(1-p)^{n_v} q_{v,j}^t + (1-p)(1-p)^{n_v} q_{v,\infty}^t) \right) \quad (31)$$

$$= \frac{1}{mq_{v,\infty}^t} \left(p(1-p)^{n_v} q_{v,\infty}^t - p(1-p)^{n_v} q_{v,j}^t \right) \quad (32)$$

$$= \frac{1}{mq_{v,\infty}^t} p(1-p)^{n_v} (q_{v,\infty}^t - q_{v,j}^t) \quad (33)$$

In the same way, $\Gamma_{(v,j)}^T(1)$ and $\Gamma_{(v,j)}^T(2)$ can be calculated.

$$\Gamma_{(v,j)}^T(1) = \frac{1}{mq_{v,\infty}^t} p(1-p)^{n_v-1} \left((1-p)q_{v,\infty}^t + pq_{v,L_v^t(n_v)} - q_{v,j}^t \right) \quad (34)$$

$$\Gamma_{(v,j)}^T(2) = \frac{1}{mq_{v,\infty}^t} p(1-p)^{n_v-2} \left((1-p)^2 q_{v,\infty}^t + p(1-p)q_{v,L_v^t(n_v)} + pq_{v,L_v^t(n_v-1)} - q_{v,j}^t \right) \quad (35)$$

Some pattern starts to become visible in these formulas. For larger values of k , more original sensor contributions change and more terms need to be added. When $k = n_v$, all sensor contributions change. In general, we can define the weight formula for arc (v, j) and position variable k as follows.

$$\Gamma_{(v,j)}^T(k) = \frac{1}{mq_{v,\infty}^t} p(1-p)^{n_v-k} \left[(1-p)^k q_{v,\infty}^t + \sum_{h=1}^k \left(p(1-p)^{h-1} q_{v,L_v^t(n_v-k+h)} \right) - q_{v,j}^t \right] \quad (36)$$

In the same way, the weight formula of objective $(1 - Z)$ can be derived. The main difference is that we should account for the weighing of the impact of an attack on each vulnerable node. The formula for $\Gamma_{(v,j)}^Z(k)$ can be seen in Equation (37)

$$\Gamma_{(v,j)}^Z(k) = \frac{1}{\sum_{u \in V'} q_{u,\infty}^z} p(1-p)^{n_v-k} \left[(1-p)^k q_{v,\infty}^z + \sum_{h=1}^k \left(p(1-p)^{h-1} q_{v,L_v^t(n_v-k+h)} \right) - q_{v,j}^z \right] \quad (37)$$

For the probability of identification, no simple formula could be determined. For now, the addition of sensor j on the identification probability should every time be calculated from the complete formula of F_v . F_v with set $S_v \cup j$ minus F_v with set S_v .

$$\Gamma_{(v,j)}^F = \frac{1}{m} ((F_v \text{ with set } S_v \cup j) - (F_v \text{ with set } S_v)) \quad (38)$$

As all weights are obtained from the difference between the objective value of the sensor network with sensor j minus the one without the sensor j it is guaranteed that a weight will always be between 0 and 1 and the objective value can never grow larger than 1 when adding a sum of these weights. For clarification, this will be formally shown for the weights $\Gamma_{(v,j)}^T(k)$ of Equation (36). Due to the minus sign and the many terms inside the large brackets of this formula, it is intuitively not immediately visible that the weights and the objective value are between 0 and 1. For T , the two rules of Propositions 1 and 2 need to be proven.

Proposition 1. $\Gamma_{(v,j)}^T(k)$ of Equation (36) is always larger or equal than zero. For this, it is only necessary to show that the part within the square brackets of that equation is larger than zero. Therefore, the following equation must hold and must be proven:

$$(1-p)^k q_{v,\infty}^t + \sum_{h=1}^k \left(p(1-p)^{h-1} q_{v,L_v^t(n_v-k+h)}^t \right) \geq q_{v,j}^t \quad \forall k$$

Proposition 2. The sum of the objective value for the time to detection plus all weights of an added sensor can never grow larger than one. Suppose this added sensor is j . The following equation needs to be proven:

$$(1-T) + \sum_{v \in V'} \Gamma_{(v,j)}^T(k) \leq 1 \quad \forall j \forall k$$

Let us first prove Proposition 1.

Proof of Proposition 1. The following lemma, Lemma 1, will be used in the proof of Proposition 1.

Lemma 1. For any k , it holds that $\sum_{h=1}^k p(1-p)^{h-1} + (1-p)^k = 1$.

Proof of Lemma 1. Mathematical induction is used to prove that this holds for all values of k .

| | | |
|-----------------|---------------|--|
| Base case: | $k = 0$ | $\sum_{h=1}^0 p(1-p)^{h-1} + (1-p)^0$ $= 0 + 1 = 1$ |
| Inductive Step: | $k = N$ holds | $\sum_{h=1}^N p(1-p)^{h-1} + (1-p)^N = 1$ |
| | $k = N + 1$: | $\sum_{h=1}^{N+1} p(1-p)^{h-1} + (1-p)^{N+1}$ $= \sum_{h=1}^N p(1-p)^{h-1} + p(1-p)^{N+1-1} + (1-p)^{N+1}$ |
| | Using $k = N$ | $= 1 - (1-p)^N + p(1-p)^N + (1-p)(1-p)^N$ |

$$= 1 - (1 - p)^N + (1 - p)^N = 1$$

□

Besides that, we use the fact that $q_{v,\infty}^t \geq q_{v,L_v^t(n_v)}^t \geq q_{v,L_v^t(n_v-1)}^t \geq \dots \geq q_{v,L_v^t(n_v-k+1)}^t \geq q_{v,j}^t$. This holds because position k in the ordered list L is based on this order of detection times. Using this fact and Lemma 1, the proof of the equation shown in Proposition 1 can be completed:

$$\begin{aligned}
& (1-p)^k q_{v,\infty}^t + \sum_{h=1}^k \left(p(1-p)^{h-1} q_{v,L_v^t(n_v-k+h)}^t \right) \\
\geq & (1-p)^k q_{v,j}^t + \sum_{h=1}^k \left(p(1-p)^{h-1} q_{v,j}^t \right) \\
= & \left(\sum_{h=1}^k p(1-p)^{h-1} + (1-p)^k \right) q_{v,j}^t \\
\text{Using Lemma 1:} \quad & = 1 * q_{v,j}^t = q_{v,j}^t
\end{aligned}$$

which finalizes the proof of Proposition 1 such that it is shown that the weights of $\Gamma_{(v,j)}^T(k)$ are always greater than or equal to zero. □

The same proof can also be used to show the same result for $\Gamma_{(v,j)}^Z(k)$. For D , all terms used are already positive. The proof of Proposition 2 that the objective value can not grow larger than one can be found in Appendix B.

A positive advantage for the run time of our algorithm is that for weights $\Gamma_{(v,j)}^D$, $\Gamma_{(v,j)}^T$ and $\Gamma_{(v,j)}^Z$ we do not need to calculate every weight again after adding one sensor. If the added sensor is only connected to some vulnerable nodes, only the weights on the arcs leaving those influenced vulnerable nodes need to be changed. This also applies for $\Gamma_{(v,j)}^F$ when the definition of F of Equation (12) is used. For F^α , an extra sensor can also affect the probabilities of identification of other vulnerable nodes.

4.4.3 The Greedy Part

A greedy algorithm will be used to place sensors in the network based on the weighted set covering formulation. After the placement of each sensor, the weights are updated. Therefore, at each iteration of the algorithm, we choose to pick the best possible sensor location given the weights at that time and place a new sensor at that location.

Before we can start placing sensors, some initial work is done. A water network is simulated in EPANET for one demand pattern. EPANET, the software developed for research on water distribution networks, can be used to perform hydraulic network simulations for a given network with demand patterns and information of all the pipes and nodes. EPANET can also be used to monitor water quality, pressure, velocity and many other parameters. Using the simulation in EPANET, it is possible to construct the corresponding directed graph for a certain demand pattern and to obtain several other necessary values which are needed to compute time and impact. Using a depth-first search algorithm as described by Deo (1974) [38] on the directed graph, a bipartite

graph can be constructed. The initial weights in the bipartite graph can be calculated from the time and impact values from EPANET and the weight functions with $n_v = k = 0$.

Given a weighted bipartite graph, the best sensor candidate j can be chosen. This is the sensor location with the largest weighted sum when summing all weights $\Gamma_{(v,j)}^\Delta$ of the incoming arcs on that node j , multiplied with the corresponding weight w_Δ of each objective Δ . After a sensor is placed on node j and added to the set of sensors, the following steps need to be taken. The objective function is updated and all weights of the arcs entering node j are put to zero. After that, all remaining weights which need to be updated, are updated using the formulas of the previous section. After that, it is again investigated which sensor location is the best to pick and this will continue until B sensors have been placed. The output of the algorithm will be the resulting sensor placement and the corresponding objective values. The pseudo code for the full greedy algorithm can be found in Algorithm 1.

Algorithm 1 Greedy algorithm

```

1: procedure SENSORPLACEMENT
2:   Simulate Water Network in EPANET
3:   Construct Bipartite Graph
4:
5:    $X \leftarrow \emptyset$  ▷ The sensor placement
6:    $Obj \leftarrow 0$  ▷ The objective value
7:   Initialize  $S_v, L_v^t, n_v$ 
8:    $\Gamma_{(v,j)}^\Delta \leftarrow \text{INITIALIZE}(n_v, L_v^t, k = 0)$ 
9:   for  $it = 0$  to  $B$  do
10:      $Max \leftarrow 0$ 
11:      $Bestj \leftarrow 0$ 
12:     for  $j$  in  $V \setminus V'$  do ▷ Find the best sensor location to add.
13:        $Sumj \leftarrow \sum_v \sum_\Delta (w_\Delta * \Gamma_{(v,j)}^\Delta)$ 
14:       if  $Sumj > Max$  then
15:          $Max \leftarrow Sumj$ 
16:          $Bestj \leftarrow j$ 
17:      $X \leftarrow X + Bestj$ 
18:      $Obj \leftarrow Obj + Max$ 
19:     Update  $S_v, L_v^t, n_v$ 
20:      $\Gamma_{(v,j)}^\Delta \leftarrow \text{UPDATE}(n_v, L_v^t, k)$ 
21:   return  $X, Obj$ 

```

We will now demonstrate the greedy algorithm on the general example of the Ex-1 water distribution network of Figure 2. We have already seen the bipartite graph of the network in Figure 4a. All initial weights are shown in Table 3.

The weights for D are straightforward: $p/2 = 0.4$. It can be seen that only sensors on node j_1 or j_2 add to objective F , as it is certain that the attack happened at node v_1 if one of these sensors raises an alarm. For T and Z , $q_{v_1,\infty}^t$ and $q_{v_2,\infty}^z$ are necessary. In Sections 4.2.3 and 4.2.4, we showed that $q_{v_1,\infty}^t = q_{v_2,\infty}^t = 48$, $q_{v_1,\infty}^z = 129$ and $q_{v_2,\infty}^z = 46$ for this example. The total impact of a contamination on node v_1 is larger, which can also be seen in the corresponding weights.

Table 3: The initial weights for the Ex-1 Network.

| | Γ^D | Γ^F | Γ^T | Γ^Z |
|--------------|------------|------------|------------|------------|
| (v_1, j_1) | 0.4 | 0.4 | 0.375 | 0.590 |
| (v_1, j_2) | 0.4 | 0.4 | 0.367 | 0.585 |
| (v_1, j_3) | 0.4 | 0.0 | 0.333 | 0.549 |
| (v_2, j_3) | 0.4 | 0.0 | 0.383 | 0.210 |

Using equally weighted objectives, the best sensor to add can now be calculated. A sensor on node j_1 adds 0.4 to the objective D , 0.4 to F , 0.375 to $(1 - T)$ and 0.590 to $(1 - Z)$. The total contribution of that sensor to the objectives is the average, which is equal to 0.441. In the same way, it is calculated that a sensor on j_2 contributes 0.438. As a sensor on node j_3 can detect attacks on both vulnerable nodes, the contribution of a sensor placed here is larger. A sensor on node j_3 adds 0.8 to D , 0 to F , 0.716 to $(1 - T)$ and 0.759 to $(1 - Z)$ which averages to 0.569. As $0.569 > 0.441 > 0.438$, a sensor is added to node j_3 .

Finally, we should update the weights of arcs (v_1, j_1) and (v_1, j_2) and put the weights of the other two arcs to zero. The updated weights are shown in Table 4. $k = 1$ is used as the sensor on nodes j_1 or j_2 can detect a contamination on v_1 before the sensor on node j_3 can detect the contamination. It can be seen that a sensor on node j_1 will perform slightly better than placing one on j_2 and will therefore be picked as the second sensor.

Table 4: The updated weights in the Ex-1 Network for the remaining two arcs.

| | Γ^D | Γ^F | Γ^T | Γ^Z |
|--------------|------------|------------|------------|------------|
| (v_1, j_1) | 0.08 | 0.4 | 0.108 | 0.151 |
| (v_1, j_2) | 0.08 | 0.4 | 0.100 | 0.146 |

4.5 Local Search Algorithm

Greedy algorithms are in general very fast and may result in good solutions. However, the optimal solution is not always found. This will be shown with a small example in which the optimal sensor placement is not found. Consider the Ex-2 Network of Figure 5.

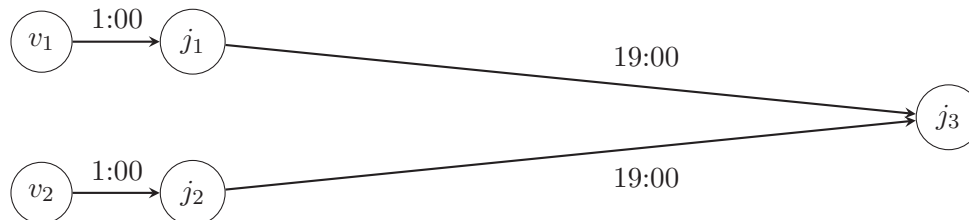


Figure 5: The Ex-2 Network with two vulnerable nodes and three demand nodes. The flow time of each arc is shown at each arc.

The time it takes for the water in the network to go from one node to another is shown in the figure. The demand of each node is again one per hour. It is our task to place two sensors in this network. In Table 5, the initial weights of the corresponding bipartite graph are shown, using $q_{v,\infty}^t = 48$ and the calculated $q_{v,\infty}^z = 75$ for each vulnerable node.

Table 5: The initial weights and the average contribution of each arc for the Ex-2 Network.

| | Γ^D | Γ^F | Γ^T | Γ^Z | Average |
|--------------|------------|------------|------------|------------|---------|
| (v_1, j_1) | 0.4 | 0.4 | 0.392 | 0.4 | 0.398 |
| (v_2, j_2) | 0.4 | 0.4 | 0.392 | 0.4 | 0.398 |
| (v_1, j_3) | 0.4 | 0 | 0.233 | 0.299 | 0.233 |
| (v_2, j_3) | 0.4 | 0 | 0.233 | 0.299 | 0.233 |

Again, all w 's are equal. A sensor on node j_1 or j_2 adds 0.398 to the total objective, whereas a sensor on node j_3 adds $0.233 * 2 = 0.466$. So, in the greedy algorithm, we first place a sensor on node j_3 . However, the optimal sensor placement of two sensors consists of sensors on nodes j_1 and j_2 . This can be seen in Table 6. A sensor network with a sensor on node j_3 performs only better on the probability of detection of the network.

Table 6: Enumeration of all possible sensor placements of two sensors in the Ex-2 Network.

| Sensor Placement | D | F | $1 - T$ | $1 - Z$ | Average |
|------------------|------|-----|---------|---------|---------|
| (j_1, j_2) | 0.8 | 0.8 | 0.783 | 0.8 | 0.796 |
| (j_1, j_3) | 0.88 | 0.4 | 0.672 | 0.758 | 0.678 |
| (j_2, j_3) | 0.88 | 0.4 | 0.672 | 0.758 | 0.678 |

One of the many ways to try and find better solutions is to use a local search algorithm in order to improve the solution found by the greedy algorithm. This local search method is defined as follows. Consider a sensor placement X . In the local search step, each neighboring solution is investigated to check whether it has a better objective value than the current sensor placement. If so, the best-found neighboring placement will be the new placement X and we perform another

local search step until a local optimum is found. Neighboring solutions are defined by replacing one sensor node in X with one of the set $\{\{V \setminus V'\} \setminus X\}$. The pseudo code of the full local search procedure can be seen in Algorithm 2 in which $\text{OBJECTIVEVALUE}(X, w)$ is a function in which the objective value will be calculated, given a sensor placement X and weights w for each objective.

Algorithm 2 Local Search algorithm

```

1: procedure LOCALIMPROVEMENT( $X, w$ )
2:    $Obj \leftarrow \text{OBJECTIVEVALUE}(X, w)$ 
3:   while Improvement = True do           ▷ If improvement is found, execute algorithm again.
4:     Improvement  $\leftarrow$  False
5:     Nodes  $\leftarrow \{V \setminus V'\} \setminus X$ 
6:     for  $i$  in  $X$  do
7:       for  $j$  in Nodes do
8:          $Obj^* \leftarrow \text{OBJECTIVEVALUE}(X - i + j, w)$ 
9:         if  $Obj^* > Obj$  then
10:           $X \leftarrow X - i + j$ 
11:           $Obj \leftarrow Obj^*$ 
12:          Improvement  $\leftarrow$  True
13:   return  $X, Obj$ 

```

The example of the Ex-2 Network can now be completed. When the outcome of the greedy algorithm is the sensor placement (j_1, j_3) , the local search algorithm can find a better solutions by swapping sensor location j_3 by location j_2 .

4.6 Non-Dominated Greedy Algorithm

The local search algorithm is a very simple method to find better solutions than the solution of the greedy algorithm of Section 4.4. It is also possible to alter the greedy algorithm itself to find better solutions. Only picking one sensor based on the objectives and its weights and thus ignoring all other sensors can cause some problems, as was shown with the Ex-2 Network of Figure 5. By choosing sensor location j_3 in the first step, all further solutions of the greedy algorithm contain that sensor. As multiple objectives are being considered, it might be a good idea to not only add the best overall sensor to form a new partial solution but to look at all sensors which produce a non-dominated solution. All of these sensors which form a non-dominated solution are added separately to the sensor placement such that multiple partial solutions are formed. We refer to this algorithm as the non-dominated greedy algorithm.

So, after finding multiple partial solutions, the weights are updated for each of the partial sensor placements and each placement with its related information is taken to the next step of the algorithm. For every placement, it is investigated if new non-dominated solutions can be reached by adding one sensor. A sensor placement X_1 is dominated by another placement X_2 according to the following dominating decision rules:

$$\Delta(X_1) \leq \Delta(X_2) \quad \forall \Delta \in \{D, F, T, Z\} \quad (39)$$

$$\Delta(X_1) < \Delta(X_2) \quad \text{for at least one } \Delta \in \{D, F, T, Z\} \quad (40)$$

X_1 is added to the set of non-dominated partial solutions if for all current sensor placements, one of the rules of Equation (39) does not apply. Finally, we will end up with many sensor placements of size B . Using the weights w_Δ , the final sensor placement will be picked out of these sensor placements. In the Ex-2 Network of Figure 5, it can be seen that a sensor on node j_3 does not dominate a sensor on node j_1 or j_2 , due to the objective of identification. Using the non-dominated greedy algorithm, all sensor placements of Table 6 are reached as none of these dominate each other. Given equal weights, the best sensor placement of (j_1, j_2) is found.

The main advantage of taking multiple partial solutions to the next step is that a broader region of the state space is considered. In general, the objective value will almost certainly be larger or equal than the solution of the normal greedy algorithm. Of course, some drawbacks are also present. The number of solutions to take to the next step can increase rapidly which results in higher run times. Another drawback is that the algorithm is still a heuristic as not necessarily all non-dominated solutions are found. It is possible that all subsets of some size of a non-dominated solution are at that size dominated by other partial solutions. This problem would be solved by using all D_v 's, F_v 's etcetera instead of just D , F T and Z to determine the non-dominated solutions. This would even more explode the number of partial solutions, especially with many vulnerable nodes. To ensure that the algorithm terminates in reasonable time, it is possible to restrict the number of partial solutions which are taken to the next step. As it is not sure that all non-dominated solutions are being found by the non-dominated greedy algorithm, it is not guaranteed that the optimal solution is amongst them.

The non-dominated greedy algorithm and the local search algorithm can also be combined into a more sophisticated algorithm. In this new algorithm, a local search step is added in each greedy step to find neighboring non-dominated solutions. For each of the multiple partial sensor placements, it is checked if neighboring solutions are also non-dominated solutions which can also be added to the set of non-dominated sensor placements. With this addition, it is expected that the number of non-dominated solutions and the run time will clearly be even higher while better solutions can be found.

4.7 Dynamic Demand

In the previous sections, several methods were introduced to find a good sensor placement given a level of sensor uncertainty with static demand and flow directions. In this section, dynamic demand or flow directions will be added into this framework. The effect of different demand loadings has been studied before, but never with the addition of sensor uncertainty.

First, Assumption (viii) should be recalled. It is stated in this assumption that transitions between different demand patterns are ignored such that each demand pattern is treated independently. Demand can possibly change because consumers, for example companies or people at their homes, use different volumes of water at different times of the day or week.

Consider a water network with a set of different demand patterns. Define this set of demand patterns as Θ which contains $|\Theta|$ different patterns. For each pattern $\theta \in \Theta$, the bipartite graph with the corresponding weights can be created. It is possible that the different bipartite graphs differ a lot due to various water flow directions. It is even possible that the bipartite graphs have different sets of vulnerable nodes. Some reservoirs will not be operational all the time and tanks in the network alternate between filling up and emptying.

These different demand levels and flow patterns can have the consequence that a best sensor location in one bipartite graph is not the best for the other ones. In the greedy algorithm, we

should now pick the best overall sensor for all $|\Theta|$ demand patterns. As it is assumed that it is known how long each flow pattern occurs in the network during a day or week, we can also use this information to weigh the objective values per flow pattern. In each step of the greedy algorithm, the best sensor location over all time periods will be chosen and the weights for all bipartite graphs should thereafter be updated.

We can show an example to illustrate this idea. Consider again the Ex-2 Network of Figure 5, but now there are two flow scenarios. The first scenario, scenario θ_1 , is the same as the one which was considered previously. In the second scenario, scenario θ_2 , the water resource node v_2 is shut down. Water now flows from v_1 to j_1, j_3 and j_2 . Besides that, nothing changes with the water velocities or demands on each node in this scenario. In Figure 6, the bipartite graphs of both scenarios are shown, scenario θ_1 in Figure 6a and scenario θ_2 in 6b. Both scenarios occur half of the day.

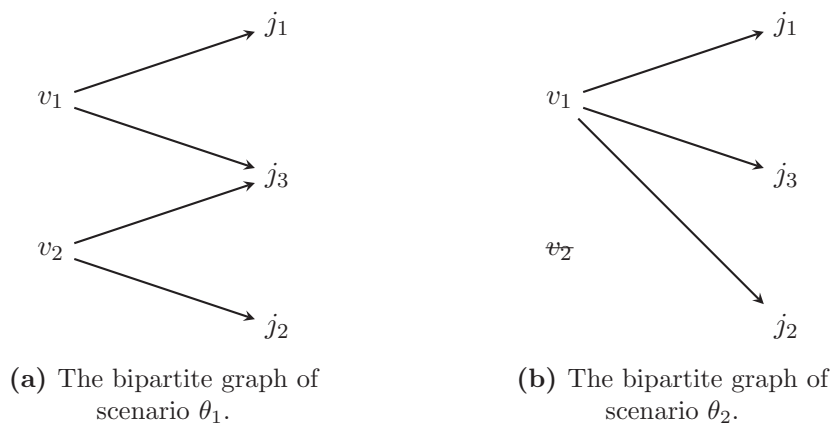


Figure 6: Both bipartite graphs.

Placing one sensor in scenario θ_1 would result in a sensor on node j_3 , as was already established in Section 4.5. j_3 contributes 0.466 and j_1 contributes 0.398 to the objective value. In scenario θ_2 , a sensor would be placed on node j_1 as it is the first sensor in time which can detect a contamination on node v_1 . When the contributions of j_1 and j_3 for the objective value are again compared, it gives 0.796 and 0.672 respectively. A different sensor location is the best one for each scenario. As both scenarios occur with equal probability, the best sensor location can be calculated. $(0.398 + 0.796)/2$ is larger than $(0.466 + 0.672)/2$, so the best overall sensor location to pick for both scenarios will be node j_1 . This is mainly due to the identification in scenario θ_1 and the time and impact in scenario θ_2 .

4.8 Sensor Degradation

It is known that the sensors in a WDN need to be maintained quite often and even need to be replaced every year. Sensor degradation can be a problem as the sensors are placed in extreme conditions with large flows of water keep on passing the sensor probes. It is not expected that a certain probability of detection p by a sensor is steady during the whole lifetime of the sensor. In this section, we would like to present a method to take this degradation into account while deciding for a sensor placement. Later, it is investigated if it is useful to take this into account or not.

When new sensors are being placed, it is assumed that their performance is close to perfect, so the probability of detection will be close to 1, denoted as p^+ . At the end of the lifetime of the sensor, this probability will be smaller. This end probability will be denoted as p^- . In between, the sensor becomes steadily more and more imperfect. This can be taken into account when deciding for a sensor placement is a way which is similar to the way dynamic demand was handled.

Consider p^+ and p^- and some small step size δ . All sensor detection probabilities in the set $\{p^+, p^+ - \delta, p^+ - 2\delta, \dots, p^- + \delta, p^-\}$ are considered. For each of these probabilities, it is theoretically possible that a different sensor network is found. Previously, with dynamic demand, multiple bipartite graphs were defined and the best sensor location over all scenarios was picked. With sensor degradation, for each different p a bipartite graph with different weights is formed and the best sensor location over all bipartite graphs is picked. Afterwards, the weights of all graphs should again be updated.

When for example $p^+ = 0.99$, $p^- = 0.90$ and $\delta = 0.01$, ten bipartite graphs with different weights are used to determine the best sensor placement during the whole lifetime of the sensor. The main interest when this method is tested is to see if it is useful to consider this sensor degradation. Therefore, the resulting sensor placement of all different values of p is compared to the sensor placement when the mean p is considered. This mean p will be denoted as \bar{p} , $\bar{p} = (p^+ + p^-)/2$.

4.9 Extensions

In this section, some descriptions of other possible extensions are presented and the way to handle these ideas within our framework is defined. Results of these extensions will not be given.

Vulnerable Nodes In this research, only a small subset of nodes are considered to be vulnerable. Contaminations can only take place on these nodes. In reality, it is possible that contaminations also occur on other nodes, but this probability is assumed to be much smaller. Deliberate attacks or accidental contaminations will more likely happen at the special vulnerable nodes. Even though the probability that an attack occurs at a normal junction is smaller, a water agency might also want to take this into account. The following changes should be made in order to incorporate this.

In our model, it is possible to state that all nodes in V are vulnerable and give the vulnerable nodes weights which show how vulnerable a node is. As $V' = V$, the set of possible sensor locations can not be $\{V \setminus V'\}$ anymore. The formulation is changed such that each node in V is also a possible sensor location. When a contamination occurs on node $v \in V$, it is assumed that it can not be detected by a sensor on the same node v . The weights which show how vulnerable a certain node is can be incorporated in the weights Γ on the arcs in the bipartite graph. These weights are necessary as it is still believed that contaminations will more likely occur at for example reservoirs or pumping stations. When this extension is implemented, one should think about these weights by determining how much more vulnerable a reservoir is, compared to a normal junction. The sensor placement will also depend on these weights. Weights for the different vulnerable nodes can also be added for the small set of vulnerable nodes considered in this research.

Existing Sensor Placements In Chapter 1, it was stated that the price of water quality sensors is quite high. Furthermore, it is assumed that placing a new sensor on a new position in the water distribution network will cost some money. Logically, this cost will be higher than replacing a sensor at an already used sensor location. In this research, we want to find out if water agencies

should choose a different sensor placement due to the imperfection of sensors. If this is the case, sensors should be deployed on different positions or extra sensors can be placed.

A water agency can choose to keep its current sensor placement X_0 of size B and only place B^+ extra sensors. In this case, the current placement will be used as input for our heuristics and the weights on the bipartite graph will be calculated according to the current network. The B^+ extra sensor locations will be chosen from the remaining set of possible sensor locations. This scenario is very much the same to the normal scenario.

When changes in the current sensor placement are allowed, it is different as it was just mentioned that replacing a sensor is cheaper than placing a sensor in a completely new position of the water network. The cost of this change in the sensor placement can be implemented in the problem formulation. Consider two sensor placements X_1 and X_2 with a better objective value than the already used sensor placement X_0 in a WDN. All networks contain B sensors. X_1 has a slightly better objective value than X_2 , but X_1 differs more from X_0 than X_2 . Both sensor placements perform so much better than the current sensor placement that the water agency wants to change their sensor placement. Due to the costs, it is more likely that a water agency would want to choose placement X_2 over X_1 if the difference in the performance is not that big. This can be implemented into our greedy algorithm by including X_0 as an input parameter and adding a small penalty cost to the weights of arcs to sensor locations which are not in X_0 . This can also be used when the new sensor placement should contain more than the current number of B sensors.

A simple, different solution for this is to limit the maximum number of changes from the current sensor placement X_0 . When only two deviations are allowed, the algorithm should make sure that X_1 only differs from two sensor locations compared to X_0 .

5 Benchmark Instances

Several available EPANET benchmark instances of WDNs will be used in our research to demonstrate the methodology of Chapter 4. The WaDi network used for experiments will be thoroughly explained in Chapter 7. In this chapter, we will introduce three water networks which are based on real-life networks. For each of these networks, the specifications are given and it is described for which case studies the network will be used.

In Section 5.1, the Bangalore Network is introduced. This is a medium-sized network with a lot of vulnerable nodes relative to other networks. The Richmond Network, introduced in Section 5.2, is a medium-sized network with major differences in the different demand patterns. The BWSN-2 Network is a very large-sized network, used in the Battle of the Water Sensor Networks of Ostfeld et al. (2008) [16]. This network will be shown in Section 5.3. The Bangalore Network can be found in Datta (1992) [39] and the other two networks are downloaded from the Centre for Water Systems [40].

5.1 Bangalore Network

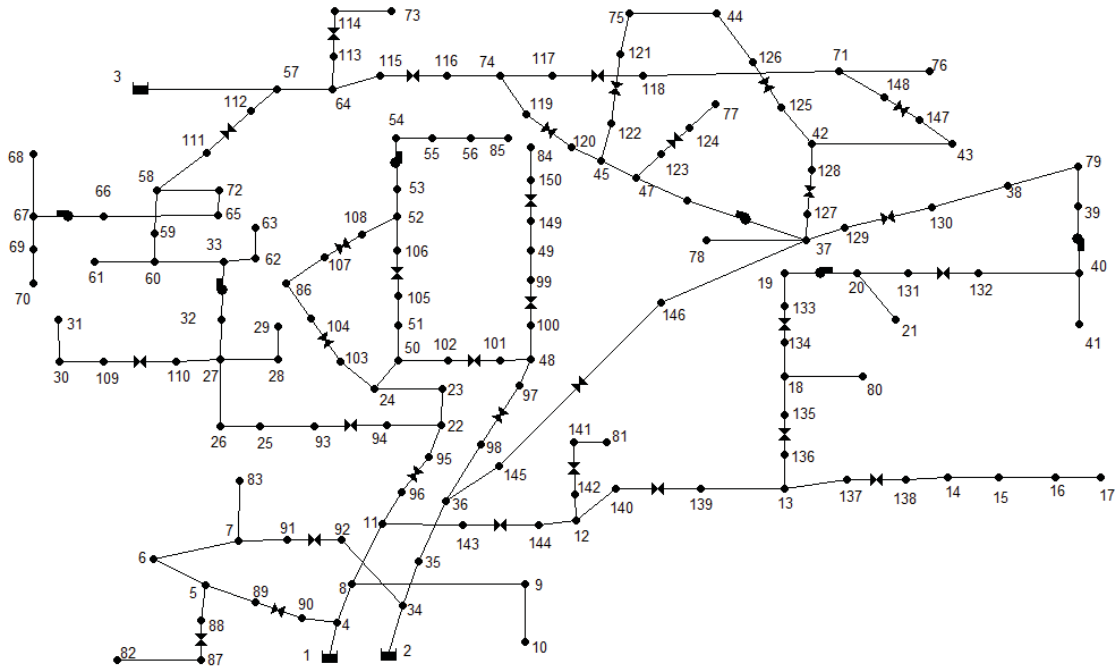


Figure 7: The Bangalore Network.

The Bangalore Network, as seen in Figure 7, is a reduced representation of the water distribution network of Bangalore, a city in India. The network has three sources, which are represented in the network by the reservoir nodes 1, 2 and 3. A clarification of all symbols used in the representation of WDNs can be found in Table A.2 in Appendix A. This network contains 150 nodes in total, 116 normal pipes, 32 valves and six pumps. Pumps and valves are represented in the networks with an edge between two nodes with the specific symbol on the edge. For this reason, when pumps or valves are considered as a vulnerable node, the node before the pump or valve is labeled as

vulnerable. For example, consider the pump between node 66 and 67 on the far left of the network. As water flows from node 66 to node 67, node 66 is considered as a vulnerable node and nodes 67, 68, 69 and 70 will be affected by an attack on this node. Even though the network contains several different demand conditions, the consequences for the flow directions are minimal.

This network was also used by Palleti et al. (2016) [12] in their research focusing on detection and identification with perfect sensors. The main advantage of this network is the large number of reservoirs from which water flows and possible vulnerable nodes, such that the sets of nodes affected by an attack can be very different. Palleti et al. used nine vulnerable nodes in their research: the three reservoirs and the six pumps. The large number of 32 valves were omitted for simplicity of presenting the results as their target was 100% detection and identification, which would otherwise require around 30 sensors. In our research, both 9 and 41 vulnerable nodes will be used. The one with 9 vulnerable nodes will be referred to as the standard Bangalore Network. The last one will mainly be used to see how well our methods and objectives can handle larger numbers of vulnerable nodes.

5.2 Richmond Network

The Richmond Network of Figure 8, see Van Zyl (2001) [41], is a water distribution network based on the water network of Richmond (North Yorkshire), UK. It is comprised of one reservoir, on the far right of the figure, and six tanks from which water can flow. Furthermore, there are 865 nodes, 949 pipes, seven pumps and one valve present in the network.



Figure 8: The Richmond Network.

The main advantage of this network is the number of different flow patterns present in the EPANET data file. In each pattern, a different subset of the reservoir and tanks can be pumping water to the consumers while other tanks are being filled up. Therefore, this network can be used well for research on dynamic demand as it is expected that for different flow patterns, different sensor locations are the best to choose. The network contains a total of 44 different demand and corresponding flow conditions. Furthermore, this medium-sized network will be used to make sure that the results and conclusions we obtain do not only rely on the Bangalore Network.

5.3 BWSN-2 Network

The largest known benchmark instance of water distribution networks is the BWSN-2 Network. This network was presented as one of the two considered networks in the Battle of the Water Sensor Networks of Ostfeld et al. (2008) [16]. To preserve anonymity, it is not known from which real water distribution network it was derived. The complete network is shown in Figure 9a and a close-up of a small part of the network in Figure 9b.

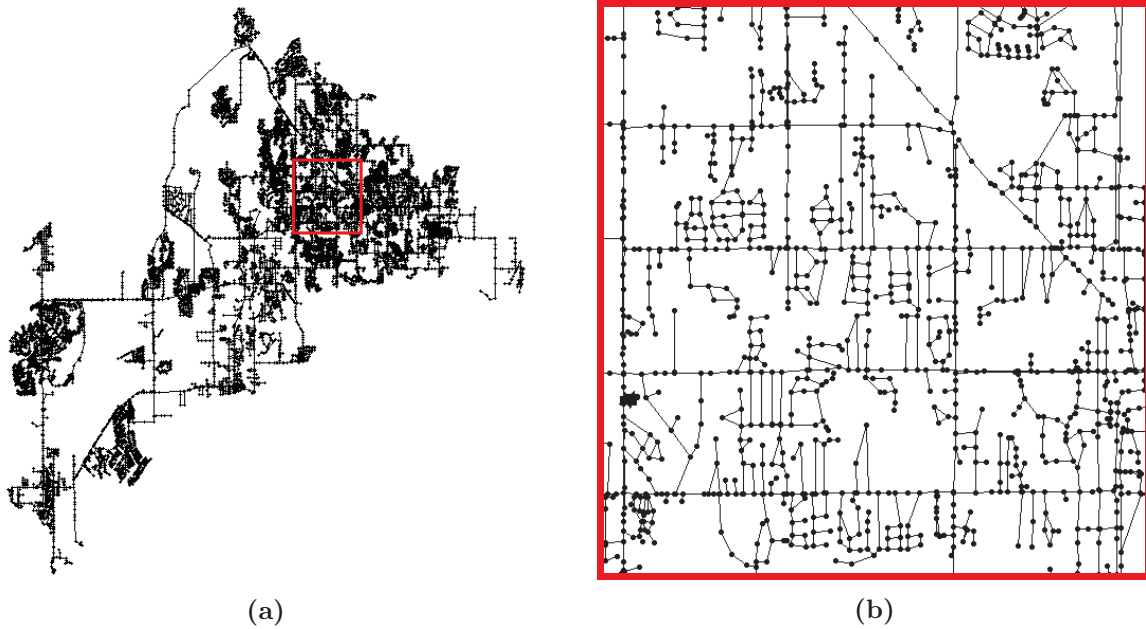


Figure 9: The full BWSN-2 Network in 9a from which we enlarge the red square in 9b for clarification.

The BWSN-2 Network consists of 12,523 nodes, 14,822 pipes, two reservoir sources, two tanks, four pumps and five valves. This enormous network will be mainly used to check the speed of our heuristic approaches as this network resembles real large city networks in which no parts are omitted.

6 Results of Proposed Methodology

All the main results found using the methodology of Chapter 4 with the networks of Chapter 5 will be given in this chapter. In Sections 6.1 and 6.2, the main results regarding the effect of imperfect sensors on the sensor placement and the performance of the presented heuristics will be given. In these sections, the objective of identification F is reviewed using the definition of certain identification. We will go more in-depth into identification and its different versions in Section 6.3. For simplicity, the objectives of maximizing $1 - T$ and $1 - Z$ are just referred to as T and Z in this chapter. The results of the inclusion of dynamic demand and sensor degradation are presented in Sections 6.4 and 6.5 respectively. All algorithms were implemented in Python 2.7 and were run on a computer with an Intel Core i5-5300U processor with 2.30 GHz CPU and 8 GB of RAM.

6.1 Perfect-Sensor Placement vs Imperfect-Sensor Placement

In this section, our main results regarding imperfect-sensor placements in comparison to perfect-sensor placements will be given. In Section 6.1.1, a perfect-sensor network of a WDN will be presented. After that, in Section 6.1.2, the probability of detection p will be lowered to see when and how the best-found sensor placements change. In these sections, equal weights for each objective will be considered. In Section 6.1.3, the best sensor placement per objective will be considered.

6.1.1 Perfect-Sensor Placement

As this research is inspired by the findings of Palleti et al. (2016) [12], we will first compare our results with theirs. In our research, the same static loading condition and flow pattern is used. Palleti et al. assumed perfect sensors, considered the objectives regarding detection and identification and performed a case study on the Bangalore Network of Figure 7. The nodes before the pumps and the three reservoirs were considered to be vulnerable nodes. The resulting nine vulnerable nodes were nodes 1, 2, 3, 19, 32, 37, 39, 53 and 66. As a consequence, there are 141 nodes left where sensors can be placed.

They found out that six sensors are necessary in this network to detect and identify all possible attacks on the nine vulnerable nodes. The sensor network they present is the sensor set $\{20, 33, 40, 54, 67, 71\}$, which indeed gives value 1 to objectives D and F . However, using the presented greedy algorithm with $p = 1$ and adding objectives T and Z , a slightly different sensor placement is found. The sensor on location 71 is moved to location 74 as this location is reached somewhat earlier in the network. The small differences for the objectives can be seen in Table 7.

Table 7: Comparison of the solution found by Palleti et al. and the solution found by the greedy algorithm of Algorithm 1 with $p = 1$.

| Sensor Placement | D | F | T | Z | Average |
|--|-------|-------|-------|-------|---------|
| Palleti et al.: $\{20, 33, 40, 54, 67, 71\}$ | 1.000 | 1.000 | 0.970 | 0.981 | 0.988 |
| Algorithm 1: $\{20, 33, 40, 54, 67, 74\}$ | 1.000 | 1.000 | 0.980 | 0.995 | 0.994 |

The sensor placement $\{20, 33, 40, 54, 67, 74\}$ will be used in the next sections as the perfect-sensor placement for comparison and to see how this placement changes when p decreases. This placement will be referred to as X_P . In Figure 10, the Bangalore Network is represented in which the nine vulnerable nodes are shown as red nodes and the six sensor locations from X_P are shown

as green triangles. The flow directions are also represented in this network by arrows, such that the paths from the vulnerable nodes to the sensors are visible.

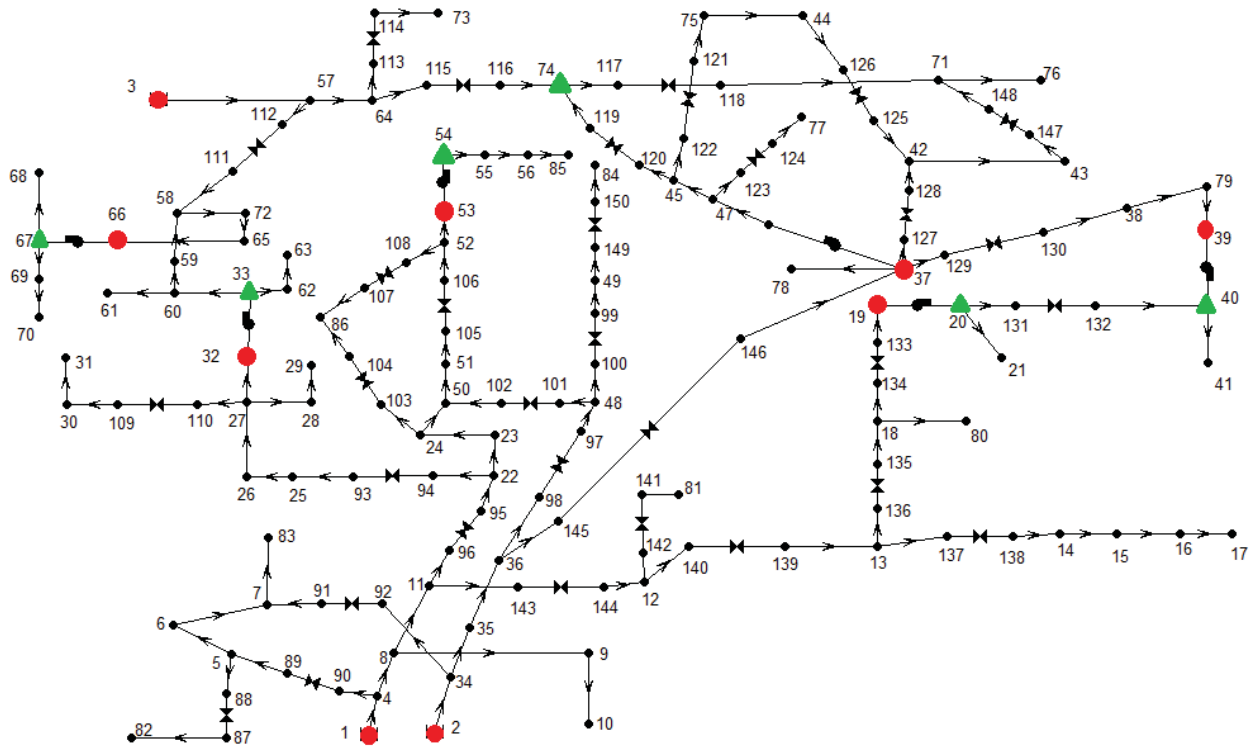


Figure 10: The Bangalore Network with V' in red and the perfect-sensor placement X_P in green.

6.1.2 Imperfect-Sensor Placement

We will now continue by gradually decreasing the probability p that a sensor is able to detect the contamination from 1 down to 0.9. Of course, p can also be lower than 0.9, but we are mainly interested in the switch from perfect-sensor placements to imperfect-sensor placements. In Chapter 7, it will be investigated how imperfect sensors are. The used step size was 0.1% or 0.001. For each value of p , the greedy algorithm can be executed to find out what sensor placement should be chosen. This results in a range of p for which a certain sensor placement is the outcome of the algorithm. In Table 8, the resulting sensor placements are given along with its range in which the sensor placement is found using the greedy algorithm.

Table 8: Different sensor placements and the range of p in which that sensor placement is found with the greedy algorithm.

| Sensor Placement | Range of p in which the sensor placement is the found sensor placement |
|---------------------------------------|--|
| $X_P : \{20, 33, 40, 54, 67, 74\}$ | $p = 1.000$ to 0.981 |
| $X_{G1} : \{20, 40, 54, 58, 67, 74\}$ | $p = 0.980$ to 0.955 |
| $X_{G2} : \{20, 40, 54, 67, 68, 74\}$ | $p = 0.954$ to 0.932 |
| $X_{G3} : \{40, 41, 54, 67, 68, 74\}$ | $p = 0.931$ to 0.900 |

The perfect-sensor placement is only the best-found sensor placement until the probability of detection is 0.98. At that point, changing one sensor location results in a better overall objective value. In Table 9, it can be seen how this objective value changes when p declines for each of the four sensor placements. The best one for a given p is shown in green.

Table 9: Objective value for the four sensor placement for different values of p .

| Sensor Placement | $p = 1$ | 0.99 | 0.98 | 0.96 | 0.94 | 0.92 | 0.90 |
|------------------|---------|---------|---------|---------|---------|---------|---------|
| X_P | 0.99372 | 0.82391 | 0.82071 | 0.81414 | 0.80733 | 0.80028 | 0.79297 |
| X_{G1} | 0.99316 | 0.82364 | 0.82073 | 0.81477 | 0.80860 | 0.80220 | 0.79556 |
| X_{G2} | 0.93527 | 0.82184 | 0.81946 | 0.81452 | 0.80932 | 0.80386 | 0.79811 |
| X_{G3} | 0.87552 | 0.81824 | 0.81646 | 0.81273 | 0.80876 | 0.80455 | 0.80007 |

When p decreases, it can be seen that the objective value of sensor placement X_{G3} decreases less fast than the one of X_P . The reason behind this can clearly be seen when considering the sensor network X_{G3} (see Figure 11). At some places in the network, two sensors are placed next to each other to compensate the imperfection of the other sensor. This occurs with location pairs 40 & 41 and 67 & 68 in X_{G3} .

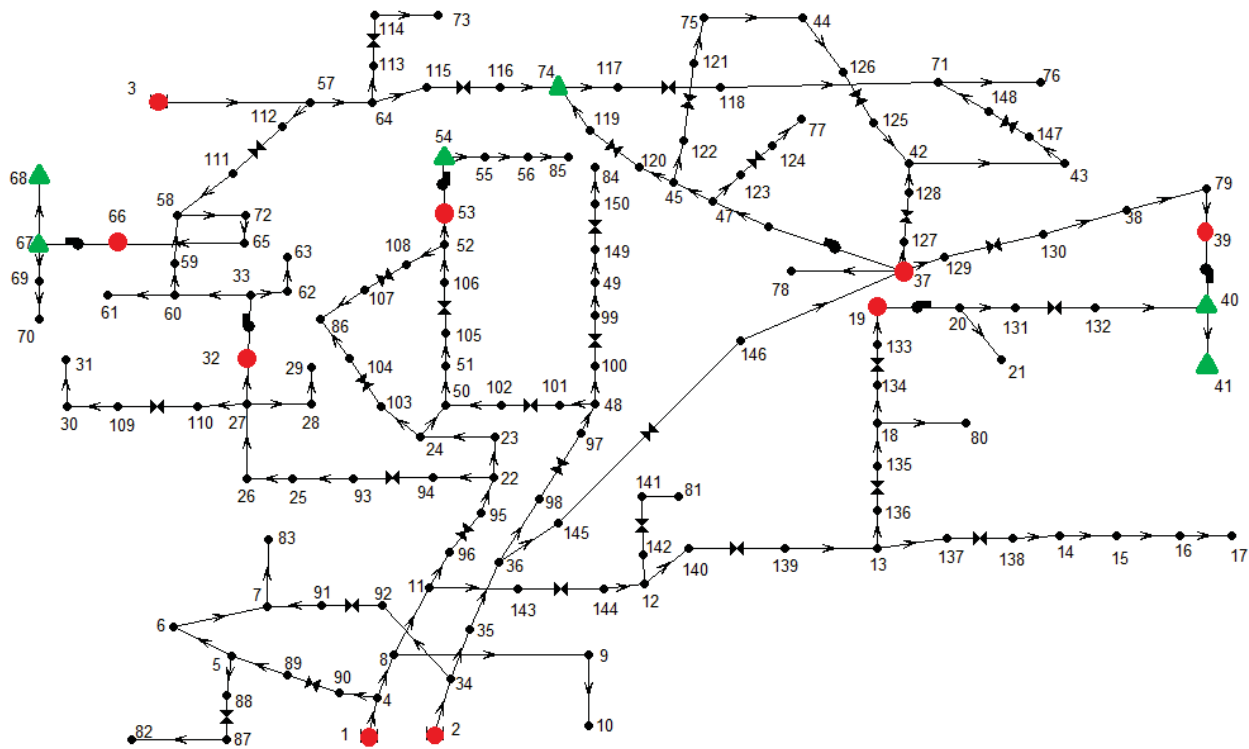


Figure 11: The Bangalore Network with the more robust sensor placement X_{G3} given in green.

Another thing that stands out by looking at these numbers is the large drop between $p = 1$ and $p = 0.99$. This is primarily due to a large drop in the probability of identification, which will be more elaborated on in Section 6.3. Besides that, the relation between p and $f(X)$ is not linear for $p < 1$. It can be seen that the overall objective value decreases slightly more when p gets lower.

So, when six sensors are placed in this network, it is now known that the perfect-sensor placement is not the best sensor placement anymore when the probability of detection is lower than 0.98. A follow-up question is to what extent and in what manner the transition probability depends on the number of sensors to be placed, the network or the number of vulnerable nodes. The level of probability of detection for which the sensor placement switches to a different sensor placement is shown for different settings in Table 10. It shows the transition level for four up to nine sensors for the Bangalore and the larger Richmond Network with both a few and more vulnerable nodes.

Table 10: The probability of detection level for which the perfect-sensor placement is not the best solution anymore. m is number of vulnerable nodes and M is number of possible sensor locations.

| Network | Bangalore | Bangalore | Richmond | Richmond | Richmond |
|---------|-----------|-----------|----------------|-----------------|-----------------|
| m | 9 | 41 | 7 | 13 ^a | 18 ^b |
| M | 141 | 109 | 865 | 859 | 854 |
| $B = 4$ | < 0.5 | 0.514 | < 0.5 | < 0.5 | < 0.5 |
| $B = 5$ | 0.957 | 0.999 | < 0.5 | 0.999 | < 0.5 |
| $B = 6$ | 0.980 | 0.999 | < 0.5 | 0.999 | 0.999 |
| $B = 7$ | 0.980 | 0.999 | 0.557 | 0.642 | 0.999 |
| $B = 8$ | 0.980 | 0.999 | - ^c | 0.655 | 0.999 |
| $B = 9$ | 0.990 | 0.999 | - ^c | 0.999 | 0.999 |

^aExtra randomly selected vulnerable node locations: 12, 78, 143, 220, 269, 401

^bExtra randomly selected vulnerable node locations: 12, 18, 78, 143, 204, 220, 269, 375, 401, 529, 707

^cNo perfect-sensor network exists with more than seven sensors as only seven vulnerable nodes are considered.

When the found sensor network with B sensors and a certain level of p differs from the perfect-sensor network, it is plausible that the sensor network of $B + 1$ sensors will also be different as a different sensor placement is taken to the next step of the greedy algorithm. Because of this, it can be seen in the table that for the standard Bangalore Network the sensor placement differs from the perfect placement for p closer to 1 when B increases. For the Bangalore Network with more vulnerable nodes, the sensor placement differs almost immediately from the perfect-sensor placement with only five sensors deployed in the network. At nine sensors, three out of the nine sensors are already different using $p = 0.999$ instead of $p = 1$.

For the standard Richmond Network, it can be seen that the sensor placement does not tend to change when p drops down. Different sensor networks were only being found for very low values of p . As the Richmond Network only contained seven standard vulnerable node locations, perfect-sensor placements did not exist with more than seven sensors. For this reason, tests have also been executed with extra, randomly selected vulnerable nodes to see if the sensor networks also change for higher values of p when more vulnerable nodes are considered. The results for 13 and 18 vulnerable nodes are also given in Table 10. With this higher number of vulnerable nodes, the sensor placement for this WDN also started to differ from the perfect-sensor placement for higher values of p . Something remarkable happened with the Richmond Network with 13 vulnerable nodes. The fifth, sixth and seventh sensor were chosen in a different order when sensors were assumed to be slightly imperfect compared to when sensors assumed to be perfect, but the total set of seven sensors was the same. For that reason, the value of p dropped back to 0.64 for a sensor network with seven sensors.

From these results, we can conclude that there is an indication that a higher number of sensors

or a higher number of vulnerable nodes most likely results in more changes in the sensor placement when sensors are assumed to be imperfect instead of perfect, even for values of p close to 1. The first sensors the greedy algorithm picks, which contribute the most to the final objective values, are almost always the same for each p . Differences occur when B grows. Besides that, it should be noted that a change in the sensor placement also depends on the network itself. As water agencies of large real-life networks probably want to consider a large number of vulnerable nodes and place a large number of sensors, the sensor placement in those real WDNs probably changes very quickly when the assumption of perfect sensors is relaxed.

6.1.3 Sensor Placements for each Objective Separately

We will now look deeper into the four different objective values and investigate differences in the chosen sensor placements when only one of the objectives is considered. Consider Tables 11 and 12. In Table 11, we observe the changes for the objective values of the perfect-sensor placement X_P when the sensors become imperfect ($p = 0.95$) instead of perfect ($p = 1$). Table 12 shows the different resulting sensor sets and objective values when only one of the four objective values is optimized with $p = 0.95$. The best-found sensor placement with equal weights, X_{G2} , is presented in the bottom column.

Table 11: The different objective values for sensor placement X_P for $p = 1$ and $p = 0.95$.

| Sensor Placement | D | F | T | Z | Average |
|------------------|-------|-------|-------|-------|---------|
| $X_P; p = 1$ | 1.000 | 1.000 | 0.980 | 0.995 | 0.994 |
| $X_P; p = 0.95$ | 0.982 | 0.312 | 0.959 | 0.990 | 0.811 |

Table 12: The different objective values and sensor sets when using different weights and $p = 0.95$. The objective value corresponding to the one which is maximized, is shown in green.

| Weight settings | Sensor set | D | F | T | Z | Average |
|----------------------------|---------------------------------------|-------|-------|-------|-------|---------|
| $w_D = 1$ | {40, 41, 54, 55, 67, 68} | 0.998 | 0.111 | 0.936 | 0.963 | 0.752 |
| $w_F = 1$ | {4, 34, 35, 57, 90, 111} | 0.333 | 0.333 | 0.327 | 0.689 | 0.420 |
| $w_T = 1$ | {20, 33, 40, 54, 67, 74} | 0.982 | 0.312 | 0.959 | 0.990 | 0.811 |
| $w_Z = 1$ | {40, 47, 54, 58, 67, 74} | 0.977 | 0.321 | 0.942 | 0.993 | 0.808 |
| Average: $w_\Delta = 0.25$ | {20, 40, 54, 67, 68, 74}(= X_{G2}) | 0.988 | 0.317 | 0.956 | 0.988 | 0.812 |

We already mentioned the large decrease in the probability of identification when we assume that $p < 1$. It can be seen in both tables that F will not go higher than 0.333 because it is only possible to certainly identify attacks on three out of the nine vulnerable nodes. Objective Z is the least affected objective when sensors become imperfect.

From Table 12, it can clearly be seen that some trade-off between objectives is present with these objectives. For example, when D is maximized, Z decreases more than twice as hard as D increases, compared to the solution values of X_{G2} . In the same way, when Z is maximized, D decreases twice as hard as Z increases compared to X_{G2} . For D , T and Z , three out of the six sensors are always the same: locations 40, 54 and 67 are always chosen. At least four out of the six sensors are also present in the average best sensor set X_{G2} . For F , this is completely different. As there are only three vulnerable nodes which can be certainly identified, the algorithm will only place sensors close

to those vulnerable nodes. This has the consequence that the other three objectives values are very low. Only considering F is thus not a good idea. If only one objective can be considered, T seems to be the best choice as the objective values are very close to the best-found average sensor placement. However, this is probably case-specific and these results show as expected that multiple objectives must be considered when deciding on the sensor placement.

6.2 Performance of the Greedy Algorithm

In this section, we will look deeper into the performance of the greedy algorithm. We would like to find out how good the sensor placements are compared to the other more sophisticated heuristics presented in Chapter 4 and what the consequences for the running time are. This will be presented in Section 6.2.1. In Sections 6.2.2 and 6.2.3, the greedy algorithm will be evaluated in different ways.

6.2.1 Comparison with other Presented Heuristics

The performance of the basic greedy algorithm of Algorithm 1 (also to be referred to as Greedy) will be compared with three other heuristics: the greedy algorithm with the local search algorithm of Section 4.5 (Greedy + LS) and the non-dominated greedy algorithm (NonDom) and the non-dominated greedy algorithm with local search (NonDom + LS) which are shown in Section 4.6.

First, the Bangalore Network with nine vulnerable nodes will be used to compare the four heuristics. They will be compared using 100 different cases by using ten different values for p and ten for B , the number of sensors. p will be between 0.90 and 0.99 and B between 5 and 14. For each case and heuristic, the final objective value and run time are saved. The weights are again all chosen equal for the different objectives. In Table 13, the results for these 100 cases can be found. The average objective value and the average run time are given along with the number of times the solution of the algorithm was equal to the best-found solution by all heuristics. For convenience, the results for the 50 smallest cases and the 50 largest cases are split up in the table. All individual objective values can be found in Appendix C.1.

Table 13: A full comparison of the four presented heuristics on the Bangalore Network with nine vulnerable nodes on 100 different cases. For convenience, the table is split into a part with a small number of sensors (5 to 9) and a larger number (10 to 14). Ten different values of p were used: $p \in \{0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99\}$. The average objective value, the number of times the best solution was found and the average run time by each algorithm are given.

| Bangalore $m = 9$ | 5-9 sensors, 10 different p 's | | | 10-14 sensors, 10 different p 's | | |
|----------------------|----------------------------------|--------------------------|-----------|------------------------------------|-------------|------------|
| | Avg. obj. | #best-found ^a | Avg. time | Avg. obj. | #best-found | Avg. time |
| Greedy | 0.81509 | 41 | 0.251 s | 0.82853 | 13 | 1.905 s |
| Greedy + LS | 0.81511 | 44 | 3.226 s | 0.82887 | 48 | 64.970 s |
| NonDom | 0.81514 | 46 | 12.343 s | 0.82887 | 47 | 443.416 s |
| NonDom + LS | 0.81516 | 50 | 210.708 s | 0.82888 | 50 | 9098.782 s |

^aNumber of solutions found with the heuristic, which are not outperformed by the other heuristics.

It can be seen that the NonDom + LS heuristic always finds the best solution among the four used heuristics. This is as expected, as it is a combination of the other heuristics. The other two

extended heuristics, Greedy + LS and NonDom, are able to find this best solution in approximately 90 out of the 100 cases and the basic greedy algorithm only in 54 cases. The more sensors need to be placed, the less likely it is that Greedy is able to find the best-known solution. However, the differences are very small. For the 50 biggest cases, the difference between the average objective value of Greedy and NonDom + LS is 0.00035 while the maximum deviation in one case from the best-known objective value is 0.00144. This largest deviation occurs with the case $p = 0.96$ and $B = 12$ for which the best-found objective value was 0.83038. To put the difference of 0.00144 in perspective, it means that the score of the solution of Greedy is 99.83% if the best-found solution is considered to be 100 percent.

While this is only a fairly small difference, the differences in run times between the heuristics are much larger. The run time of Greedy is on average 1 second over all 100 cases in this small network with a maximum run time of 4 seconds. The run times of Greedy + LS and NonDom increase much faster, the average run time for these heuristics for the 50 largest cases is over a minute and over seven minutes respectively. For the NonDom + LS heuristic, this is even more extreme with an average run time of over 9000 seconds, which is two and a half hours, and a maximum run time of almost eight hours. In practice, the run time of these algorithms is not a big limitation as finding a sensor placement is not a real-time application. However, this network is only quite small such that problems will occur with larger networks. The best-known solution is always found with this algorithm, but there is a clear price for that. The enormous difference between the run times of the Greedy algorithm and the two NonDom algorithms is mainly due to the large number of non-dominated sensor placements which are taken to the next step. For example for $p = 0.95$ and $B = 10$, there are 238 different non-dominated sensor placements found with the NonDom heuristic and even 373 unique solutions with the NonDom + LS heuristic.

For this Bangalore Network with 9 vulnerable nodes, the basic greedy algorithm with the changing weighted bipartite graph already works fast and good, but other heuristics can still find slightly better sensor placements in reasonable time. We will now look into the performance of these heuristics on other, especially larger, networks. Results for these other networks can again be found in Appendix C.1. First, the effect of more vulnerable nodes is investigated with the Bangalore Network with $m = 41$. Using the four heuristics, the average run times approximately doubles in all cases. Surprisingly, the sensor placements found with the basic greedy algorithm are not outperformed by any sensor placement found with the other heuristics.

Finally, the larger Richmond Network and BWSN-2 Network are used as input for the heuristics. In Table 14, the objective values and run times for input parameters $p = 0.95$ and $B = 6$ are shown. A solution for the BWSN-2 Network with the NonDom + LS algorithm was not found within a day, and not even close to the end, such that the execution of the algorithm was terminated.

Table 14: A comparison of the performance of the heuristics for larger water distribution networks.

| | Richmond $p = 0.95, B = 6$ | | BWSN-2 $p = 0.95, B = 6$ | |
|-------------|-------------------------------|----------|-----------------------------|-------------|
| | Objective Value | Run Time | Objective Value | Run Time |
| Greedy | 0.86699 | 0.583 s | 0.77507 | 92.766 s |
| Greedy + LS | 0.86699 | 5.650 s | 0.77507 | 10901.984 s |
| NonDom | 0.86699 | 2.460 s | 0.77507 | 357.235 s |
| NonDom + LS | 0.86699 | 49.639 s | - | > 1 day |

It can be seen that for these parameters, the solution of the fastest greedy algorithm was not improved by any other heuristic. In fact, no improvement of the Greedy solution could again be found for all other considered parameter values of B and p . Another interesting observation from the results of these larger networks has to do with the run times. In the smaller Bangalore Network, the local search algorithm was faster than the non-dominated greedy algorithm. With larger networks, it can be seen that local search is much slower. Due to the larger number of possible sensor locations, more local deviations need to be checked with the local search algorithm whereas checking if a solution dominates another solution does not take more time with larger networks. Besides that, fewer non-dominated sensor placements are found in the larger networks compared to the Bangalore Network.

6.2.2 Comparison with Improved Random Solutions

According to the results in Section 6.2.1, the basic greedy algorithm almost always finds the best-found sensor placement in a much faster way than the more sophisticated heuristics. These extended methods can hardly find better sensor placements. Given the results, it can be questioned if the heuristics get stuck in the same local optima and even whether so much effort should be put into this. Therefore, a new method is introduced in this section which can find sensor placements without much effort or knowledge.

In this method, a random sensor placement X of B sensors is formed after which the local search algorithm tries to find improvements until no improvements can be found. This will be done with ten random starting solutions to evaluate how often the best-found solution is found and how long this will take. The best resulting sensor placement of the ten improved starting placements will be chosen as sensor placement. In Table 15, the four heuristic methods of the previous section are compared with the random method (Random) and the improved random method using local search (Random + LS). The average objective value and run time of the best-found sensor placement using the same 100 cases as in Table 13 are shown as well as the number of times a heuristic has found the best solution over all heuristics.

Table 15: A comparison of the greedy-based heuristics with the improved random solutions using the same 100 cases as in Table 13. The time given for the random solution methods is the sum of the run times of evaluating all ten random starting points.

| Bangalore $m = 9$ | Same 100 cases as in Table 13 | | |
|----------------------|-------------------------------|-------------|------------|
| | Avg. objective value | #best-found | Avg. time |
| Greedy | 0.82181 | 54 | 1.078 s |
| Greedy + LS | 0.82199 | 92 | 34.098 s |
| NonDom | 0.82201 | 93 | 227.879 s |
| NonDom + LS | 0.82202 | 100 | 4654.745 s |
| Random | 0.67971 | 0 | 0.066 s |
| Random + LS | 0.82201 | 98 | 1434.301 s |

It can be seen that the improved random solution method is not able to find better solutions than previously found using the NonDom + LS method. Besides that, the Random + LS method ended in local optima which were not as good as the best-found solution twice for all ten starting solutions which are not as good as the best-found solution. The multiple starting solutions are

necessary as the best solution was only found with on average eight out of the ten starting sensor placements, sometimes only once or twice. The difference between the average objective value of the Random + LS method and the greedy-based methods is negligible.

The best random starting sensor placement has an average objective value of 0.67971 whereas the basic greedy algorithm has an average objective value of 0.82181. Due to the less good starting solutions, the local search algorithm needs much more iterations to find a local optimum. Therefore, the total run time is much larger, even for only one starting solution instead of ten. Even if starting sensor placements are generated until the run time of the basic greedy algorithm is matched (which happens around 164 generated random solutions on average), the best random sensor placement still has an average objective value of 0.76479. The necessity of a better starting solution is more visible with larger networks. As was shown in Table 14, one iteration of the local search algorithm takes much more time for the Richmond and the BWSN-2 Network. Furthermore, better solutions were also not found using the Random + LS method on the Richmond Network.

It can be concluded that the algorithms introduced in Chapter 4 have some clear advantages over this random-based method. A greedy algorithm is definitely preferred over the random-based method as good solutions are being found very fast, especially with larger networks in which the local search steps are time-consuming. For smaller networks, the Greedy, Greedy + LS or NonDom method is preferred over the Random + LS method as it is uncertain if a good sensor placement is found with only some starting points.

6.2.3 Comparison with Optimal Solutions and Upper-bounds

It can be seen in Section 6.2.2 that better solutions were not found using random starting points. However, it is still not sure if the best-found sensor placement is the optimal solution. To show how well the previously described heuristics perform in general, compared to the optimal sensor placement, the heuristics will be compared with the presented exact brute force solution method of Section 4.3 and with theoretical upper-bounds on the objective value.

In order to make sure that the brute force algorithm terminates within reasonable time, the bipartite graph of the Bangalore Network is largely reduced. This is done in a way that the most important sensor locations are still in the bipartite graph. The non-dominated greedy algorithm is used to place 12 sensors in the water network. Afterwards, the union of all non-dominated sensor placements forms the new set of possible sensor locations $\{V \setminus V'\}$ and all other nodes are deleted from the bipartite graph. As a result, only 36 out of the 141 possible sensor locations remain. In Table 16, the objective value and run time of the four heuristics and the exact method are given.

Table 16: The results of the four heuristics and the exact method with the reduced Bangalore Network, 6 sensors and $p = 0.95$

| 6 sensors, $p = 0.95$ | Objective value | Run time |
|-----------------------|-----------------|-----------|
| Greedy | 0.81195 | 0.061 s |
| Greedy + LS | 0.81195 | 0.438 s |
| NonDom | 0.81195 | 0.905 s |
| NonDom + LS | 0.81195 | 6.751 s |
| Brute Force | 0.81195 | 703.838 s |

It can be seen that the solution found with the heuristics in this smaller case is optimal as the

brute force method found the same sensor placement. By comparing the run times, it is clear that the heuristics are able to find good solutions very fast. In this case, the greedy algorithm is more than 10,000 times as fast while it is able to find the same sensor placement. For larger networks or higher values of B or m , the run time of the brute force method would blow up even more. As mentioned before, there must exist small adaptations or short-cuts for the method while still preserving optimality, but this has never been investigated before and it is also not expected that exact solution methods will find sensor placement within reasonable time for larger WDNs.

So, there are no fast methods present to find the optimal sensor placement in larger networks or to check if a certain sensor placement is optimal. However, we can compare our heuristic solutions with theoretical upper-bounds on the objective value for each water distribution network. In general, when an infinite amount of sensors are being placed in the network, i.e. on every centimeter of pipeline in the network, the objective values D , T and Z will be close to 1. For F , this is slightly different as it is not possible to certainly identify attacks on some vulnerable nodes. For example in the standard Bangalore Network, it was stated that F cannot go higher than $\frac{1}{3}$. A theoretical upper-bound in this network is thus an objective value of $3\frac{1}{3}$ divided by 4 equals $\frac{5}{6} = 0.83333$. The same idea can be used for all other networks. In Table 17, this upper-bound is compared to the solution value found with the greedy algorithm for 15 sensors and $p = 0.95$. Using these results, the optimality gap bound can be calculated. The optimality gap bound shows the difference between the greedy solution and the upper-bound as a percentage of the upper-bound.

Table 17: A comparison of the greedy solution with 15 sensors and $p = 0.95$ with the theoretical upper-bound. With these values, the optimality gap bound is calculated.

| Instance | Greedy: 15 sensors, $p = 0.95$ | Upper-bound | Optimality gap bound |
|---------------------|--------------------------------|-------------|----------------------|
| Bangalore, $m = 9$ | 0.83110 | 0.83333 | 0.27% |
| Bangalore, $m = 41$ | 0.74586 | 0.76829 | 2.982% |
| Richmond | 0.96245 | 0.96429 | 0.19% |
| BWSN-2 | 0.78664 | 0.83333 | 5.60% |

For these networks and this number of sensors, we are not sure if the optimal sensor placements are found using the greedy algorithm. However, using the upper-bounds, it can be shown what the maximum possible difference between the greedy solution and the optimal solution will be. For two networks, the standard Bangalore and the Richmond Network, the solution value with only 15 sensors is very close to the upper-bounds. In the other two cases, the gap is a bit more significant, probably due to the larger number of vulnerable nodes or the larger network. Of course, with only 15 sensors and a p of 0.95, it is not possible to reach the upper-bound, but the results provided by the greedy solution are not that far off. This is another result which demonstrates the performance of the sensor placements found with the greedy algorithms.

6.3 Identification Probability

In the previous sections, it was stated that the probability of certain identification cannot go higher than 0.333 in the Bangalore Network with 9 vulnerable nodes when sensors are considered to be imperfect. Only three out of the nine vulnerable nodes can be perfectly identified when sensors are placed. For the other six vulnerable nodes, the set of possible sensor locations is a subset of the set of possible sensor locations for another vulnerable node. Formally written, this means that if $A_v \subset A_u$ for at least one $u \in V'$, $F_v = 0$ for all sensor placements X . In the Bangalore Network, only the water reservoirs 1, 2 and 3 are identifiable as all pumps and valves are downstream of one of these three nodes. This can be seen for sensor placement X_P in Table 18.

Table 18: The source identification probability F_v for all vulnerable nodes with the standard Bangalore Network using $X = X_P$ and different values of p .

| Vuln. node | | 1 | 2 | 3 | 19 | 32 | 37 | 39 | 53 | 66 |
|------------|------------|--------|--------|--------|----|----|----|----|----|----|
| F_v | $p = 1$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | $p = 0.95$ | 0.9996 | 0.9476 | 0.9025 | 0 | 0 | 0 | 0 | 0 | 0 |

The objective of certain identification therefore ensures that some importance is placed by the algorithm on the quick detection of contaminations located in these water reservoirs. However, this can also be achieved by weighing the vulnerable nodes differently as was described in one of the extensions in Section 4.9. Using the objective of certain identification with uncertain sensor readings seems not suitable, especially with a large number of vulnerable nodes. When for example 100 vulnerable nodes are considered and the network contains two identifiable vulnerable nodes, the objective F cannot grow larger than 0.02. In this section, we will thoroughly look at the interpretation of source identification in the imperfect setting, the effect of using objective F^α instead of F and the effect of not considering source identification as one of the objectives at all.

Let us first return to the reasons why source identification is considered to be an important objective. When it can be determined based on the sensor readings where the contaminant has been introduced into the water network, action can be taken immediately. Practically, this has many advantages. The source of the problem can be fixed sooner, the damage to other parts of the network is minimized and the water network can get back to operation sooner such that the distribution of drinking water to the customers has as little obstruction as possible. This is clearly relevant for water agencies and a good ability of a sensor network to determine where the contamination was introduced is therefore important.

With the imperfection of sensors, being certain on source identification is not possible for many potential attacks on the water distribution network. For that reason, we also introduced F^α in Section 4.2.2 which is the probability to be at least $\alpha\%$ certain about identifying the source for any attack. With this definition, a probability of 1 is also achievable for this objective when an infinite amount of sensors is placed in the network. In Table 19, the previous Table 18 is extended for placement X_P with $p = 0.95$ and multiple values of α . As mentioned earlier, $\alpha = 1$ is the same as certain identification and the results are therefore the same as the bottom row in Table 18.

It can be seen that when X_P is used and the probability of detection is 0.95, most attacks on all vulnerable nodes can be identified with 90% certainty. For higher values of α , some F_v^α s are again zero as it is not at least 95% or 99% certain that a contamination was introduced on that v . On average, the source can be identified with at least 90% certainty in 94% of the attacks and it can be identified with at least 95% certainty in 73% of the attacks. It can be seen that F^α does

Table 19: The source identification probability F_v^α for all vulnerable nodes with the standard Bangalore Network using $X = X_P$ and $p = 0.95$ with different values of α .

| Vuln. node | 1 | 2 | 3 | 19 | 32 | 37 | 39 | 53 | 66 | Average |
|------------------------------|--------|--------|--------|------|------|--------|------|------|------|---------|
| $\alpha = 1$ | 0.9996 | 0.9476 | 0.9025 | 0 | 0 | 0 | 0 | 0 | 0 | 0.3116 |
| F_v^α $\alpha = 0.99$ | 0.9996 | 0.9476 | 0.9025 | 0.95 | 0.95 | 0 | 0 | 0.95 | 0 | 0.6333 |
| $\alpha = 0.96$ | 0.9996 | 0.9476 | 0.9025 | 0.95 | 0.95 | 0 | 0 | 0.95 | 0 | 0.6333 |
| $\alpha = 0.95$ | 0.9996 | 0.9476 | 0.9025 | 0.95 | 0.95 | 0.9025 | 0 | 0.95 | 0 | 0.7336 |
| $\alpha = 0.90$ | 0.9996 | 0.9476 | 0.9025 | 0.95 | 0.95 | 0.9025 | 0.95 | 0.95 | 0.95 | 0.9447 |

not change between $\alpha = 0.96$ and $\alpha = 0.99$. Due to the indicator function, there is a stepwise decreasing pattern between α and F^α .

The value of F^α can be different for every combination of p and α and it is therefore hard to interpret what these values really indicate. An additional drawback of using F^α is the extra level of uncertainty. When a real attack occurs and the wrong conclusion on the place of intrusion is drawn, the wrong actions could be taken and the consequences could be severe. On the other hand, the use F^α will influence the sensor placement more by spreading the sensors more throughout the network such that source identification will be easier. This can be illustrated when comparing X_P to for example X_{G2} . Both sensor placements score around 0.31 on certain identification whereas F^α with $\alpha = 0.95$ for X_P and X_{G2} is 0.734 and 0.633 respectively. The conclusion which can be taken from F is more clear than from F^α , whereas F^α is in some way more meaningful or useful. Making sure that no zeros are present in Table 18 can also be used as a design principal to decide how many sensors should be placed in the network.

In Table 20, we focus on the consequence of using different versions of F or not using F at all for the resulting sensor placement when the Greedy + LS algorithm is used on the standard Bangalore Network. Greedy + LS is chosen to make sure the resulting sensor placement is closer to the optimal one while the run time is still reasonable. These average run times are shown. The resulting sensor placements are evaluated by calculating F , F^α with $\alpha = 0.90$ and 0.95 and the average of objectives D , T and Z .

Table 20: A comparison between the resulting sensor placements and run times when using different versions of F or completely ignoring that objective ($w_F = 0$, other weights are 1/3). The same 100 cases with the standard Bangalore Network are considered and the sensor placement was found using the Greedy + LS method. For the resulting sensor placements, the average value for F , for different F^α 's and for the three objectives D , T and Z are given.

| Bangalore $m = 9$ | Same 100 cases as in Table 13 | | | | |
|------------------------------|-------------------------------|-----------------|-----------------|--------------|---------------|
| | Avg. F | Avg. $F^{0.95}$ | Avg. $F^{0.90}$ | Avg. D,T,Z | Avg. run time |
| F | 0.32665 | 0.77135 | 0.86722 | 0.98710 | 34.544 s |
| F^α , $\alpha = 0.99$ | 0.32304 | 0.86660 | 0.90689 | 0.98465 | 65.645 s |
| F^α , $\alpha = 0.95$ | 0.32378 | 0.90197 | 0.91562 | 0.98458 | 63.308 s |
| F^α , $\alpha = 0.90$ | 0.32614 | 0.82914 | 0.94901 | 0.98577 | 84.758 s |
| no F , $w_F = 0$ | 0.31155 | 0.76937 | 0.84586 | 0.98793 | 1.363 s |

It can be seen that only very small differences are being found when the value of F or the other three objectives D , T and Z are compared for the different resulting sensor placement. As expected, when F was not considered at all, the values for source identification are the lowest whereas the values for the other three objectives are the highest. On the other hand, the differences for F^α are way larger, even in between the sensor placements found when a different α was considered. This is due to the surprising result that many different sensor placements are found when F or F^α is used. The sensor placement was for example only seven times identical for all three considered values of α . So, the sensor placement highly depends on the different α and is therefore very unstable.

Let us consider the resulting sensor placement when $F^{0.95}$ is considered compared to when F is considered. A water agency will see that on average, both sensor placements perform about equally well on certain identification and the objectives D , T and Z . However, when F^α is considered, the sensor placement is better able to almost certainly identify the source of the attack. The question remains what value for α should be used and how pleased a water agency is with 'almost certain'.

Another observation is the result that the run times increase when using F^α instead of F . This is mainly due to the fact that the bipartite graph changes to a complete bipartite graph as placing a sensor on a node can theoretically affect the identification probability of all vulnerable nodes. This can cause problems for larger networks.

As different values of α have a large impact on the sensor placement, the run times and the interpretation of the resulting value for source identification, it can be concluded that F^α should not be used in the optimization phase as it is too unstable.

Completely omitting F as one of the objectives makes the algorithm significantly faster. Using F with many sensors to place is slow as the number of sets in the power set $\mathcal{P}(S_v)$ of each vulnerable node increases rapidly. For the other three objectives, weights and weight updates can easily be calculated which makes it very fast. Therefore, the more sensors are being placed, the more the run times differ. Placing 15 sensors in the Bangalore Network with the basic Greedy algorithm takes about five seconds while placing 20 sensors already takes about a minute. Without F , the run times for placing 15 and 20 sensors are approximately 0.05 and 0.06 seconds respectively. Large decreases in the run time, but not as gigantic as in this example, are also found when larger networks or more sophisticated heuristics are considered. Between sensor placements when F was considered versus placements for which F was omitted, only very small differences are being found.

Due to all of these results about source identification and its shortcomings in the imperfect setting, we decide that it is best to omit the practical objective of source identification when determining a good sensor placement. This objective, with α or without, can be used when the sensor placement is constructed, but one has to realize the drawbacks of using them.

On the other hand, the concept source identification can be very useful after an attack took place in the water network. By looking at the set of sensors which raised an alarm, water agencies can see where the point of intrusion could have possibly been and take some measurements accordingly. If for example three sensors raised an alarm, it can be deduced that with some percentage the attack should have taken place on vulnerable node v_1 , with some percentage on node v_2 , etcetera. When one of these percentages is 100%, the attack can certainly be identified. Otherwise, the vulnerable node with the highest chance can first be investigated.

Finally, the conclusions which were made in the previous sections should be looked at again as these might be different now without considering F as an objective. Most importantly, the sensor placement still changes for high values of p when imperfect sensors are considered instead of perfect sensors. However, it is in general less high than before. The values 0.990 and 0.999 for

p with the two Bangalore Networks and $B = 9$ in Table 10 become 0.989 and 0.986 respectively when only three objectives are considered. The extreme 0.999 is found less often. For the sake of completeness, when F^α is used, the sensor placement also quickly changes when p is not perfect.

The conclusions drawn when comparing the different heuristics are mostly the same without F . For the standard Bangalore Network, the basic Greedy algorithm could again not always find the best-found solution by the other heuristics. The gap between the solutions of the Greedy algorithm and the other algorithms is still very low for the standard Bangalore Network. These results can be found in the Appendix C.1. There is again no difference between the sensor placements when larger networks are considered. The main difference are the run times of the different algorithms. The NonDom + LS heuristics can now determine sensor placements for the Bangalore Network in minutes instead of hours. However, the local search steps still take a lot of time on a large network as the BWSN-2 Network.

6.4 Dynamic Demand

In this section, dynamic demand will be considered instead of static demand and flow directions. It will be combined with imperfect sensors to show if it is necessary to always take dynamic demand into account when this information is known. As was concluded in the previous section, only the objectives D , T and Z are considered and F is omitted. An extra disadvantage of using the objective value of source identification is that it would be even more difficult to determine or interpret source identification when the flow directions are often changing in a setting where the sensors are also considered to be imperfect.

Sensor placements are created for the Richmond Network, which contained $|\Theta| = 44$ highly variable demand patterns. This means that 44 different bipartite graphs must be constructed. The greedy algorithm can be used to decide where to place sensors for all patterns together using the methods described in Section 4.7 or for each pattern separately. If only one demand pattern with one bipartite graph is considered, Algorithm 1 can just be used in the same way as in the previous sections. The best sensor placement using all patterns can be compared with the found sensor placement for each separate pattern and for multiple values of p in order to find out if it is useful to take both dynamic demand and imperfect sensors into account.

Define a sensor placement found with the greedy algorithm using the information of all Θ different patterns as X_Θ and a sensor placement found using only one of those patterns as X_θ . For any value of p and B , it is possible to determine X_θ for all 44 patterns. Some of these sensor placements will probably be the same. Any X_θ can thereafter be tested on all 44 patterns to see how well that placement performs in reality on all demand patterns. By comparing the overall objective values of X_Θ and each X_θ , the difference will be clear between a sensor placement which takes the dynamic demand into account and a sensor placement which only looks at a static demand pattern.

In Table 21, the objective value of the found X_Θ , which was based on all demand patterns, is compared to several sensor placements which are only based on one pattern. The idea behind these four other ways to find a sensor placement is based on the concept that when a water agency ignores dynamic demand, their sensor placement is only based on one demand loading, probably a demand loading which is very common, while the demand is in fact dynamic. We consider X_θ^{avg} (the average overall objective value when a random θ is chosen to base the overall sensor placement on), X_θ^{best} (the best overall placement which is only based on one pattern θ), X_θ^{worst} (the worst overall placement based on only one pattern) and X_θ^{mode} (the sensor placement which is found

for the most number of θ 's). Furthermore, two values of p (0.95 and 1.0) and B (6 and 10) are considered.

Table 21: A comparison of using the information of dynamic demand when generating a sensor placement with only using the information of one demand pattern. The sensor placements are tested against all 44 known demand patterns of the Richmond Network.

| Richmond Sensor Placement | Overall Objective Value | | | |
|------------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| | $B=6, p=0.95$ | $B=6, p=1.0$ | $B=10, p=0.95$ | $B=10, p=1.0$ |
| X_{Θ} | 0.90990 | 0.93876 | 0.97218 | 0.98839 |
| X_{θ}^{avg} | 0.81774 | 0.85568 | 0.86049 | 0.92941 |
| X_{θ}^{best} | 0.90415 | 0.93305 | 0.95789 | 0.97256 |
| X_{θ}^{worst} | 0.71557 | 0.76803 | 0.73041 | 0.86243 |
| X_{θ}^{mode} | 0.76725 (8 θ 's) | 0.79440 (8 θ 's) | 0.82023 (6 θ 's) | 0.89089 (7 θ 's) |

With the results of Table 21, conclusions can be drawn about both the necessity of taking dynamic demand into account as well as the difference between perfect and imperfect sensors on dynamic demand. It can be seen that X_{Θ} , the sensor placement found by taking dynamic demand into account, is never found when only one pattern is considered. In every setting, there is a small difference between the overall performance of X_{Θ} and the best X_{θ} . Furthermore, only looking at the sensor placement which is found for the most number of demand patterns (X_{θ}^{mode}), and thus probably for the most common demand pattern, results in an overall performance which is below average and thus not very good.

Besides that, there are also some differences in using perfect or imperfect sensors with dynamic demand. For example, the X_{Θ} of ten sensors with $p = 0.95$ differs from the sensor placement found using $p = 1$. The sensor placements are the same for six sensors. When sensors are imperfect, the differences between the overall objective values of the average or worst X_{θ} and the overall sensor placement X_{Θ} are much larger, especially when more sensors are being placed. This result indicates that when sensors are considered to be imperfect, the necessity of also taking dynamic demand into account is bigger. As expected, the run times increase with approximately a factor $|\Theta|$, or a factor 44 in the Richmond Network.

Of course, these demand patterns are not always certain in reality and can even be different a year or two years later. Therefore, it is advised to generate a larger set of possible demand patterns when deciding on the sensor placement in a water network.

6.5 Sensor Degradation

In Table 9, it can be seen that different sensor placements perform best for different values of p . Besides that, it is expected that the performance of sensors in real water distribution networks decreases over time. This is called sensor degradation. This means that the best sensor placement at the start of the lifetime of the sensor might be different than the one at the end of its lifetime. In this section, it is investigated if it is useful to take this degradation into account when a sensor placement is decided. Source identification is again ignored in this section.

First, let us consider the example which was already given in Section 4.8: $p^+ = 0.99$, $p^- = 0.90$ and $\delta = 0.01$. This means that p drops from 0.99 to 0.90 during its lifetime. Ten bipartite graphs are created, all with a different p to find different weights Γ . Using those 10 graphs, the sensor placement can be determined. Besides that, the mean p , $\bar{p} = 0.945$, can also be used to determine the sensor placement using only one bipartite graph. In Table 22, the resulting sensor placement and its objective value are shown.

Table 22: The sensor placement and corresponding objective value when multiple p 's or just the mean p is considered.

| Bangalore | Sensor Placement (6 sensors) | Overall objective Value |
|-----------------|------------------------------|-------------------------|
| Multiple p 's | {20, 40, 54, 67, 68, 74} | 0.97552 |
| \bar{p} | {20, 40, 54, 67, 68, 74} | 0.97587 |

It can be seen that the sensor placement does not change when only 6 sensors are placed. In fact, this is the case for all values of B considered with these settings (up to $B = 14$). The only difference is that the objective value of only using \bar{p} is slightly higher. This is valid as it was previously established that the objective value of a sensor placement decreases more when p gets lower which means that the average objective value with multiple p 's is lower.

In this example, the sensor placement does not change when sensor degradation is taken into account. We can try to generalize this by looking at many more cases. Both p^+ and p^- can be randomly generated between 0.9 and 1.0 from which 10 bipartite graphs can be constructed. Using these multiple values of p and \bar{p} , both sensor placements can be build for a certain number of sensors. The resulting sensor placements can then be compared to see if the sensor placement is different. The values of p were randomly generated 100 times and 10 different values of B were used, 5 up to 14. Out of these thousand considered cases, the sensor placement only differed 8 times. Each time, only one sensor was different.

It can be concluded that it is not necessary to literally take sensor degradation into account as hardly any differences are found. It is sufficient to only use $p = \bar{p}$ when deciding on the sensor placement. Using this \bar{p} , the sensor degradation is taken into account in a much easier way than looking at many different values for p . However, it is good to realize that the resulting objective value of using that mean p is a bit higher than the real objective value.

7 Experiments

In the previous chapter, it was shown that different sensor placements should be used compared to the perfect-sensor placement when sensors become slightly imperfect. However, little to nothing is known about the imperfection of real sensors in water networks as to our knowledge, an investigation into this has never been done or published. It is assumed that failures and errors can occur within sensor networks in the detection or communication as they can occur in every technical system. Besides that, it is likely that sensor readings get more uncertain over time as sensors need to be replaced after approximately a year. Sensors should be tested for imperfection to make sure that a good indication for p can be made.

To learn to what extent sensors are in fact imperfect, sensitive or vulnerable to degradation, we have performed several experiments on the WaDi testbed in Singapore. The WaDi testbed is a small, world-class water distribution network designed for research purposes at the Singapore University of Technology and Design (SUTD). In this chapter, all of the experiments performed to find evidence for the unreliability of sensors are explained and the results and insights are given in Section 7.2. First, all necessary information about the testbed will be presented in Section 7.1.

7.1 The WaDi Testbed

In order to understand the performed experiments on the WaDi testbed and to understand its possibilities, an overview of the WDN located at the SUTD will be given in Section 7.1.1. Afterwards, more details on the contaminations and sensors used in our research will be given in Section 7.1.2.

7.1.1 Overview of the Testbed

The WaDi testbed is a small real-life water distribution network with tanks, pipes, valves, pumps, and sensors. As we are mostly interested in the sensors and their location in the network, the focus of this overview will be on those sensors. In Figure 12, the complete overview of the WaDi testbed is given. In this overview, the sensor locations are marked with a purple cross. The sensors are located at fixed positions. Some photographs of the testbed and more detailed, zoomed in displays of the overview of Figure 12 are presented in Appendix D.1 for further clarification.

Water flows in the representation of Figure 12 from the top left corner in a clockwise direction back to the top left corner. The route the water takes is the following:

1. There are two different sources. One provides tap water from the Singaporean water distribution network and one provides cleaned return water that has traveled through all the pipelines before. From these points, water travels past a first water quality sensor (P1) to the raw water tanks. The task of this sensor is to decide if the quality of the water is good enough. If the quality is not sufficient, the first tanks will be drained.
2. After the raw water tanks, water is pumped to some other tanks, the elevated reservoirs. From these tanks, the water flows further through the system. The amount of water which exits these elevated reservoirs is equal to the required water demand by the consumer tanks described in Step 4. After the reservoirs, the water first passes sensor P2A after which contaminations can be injected into the pipe from the organic and inorganic tanks and are mixed with the water.

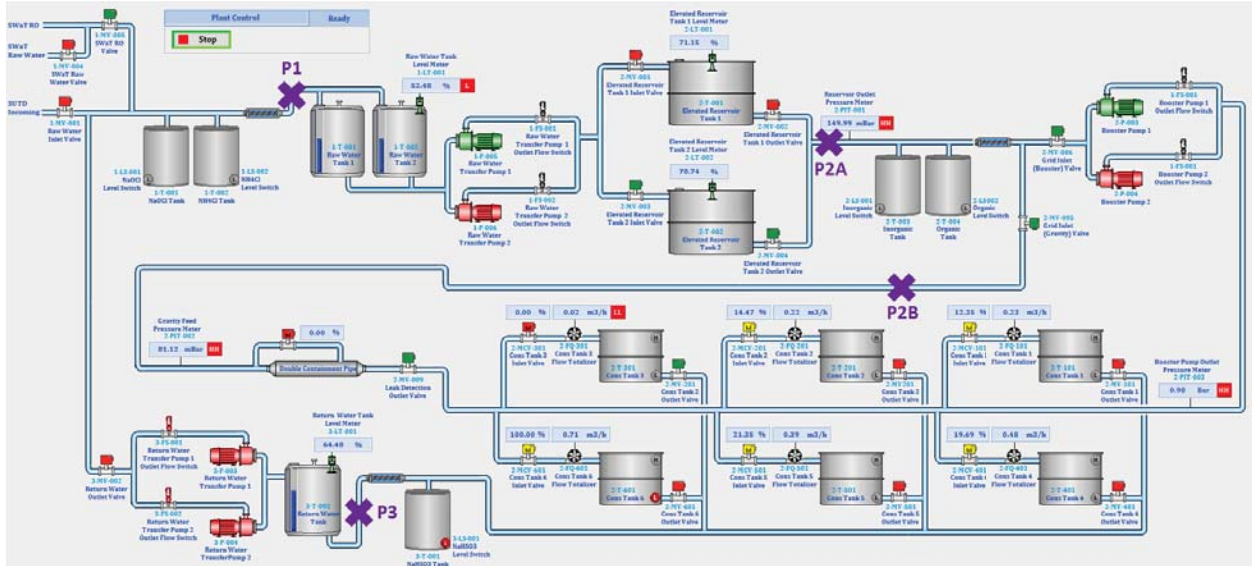


Figure 12: An overview of the WaDi testbed. The four sensors are presented with purple crosses.

3. At that point, the water can take two routes to the consumer tanks, via the gravity feed or via the booster pump stations. It is possible to regulate which of these routes is used by closing certain valves. In the gravity feed, some interesting features are present, including another water quality sensor (P2B). The contaminations which were inserted into the system at the end of Step 2, can be detected by sensor P2B. Due to this, the booster station valve was always closed in our research. The gravity feed also includes a draining option after sensor P2B for situations in which the toxic water should be drained immediately. More on this part from the injection of a contamination to sensor P2B will be given in Section 7.1.2.
4. After the gravity feed, or the booster pumps, the water will reach the consumer tanks. The amount of water going to each consumer tank and thus the flow through the system can be regulated beforehand. The required demand of each consumer tank can change every 5 minutes to simulate dynamic demand in some extent.
5. Finally, the water coming through the consumer tanks will reach the return water system. In this last part, the water can be cleaned in order for it to return to the start of the process and the final sensor P3 is located. This sensor is used to check if the water is clean enough to return to the first part of the system.

7.1.2 Contaminations and Sensors

As can be seen in the previous section, contaminations can be inserted into the WaDi testbed after the elevated reservoirs between sensors P2A and P2B. After a small section of pipes, this contamination can be detected by sensor P2B located in the gravity feed. Sensors further in the network, for example sensor P3, cannot easily detect those contaminants as they are largely diluted due to the tanks in between. In this section, it will be explained how contaminations can be inserted and in what way a sensor can detect contaminations.

When an unhealthy liquid is placed in the contamination tanks, it can be pumped into the network with a chosen flow rate and for a chosen duration. The dosing pumps present at the WaDi testbed can pump a contamination into the pipes with a maximum dosing rate of 0.59 liters per hour. The contaminated water will then flow through some pipes to sensor P2B. In our research, we used three different solutions as contamination: sodium hypochlorite, hydrogen chloride and ammonia. To get an idea of how harmful these liquids are for humans, it was given that their pH levels were approximately 12, 12.5 and 2 respectively.

Information on the length and diameter of these pipes is given in Table 23. It is also good to note there are eleven 90 degrees bends present in this section of pipes. The effect of these bends are ignored for simplicity reasons as this is also not taken into account by EPANET. The total volume of the pipes between the point of injection and the sensor is $10.659 \text{ dm}^3 = 10.659 \text{ liters}$ and can be calculated with the length and diameter. As the flow rate in these pipes is also continuously measured, it can be calculated how long it should take for sensor P2B to detect the contamination after the injection of a certain contamination is started. If the flow rate is for example 0.1 liter per second, it is expected that the contamination reaches sensor P2B after 106.59 seconds.

Table 23: Measurements on the piping between the injection point and sensor P2B which could detect the contamination.

| | From injection point to bifurcation to smaller pipe | From bifurcation to sensor |
|----------------|---|----------------------------|
| Length | 5.24 meter | 1.08 meter |
| Inner diameter | 5.080 cm (2 inches) | 0.635 cm (0.25 inches) |
| Volume | 10.625 dm^3 | 0.034 dm^3 |

The four sensors in the WaDi testbed are the same sensors used in the water network of Singapore. Each sensor consists of multiple probes and measures four different parameters which can all be used to give an indication to the quality of the water. These parameters are conductivity, turbidity, pH and ORP. For each parameter, the range for which the water is healthy is shown Table 24. Conductivity sensors can measure the ability of a liquid to conduct electricity. This value changes when more ions are present in the water. Turbidity readings state something about the particle load in the water. It is an optical measurement of the diffraction of light in the water. pH measures how acidic or basic a liquid is. ORP, or Oxidation Reduction Potential in full, is used as a measure of disinfection potential which is related to how long bacteria can stay alive in that liquid. The units of conductivity, turbidity and ORP are mS/cm (milliSiemens per centimeter), NTU (Nephelometric Turbidity Unit) and mV (millivolt) respectively. pH is unitless. Table 24 also shows an estimation of the four parameters in normal tap water.

Table 24: Ranges of the sensor parameters for which the quality is considered to be sufficient together with the normal reading of tap water.

| | Healthy Range | Normal tap water |
|--------------|-----------------|--------------------|
| Conductivity | 0 to 0.32 mS/cm | 0.15 to 0.20 mS/cm |
| Turbidity | 0 to 4 NTU | 0 to 0.5 NTU |
| pH | 6 to 9 | 7.5 to 8 |
| ORP | -500 to 500 mV | 300 to 500 mV |

The sensors in the WaDi testbed are programmed in a way that an alarm is raised when the

measured value of the sensor is outside of this healthy range. Furthermore, as P2A and P2B are located close to each other, it is also tested if the difference between a reading of P2A and P2B is too large.

The sensors used in the testbed have a lifetime of approximately one year and maintenance, for example calibration of the sensors, should be done every couple of months. In this calibration, it is checked for each of the four parameters with several standard solutions if the performance of the sensors is still good. Furthermore, there is a lot of knowledge by researchers at the WaDi testbed on how to hack the sensors and how to detect these hacks. Even though this is a form of imperfection of the sensors, it is not considered in our research. The information that it is possible to hack water quality sensors only proves that the sensors are not perfect.

7.2 Experiments and Results

We designed three different experiments which all state something about the imperfection of sensors and the way contaminants are being detected. First, the WaDi testbed was put into operation for multiple days to check how sensors behaved in the long run and to get an initial idea on imperfection. Second, the performance of the sensors was evaluated by comparing known levels of standard solutions with the sensor readings several times. Finally, contaminations were added into the network to see how all parameters behaved and if the contaminants could be detected. These different experiments and their results are presented in Sections 7.2.1, 7.2.2 and 7.2.3 respectively.

7.2.1 Long Runs

The first experiment is a simple procedure in which the WaDi testbed is put into operation for a couple of days such that water is continuously flowing through the system and a large amount of sensor data is being collected. Our main focus is on the sensor data of P2A and P2B as these are very close to each other. The same water that passes P2A will reach P2B within a couple of minutes, depending on the flow rate. Therefore, it is expected that the long run sensor data is similar to each other and highly correlated. Another point of interest is finding indications of imperfection in the long resulting time series.

Two different long runs are being considered. The first one consists of data of four days and the second one of almost three days. We will refer to them as LR1 and LR2 respectively. The two long runs were separated by several weeks. In Figures 13a and 13b, the conductivity and pH levels over time of LR1 are being shown. In each figure, the red line represents the time series of sensor P2A and the blue one of P2B.

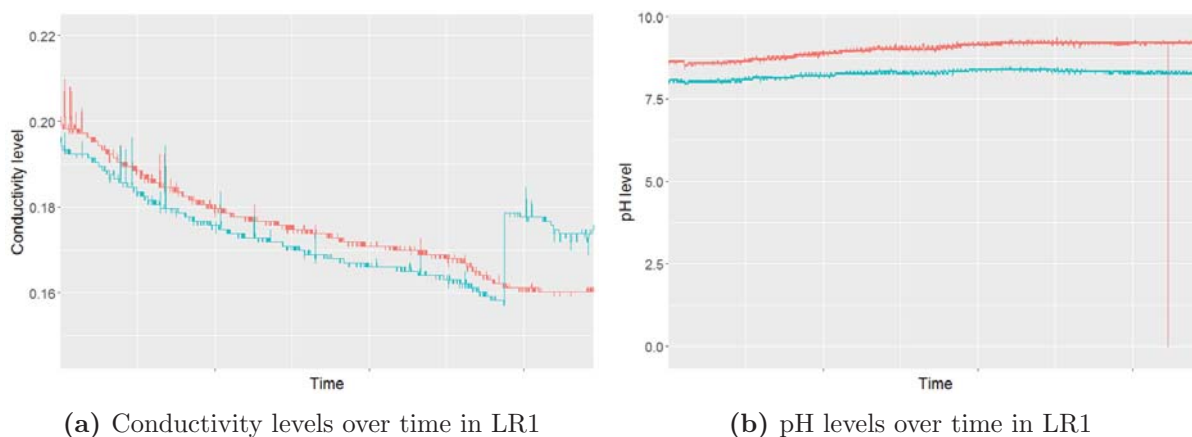


Figure 13: Two time series of sensor readings of P2A (red) and P2B (blue) in LR1.

Even though the time series of both sensors are quite similar in these two figures of Figure 13, some strange events are visible. There is an inexplicable jump present in the P2B time series of conductivity in which the conductivity levels suddenly increased with 0.021. This jump can cause the sensor to raise an alarm in real WDNs even though no contamination was added into the system. Some missing values were present close to the end in the P2A time series of pH in Figure 13b. This is probably due to measurement or communication errors. In total, there were 2 inexplicable jumps and 4 time periods with missing values in all collected data. The longest time

period in which a sensor reading was missing in the time series was a period of almost two minutes in the P2B time series of turbidity. If an attack passes a sensor during one of the time periods with measurement errors, the contamination will probably not be detected and the consequences can be severe.

In Figures 14a and 14b, two LR2 time series of respectively pH and turbidity are given.

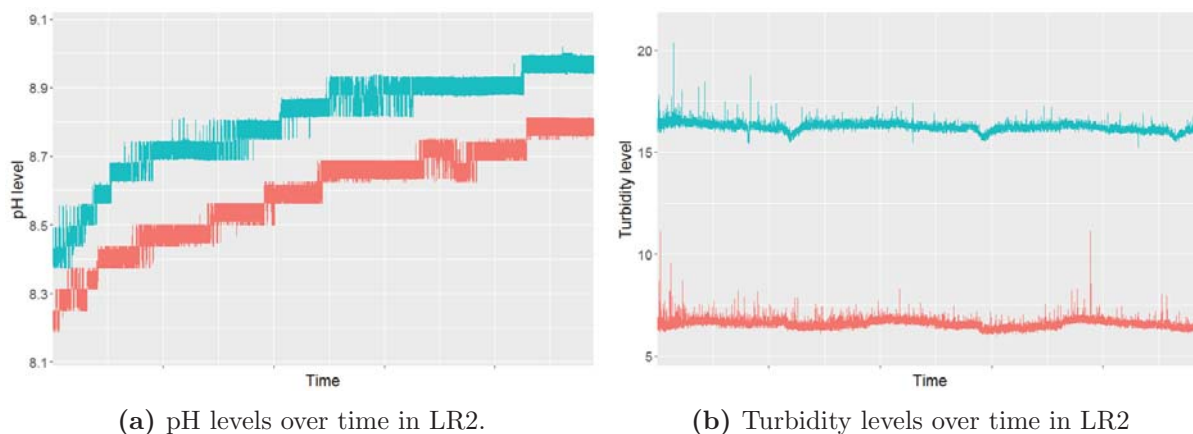


Figure 14: Two time series of sensor readings of P2A (red) and P2B (blue) in LR2.

If the sensors are working accurately, it is expected that the time series of sensors P2A and P2B should approximately be the same because the water which passes both sensors is the same. In Figure 14a, it looks like this is approximately correct whereas the turbidity long run of Figure 14b shows more instability in which the peaks and changes are mostly only visible in one time series. The Pearson’s correlation coefficient (see: [42]) between both time series can be calculated as a formal measurement to see in what extent the time series follow the same patterns. For this, the time it takes for water to travel from sensor P2A to P2B has to be taken into account. In LR1, the average flow rate was 313 liters per hour and in LR2 it was 630 liters per hour. Using these values and the volume of the pipelines in between, it can be calculated that the water should reach P2B in approximately two minutes and one minute in LR1 and LR2 respectively. Furthermore, as outliers can have a large impact on the correlation, the missing values were replaced with the last correctly measured value. The time series with a sudden large jump are split in two time series in order to evaluate the correlation before and after the permanent increase.

Correlation is measured on a scale from -1 to 1 in which a 1 means that there is a perfect positive linear relationship between two variables. If one variable increases, the other variable increases with the same magnitude. A value close to 0 means that there is no linear relationship and a negative value represents a negative relationship in which both variables move in opposite directions. It is expected that the sensor readings of P2A and P2B show a high correlation. The highest correlation was found with the conductivity time series of LR1 before the jump which was shown in Figure 13a. After the jump, the correlation was much lower (0.995 versus 0.652). The correlation of the other conductivity time series was just below 0.9 and all pH and ORP time series were highly correlated with values higher than 0.9.

In contrast to these results of pH, ORP and conductivity, the turbidity time series hardly showed any correlation. The correlation between the time series of P2A and P2B were -0.126 and 0.288 for LR1 and LR2 respectively. This is a first indication that the turbidity sensor is quite unstable.

As can be seen in Figure 14b, the turbidity sensor will often raise a false alarm due to the large number of small peaks. Therefore, the turbidity sensors will not be trusted by operators and might be ignored when a real contamination passes that sensor.

So, using these long runs, the first proof of imperfection and inaccuracy is found as the existence of measurement or communication errors, inexplicable jumps and instability in some sensor readings is shown. The four time series which were not presented in this section can be found in Appendix D.2 along with all correlation values.

7.2.2 Standard Solutions

In Section 7.1.2, it was mentioned that the performance or accuracy of the sensors should be checked every couple of months using some standard solutions and that the sensors should be calibrated if necessary. This procedure can be illustrated with an example. Suppose we have a standard solution with a known pH level of 7. A sensor which is put into a sample of this liquid should measure a pH of 7. If this measured value is not 7 but 5 or 8.5, the sensor must be recalibrated such that it will give the accurate value in the near future. This calibration should be done every three months. Most of the parameters measured at a sensor were calibrated using a two-point calibration and thus with two standard solutions.

This calibration was done at the WaDi testbed before and in between several other experiments. At some point, it was clear that a parameter for a sensor could become unreliable very fast. For example, the ORP value of sensor P2A was showing 530 instead of the standard 430 within a week after a previous calibration. Therefore, experiments were done to formally show this inaccuracy or instability over time and to find evidence for sensor degradation or drifting. During these experiments, the sensors were not calibrated and the readings were tested every couple of days for all parameters and sensors to check how far each sensor reading was off. Before the start of these experiments, sensors P2A and P2B were checked and, if necessary, calibrated approximately a month after the other two sensors were last checked.

In Table 25, the results for all four sensors for the standard solution with a pH level of 10.01 over time are shown. The first time, on September 7, P2A and P2B were the only sensors which were being checked. After September 7, the accuracy of all sensors was checked every six days.

Table 25: The performance of the pH sensors over some time.

| Sensor | Standard Solution: pH = 10.01 | | | |
|--------------|-------------------------------|-----------|-----------|---------|
| | P1 | P2A | P2B | P3 |
| Last checked | August 10 | August 29 | August 29 | July 28 |
| September 7 | - | 9.68 | 10.05 | - |
| September 13 | 12.31 | 9.79 | 9.95 | 9.82 |
| September 19 | 12.71 | 9.93 | 10.12 | 10.09 |
| September 25 | 12.98 | 10.10 | 10.31 | 10.15 |

In order to evaluate if a sensor does not give accurate readings anymore, we allow for a five percent margin. If the readings are within 5% of the known value of the standard solution, the performance of the sensor is still considered to be sufficient. For the standard solution with pH level of 10.01, the sensor reading should be roughly between 9.5 and 10.5. It can be seen in Table 25 that all sensor readings are still within this range except for sensor P1. The readings of this

sensor are between 12 and 13. For the other standard solution with a known pH level, a pH of 7, this sensor also shows too high readings. This means that the pH sensor of P1 is unreliable after just a month and needs to be recalibrated.

Even though the other three sensors are within the right range, they all show some upwards trend. These pH sensors are all slowly drifting away from the right value which can also cause problems in the long-term. When sensors are programmed that they raise an alarm when a sensor reading is outside a certain range, drifting can lead to false positives or even to non-detections of attacks. Non-detections can for example occur when a sensor reading of pH drifts towards the maximum value within the healthy range when the quality of the tap water is in fact still in the middle of this healthy range. An acidic attack causes a large decrease in the pH value such that the sensor reading is close to the minimum value of the healthy range when it is in fact harmful for humans.

For sensor P2A, the readings over time for conductivity, ORP and turbidity can be seen in Table 26. ORP is the only parameter for which only one standard solution was available.

Table 26: The sensor readings of sensor P2A over time for conductivity, ORP and turbidity.

| P2A Std. Solution value | Conductivity | | ORP | Turbidity | |
|----------------------------|--------------|-------|-----|-----------|------|
| | < 0.1 | 0.447 | 430 | 0.1 | 20 |
| September 7 | 0.066 | 0.514 | 425 | 2.7 | 37.0 |
| September 13 | 0.059 | 0.590 | 421 | 6.7 | 28.8 |
| September 19 | 0.055 | 0.505 | 420 | 7.8 | 21.8 |
| September 25 | 0.057 | 0.587 | 412 | 8.1 | 47.4 |

A downwards drifting pattern was found for the ORP reading. This was the case for all four sensors. The conductivity readings were more stable but in general a bit too high for the second standard solution compared to the value of 0.447. We suspect that this is mainly due to the fact that conductivity was checked every time, but the last time it was calibrated was more than half a year ago. The values around 0.5 were probably tolerated in combination with the good sensor readings of the other " < 0.1 " standard solution. The turbidity readings were, just as we concluded in Section 7.2.1, very unstable.

Finally, Table 27 shows for each sensor and parameter which readings were still correct (+) after a month and which were inaccurate (-). A plus-minus sign means that the sensor reading was still stable and within or close to the five percent range for most of the evaluations or that it was unable to form a conclusion with just three or four data points.

Table 27: A summary of all sensors and parameters which shows which sensor readings were still accurate (+), inaccurate (-) or indeterminable (\pm) after a month.

| | Conductivity | ORP | pH | Turbidity |
|-----|--------------|-----|----|-----------|
| P1 | \pm | - | - | - |
| P2A | \pm | + | + | - |
| P2B | \pm | + | + | \pm |
| P3 | \pm | - | + | - |

As all conductivity readings were somewhat too high, a \pm sign was given for all of them as the last calibration happened more than six months ago and a formal conclusion can therefore

not be drawn. Remember that according to the guidelines, the sensors should be calibrated every three months. After one month, only two ORP and three pH readings were still accurate and the turbidity sensor was unstable and inaccurate for most sensors. Even though the considered sensors are close to the end of their lifetime, the results are quite evident. Maintenance or calibration should happen more often than currently recommended because the sensors are drifting quite fast. As we suspect that the performance of the sensors is better in the beginning of their lifetime, sensor degradation also probably exists. All in all, this is another result indicating that the assumption that sensors can perfectly detect contaminations is wrong. The results of all parameters and sensor readings including the date of the last check and last calibration for each parameter can be found in Appendix D.3.

7.2.3 Adding Contaminations

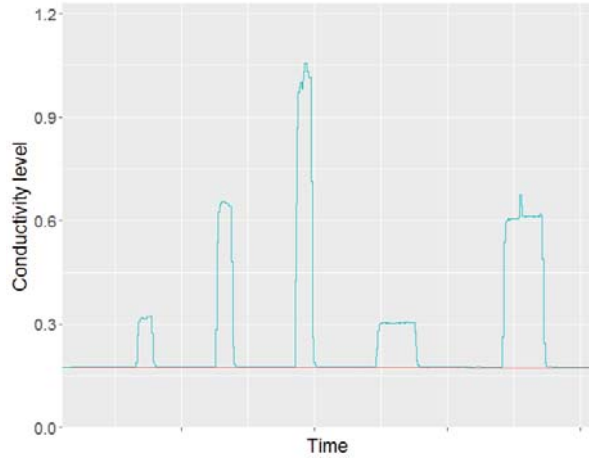
As mentioned earlier, three different liquids were used in our research as a contamination and added into the WaDi testbed. A sodium hypochlorite solution (Hypo), an ammonia solution (Ammonia) and a hydrogen chloride solution (HCl) were available. Our main interests when adding these contaminations into the water network was to see which sensor parameters could detect the contaminations, how long it took to detect a contamination and how long the contamination stayed in the pipelines. Besides that, we focus on the effects of factors such as flow rate, dosing rate, dosing duration or the type of contamination on the detection.

Multiple experiments were performed with each contamination. The liquid was added into the network in a different setting every time, which means a different combination of the water flow rate, the dosing rate and the dosing duration. All different values used for the different setting can be found in Table 28. Recall that the maximum dosing rate for the contamination pumps is 0.59 liters per hour. The maximum flow rate for the water through the gravity feed of the WaDi testbed was 1000 liters per hour. Only two dosing durations were used as the supply of these contaminations was limited.

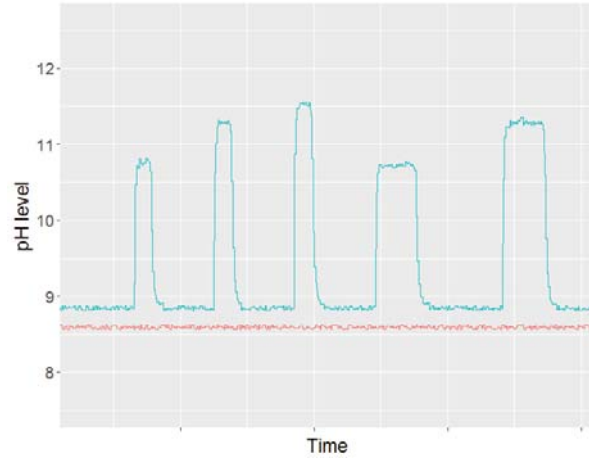
Table 28: Different values of flow rate, dosing rate and dosing duration used.

| | Water flow rate | Dosing rate | Dosing duration |
|--------|-----------------------|-----------------------|-----------------|
| High | ± 950 liters per hour | 0.472 liters per hour | 5 minutes |
| Medium | ± 600 liters per hour | 0.354 liters per hour | - |
| Low | ± 300 liters per hour | 0.236 liters per hour | 2 minutes |

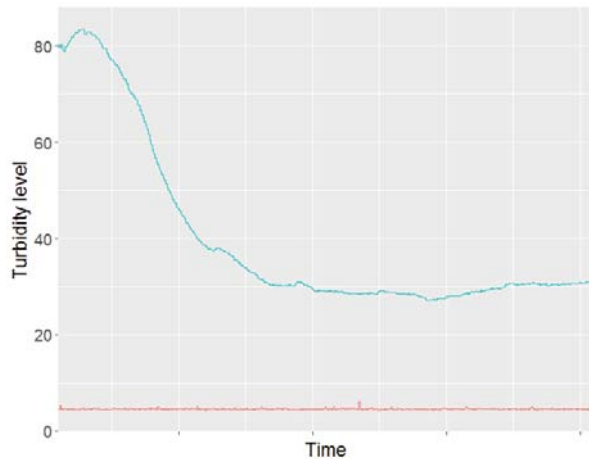
As a contamination is inserted between sensors P2A and P2B, sensor P2B should be able to detect the contamination. Figure 15 shows what happens to the time series of the four sensor readings of P2B when Ammonia is added. The time series of sensor P2A is also added to the picture as a benchmark. Ammonia was added five times in these figures, each time in a different setting. In each setting, the water flow rate was high. The combination (Dosing rate, dosing duration) was (Low, Low), (Medium, Low), (High, Low), (Low, High) and (Medium, High) for settings one to five respectively.



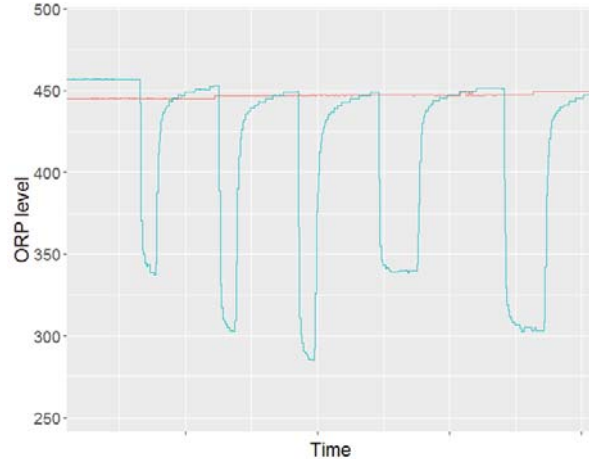
(a) Conductivity levels over time



(b) pH levels over time



(c) Turbidity levels over time



(d) ORP levels over time

Figure 15: Time series of the four sensor readings of P2A (red) and P2B (blue) over time when Ammonia is added into the network with five settings which differed in dosing rate and dosing duration.

The most remarkable result can be found in the turbidity levels in Figure 15c. All other three time series show clear peaks when the contamination is inserted and the influence of the dosing rate and dosing duration is very clear. Take for example the conductivity readings of Figure 15a. The higher the dose, the higher the peak and the longer the duration, the wider the peak. These findings are the same for all three contaminations. In these figures, the flow rate was high each time. When less normal water was flowing through the pipelines per hour, the peaks were somewhat higher.

So, the turbidity sensors are not able to detect Ammonia. In Table 29, a summary is presented for all three contaminations showing which parameters can detect a certain contamination and if the sensor reading increases or decreases. The corresponding figures for the other two contaminants with high flow rates can be found in Appendix D.4.

As can be seen in Table 29, turbidity is not able to detect any of these three contaminants. The turbidity sensor measures the diffraction of light in the water which means that a reading

Table 29: A summary which shows which parameters reacted (+) to each of the contaminations. The arrow shows if it is a positive peak or a negative peak. A minus sign means that the sensor parameter was unable to detect the contamination and the plus-minus sign shows that there were some peaks present, but not as clear as with the other parameters.

| | Conductivity | Turbidity | pH | ORP |
|------|--------------|-----------|-------|-------|
| Hypo | + (↑) | - | + (↑) | ± (↑) |
| Ammo | + (↑) | - | + (↑) | + (↓) |
| HCl | + (↑) | - | + (↓) | + (↑) |

states something about the particle load in the water. As all three considered contaminants are transparent liquids, none of these make the contaminated water turbid. However, the resulting time series of these experiments showed again that the turbidity sensor readings are quite unstable as was also shown in the previous sections. Conductivity, pH and ORP all showed some peaks when the contamination was added. The peaks in the ORP time series when Hypo was added were not as clear as the ones of conductivity and pH such that ORP alone is probably not enough to detect that contaminant. This indicates that multiple sensor parameters are really necessary to make a decision.

Finally, we are interested in the time it takes for a contamination to reach the sensor and how long the contamination remains visible in the water. Using the calculated volume between the point of injection and sensor P2B in 7.1.2 and the measured flow rate, the expected detection time in seconds can be calculated. This formula is shown in Equation 41.

$$\text{Expected detection time in seconds} = \frac{\text{Volume of pipelines (dm}^3\text{)}}{\text{Flow rate (dm}^3\text{/s)}} \quad (41)$$

The volume of the pipelines is 10.659 dm³ and three different flow rates were used in our research. In Table 30, the average flow rate in the three settings is shown. With each flow rate, the expected detection time can be calculated. This value is compared with the average observed detection time of the added contamination. This observed detection time is based on the first observation after contamination was added for which the sensor readings differ more than 5% from the mean of the previous 100 observation points. So, not on the range or difference between the sensor reading of P2A and P2B as it was described in Section 7.1.2. This was chosen due to the instability and inaccuracy of the time series which were observed in Section 7.2.1.

Table 30: An overview of the detection times of contaminants in the WaDi testbed for different flow rates.

| | Average flow rate | Expected detection time | Observed detection time |
|------------------|---------------------|-------------------------|-------------------------|
| High flow rate | 960 liters per hour | 40 seconds | 49 seconds |
| Medium flow rate | 610 liters per hour | 63 seconds | 67 seconds |
| Low flow rate | 312 liters per hour | 123 seconds | 111 seconds |

The difference between the observed and expected detection time is quite interesting. When the flow rate is high, the expected time is too low and when the flow rate is low, the expected time is too high. The average observed detection time for all different flow rates is very stable as the

variance is quite small. The detection of contaminants when the flow rate was high was for example always close to 50 seconds. There was hardly any difference in the detection times for the different contaminations or different dosing rates.

This result is somewhat strange and inexplicable and we are not able to give a solid explanation for this difference. It is possible that the measured flow rate is not correct as this flow rate was only measured in one point of the pipelines. The consequence however is that the detection time, which is used as one of the objectives in our research, might be calculated in an incorrect way. More research should be done to clarify this. Furthermore, after the adding of the contaminants was stopped, most sensor readings, especially conductivity and pH, quickly returned to their old values. This can also be seen in Figure 15.

The resulting time series when contaminations are added into the water are mainly interesting as a validation of the water network simulations done in EPANET. Even though some time periods of missing values were again observed in the time series of the sensor readings, little can be said about the imperfection of sensors using these results. All added contaminants were detected by at least two sensor parameters. After the contaminant is added, the sensor readings quickly increase or decrease to a new static level. This means that when the sensor misses this initial change due to errors, the contamination will possibly not be immediately detected.

Furthermore, as lower water flow rates and higher contamination dosing rates result in higher peaks, it can be said that this can slightly influence the probability of detecting a contamination. This result can mean that Assumption (iii), in which it is stated that all sensors have the same probability of detecting a contamination and the effect of dilution is ignored, is not valid. In reality, sensors which are placed further away in the network might observe such small peaks due to branching and dilution effects that some attacks will not be detected. Therefore, the probability of detection p by a sensor might also differ throughout the network. Due to the lay-out of the WaDi testbed, experiments about these effects of dilution or branching are not possible.

As a final conclusion of the experiments performed on the WaDi testbed, it can be stated that the first evidence for sensor imperfection has been found. Failing to detect a contamination can occur as missing values were observed in the data series. Besides that, the sensor readings are quite unstable as inexplicable peaks and jumps in the sensor readings were found and the sensors become inaccurate and unreliable within several weeks due to drifting. Especially the sensor readings for turbidity are very unstable. However, using these experiments, it is not yet possible to provide water agencies with a clear indication of the value of p . One minus the time periods with missing data in the long run divided by the total number of data points can for example be used, but it does not capture everything.

8 Discussion and Further Research

In this research, we have constructed a simple greedy algorithm and several extended algorithms which were able to find a sensor placement for a given network and a given probability of detection p . If this value for p , or a mean value as \bar{p} , is known, the presented methods will return a sensor placement with a high performance for that given p .

In reality, this value p is not known at the moment. Consider two research groups who want to convince a water agency to use their sensor placement. One is based on $p = 0.97$ and the other one on $p = 0.99$. At this moment, no results exist which validate one of these claims. Using the WaDi testbed, it was shown that there is a chance of not detecting contaminations due to for example errors, instability or drifting such that p is not equal to 1. However, the available time for experiments was too little to provide a clear statement of the performance of these sensors. Especially because the used sensors were not new when our experiments started.

Furthermore, p is believed to be not equal for all sensors in the network and not every sensor degrades at the same rate. In future research, probability distributions should be defined for the detection probability p . When the value of p is different for each sensor or follows a probability distribution, our presented methods are still valid. More research must be done using this testbed or other testbeds. It is recommended that many, brand new and perfectly calibrated sensors are evaluated over a long time by independent researchers. Real testbeds can also be used for research on dilution, branching, finding out how different contaminants behave in water networks and dynamic flow directions.

The results presented for the methods which incorporate dynamic demand and flow patterns look promising. However, using our assumptions, the interchange between two different patterns is ignored. This assumption has always been used by researchers. The interchanges can however have a large impact on many objectives, but little to nothing is known about this. It can be possible that a contaminated flow reaches a completely different part of the network then the part it should have reached when the demand pattern was considered to be static at the time of intrusion.

Finally, using the data of the experiments on the WaDi testbed, it can be shown that many false positives occurred. A sensor can measure a rapid change in a certain parameter and therefore raise an alarm when no contaminants are being added. Due to the large number of false positives and the fact that a water network will not be shut down for each positive sensor reading, it is possible that even when a contaminant is added and the sensor is detecting something, it can be considered as a false positive by the system or by employees. This can result in a non-detection even when it was detected. The occurrence of false positives should therefore perhaps also be taken into account in further research.

Overall, in this subfield of research, it is recommended to perform more research on real-life water networks and testbeds to gain more insights in water networks and to better decide on the value or distribution of p . Besides that, this research can also initiate studies into other sensor networks. Sensors are not only used to monitor water distribution networks. Even though other sensor networks are designed differently, it is worth looking into the question if it is good to account for the possible failure or non-detection of a sensor.

9 Conclusion

Worldwide, there is an upcoming threat of water pollution and terrorist attacks on water distribution networks. Accidental or deliberate incidents will affect the quality of the drinking water and can cause many fatalities and a huge economic impact. To prevent these consequences, WDNs should use a sensor network to monitor the quality of the drinking water.

In this study, we have focused on making these sensor networks more robust by incorporating sensor imperfection, multiple objectives and dynamic demand. It is assumed that attacks can only occur at accessible or vulnerable nodes in the water network. The considered objectives in this research are minimizing the time to detection and the impact of an attack and maximizing the probability of detection and the probability of identifying the source of the attack. This last objective is quite new and was never studied in the imperfect setting before. Two ways to calculate this objective have been considered. However, several drawbacks were present when one of these two objective formulations was used. It has been decided that the concept of source identification is useful after an attack occurred in a water network, but not in the design phase of a sensor network when sensors are considered to be imperfect.

In previous studies, it was mostly assumed that sensors are perfect. However, sensors can fail to detect a contamination due to errors, failures, maintenance difficulties, degradation, drifting or hacking. The first evidence for this imperfection has been found after performing several experiments on the WaDi testbed located in Singapore. Therefore, it can be concluded that assuming perfect sensors is not correct, but more research should be done to specify the level of uncertainty.

In order to design a sensor placement, the problem has been converted to a changing weighted set covering formulation with a bipartite graph. A greedy algorithm and several extended algorithms can be used to solve this problem. This greedy algorithm can incorporate sensor imperfection, dynamic demand and multiple objectives and is able to find sensor placements for large networks in reasonable time. The algorithms have been tested using several water distribution networks based on real networks.

It was found that large differences in the sensor locations can be found when considering slightly imperfect sensors compared to perfect sensors. When many sensors are placed and many vulnerable nodes are considered, the sensor network already differs when using p close to 0.999 instead of $p = 1$. Together with the found evidence of sensor imperfection on the WaDi testbed, it is recommended to take sensor imperfection into account. It has been shown that when there is dynamic demand and the sensors are imperfect, taking this imperfection into account is even more important. On the other hand, taking sensor degradation into account when designing a sensor placement is not necessary. Using the mean sensor probability during its lifetime is enough to find good sensor placements.

Finally, it can be stated that the introduced greedy algorithm guarantees a high level of performance. For large networks, no improvements could be found by the more sophisticated and more time-consuming algorithms. The differences between the objective values in the small Bangalore network are very small. In smaller networks, it is still possible to use the non-dominated greedy algorithm or some local search steps to find improvements within reasonable time. However, for large real water networks, this is not recommended. The new framework with the greedy algorithm and the changing weights is fast and able to find good solutions and therefore recommended to use when a sensor placement should be chosen.

References

- [1] P. U. B. Singapore. Managing the water distribution network with a smart water grid. Smart Water, 1:1–13, 2016.
- [2] R. Murray, W. E. Hart, C. A. Phillips, J. Berry, E. G. Boman, R. D. Carr, L. A. Riesen, J.-P. Watson, T. Haxton, J. G. Herrmann, et al. Us environmental protection agency uses operations research to reduce contamination risks in drinking water. Interfaces, 39(1):57–68, 2009.
- [3] W. Yan, J. Li, and X. Bai. Comprehensive assessment and visualized monitoring of urban drinking water quality. Chemometrics and Intelligent Laboratory Systems, 155:26–35, 2016.
- [4] P. Fowler. Tests reveal source of havelock north water contamination. New Zealand Radio, February 2017. URL <http://bit.ly/2pdGULk>. Accessed: 2017-04-14.
- [5] M. LeBien. Corpus christi water restrictions lifted for some residents. CNN, December 2016. URL <http://edition.cnn.com/2016/12/15/health/corpus-christi-water/>. Accessed: 2017-04-14.
- [6] C. B. S. News. Chemical spill shuts off water to 300K in West Virginia, 2014. URL <https://www.cbsnews.com/news/some-people-treated-for-water-related-issues-in-w-va/>. Accessed: 2017-04-14.
- [7] S. Fenton. 5,000 scottish homes hit with contaminated water. Independent, June 2015. URL <http://www.independent.co.uk/news/uk/home-news/5000-scottish-homes-hit-with-contaminated-water-10328100.html>. Accessed: 2017-04-14.
- [8] P. H. Gleick and M. Heberger. Water conflict chronology. In The world’s water, pages 173–219. Springer, 2014.
- [9] C. Cameron. Feds arrest al qaeda suspects with plans to poison water supplies, 2002. URL <http://fxn.ws/1lp5hQa>. Accessed: 2017-05-12.
- [10] J. Ware. Tens of thousands without water in kosovo after fears of isis plot to poison reservoir, 2015. URL <https://ind.pn/1RY74GM>. Accessed: 2017-05-12.
- [11] J. Raich. Review of sensors to monitor water quality. European Reference Network for Critical Infrastructure Protection (ERNICIP) project, 2013.
- [12] V. R. Palleti, S. Narasimhan, R. Rengaswamy, R. Teja, and S. M. Bhallamudi. Sensor network design for contaminant detection and identification in water distribution networks. Computers & Chemical Engineering, 87:246–256, 2016.
- [13] L. A. Rossman et al. Epanet 2: users manual. 2000.
- [14] B. H. Lee, R. A. Deininger, and R. M. Clark. Locating monitoring stations in water distribution systems. Journal (American Water Works Association), pages 60–66, 1991.

- [15] B. H. Lee and R. A. Deininger. Optimal locations of monitoring stations in water distribution system. Journal of Environmental Engineering, 118(1):4–16, 1992.
- [16] A. Ostfeld, J. G. Uber, E. Salomons, J. W. Berry, W. E. Hart, C. A. Phillips, J.-P. Watson, G. Dorini, P. Jonkergouw, Z. Kapelan, et al. The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. Journal of Water Resources Planning and Management, 134(6):556–568, 2008.
- [17] J. Berry, W. E. Hart, C. A. Phillips, J. G. Uber, and J.-P. Watson. Sensor placement in municipal water networks with temporal integer programming models. Journal of water resources planning and management, 132(4):218–224, 2006.
- [18] A. Krause, J. Leskovec, S. Isovitsch, J. Xu, C. Guestrin, J. VanBriesen, M. Small, and P. Fischbeck. Optimizing sensor placements in water distribution systems using submodular function maximization. In Water Distribution Systems Analysis Symposium 2006, pages 1–17, 2006.
- [19] G. Dorini, P. Jonkergouw, Z. Kapelan, F. Di Pierro, S. Khu, and D. Savic. An efficient algorithm for sensor placement in water distribution systems. In Water Distribution Systems Analysis Symposium 2006, pages 1–13, 2006.
- [20] Z. Y. Wu and T. Walski. Multiobjective optimization of sensor placement in water distribution systems. In Water Distribution System Analysis Symposium 2006, 2006.
- [21] W. E. Hart, J. W. Berry, E. G. Boman, R. Murray, C. A. Phillips, L. A. Riesen, and J.-P. Watson. The teva-spot toolkit for drinking water contaminant warning system design. In World Environmental and Water Resources Congress 2008: Ahupua’A, pages 1–12, 2008.
- [22] V. Chvatal. A greedy heuristic for the set-covering problem. Mathematics of operations research, 4(3):233–235, 1979.
- [23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation, 6(2):182–197, 2002.
- [24] A. Preis and A. Ostfeld. Multiobjective contaminant sensor network design for water distribution systems. Journal of Water Resources Planning and Management, 134(4):366–377, 2008.
- [25] J. W. Berry, L. Fleischer, W. E. Hart, C. A. Phillips, and J.-P. Watson. Sensor placement in municipal water networks. Journal of Water Resources Planning and Management, 131(3): 237–243, 2005.
- [26] Y. Shastri and U. Diwekar. Sensor placement in water networks: A stochastic programming approach. Journal of water resources planning and management, 132(3):192–203, 2006.
- [27] H. Shen, E. McBean, and Y. Wang. Sensor placement under nodal demand uncertainty for water distribution systems. In Securing Water and Wastewater Systems, pages 123–133. Springer, 2014.
- [28] A. Ostfeld and E. Salomons. Optimal layout of early warning detection stations for water distribution systems security. Journal of Water Resources Planning and Management, 130(5): 377–385, 2004.

- [29] M. Weickgenannt, Z. Kapelan, M. Blokker, and D. A. Savic. Risk-based sensor placement for contaminant detection in water distribution systems. Journal of Water Resources Planning and Management, 136(6):629–636, 2010.
- [30] J. Berry, R. D. Carr, W. E. Hart, V. J. Leung, C. A. Phillips, and J.-P. Watson. On the placement of imperfect sensors in municipal water networks. In Water Distribution Systems Analysis Symposium 2006, pages 1–13, 2008.
- [31] J. Berry, R. D. Carr, W. E. Hart, V. J. Leung, C. A. Phillips, and J.-P. Watson. Designing contamination warning systems for municipal water networks using imperfect sensors. Journal of Water Resources Planning and Management, 135(4):253–263, 2009.
- [32] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. Journal of Water Resources Planning and Management, 134(6):516–526, 2008.
- [33] J. Xu, M. Small, P. Fischbeck, and J. VanBriesen. Integrating location models with bayesian analysis to inform decision making. Journal of Water Resources Planning and Management, 136(2):209–216, 2010.
- [34] A. Krause and C. Guestrin. Robust sensor placement for detecting adversarial contaminations in water distribution systems. In World Environmental and Water Resources Congress 2009: Great Rivers, pages 1–10, 2009.
- [35] X. Xu, Y. Lu, S. Huang, Y. Xiao, and W. Wang. Incremental sensor placement optimization on water network. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 467–482. Springer, 2013.
- [36] J.-P. Watson, R. Murray, and W. E. Hart. Formulation and optimization of robust sensor placement problems for drinking water contamination warning systems. Journal of Infrastructure Systems, 15(4):330–339, 2009.
- [37] A. Aziz, T.-H. Lee, and A. Prakash. Elements of Programming Interviews: The Insiders’ Guide. EPI, 2012.
- [38] N. Deo. Graph Theory with Applications to Engineering and Computer Science (Prentice Hall Series in Automatic Computation). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1974. ISBN 0133634736.
- [39] R. Datta. General and sensitivity analysis of water distribution networks. PhD thesis, Ph. D. dissertation, Indian Institute of Science, Bangalore, India, 1992.
- [40] C. W. S. Centre for Water Systems. CWS benchmarks, 2017. URL <http://emps.exeter.ac.uk/engineering/research/cws/resources/benchmarks>. Accessed: 2017-08-23.
- [41] J. E. Van Zyl. A methodology for improved operational optimization of water distribution systems. PhD thesis, University of Exeter UK, 2001.
- [42] K. Pearson. Note on regression and inheritance in the case of two parents. Proceedings of the Royal Society of London, 58:240–242, 1895.









Appendix

A List of Symbols

Table A.1: An overview of all symbols used in this report and their meaning.

| Symbol | Meaning |
|-------------------------|--|
| V | The set of all nodes in the water distribution network |
| E | The set of all edges or pipelines in the water distribution network |
| V' | The set of all m vulnerable nodes. $V' \subset V$. Index of an element = v (or u) |
| $V \setminus V'$ | The set of all M possible sensor locations. Index of an element = j |
| A_v | The set of all nodes in $V \setminus V'$ downstream of vulnerable node v |
| B | Number of sensors to be placed in the network |
| p | Probability that a sensor detects a contamination |
| X | The decision variable denoting the sensor placement. $X \subseteq V \setminus V'$, $ X = B$ |
| S_v | The set of sensors able to detect an attack on node v . $S_v = X \cap A_v$ |
| n_v | The set of sensors able to detect an attack on node v . $n_v = S_v $ |
| $\mathcal{P}(S_v)$ | The powerset of S_v excluding the null set. Index of an element = c |
| V'_c | A subset of V' with elements v for which c is a subset of S_v |
| $I(v, c)$ | Indicator function which is 1 if combination c can only happen for node v |
| $I^\alpha(v, c)$ | Indicator function which is 1 if the probability that c happens after an attack took place on node v is larger than α |
| $q_{v,j}^t$ | The time it takes before a contamination on node v reaches node j |
| $q_{v,j}^z$ | The amount of water consumed after time $q_{v,j}^t$ |
| $q_{v,\infty}^t$ | Predetermined time or penalty cost for a non-detection |
| $q_{v,\infty}^z$ | The estimated impact of the contamination for a non-detection |
| L_v^t | Ordered list of all S_v sensors based on detection time. Index of item = i |
| k | Position from the back of possible sensor location j in L_v^t , $j \notin L_v^t$ |
| D | Probability of detection of the whole network |
| D_v | Probability of detection for a contamination on node v |
| F | Probability of perfect identification of the whole network |
| F_v | Probability of perfect identification for a contamination on node v |
| F^α | Probability of being α -certain on identification of the whole network |
| F_v^α | Probability of being α -certain on identification for a contamination on v |
| T | Normalized reduction of detection time of the whole network |
| T_v | Normalized reduction of detection time for a contamination on node v |
| Z | Normalized reduction of amount of contaminated water consumed of the whole network |
| Z_v | Normalized reduction of amount of contaminated water consumed for a contamination on node v |
| w_Δ | Weight of objective Δ in the objective formulation. $\Delta \in D, F, T, Z$ |
| $\Gamma_{(v,j)}^\Delta$ | Weight on arc (v, j) for objective Δ . $\Delta \in D, F, T, Z$ |
| Θ | Set of different demand or flow patterns in the WDN. Index of item in set = θ . |

Table A.2: An overview of all EPANET symbols to clarify the representation of the different water distribution networks.

| Symbol | Meaning |
|---|--|
|  | A normal demand node or junction. |
|  | A pipe between two nodes. |
|  | A water reservoir. Water from the reservoirs is used to distribute water to the households and to fill tanks. |
|  | A water tank. Used to collect, store and distribute water in the water network. |
|  | A water pump. Used to pump water through the water network. |
|  | A valve. Many different types of valves can be used in a water distribution network. They can control things as flow and pressure. |
|  | A check valve. A valve which restricts flow to one direction. |
|  | The flow direction of the water in the pipe. |

B Proof of Proposition 2

Proof of Proposition 2. Let us first recall the equation which was given in Proposition 2 which should be proven:

$$\text{To prove:} \quad (1 - T) + \sum_{v \in V'} \Gamma_{(v,j)}^T(k) \leq 1 \quad \forall j \forall k$$

This equation can be rewritten into:

$$\text{To prove:} \quad T - \sum_{v \in V'} \Gamma_{(v,j)}^T(k) \geq 0 \quad \forall j \forall k$$

such that it is necessary to show that the the left side of the equation is larger or equal than zero. By implementing the full formulas of T and $\Gamma_{(v,j)}^T(k)$, putting everything within one summation and combining several terms, this proof can be continued as follows:

$$\begin{aligned} & T - \sum_{v \in V'} \Gamma_{(v,j)}^T(k) \\ &= \sum_{v \in V'} \frac{1}{mq_{v,\infty}^t} \left(\sum_{i=1}^{n_v} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^t \right) + (1-p)^{n_v} q_{v,\infty}^t \right) - \sum_{v \in V'} \frac{1}{mq_{v,\infty}^t} p(1-p)^{n_v-k} \left((1-p)^k q_{v,\infty}^t + \sum_{h=1}^k \left(p(1-p)^{h-1} q_{v,L_v^t(n_v-k+h)}^t \right) - q_{v,j}^t \right) \\ &= \sum_{v \in V'} \frac{1}{mq_{v,\infty}^t} \left((1-p)^{n_v+1} q_{v,\infty}^t + \sum_{i=1}^{n_v} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^t \right) - p(1-p)^{n_v-k} \sum_{h=1}^k \left(p(1-p)^{h-1} q_{v,L_v^t(n_v-k+h)}^t \right) + p(1-p)^{n_v-k} q_{v,j}^t \right) \end{aligned}$$

The terms with $q_{v,\infty}^t$ and $q_{v,j}^t$ are strictly positive and can be left out for simplicity.

$$\geq \sum_{v \in V'} \frac{1}{mq_{v,\infty}^t} \left(\sum_{i=1}^{n_v} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^t \right) - p(1-p)^{n_v-k} \sum_{h=1}^k \left(p(1-p)^{h-1} q_{v,L_v^t(n_v-k+h)}^t \right) \right)$$

The first summation within the brackets from 1 to n_v is now split into a part until $n_v - k$ and one from $n_v - k$ to n_v . Afterwards, one of these parts can be combined with the last summation.

$$\begin{aligned} &= \sum_{v \in V'} \frac{1}{mq_{v,\infty}^t} \left(\sum_{i=1}^{n_v-k} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^t \right) + p(1-p)^{n_v-k} \sum_{h=1}^k \left((1-p)^{h-1} q_{v,L_v^t(n_v-k+h)}^t \right) - p(1-p)^{n_v-k} \sum_{h=1}^k \left(p(1-p)^{h-1} q_{v,L_v^t(n_v-k+h)}^t \right) \right) \\ &= \sum_{v \in V'} \frac{1}{mq_{v,\infty}^t} \left(\sum_{i=1}^{n_v-k} \left(p(1-p)^{i-1} q_{v,L_v^t(i)}^t \right) + p(1-p)^{n_v-k+1} \sum_{h=1}^k \left((1-p)^{h-1} q_{v,L_v^t(n_v-k+h)}^t \right) \right) \geq 0 \end{aligned}$$

The last equation holds because it is a summation of strictly positive terms. This concludes the proof of Proposition 2 to show that the objective value for time to detection can never grow larger than one. Again, the same steps can be taken to prove this result for the other objectives. \square

C Results

C.1 All Results Comparing the Different Heuristics

Table C.1: All individual results, i.e. the objective values, with the standard Bangalore Network for the 100 considered cases and all four objectives. A solution in green is the best-found solution.

| Greedy | 0,90 | 0,91 | 0,92 | 0,93 | 0,94 | 0,95 | 0,96 | 0,97 | 0,98 | 0,99 |
|-------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 5 | 0,78882 | 0,79223 | 0,79557 | 0,79884 | 0,80204 | 0,80519 | 0,80868 | 0,81263 | 0,81653 | 0,82037 |
| 6 | 0,80007 | 0,80234 | 0,80455 | 0,80669 | 0,80932 | 0,81195 | 0,81477 | 0,81778 | 0,82073 | 0,82391 |
| 7 | 0,80912 | 0,81058 | 0,81196 | 0,81325 | 0,81496 | 0,81670 | 0,81870 | 0,82117 | 0,82364 | 0,82646 |
| 8 | 0,81497 | 0,81588 | 0,81670 | 0,81754 | 0,81888 | 0,81998 | 0,82127 | 0,82311 | 0,82574 | 0,82819 |
| 9 | 0,81951 | 0,82033 | 0,82106 | 0,82172 | 0,82230 | 0,82289 | 0,82370 | 0,82490 | 0,82697 | 0,82936 |
| 10 | 0,82276 | 0,82348 | 0,82414 | 0,82473 | 0,82525 | 0,82574 | 0,82602 | 0,82663 | 0,82811 | 0,82997 |
| 11 | 0,82514 | 0,82583 | 0,82645 | 0,82701 | 0,82751 | 0,82797 | 0,82772 | 0,82824 | 0,82925 | 0,83056 |
| 12 | 0,82685 | 0,82748 | 0,82804 | 0,82856 | 0,82901 | 0,83004 | 0,82895 | 0,82948 | 0,83036 | 0,83113 |
| 13 | 0,82804 | 0,82856 | 0,82904 | 0,82947 | 0,82985 | 0,83071 | 0,83009 | 0,83059 | 0,83097 | 0,83170 |
| 14 | 0,82872 | 0,82923 | 0,82969 | 0,83010 | 0,83047 | 0,83092 | 0,83121 | 0,83147 | 0,83157 | 0,83185 |
| Greedy + LS | | | | | | | | | | |
| 5 | 0,78882 | 0,79223 | 0,79557 | 0,79884 | 0,80204 | 0,80519 | 0,80868 | 0,81263 | 0,81653 | 0,82037 |
| 6 | 0,80007 | 0,80234 | 0,80455 | 0,80669 | 0,80932 | 0,81195 | 0,81477 | 0,81778 | 0,82073 | 0,82391 |
| 7 | 0,80912 | 0,81058 | 0,81196 | 0,81325 | 0,81496 | 0,81670 | 0,81870 | 0,82117 | 0,82364 | 0,82646 |
| 8 | 0,81497 | 0,81588 | 0,81670 | 0,81773 | 0,81888 | 0,81998 | 0,82139 | 0,82311 | 0,82574 | 0,82868 |
| 9 | 0,81951 | 0,82033 | 0,82106 | 0,82172 | 0,82230 | 0,82289 | 0,82370 | 0,82490 | 0,82697 | 0,82966 |
| 10 | 0,82276 | 0,82348 | 0,82414 | 0,82473 | 0,82525 | 0,82574 | 0,82642 | 0,82708 | 0,82864 | 0,83028 |
| 11 | 0,82514 | 0,82583 | 0,82645 | 0,82701 | 0,82751 | 0,82797 | 0,82861 | 0,82920 | 0,82978 | 0,83086 |
| 12 | 0,82745 | 0,82809 | 0,82866 | 0,82918 | 0,82963 | 0,83004 | 0,83038 | 0,83024 | 0,83092 | 0,83144 |
| 13 | 0,82836 | 0,82895 | 0,82949 | 0,82997 | 0,83040 | 0,83078 | 0,83112 | 0,83140 | 0,83163 | 0,83183 |
| 14 | 0,82894 | 0,82942 | 0,82986 | 0,83026 | 0,83065 | 0,83099 | 0,83128 | 0,83152 | 0,83172 | 0,83187 |
| NonDom | | | | | | | | | | |
| 5 | 0,78882 | 0,79223 | 0,79557 | 0,79884 | 0,80204 | 0,80519 | 0,80868 | 0,81263 | 0,81653 | 0,82037 |
| 6 | 0,80007 | 0,80234 | 0,80455 | 0,80669 | 0,80932 | 0,81195 | 0,81477 | 0,81778 | 0,82073 | 0,82391 |
| 7 | 0,80912 | 0,81058 | 0,81196 | 0,81325 | 0,81496 | 0,81670 | 0,81870 | 0,82117 | 0,82364 | 0,82646 |
| 8 | 0,81497 | 0,81588 | 0,81670 | 0,81773 | 0,81888 | 0,81998 | 0,82151 | 0,82348 | 0,82574 | 0,82819 |
| 9 | 0,81951 | 0,82033 | 0,82106 | 0,82172 | 0,82230 | 0,82289 | 0,82411 | 0,82566 | 0,82741 | 0,82936 |
| 10 | 0,82276 | 0,82348 | 0,82414 | 0,82473 | 0,82525 | 0,82574 | 0,82642 | 0,82750 | 0,82864 | 0,83007 |
| 11 | 0,82514 | 0,82583 | 0,82645 | 0,82701 | 0,82751 | 0,82797 | 0,82861 | 0,82920 | 0,82978 | 0,83069 |
| 12 | 0,82745 | 0,82809 | 0,82866 | 0,82918 | 0,82963 | 0,83004 | 0,83038 | 0,83068 | 0,83092 | 0,83126 |
| 13 | 0,82836 | 0,82895 | 0,82949 | 0,82997 | 0,83040 | 0,83078 | 0,83112 | 0,83140 | 0,83163 | 0,83183 |
| 14 | 0,82894 | 0,82942 | 0,82986 | 0,83026 | 0,83065 | 0,83099 | 0,83128 | 0,83152 | 0,83172 | 0,83187 |
| NonDom + LS | | | | | | | | | | |
| 5 | 0,78882 | 0,79223 | 0,79557 | 0,79884 | 0,80204 | 0,80519 | 0,80868 | 0,81263 | 0,81653 | 0,82037 |
| 6 | 0,80007 | 0,80234 | 0,80455 | 0,80669 | 0,80932 | 0,81195 | 0,81477 | 0,81778 | 0,82073 | 0,82391 |
| 7 | 0,80912 | 0,81058 | 0,81196 | 0,81325 | 0,81496 | 0,81670 | 0,81870 | 0,82117 | 0,82364 | 0,82646 |
| 8 | 0,81497 | 0,81588 | 0,81670 | 0,81773 | 0,81888 | 0,81998 | 0,82151 | 0,82356 | 0,82603 | 0,82868 |
| 9 | 0,81951 | 0,82033 | 0,82106 | 0,82172 | 0,82230 | 0,82289 | 0,82411 | 0,82566 | 0,82741 | 0,82966 |
| 10 | 0,82276 | 0,82348 | 0,82414 | 0,82473 | 0,82525 | 0,82574 | 0,82642 | 0,82750 | 0,82864 | 0,83028 |
| 11 | 0,82514 | 0,82583 | 0,82645 | 0,82701 | 0,82751 | 0,82797 | 0,82861 | 0,82920 | 0,82978 | 0,83086 |
| 12 | 0,82745 | 0,82809 | 0,82866 | 0,82918 | 0,82963 | 0,83004 | 0,83038 | 0,83068 | 0,83092 | 0,83144 |
| 13 | 0,82836 | 0,82895 | 0,82949 | 0,82997 | 0,83040 | 0,83078 | 0,83112 | 0,83140 | 0,83163 | 0,83183 |
| 14 | 0,82894 | 0,82942 | 0,82986 | 0,83026 | 0,83065 | 0,83099 | 0,83128 | 0,83152 | 0,83172 | 0,83187 |

Table C.2: All individual results, i.e. the objective values, with the Bangalore Network with 41 vulnerable nodes for the 100 considered cases and all four objectives. A solution in green is the best-found solution.

| Greedy | 0,90 | 0,91 | 0,92 | 0,93 | 0,94 | 0,95 | 0,96 | 0,97 | 0,98 | 0,99 |
|-------------|---|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 5 | 0,58042 | 0,58514 | 0,58983 | 0,59449 | 0,59913 | 0,60375 | 0,60834 | 0,61291 | 0,61745 | 0,62197 |
| 6 | 0,60730 | 0,61205 | 0,61678 | 0,62148 | 0,62615 | 0,63080 | 0,63541 | 0,64000 | 0,64455 | 0,64908 |
| 7 | 0,62620 | 0,63083 | 0,63542 | 0,63998 | 0,64450 | 0,64898 | 0,65344 | 0,65786 | 0,66224 | 0,66659 |
| 8 | 0,64364 | 0,64801 | 0,65236 | 0,65666 | 0,66093 | 0,66517 | 0,66937 | 0,67383 | 0,67825 | 0,68263 |
| 9 | 0,65937 | 0,66378 | 0,66816 | 0,67250 | 0,67680 | 0,68107 | 0,68530 | 0,68950 | 0,69367 | 0,69792 |
| 10 | 0,67416 | 0,67858 | 0,68298 | 0,68736 | 0,69171 | 0,69603 | 0,70033 | 0,70461 | 0,70887 | 0,71310 |
| 11 | 0,68771 | 0,69212 | 0,69651 | 0,70087 | 0,70520 | 0,70951 | 0,71379 | 0,71805 | 0,72229 | 0,72650 |
| 12 | 0,70111 | 0,70552 | 0,70990 | 0,71425 | 0,71857 | 0,72287 | 0,72713 | 0,73137 | 0,73558 | 0,73977 |
| 13 | 0,71276 | 0,71666 | 0,72116 | 0,72562 | 0,73006 | 0,73447 | 0,73885 | 0,74321 | 0,74754 | 0,75184 |
| 14 | 0,72379 | 0,72724 | 0,73064 | 0,73399 | 0,73730 | 0,74055 | 0,74379 | 0,74777 | 0,75172 | 0,75564 |
| Greedy + LS | All Solutions the same | | | | | | | | | |
| NonDom | All Solutions the same | | | | | | | | | |
| NonDom + LS | All Solutions the same (evaluated until $B = 8$) | | | | | | | | | |

Table C.3: All individual results, i.e. the objective values, with the Richmond Network for the 100 considered cases and all four objectives. A solution in green is the best-found solution.

| Greedy | 0,90 | 0,91 | 0,92 | 0,93 | 0,94 | 0,95 | 0,96 | 0,97 | 0,98 | 0,99 |
|-------------|---|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 5 | 0,72444 | 0,73249 | 0,74054 | 0,74859 | 0,75664 | 0,76469 | 0,77274 | 0,78079 | 0,78884 | 0,79689 |
| 6 | 0,82136 | 0,83049 | 0,83961 | 0,84874 | 0,85786 | 0,86699 | 0,87612 | 0,88524 | 0,89437 | 0,90350 |
| 7 | 0,87618 | 0,88506 | 0,89393 | 0,90277 | 0,91160 | 0,92041 | 0,92919 | 0,93796 | 0,94672 | 0,95545 |
| 8 | 0,89509 | 0,90227 | 0,90939 | 0,91645 | 0,92345 | 0,93039 | 0,93726 | 0,94408 | 0,95083 | 0,95753 |
| 9 | 0,91320 | 0,91875 | 0,92420 | 0,92955 | 0,93480 | 0,93994 | 0,94499 | 0,94993 | 0,95478 | 0,95952 |
| 10 | 0,92361 | 0,92823 | 0,93272 | 0,93708 | 0,94132 | 0,94544 | 0,94943 | 0,95330 | 0,95704 | 0,96067 |
| 11 | 0,93363 | 0,93735 | 0,94091 | 0,94433 | 0,94760 | 0,95073 | 0,95371 | 0,95654 | 0,95923 | 0,96177 |
| 12 | 0,94332 | 0,94616 | 0,94883 | 0,95134 | 0,95367 | 0,95584 | 0,95784 | 0,95967 | 0,96134 | 0,96283 |
| 13 | 0,95065 | 0,95276 | 0,95469 | 0,95646 | 0,95806 | 0,95949 | 0,96075 | 0,96185 | 0,96279 | 0,96356 |
| 14 | 0,95538 | 0,95705 | 0,95853 | 0,95985 | 0,96099 | 0,96195 | 0,96274 | 0,96336 | 0,96380 | 0,96407 |
| Greedy + LS | All Solutions the same | | | | | | | | | |
| NonDom | All Solutions the same | | | | | | | | | |
| NonDom + LS | All Solutions the same (evaluated until $B = 7$) | | | | | | | | | |

Table C.4: All individual results, i.e. the objective values, with the standard Bangalore Network for the 100 considered cases without the objective of source identification. A solution in green is the best-found solution.

| Greedy | 0,90 | 0,91 | 0,92 | 0,93 | 0,94 | 0,95 | 0,96 | 0,97 | 0,98 | 0,99 |
|-------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 5 | 0,95246 | 0,95577 | 0,95899 | 0,96213 | 0,96526 | 0,96920 | 0,97306 | 0,97684 | 0,98056 | 0,98420 |
| 6 | 0,96713 | 0,96898 | 0,97074 | 0,97243 | 0,97466 | 0,97715 | 0,98023 | 0,98321 | 0,98610 | 0,98891 |
| 7 | 0,97617 | 0,97718 | 0,97810 | 0,97892 | 0,98020 | 0,98149 | 0,98372 | 0,98637 | 0,98920 | 0,99195 |
| 8 | 0,98190 | 0,98284 | 0,98370 | 0,98448 | 0,98518 | 0,98559 | 0,98696 | 0,98879 | 0,99127 | 0,99389 |
| 9 | 0,98618 | 0,98702 | 0,98779 | 0,98848 | 0,98911 | 0,98968 | 0,99018 | 0,99117 | 0,99290 | 0,99471 |
| 10 | 0,98906 | 0,98988 | 0,99066 | 0,99136 | 0,99200 | 0,99258 | 0,99310 | 0,99355 | 0,99446 | 0,99553 |
| 11 | 0,99188 | 0,99261 | 0,99327 | 0,99387 | 0,99441 | 0,99489 | 0,99531 | 0,99567 | 0,99598 | 0,99631 |
| 12 | 0,99320 | 0,99384 | 0,99443 | 0,99496 | 0,99543 | 0,99585 | 0,99622 | 0,99653 | 0,99681 | 0,99707 |
| 13 | 0,99404 | 0,99468 | 0,99526 | 0,99578 | 0,99625 | 0,99666 | 0,99702 | 0,99733 | 0,99760 | 0,99783 |
| 14 | 0,99478 | 0,99528 | 0,99574 | 0,99617 | 0,99658 | 0,99693 | 0,99725 | 0,99754 | 0,99781 | 0,99802 |
| Greedy + LS | | | | | | | | | | |
| 5 | 0,95246 | 0,95577 | 0,95899 | 0,96213 | 0,96526 | 0,96920 | 0,97306 | 0,97684 | 0,98056 | 0,98420 |
| 6 | 0,96713 | 0,96898 | 0,97074 | 0,97243 | 0,97466 | 0,97715 | 0,98023 | 0,98321 | 0,98610 | 0,98891 |
| 7 | 0,97617 | 0,97718 | 0,97810 | 0,97892 | 0,98020 | 0,98170 | 0,98372 | 0,98637 | 0,98920 | 0,99195 |
| 8 | 0,98190 | 0,98284 | 0,98370 | 0,98448 | 0,98518 | 0,98579 | 0,98696 | 0,98879 | 0,99189 | 0,99454 |
| 9 | 0,98618 | 0,98702 | 0,98779 | 0,98848 | 0,98911 | 0,98968 | 0,99018 | 0,99117 | 0,99346 | 0,99536 |
| 10 | 0,98906 | 0,98988 | 0,99066 | 0,99136 | 0,99200 | 0,99258 | 0,99310 | 0,99355 | 0,99502 | 0,99631 |
| 11 | 0,99188 | 0,99261 | 0,99327 | 0,99391 | 0,99454 | 0,99511 | 0,99563 | 0,99609 | 0,99652 | 0,99708 |
| 12 | 0,99320 | 0,99384 | 0,99443 | 0,99496 | 0,99543 | 0,99602 | 0,99655 | 0,99702 | 0,99746 | 0,99785 |
| 13 | 0,99406 | 0,99471 | 0,99531 | 0,99585 | 0,99633 | 0,99676 | 0,99713 | 0,99746 | 0,99775 | 0,99799 |
| 14 | 0,99480 | 0,99532 | 0,99579 | 0,99624 | 0,99666 | 0,99703 | 0,99735 | 0,99762 | 0,99786 | 0,99805 |
| NonDom | | | | | | | | | | |
| 5 | 0,95246 | 0,95577 | 0,95899 | 0,96213 | 0,96526 | 0,96920 | 0,97306 | 0,97684 | 0,98056 | 0,98420 |
| 6 | 0,96713 | 0,96898 | 0,97074 | 0,97243 | 0,97466 | 0,97715 | 0,98023 | 0,98321 | 0,98610 | 0,98891 |
| 7 | 0,97617 | 0,97718 | 0,97810 | 0,97892 | 0,98020 | 0,98228 | 0,98450 | 0,98665 | 0,98920 | 0,99195 |
| 8 | 0,98190 | 0,98284 | 0,98370 | 0,98448 | 0,98518 | 0,98662 | 0,98799 | 0,98980 | 0,99183 | 0,99389 |
| 9 | 0,98716 | 0,98801 | 0,98878 | 0,98949 | 0,99013 | 0,99070 | 0,99122 | 0,99222 | 0,99346 | 0,99527 |
| 10 | 0,99002 | 0,99087 | 0,99165 | 0,99237 | 0,99302 | 0,99361 | 0,99413 | 0,99460 | 0,99502 | 0,99609 |
| 11 | 0,99162 | 0,99245 | 0,99327 | 0,99391 | 0,99454 | 0,99511 | 0,99563 | 0,99609 | 0,99649 | 0,99687 |
| 12 | 0,99320 | 0,99384 | 0,99443 | 0,99496 | 0,99543 | 0,99592 | 0,99643 | 0,99689 | 0,99728 | 0,99763 |
| 13 | 0,99404 | 0,99468 | 0,99526 | 0,99578 | 0,99625 | 0,99666 | 0,99702 | 0,99733 | 0,99760 | 0,99783 |
| 14 | 0,99478 | 0,99528 | 0,99574 | 0,99617 | 0,99658 | 0,99693 | 0,99725 | 0,99754 | 0,99781 | 0,99802 |
| NonDom + LS | | | | | | | | | | |
| 5 | 0,95246 | 0,95577 | 0,95899 | 0,96213 | 0,96563 | 0,96941 | 0,97310 | 0,97684 | 0,98056 | 0,98420 |
| 6 | 0,96713 | 0,96898 | 0,97074 | 0,97243 | 0,97499 | 0,97817 | 0,98126 | 0,98426 | 0,98716 | 0,98998 |
| 7 | 0,97617 | 0,97718 | 0,97810 | 0,97892 | 0,98020 | 0,98252 | 0,98476 | 0,98742 | 0,99026 | 0,99302 |
| 8 | 0,98190 | 0,98284 | 0,98370 | 0,98448 | 0,98518 | 0,98662 | 0,98799 | 0,98984 | 0,99189 | 0,99454 |
| 9 | 0,98716 | 0,98801 | 0,98878 | 0,98949 | 0,99013 | 0,99070 | 0,99122 | 0,99222 | 0,99346 | 0,99549 |
| 10 | 0,99002 | 0,99087 | 0,99165 | 0,99237 | 0,99302 | 0,99361 | 0,99413 | 0,99460 | 0,99502 | 0,99631 |
| 11 | 0,99162 | 0,99245 | 0,99327 | 0,99391 | 0,99454 | 0,99511 | 0,99563 | 0,99609 | 0,99652 | 0,99708 |
| 12 | 0,99320 | 0,99384 | 0,99443 | 0,99496 | 0,99544 | 0,99602 | 0,99655 | 0,99702 | 0,99746 | 0,99785 |
| 13 | 0,99406 | 0,99471 | 0,99531 | 0,99585 | 0,99633 | 0,99676 | 0,99713 | 0,99746 | 0,99775 | 0,99799 |
| 14 | 0,99480 | 0,99532 | 0,99579 | 0,99624 | 0,99666 | 0,99703 | 0,99735 | 0,99762 | 0,99786 | 0,99805 |

D Experiments

D.1 The Testbed

In this Appendix, we present the WaDi testbed in more detail. In Section 7.1.1, the outline of the whole testbed was shown in Figure 12 and explained in five steps. For each of these five steps, we can show a more detailed outline of that part of the testbed.

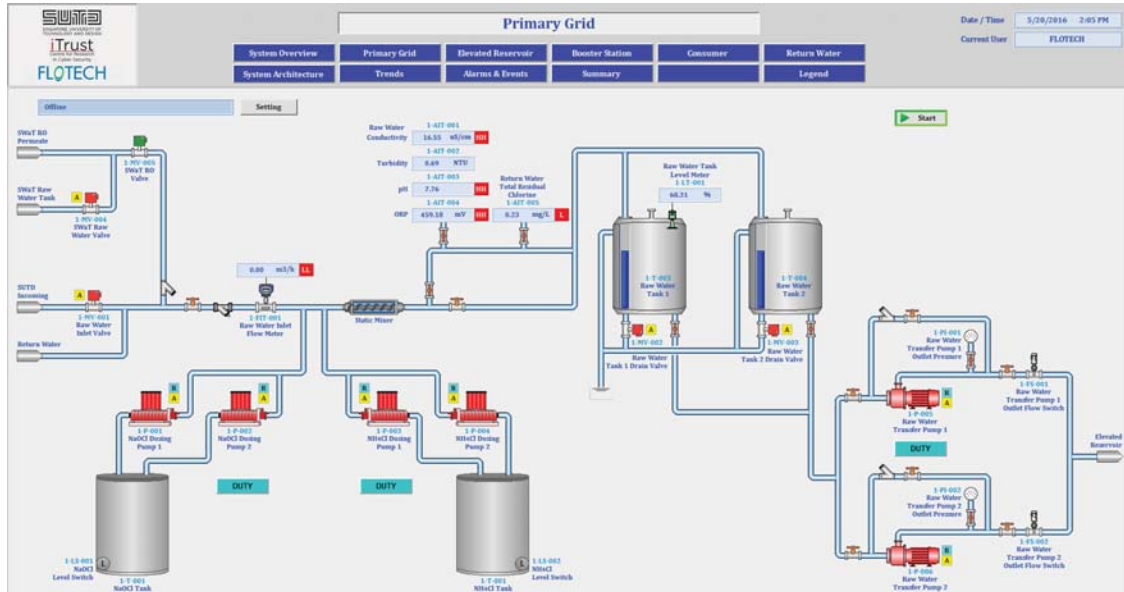


Figure D.1: An overview of the Primary Grid with the sources, raw water tanks and sensor P1.

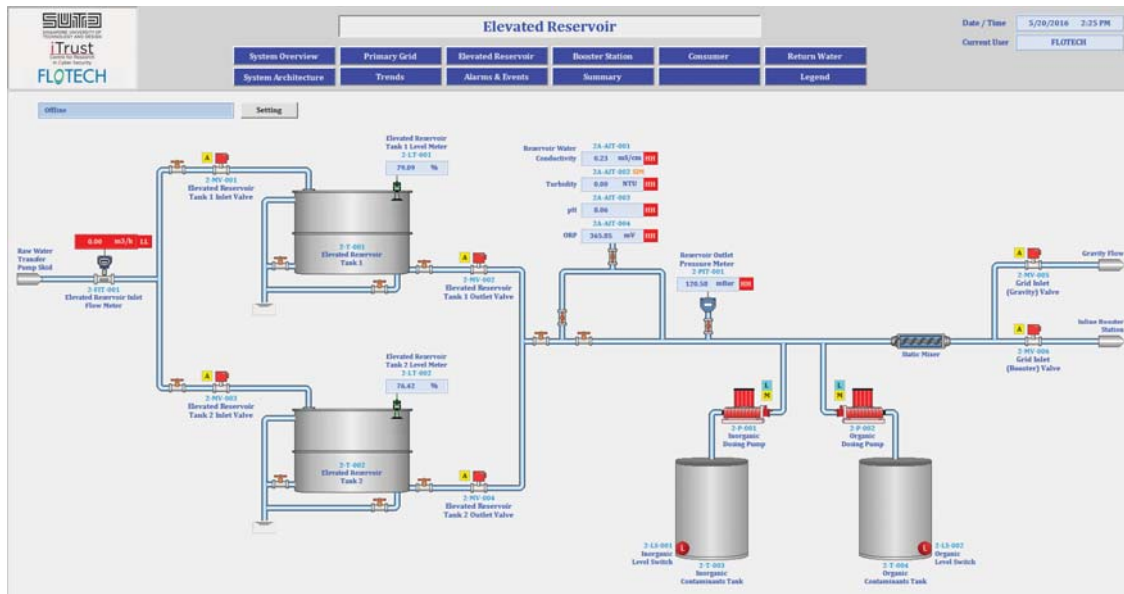


Figure D.2: An overview of the Elevated Water Tanks and P2A.

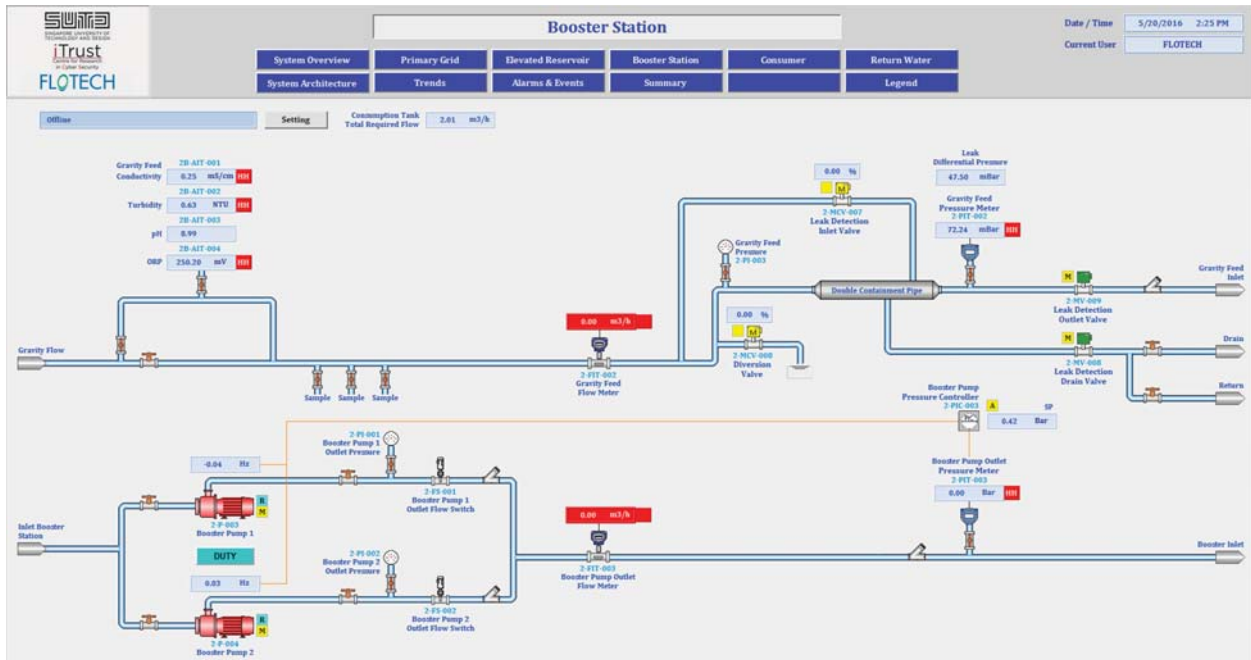


Figure D.3: An overview of the Booster Grid and Gravity Feed with sensor P2B.

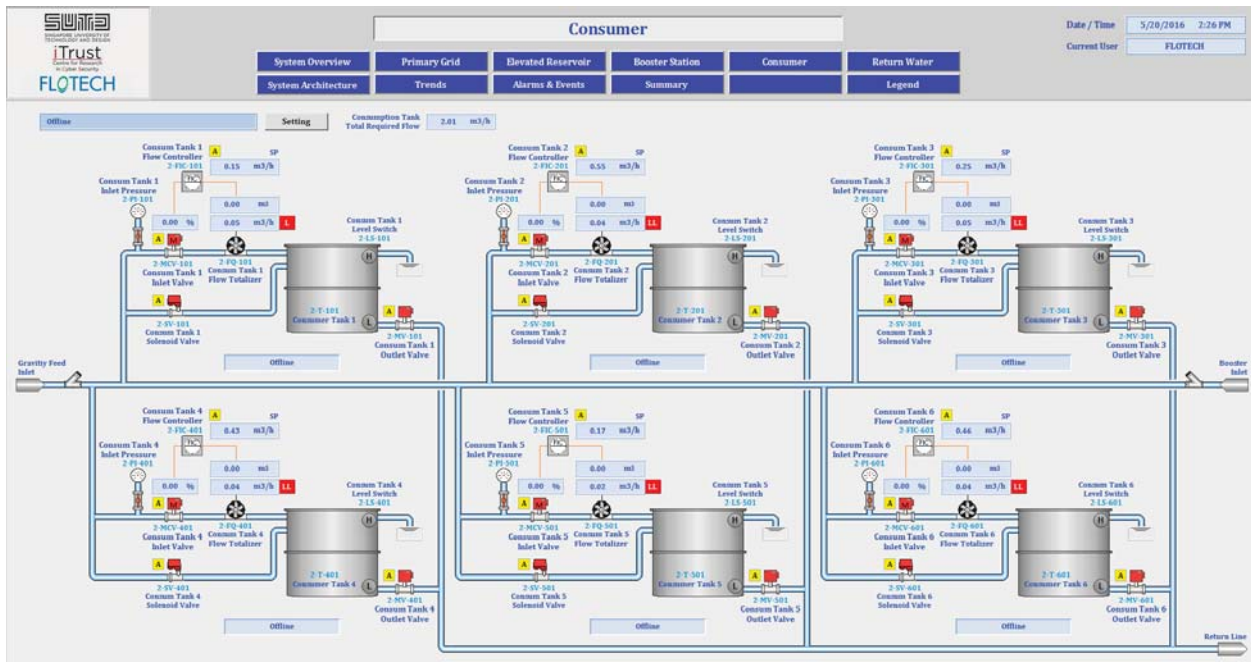


Figure D.4: An overview of the Consumer Tanks.

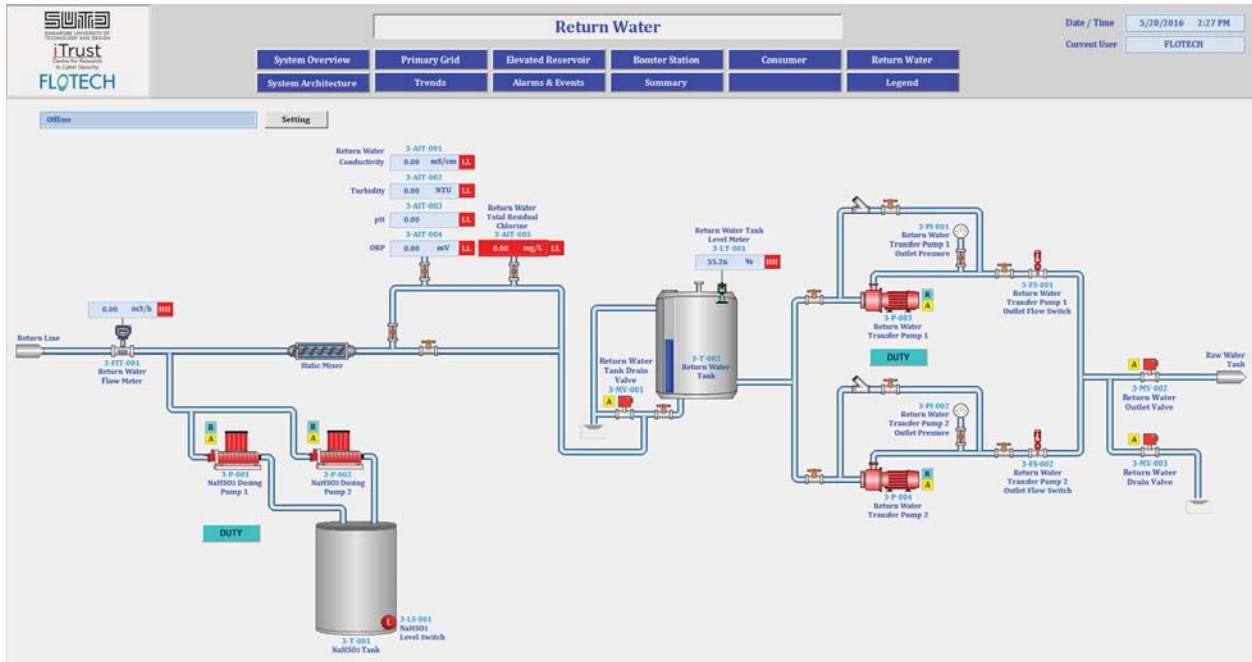


Figure D.5: An overview of the Return Water part with sensor P3.

Finally, we show some pictures of the testbed.



Figure D.6: An overview of the WaDi testbed.



Figure D.7: The contamination tanks with the dosing pumps on top.

D.2 Long Runs

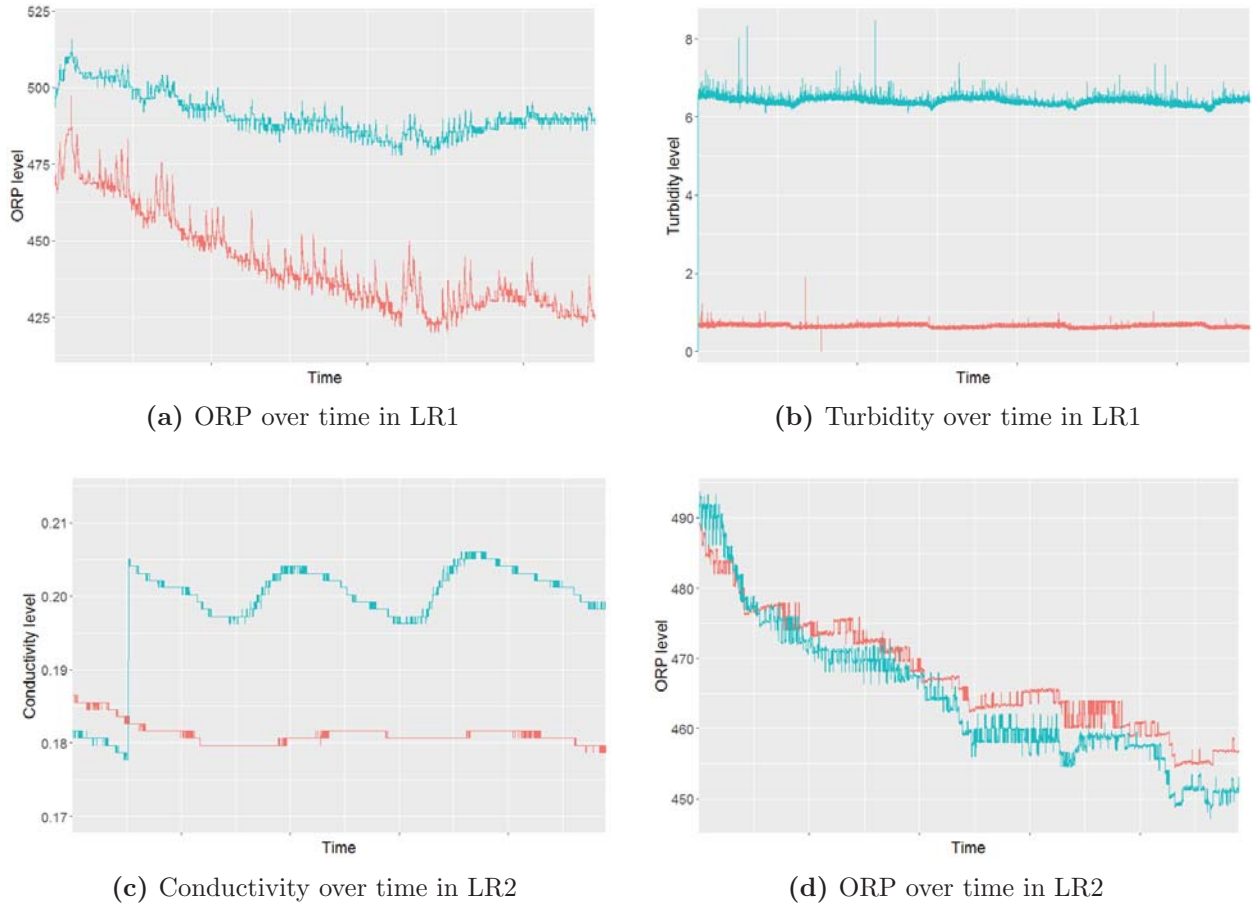


Figure D.8: The four long runs which were not shown in Section 7.2.1. Red = P2A, blue = P2B

Table D.1: Correlation levels of all sensor parameters in long runs LR1 and LR2.

| | Conductivity | | ORP | pH | Turbidity |
|-----|--------------|------------|-------|-------|-----------|
| | Before jump | After jump | | | |
| LR1 | 0.995 | 0.652 | 0.925 | 0.909 | -0.126 |
| LR2 | 0.872 | 0.486 | 0.975 | 0.953 | 0.288 |

D.3 Standard Solutions

Table D.2: The date of the last check and last calibration of all parameters and all sensors.

| | | Conductivity | ORP | pH | Turbidity |
|-----|------------------|--------------|-------------|-----------|-----------|
| P1 | Last Calibration | February 2 | August 10 | August 10 | August 10 |
| | Last Check | August 10 | August 10 | August 10 | August 10 |
| P2A | Last Calibration | February 2 | September 5 | August 10 | August 29 |
| | Last Check | August 29 | September 5 | August 29 | August 29 |
| P2B | Last Calibration | February 10 | August 29 | August 29 | August 29 |
| | Last Check | August 29 | September 5 | August 29 | August 29 |
| P3 | Last Calibration | January 30 | July 28 | July 28 | July 28 |
| | Last Check | July 28 | July 28 | July 28 | July 28 |

Table D.3: All sensor readings over time compared to the standard solution value.

| | | Conductivity | | ORP | pH | | Turbidity | |
|---------|-----|--------------|-------|-----|-------|-------|-----------|------|
| | | < 0.1 | 0.447 | 430 | 7 | 10.01 | 0.1 | 20 |
| Sept 7 | P2A | 0.034 | 0.514 | 425 | 6.05 | 9.68 | 2.7 | 37.0 |
| | P2B | 0.066 | 0.488 | 424 | 6.72 | 10.05 | 0.4 | 35.5 |
| Sept 13 | P1 | 0.059 | 0.570 | 231 | 8.69 | 12.31 | 6.0 | 9.4 |
| | P2A | 0.059 | 0.590 | 421 | 5.97 | 9.79 | 6.7 | 28.8 |
| | P2B | 0.039 | 0.565 | 422 | 6.49 | 9.95 | 0.7 | 20.6 |
| | P3 | 0.063 | 0.556 | 393 | 6.29 | 9.82 | 4.3 | 5.8 |
| Sept 19 | P1 | 0.051 | 0.549 | 220 | 9.52 | 12.71 | 3.9 | 14.0 |
| | P2A | 0.055 | 0.505 | 420 | 6.69 | 9.93 | 7.8 | 21.8 |
| | P2B | 0.025 | 0.493 | 419 | 6.56 | 10.12 | 0.3 | 19.8 |
| | P3 | 0.078 | 0.518 | 397 | 6.74 | 10.09 | 2.5 | 5.4 |
| Sept 25 | P1 | 0.064 | 0.576 | 207 | 10.14 | 12.98 | 3.4 | 12.4 |
| | P2A | 0.057 | 0.587 | 412 | 7.08 | 10.10 | 8.1 | 47.4 |
| | P2B | 0.036 | 0.540 | 414 | 6.92 | 10.31 | 0.4 | 39.2 |
| | P3 | 0.080 | 0.619 | 383 | 6.95 | 10.15 | 1.3 | 3.1 |

D.4 Adding Contaminations

The different time series for the high water flow rate when Ammonia was added were shown. In Figures D.9 and D.10, the four time series when Hypo and HCl were added are being shown.

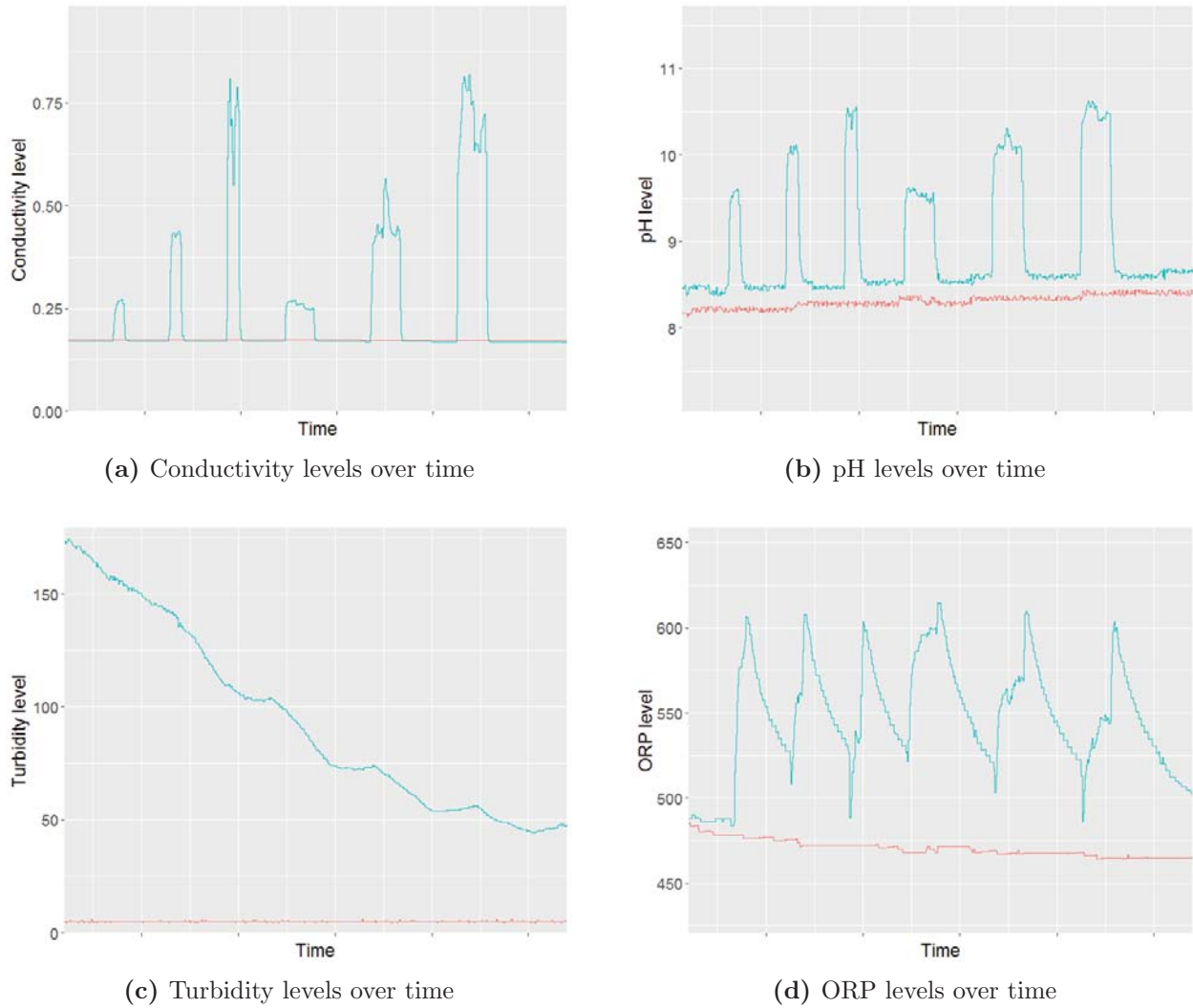
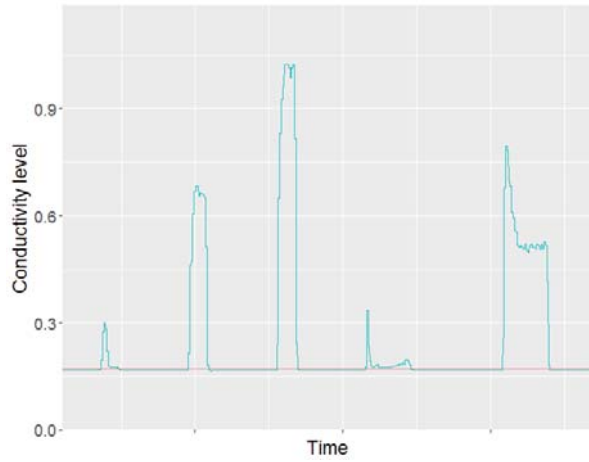
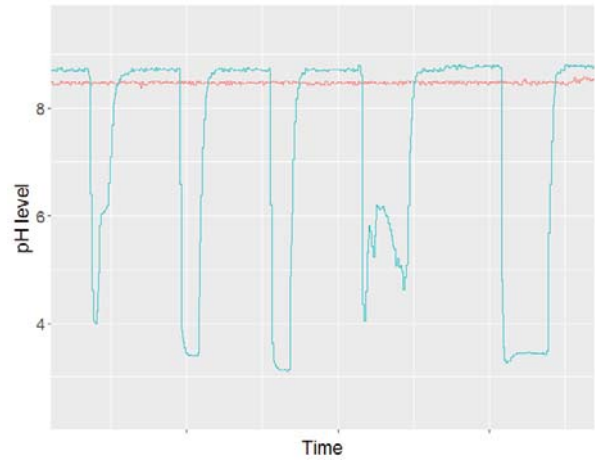


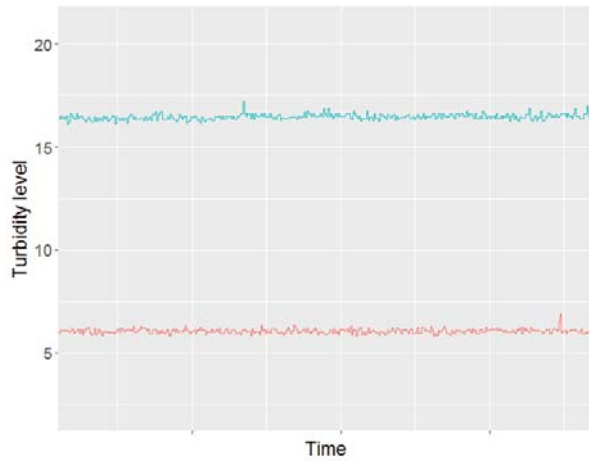
Figure D.9: Time series of the four sensor readings of P2A (red) and P2B (blue) over time when Hypo is added into the network with six settings which differed in dosing rate and dosing duration.



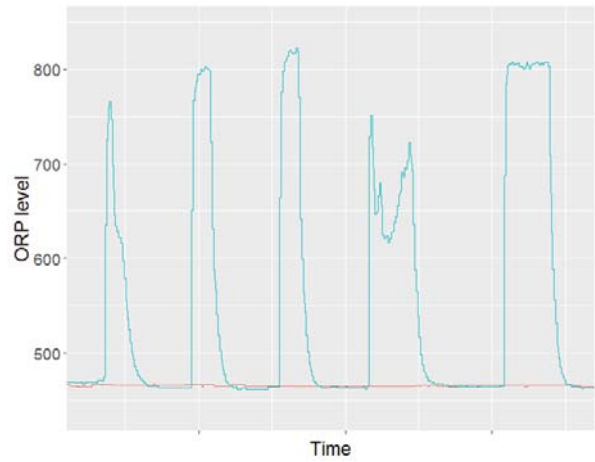
(a) Conductivity levels over time



(b) pH levels over time



(c) Turbidity levels over time



(d) ORP levels over time

Figure D.10: Time series of the four sensor readings of P2A (red) and P2B (blue) over time when HCl is added into the network with five settings which differed in dosing rate and dosing duration.