Elias Gulla Rokstad

Resilience of Water Distribution Systems Exposed to Substance Intrusions

A Topological Data Analysis

Master's thesis in Water and Wastewater Engineering
Supervisor: Christos Makropoulos
Co-supervisor: Marius Møller Rokstad
June 2021

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

Elias Gulla Rokstad

Resilience of Water Distribution Systems Exposed to Substance Intrusions

A Topological Data Analysis

Master’s thesis in Water and Wastewater Engineering
Supervisor: Christos Makropoulos
Co-supervisor: Marius Møller Rokstad
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

Summary

There is an emerging realisation in the water industry that improving resilience enhances the sustainability of water distribution systems exposed to an uncertain future with often unavoidable failures. A successful adaptation of resilience-enhancing strategies relies on the effectiveness of the water distribution system's ability to resist, absorb, and restore its functionality exposed to the effects of stress. Therefore, a deeper understanding of the underlying mechanisms promoting such abilities is of great relevance for decision-makers involved with strategic planning.

This study evaluates resilience with respect to drinking-water quality objectives. Through a simulation study, failure is quantified based on substance intrusion and the propagation of pollutants in the water distribution networks. Resilience is measured as the system's degree of performance under progressively increasing disturbance. This can be simulated through scenarios with an increasing mass of substance injected into the network. Random failure sequences are adapted to the failure mode as a multivariate stochastic modeling approach. A random selection of source point inputs addresses spatial uncertainties, while start time inputs address temporal uncertainties of the failure mode. The number of nodes used as a source point presents the level of stress inflicted on the water distribution system.

Resilience is influenced by the installed technologies and how they are connected. Yet, relatively few studies have addressed the actual relationship between resilience and topological characteristics of water distribution networks. Based on network design constraints and statistical parameters, this study presents an automatic network generation procedure to produce network variants with unique sets of topological attributes. The network variants are exposed to the aforementioned random failure sequences, and their resilience is evaluated based on 50 global failure indicators. Strong positive correlations were only observed between the rapidity of multiple failure metrics and certain topological attribute indicators. Moderate to weak correlations were observed between the other global failure metrics and some statistical properties of the topological layouts. The results indicate that there is a relationship between network design and the resilience of the generated network variants.

Keywords: Resilience, WDS, Quality, Topological Attributes, Network Generation

Sammendrag

Motstandsdyktighet er et begrep av voksende betydning i vann- og avløpssektoren. For en sektor på vei inn i en usikker fremtid preget av ofte uunngåelige svikter vil tiltak for økende motstandsdyktighet være ett viktig redskap for sikring av bærekraftige drikkevannssystem. Det vil være helt essensielt å fremme graden av drikkevannssystemene sin evne til å motstå, absorbere, og gjenopprette sin funksjonalitet selv utsatt for store mengder stress. Av den grunn, vil en dypere forståelse av de underliggende mekanismene som fremmer slike evner være av stor verdi for beslutningstakere involvert med strategisk planlegging av drikkevannssystemer.

Denne masteroppgaven undersøker motstandsdyktigheten av drikkevannssystemer utsatt for forringelse av drikkevannskvalitet. Dette er gjennomført ved å simulere inntrengende giftstoff og dens utbredelse i modeller av drikkevannsystemer. Ettersom motstandsdyktighet er ett mål på drikkevannssystemet sin evne til å opprettholde sin funksjonalitet utsatt for stadig økende stress, kan det fremstilles gjennom en rekke simuleringer med økende mengde giftstoff tilført nettverket. Metoden for «random failure sequences» er tilpasset feil mekanismen som en multivariat stokastisk modellerings metode. Variabler for start tid er tilfeldig valgt for å betrakte tidsavhengige usikkerheter av feil mekanismen, mens giftstoffkilde er tilfeldig valgt for å betrakte romlige usikkerheter. Antall giftstoffkilder brukt i hver iterasjon er ekvivalent med mengde giftstoff tilført nettverket.

Motstandsdyktigheten er påvirket av de installerte teknologiene og hvordan disse teknologiene er koblet opp mot hverandre. Til tross for dette, relativt få studier har sett på det faktiske forholdet mellom motstandsdyktighet og topologiske trekk av utformingen av drikkevannsnettverk. Som en del av denne masteroppgaven, basert på statistiske antagelser og designprinsipper, er en automatisk nettverk generator utviklet. De produserte nettverksvariantene er av unik topologi målt med statistiske indikatorer fra grafiteori. Videre er nettverksvariantene utsatt for «random failure sequences» og motstandsdyktighet er målt med «global failure indicators». En sterk positive korrelasjon er observert mellom hastigheten av giftstoffene sin utbredelse og noen topologiske indikatorer. Andre aspekter av nettverkene sin funksjonalitet viser en svak til moderat korrelasjon til flere topologiske indikatorer. Resultatene gir en indikasjon på at det finnes en sammenheng mellom den topologiske utforming og motstandsdyktighet av de genererte nettverkene.

Preface

This master thesis is conducted at the Department of Civil and Transport Engineering at the Norwegian University of Science and Technology (NTNU) and delivered in June 2021.

During this research project, I have learned a lot about the complexity of resilience and water distribution systems. Besides this, I also gained knowledge from computational science, complex network theory, big data analysis, and practice from writing this thesis.

However, without the help of other people, I would probably not accomplish this thesis. Therefore I would like to thank everyone who supported me in conducting this research. My special thanks go out to my two supervisors, Christos Makropoulos, Professor at NTNU, for giving me advice and letting me have the opportunity to do my research project. Marius Møller Rokstad, Associated Professor at NTNU, for suggestions and support for carrying out this project. I would also like to express my sincere gratitude to Georgios Moraitis, Ph. D. Candidate at the National Technical University of Athens, for countless advice and support on the data analysis. Gema Sakti Raspati, Research Scientist at SINTEF, pointed out the research directions and candid feedback. Also, I would like to thank Trondheim Bydrift for making the water network model available for this research.

Further on, I would like to thank all my friends, which have helped me in motivating me to finish this thesis. Last but not least, I would like to thank my family, who always supported me in my decisions and are always willing to help me.

I hereby declare that I am the sole author of this thesis, and there are no external or additional resources that have not been referenced within the document.

[illegible]
....

Signature of the author

Date: 07.06.2021

Contents

Summary . . . . **iii**
Sammendrag . . . . **v**
Preface . . . . **vii**
Contents . . . . **ix**
Figures . . . . **xiii**
Tables . . . . **xv**
Code Listings . . . . **xvii**
1 Introduction **1**
1.1 Research question 3
1.2 Outline of report 4
2 Background **5**
2.1 Modelling water quality 5
2.1.1 Input file 6
2.1.2 Hydraulic solver 6
2.1.3 Quality solver 7
2.2 Quantifying failure 7
2.3 Resilience as a measure of performance 9
3 Methods and tools **11**
3.1 Key Performance Indicators 11
3.1.1 Service levels 11
3.1.2 Failure dimension 12
3.1.3 Failure metrics 13
3.1.4 Global failure metrics 17
3.2 Topological attribute assessment 17

3.2.1 Complex network theory 17
3.2.2 Topological attributes 18
3.3 Correlation as a statistical method 20
3.3.1 Pearson correlation coefficient 21
3.3.2 Scatter plots 21
3.3.3 P-values 22
4 Data acquisition and processing 23
4.1 Benchmark models 23
4.1.1 Ctown configuration 24
4.2 Service areas 24
4.3 Generating network variants 26
4.3.1 Link classification 27
4.3.2 Link removal 28
4.3.3 Link addition 28
4.3.4 Model calibration 29
5 Input uncertainty analysis 31
5.1 Sensitivity of control variables 31
5.1.1 Start time of a substance intrusion 32
5.1.2 Mass flow of a substance intrusion 34
5.1.3 Duration of a substance intrusion 35
5.2 Variability of source nodes 36
5.3 Discussion 39
6 Convergence of random failure sequences 41
6.1 Random failure sequences 41
6.2 Control variables 42
6.3 Maximum number of stress-testing events 43
6.4 Minimum number of stress-testing events 44
6.5 Reliability of global failure indicators 46
6.6 Discussion 47
7 Resilience of topological attributes 49
7.1 Topological attributes and global failure of benchmark models . . . 49
7.1.1 Topological attributes 49

7.1.2 Global failure indicators 50
7.2 Topological attributes and global failure of network variants 51
7.2.1 Topological attributes 51
7.2.2 Global failure indicators 53
7.3 Correlation analysis of network variants 56
7.3.1 Ctown variants 56
7.3.2 Net3 variants 65
7.4 Resilience assessment of network variants 65
7.4.1 Ranking network variants 65
7.4.2 Fuzzy C-Means Clustering 66
7.5 Discussion 68
8 Conclusion and Future Work 71
8.1 Conclusion 71
8.2 Future Work 73
Bibliography 75
A Critical quality threshold to topological attribute indicators 81
B Ranking of network variants 91
C Computer and code listings 99
C.1 Computer and Python packages 99
C.2 Listings 100

Figures

2.1 Modelling substance intrusion simulation procedure. 5

2.2 Generic loss function as presented by Moraitis *et al.* [56]. 8

3.1 The two main types of network design structures [19] 18

4.1 Ctown nodes and links grouped by service areas. 25

4.2 Visual representation of the network generation procedure. 27

5.1 Demand multipliers of Ctown as a function of time. 33

5.2 Ctown magnitude of failure as a function of start time input. 33

5.3 Ctown magnitude of failure as a function of mass flow input. 34

5.4 Ctown magnitude of failure as a function of duration input. 35

5.5 Consequence map scaled by $PS_{L0,ratio}$ at various source points. . . . 37

5.6 Consequence map scaled by $N_{L0,n,ratio}$ at various source points. . . 37

5.7 Consequence map scaled by $C_{L0,n,ratio}$ at various source points. . . 38

5.8 Consequence map scaled by $T_{L0,t,ratio}$ at various source points. . . . 38

6.1 Random failure sequence based on illustrations of Diao *et al.* [35]. 42

6.2 Global failure indicator derived from random failure sequences rs_x. 45

6.3 Variance of $PS_{L0,ratio}$ derived from random failure sequences rs_x. . 45

7.1 Distribution of topological attributes in network variants. 52

7.2 Distribution of global performance indicators in Ctown variants. . . 54

7.3 Distribution of global performance indicators in Net3 variants. . . . 55

7.4 Ctown variants global magnitude of failure to topological attributes. 57

7.5 Net3 variants global magnitude of failure to topological attributes . 58

7.6 Ctown variants global crest factor of failure to topological attributes. 59
7.7 Net3 variants global crest factor of failure to topological attributes . 60
7.8 Ctown variants global severity of failure to topological attributes. . 61
7.9 Net3 variants global severity of failure to topological attributes. . . 62
7.10 Ctown variants global rapidity of failure to topological attributes. . 63
7.11 Net3 variants global rapidity of failure to topological attributes. . . 64
7.12 Clustering Ctown variants based on magnitude of failure. 66
7.13 Resilience grouping to topological attributes of Ctown variants. . . 67

A.1 Ctown variants global magnitude of failure to topological attributes. 82
A.2 Net3 variants global magnitude of failure to topological attributes . 83
A.3 Ctown variants global crest factor of failure to topological attributes. 84
A.4 Net3 variants global crest factor of failure to topological attributes . 85
A.5 Ctown variants global severity of failure to topological attributes. . 86
A.6 Net3 variants global severity of failure to topological attributes. . . 87
A.7 Ctown variants global rapidity of failure to topological attributes. . 88
A.8 Net3 variants global rapidity of failure to topological attributes. . . 89

B.1 Ctown variants ranked by ascending $PS_{L0,ratio}$ and $N_{L0,n,ratio}$. . . . 92
B.2 Ctown variants ranked by ascending $PS_{L0,peak}$ and $N_{L0,peak}$. 93
B.3 Ctown variants ranked by descending $PS_{L0,TEP}$ and $N_{L0,TEP}$. 94
B.4 Net3 variants ranked by ascending $PS_{L0,ratio}$ and $N_{L0,n,ratio}$. 95
B.5 Net3 variants ranked by ascending $PS_{L0,peak}$ and $N_{L0,peak}$. 96
B.6 Net3 variants ranked by descending $PS_{L0,TEP}$ and $N_{L0,TEP}$. 97

Tables

3.1 Quality ranges of service levels [56]. 12

4.1 Characteristics of the benchmark models. 23

5.1 One-At-a-Time (OAT) control variable ranges. 32

6.1 Convergence result of global failure metrics. 46

C.1 Summary of attached algorithms. 99

Code Listings

C.1 Code listing of config.py script. 100
C.2 Code listing of generation.py script. 110
C.3 Code listing of quality.py script. 114
C.4 Code listing of attributes.py script. 118

Chapter 1

Introduction

The water sector's critical infrastructures form complex cyber-physical systems [45]. With the expanded attack surface, these systems are exposed to cyber-physical threats like denial of service attacks, hacking, data manipulation, deliberate contamination events, and sabotage, along with vulnerabilities like corrosion, pipe deterioration, pressure changes, and leakages. Additional challenges related to climate change and urbanisation, make urban water systems increasingly vulnerable to uncertain yet often unavoidable failures [32].

A common practice in water engineering, or any engineering field, is to break down a problem into smaller parts to understand the underlying mechanisms of the individual technologies. Reductionism has proven to be a phenomenally successful approach in every branch of science. Consequently, the methodology is wildly applied for problem-solving and has become an integrated part of most conventional risk management procedures [11]. One could argue that the performance of individual technologies is acknowledged as more or less understood[48]. Nevertheless, it is far less clear how the overall urban water system performance is affected by a portfolio of technologies for a given set of future scenarios [48]. Furthermore, the increase in size and complexity of water distribution systems makes the hydraulic and quality analysis more challenging than in the past [12].

A great frontier of science today is concerned with what happens when the technologies come together to perform as a coherent system. The process of finding system properties given the properties of its parts introduces the study of complex systems.

In the last decades, the study of complex network theory has attracted the attention of researchers from different disciplines with applications such as social, cellular, and metabolic networks, power grids, transportation networks, and water distribution networks [9, 10]. Hence, complex network theory is becoming one of the most powerful and versatile tools to investigate, describe, and understand the topology of irregular and distributed structures [12]. In such systems, structure affects function [9].

At the same time, there has been an emerging realisation that building resilience is an essential component of enhancing the sustainability of water distribution systems [35]. Resilience is a relatively new term and has roots in different sciences. Although several definitions of resilience have been proposed, it is generally agreed upon that resilience is a property of the system [48]. E.g. in structural engineering resilience refers to a material's capability to withstand stress without being permanently deformed [8], while in ecology resilience is defined as the amount of disturbance an ecological system can withstand without changing its self-organised processes and structures [7]. The stress-testing oriented definition of Makropoulos *et al.* [48] addresses resilience as:

> *... the degree to which an urban water system continues to perform under progressively increasing disturbance*. Makropoulos *et al.* [48]

The degree of continued performance is measured through reliability and is defined by [48] as the system's ability to deliver its objective over a time span consistently. The objectives may differ between areas depending on local regulations, policies, and requirements. However, objectives typically refer to the water distribution system's ability to deliver water of sufficient quantity and quantity to customers [55]. It should also be noted that reliability can be evaluated as the loss of objectives [13]. However, for this study, any loss of objectives is referred to as failure.

Another important aspect of resilience is communicated through the progressively increasing disturbance. A disturbance typically refers to a stress-testing event with less than optimal conditions. As such, resilience can be expressed as the relationship between the reliability and level of stress inflicted on the water distribution system [48].

The last aspect of resilience relates to the system. Resilience is influenced by the installed technologies and how the components are connected to each other [48]. The connection between the components refers to the network design. Despite the topological attributes and resilience being two distinct concepts, they are closely related to the network design of water distribution systems.

In literature, it is generally assumed that the resilience of water distribution systems related to certain design principles of the network, yet relatively few studies have addressed the actual relationship [32]. E.g., Pizzol [33] refers to certain topological attributes as a direct measure of resilience. Similarly, Yazdani *et al.* [16] applies topological attributes in order to compare structural vulnerability to the cost of expansion strategies for an urban drainage system in a growing city of Africa.

Mugume *et al.* [32] presents a comprehensive global resilience assessment using random failure sequences in order to present a stress-strain relationship of resilience by increasing the number of pipe failures in urban drainage systems. Using the random failure sequences, Diao *et al.* [35] investigated the effect of several failure modes (pipe failure, fire fighting, and substance intrusion) on the water distribution system's resilience. As network design is an essential part of resilience, Meng *et al.* [65] conducted an extensive correlation analysis between the resilience of topological attributes of water distribution systems exposed to pipe failure scenarios. The study of Meng *et al.* [65] is based on 85 water distribution systems with different sizes and topological features, along with network variants of a single water distribution system. The results of Meng *et al.* [65] show that certain aspects of the quantity-related resilience (e.i. spatial and temporal scales of the failure profile) are, in fact, strongly correlated to topological attributes.

1.1 Research question

This study is based on the presented framework of Mugume *et al.* [32], Diao *et al.* [35], and Meng *et al.* [65] in order to address quality-related resilience of water distribution and its relation to the network design expressed through topological attributes. The study includes developing a network generation procedure to produce network variants based on a benchmark model. Its outcome is a series of network variants with unique sets of topological attributes. This study conducts a topological attribute and a resilience assessment of the produced network variants to answer the following research question:

> What is the relationship between the resilience of water distribution systems exposed to substance intrusions and the network design characterised by topological attributes?

The topological attribute assessment includes a total of five topological attribute indicators. In its essence, topological attribute indicators are statistical parameters used to characterise the various network designs of the network variants. The topological properties of the produced network variants are evaluated based on five indicators; link density, average shortest path length, clustering coefficient, central-point of dominance, and average closeness centrality.

The three-step methodology of Moraitis *et al.* [56] is used to quantify the failure related to water quality. An input uncertainty analysis evaluates the input parameters of substance intrusions related to the failure aspect of resilience. Global failure metrics are introduced as the mean failure value derived from multiple stress-testing events as a part of it. Furthermore, the random failure sequences procedure of Mugume *et al.* [32] is adapted to simulate the progressively increasing disturbance of substance intrusions as a multivariate stochastic modeling procedure. As a part of it, the number of nodes used as a source point for substance intrusion expresses the level of stress inflicted on the water distribution models. The model's behavioral outcome is processed to present the related global failure indicators of a water distribution system.

1.2 Outline of report

Chapter 2: presents relevant literature about impact prediction.

Chapter 3: presents the methodology of the study.

Chapter 4: presents data and processing method used in this study.

Chapter 5: address validity of the global failure metrics.

Chapter 6: address reliability of the global failure metrics.

Chapter 7: evaluates the resilience of topological attributes.

Chapter 8: concludes main findings.

Chapter 2

Background

2.1 Modelling water quality

The Water Network Tool for Resilience (WNTR) provides multiple capabilities to explore the resilience of water distribution systems, including modeling the effects of aging infrastructure, environmental emergencies, pipe breakage, and quality concerns [61]. Figure 2.1 presents the simulation procedure for modeling the behavior of a water distribution network exposed to substance intrusion.

As with any model, the simulation procedure requires well-defined system boundaries to study the artificial environment mathematically. The boundaries of the model describe physical and non-physical technologies and their connection to each other. Information on the system boundaries and environmental variables (e.i. based demand and demand patterns) is described within the input file. Information about the characteristics of a substance intrusion is added to the model before the simulation procedure. The widely employed EPANET hydraulic and water quality solver [28, 38] are applied as a part of WNTR to simulate the movement of drinking water and other substances within the defined boundaries of the system. The report file presents the output as a multidimensional matrix describing water quality and quantity for each time step of the simulation period.

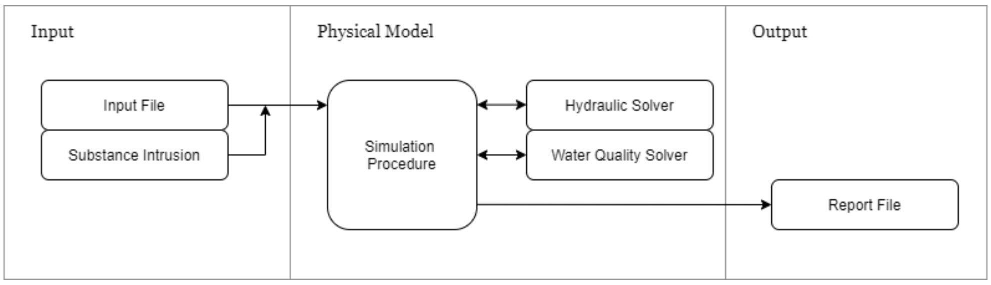


Figure 2.1: Modelling substance intrusion simulation procedure.

2.1.1 Input file

Input files describe the physical components of a model as a collection of nodes connected by links [60]. Nodes are represented as a singularity in the network and include junctions, tanks, and reservoirs. Links are the connection between the singularities and include pipes, pumps, and valves. Apart from information on the network design (e.i. how the technologies are connected to each other), nodes and links are assigned structural properties giving information of the technology in use.

Apart from the physical components, the input file also includes non-physical components. These informational objects describe the environmental and operational aspects of the model. That is, curves, patterns, and controls with the following characteristics:

- *Patterns* are a set of multipliers as a function of system time. E.g., nodes are often assigned demand patterns as a multiplier to their base demand.
- *Curves* are data pairs that describe the relationship between two quantities. E.g., pump head and flow rate, pump efficiency and flow rate, and water storage height and volume.
- *Controls* are statements that determine operational aspects of the model through the simulation period. These statements describe the state of links such as valves, pumps, and pipes based on sensor data from nodes or time-dependent thresholds.

2.1.2 Hydraulic solver

The WNTR hydraulic solver is derived from EPANET to run demand-driven analysis [61]. The solver approximates the hydrostatic and hydrodynamic behavior of pressurized water pipe networks [61]. Based on equations on continuity and energy conservation, the hydraulic solver computes the hydraulic head at junctions and flow rates in pipes at each hydraulic time step [60]. The hydraulic solver updates reservoir levels, tank levels, and water demand at junctions before the next computational step.

The continuity aspect solves the mass balance at each system node. A system of equations describes inflow and outflow at each node n for all nodes in the system N [28]. Equation (2.1) is the mass balance of node n. The set of pipes connected to node n is denoted by P_n, such that $q_{p,n}$ is the inflow (e.i. negative number describes the outflow) and D_n is the demand.

$$\sum_{p \in P_n} q_{p,n} - D_n \tag{2.1}$$

Head loss in pipes is calculated based on principles for conservation of energy

[28]. Based on Bernoulli, the energy is conserved along a given pipe length by relating friction, velocity, and pressure [28].

2.1.3 Quality solver

EPANET's quality solver is applied to simulate water quality [37]. Propagation and dilution of substance concentrations are calculated based on principles of conservation of mass and reaction kinetics, advective transport in pipes, mixing at pipe junctions and storage tanks, along with the bulk flow and pipe wall reactions [60].

Although EPANET's quality solver is widely applied to study water-quality concerns [26, 27, 35], it only provides accurate results for shorter water quality time steps [38]. Janke *et al.* [38] demonstrates that overly long time steps can yield errors in concentration estimates and can result in a surplus of constitute mass. As such, Janke *et al.* [38] presents guidelines through examples and demonstration to appropriately configure the quality solver of EPANET.

2.2 Quantifying failure

A water distribution model is designed to provide sufficient quantity and quality of water, covering customers' needs without interruptions for all system nodes. By default, a substance intrusion forms less than optimal conditions that affect the quality of supplied water to customers at specific areas of the network for an extended period. Such a scenario is referred to as a stress-testing event.

The report file from the model simulation procedure contains a pool of raw consequence data from hydraulic and quality behavior of the network performance. As a water distribution model may contain thousands of nodes and connection with temporally and spatially varying behavior, it accounts for a difficult task to translate the large volume of raw data into meaningful evaluations of the performance.

In literature, performance indicators are used as a tool to determine if the loss of service is high or low [6]. As such, Moraitis *et al.* [56] presents a collection of Key Performance Indicators (KPI) to evaluate the failure of water distribution models exposed to stress-testing events. The calculation of the failure metrics involves a comprehensive three-step procedure to characterise failure by category type, service level, and failure dimension. The three-step procedure is carried out by the following steps:

1. The first step is a determination of the service category. The service category refers to what type of service delivered to the customers. It is separated by stress-testing events relation to either water quality or quantity failures.

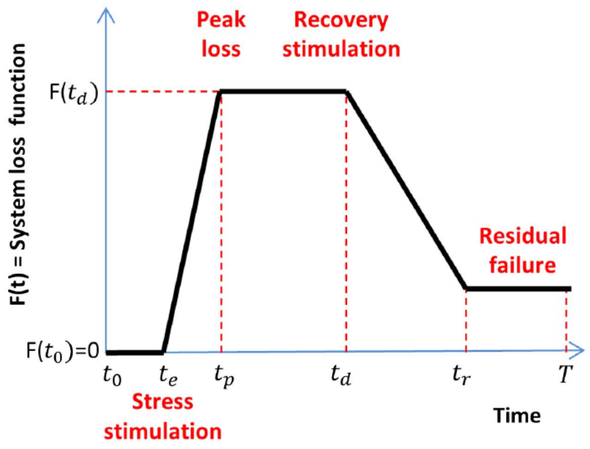


Figure 2.2: Generic loss function as presented by Moraitis *et al.* [56].

2. The second step is to identify the service levels for the selected service category. Service levels are pivotal thresholds based on regulations, standards, and policies to determine the operational state of a supply point. Based on the defined thresholds, the service of every supply node can characterised as normal, degraded, or disrupted for each step of the simulation period.
3. The third step is to differentiate between the four failure dimensions. The failure dimensions are related to service, spatial, continuity, and customer aspects of the service. Using the quality category as an example, the service dimension refers to the volume of low-quality water supplied to nodes, the spatial dimension refers to the number of nodes affected, the customer dimension refers to the number of customers that are affected, and the continuity refers to the time low-quality water is observed in the system.

The defined categories, thresholds, and dimensions are applied to produce the performance loss function. Figure 2.2 presents a generic loss function of a water distribution model exposed to a stress-testing event. E.g., a substance intrusion occurs at the time step t_e that results in a loss of performance in the following time steps. The generic loss function reflects the performance of a water distribution model on a system scale. It is observed that the failure raises to a peak loss ($t = t_p$) and decreases back towards normal operational values on a system scale. As a result, the shape of the generic loss function closely resembles the shape of the flood hydrograph [56]. Consequently, the curve can be characterised by the properties of magnitude, average propagation, crest factor, and rapidity in order to quantify the aforementioned failure metrics.

2.3 Resilience as a measure of performance

While acknowledging the importance of impact prediction, it is also essential to recognize the uncertainty of future events [48]. As future events are unknown, any assumption made to define them may affect its outcome [48]. The input of a stress-testing event or any other assumption made to determine how the existing system or alternative system designs will behave may be uncertain. Thus, an essential task of any stress-testing procedure is to capture the variability of the resulting impact through parametric changes to the input of a stress-testing event.

As resilience is a performance property of the system, it requires consideration of a wide range of threats that contribute to failure [32]. Risk analysis (e.g., the methodology of the Risk Analysis and Evaluation Toolkit [55]) is a procedure to address threat-impact relationships [35]. Risk management procedure involves threat identification, analysis, and evaluation, ranking threats based on consequences and occurrence probability, and evaluation of possible treatment options for risk reduction [57]. In terms of risk treatment, building resilience involves reducing the consequences of the threat scenario by improving the overall performance. This may be obtained by changing either the technologies (e.g., the capacity of pipes, pumps, and tanks) water distribution system or the connections between them (e.g., link connections, controls, service areas, etc.).

Conventional risk analysis typically requires likelihood estimation of threats [57]. As such, unknown events will not be captured [59], and low likelihood events are often ignored [35]. Water utilities often need to know how the system performs faced changing conditions such as asset deterioration and behavioral patterns, as well as the impact prediction of accidents and/or incidents of occurrence probability that is difficult to predict [48]. Rather than focusing on the occurrence probability, resilience assessment is a method to evaluate a wide range of threat scenarios, e.i. by including both low and high probability threats, through identification of failure modes [35].

> risk cannot be calculated; *that does not mean that such occurrences should be ignored, and resilience assessment provides a tool by which they can be considered.* Diao *et al.* [35]

The random failure sequences presented by Mugume *et al.* [32] is a stochastic stress-testing procedure to address the increasing disturbance of failure modes in urban water systems. By modeling stress-testing events with an accumulative number of link failures, Mugume *et al.* [32] expressed resilience as the relationship between the number of failed links and properties of the resulting flood hydrographs in urban drainage systems. The presented procedure of Mugume *et al.* [32] accounts for input variability by randomly selecting link failures at each stress level multiple times. Thus, the degree of lost performance may be measured as the minimum, mean, and maximum strain (e.i. strain) at each stress level.

Similarly, Diao *et al.* [35] applies the random failure sequences to evaluate the resilience of water distribution systems exposed to pipe failures, fire fighting demands, and substance intrusions. In terms of failure due to substance intrusion, Diao *et al.* [35] quantifies failure as the total volume of low-quality water (e.g., substance concentration above a predefined threshold) supplied to customers, and the total duration of the low-quality water is observed in the network.

Meng *et al.* [65] express resilience as the area under the stress-strain curve using the random failure sequences to express progressively increasing disturbance of pipe failures. Based on quantity objective, Meng *et al.* [65] quantifies resilience related to six failure aspects (e.i. start time of failure, failure duration, peak loss, the rate of failure, recovery rate, the magnitude of failure).

Chapter 3

Methods and tools

The method and tools chapter presents the modules used in the correlation analysis. It is comprised of three sections Key Performance Indicators (see Section 3.1), topological attribute metrics (see Section 3.2), and correlation as a statistical method (see Section 7.3).

3.1 Key Performance Indicators

The total simulation duration are configured to 94 hours with a report time step of 1 hour of all experiments conducted. Furthermore, hydraulic and quality time steps are configured as 3600 seconds and 60 seconds, respectively. The quality time step is significantly shorter than the hydraulic time step in order to avoid mass imbalances and errors in concentration estimate [38]. The following subsections presents service levels, failure dimension, and failure metrics used in this study [56], along with the introduction of global failure indicators for water-quality related concerns.

3.1.1 Service levels

Directive of the European Parliament and of the Council [3] defines safe water intended for human consumption as the absence of harmful micro-organisms and substances. In addition to definitions, principles, and regulations, its report presents the minimum requirement for parametric values used to assess the quality of drinking water. As water contaminants may be harmful if consumed above certain levels, concentration thresholds are set for a list of priority pollutants. E.g., legislation values range from lower concentration such as cyanide $50\mu g/L$ to $250g/L$ that is chloride. Thus, it is reasonable to expect drinking water to contain at least small amounts of contaminants, such as physical, chemical, or biological substances, without harming the customer.

Table 3.1: Quality ranges of service levels [56].

Service Level	Criticality	Alias	Quality Range (g/L)
L_0	Any	Low Quality	$c_{n,t} > 20$
L_1	Moderate	Sub-Standard Quality	$60 > c_{n,t} > 20$
L_2	Critical	Polluted Quality	$c_{n,t} > 60$

The service levels of Moraitis *et al.* [56] assigns two concentration thresholds based on a substantial degree of lethality. Moderate lethality is denoted by c_{s_e} and indicates that the water quality is above legislation. Water consumed with a smaller concentration than c_{s_e} is assumed harmless to the customer, while any water of higher consternation may affect the customer's health. As this study assesses the failure mode of substance intrusion rather than a specific threat scenario, the arbitrary value of $20g/L$ is assigned to c_{s_e} threshold. The threshold of critical lethally is denoted by c_{s_p}, and any consumed concentration above c_{s_p} is considered lethal to the customers. In a similar manner to c_{s_e} threshold, the value of c_{s_p} is configured to $60g/L$.

Quality ranges are defined from the given concentration thresholds in Table 3.1. The three service levels L_0 , L_1, and L_2 represent quality ranges that define low, sub-standard, and polluted quality respectably. Water leaving the node n at time t is classified as low quality (e.i. L_0) if the substance concentration $c_{n,t}$ is above c_{s_e}, including concentrations of both moderate and critical lethality. While polluted quality (e.i. L_0) involve any node n at time t with a substance concentration $c_{n,t}$ above the critical threshold c_{s_p}. The final service level L_0 is sub-standard water quality and includes any substance concentration $c_{n,t}$ substance concentration between the moderate and critical threshold. Hence, a source point with a substance concentration not included by the quality ranges at time t is predicted to supply water of normal quality, or at least water safe to drink. It must also be noted that $c_{n,t}$ may be included in two quality ranges as a combination of either L_0 and L_1, or L_0 and L_2.

3.1.2 Failure dimension

The failure dimension is the manifestation of service, spatial, temporal, and social aspects of performance [56]. Each dimension is quantified at a nodal and temporal level and refers to the system loss function (see Figure 2.2). A stress-testing simulation of a network that contains N number of supply points is defined for time $t \in (0, t]$ and node $n \in [1, N]$. Thus, substance concentration $c_{n,t}$ and water supply $S_{n,t}$ are defined for each time step and node in the report file.

The service dimension is the supplied volume in node n of water with quality L_i at time t and denoted as polluted supply $PS_{L_i,n,t}$. Thus, polluted supply $PS_{L_i,n,t}$ is identical to the supplied water $S_{n,t}$ if the concentration of node n at time t is in the quality range of L_i, as seen in the Equation (3.1).

$$PS_{L_i,n,t} = \begin{cases} S_{n,t} & \text{if } c_{n,t} \in L_i \\ 0 & \text{if } c_{n,t} \notin L_i \end{cases} \tag{3.1}$$

The spatial dimension $N_{L_i,n,t}$ determines if node n is affected by water of quality L_i at time t. Any node affected by degraded quality is assigned the value of 1, while those not affect assigned the value of 0 (see Equation (3.2)). An estimation of the number of nodes affected will, in turn, help to determine the spatial influence of a stress testing event.

$$N_{L_i,n,t} = \begin{cases} 1 & \text{if } c_{n,t} \in L_i \\ 0 & \text{if } c_{n,t} \notin L_i \end{cases} \tag{3.2}$$

Social metrics are closely related to the spatial dimension, as it determines the number of affected customers. The number of costumers $C_{t,n}$ at node n at time t is calculated in Equation (3.3) as a function of $D_{t,n}$ using the per capita consumption at peak business hour $D_{t_{peak}}$ and the total population P_{total}. With the estimated population at the supply point, the number of affected customers $C_{L_i,t,n}$ is calculated as the spatial dimension in Equation (3.4).

$$C_{n,t} = \frac{D_{n,t}}{D_{t_{peak}}} \cdot P_{total} \tag{3.3}$$

$$C_{L_i,n,t} = \begin{cases} C_{n,t} & \text{if } c_{n,t} \in L_i \\ 0 & \text{if } c_{n,t} \notin L_i \end{cases} \tag{3.4}$$

The continuity dimension is denoted by $T_{L_i,n,t}$ and communicates the temporal aspects of the failure. Equation (3.4) assigns nodes at time t node n is affected by water of quality L_i a value of one.

$$T_{L_i,n,t} = \begin{cases} 1 & \text{if } c_{n,t} \in L_i \\ 0 & \text{if } c_{n,t} \notin L_i \end{cases} \tag{3.5}$$

3.1.3 Failure metrics

The proposed methodology of Moraitis *et al.* [56] quantifies a water system's failure in terms of its magnitude, average propagation, severity, and peak-to-average ratio using the loss function analogous with the flood curve in hydrology.

Magnitude

A key property of the flood curve is the total runoff volume, as it directly reflects the magnitude of the flood event [25]. It also refers to the catchment's ability to absorb some of the precipitation through infiltration and evaporation [55]. Similarly, the magnitude of failure captures the total area under the system loss function for each of the failure dimensions. In terms of absorbing a substance intrusion, the low-quality water may be flushed out of the system or diluted by mixing with water from other parts of the distribution network.

In terms of the service dimension, magnitude is the total volume of degraded quality L_i supplied to customers. The system loss function is first defined by volume degraded water quality at each time step. Thus, by also adding the volume at each time step of the simulation period, as shown in Equation (3.6), grants the total volume of degraded water supplied to customers PS_{L_i}. Equation (3.7) show an alternative representation of service magnitude. $PS_{L_i,Ratio}$ forms a ratio between the total volume degraded water supplied and total demand D_{L_i} in the system [56].

$$PS_{L_i} = \sum_{t_0}^{T} \sum_{n=1}^{N} PS_{L_i,n,t} \tag{3.6}$$

$$PS_{L_i,ratio} = \frac{\sum_{t_0}^{T} \sum_{n=1}^{N} PS_{L_i,n,t}}{\sum_{t_0}^{T} \sum_{n=1}^{N} D_{L_i,n,t}} \tag{3.7}$$

Spatial metrics for magnitude give the number of nodes that experience degraded water, according to each level [56]. As such, a one dimensional vector is formed by (Equation 3.8) to quantify if node $N_{L_i,n}$ affected by degraded quality within the simulation period T. The total number of nodes affected N_{L_i} is calculated with Equation (3.9), and ratio between nodes affected and total nodes in the system is calculated with Equation (3.10).

$$N_{L_i,n} = \begin{cases} 1 & \text{if } \exists\ t\text{: } N_{L_i,n,t} = 1 \\ 0 & \text{if } \nexists\ t\text{: } N_{L_i,n,t} = 1 \end{cases} \tag{3.8}$$

$$N_{L_i} = \sum_{n} N_{L_i,n} \tag{3.9}$$

$$N_{L_i,ratio} = \frac{\sum_n N_{L_i,n}}{N} \tag{3.10}$$

The same principle holds for the customer dimension [56]. Based on customers maximum customers affected $C_{L_i,n}$ from Equation (3.11), magnitude of failure is the customers or ratio of customers that experienced failure throughout the total duration of service failure (see Equation (3.12) and Equation (3.13)).

$$C_{L_i,n} = \begin{cases} \max_{t_0 \le t \le T} C_{L_i,n,t} & \text{if } \exists\ t\text{: } C_{L_i,n,t} = C_{n,t} \\ 0 & \text{if } \nexists\ t\text{: } C_{L_i,n,t} = C_{n,t} \end{cases} \tag{3.11}$$

$$C_{L_i} = \sum_n C_{L_i,n} \tag{3.12}$$

$$C_{L_i,ratio} = \frac{\sum_n C_{L_i,n}}{P} \tag{3.13}$$

As for continuity, the time based vector $T_{L_i,t}$ is derived from Equation (3.14) to determine which time steps any nodes is supplied with degraded water quality L_i. Equation (3.15) is the summation of the $T_{L_i,t}$ vector to calculate total duration of failure T_{L_i}. That is the total time the system services below expectations even one node, while $T_{L_i,ratio}$ is the ratio between failure time and the simulation duration T (see Equation (3.16)).

$$T_{L_i,t} = \begin{cases} 1 & \text{if } \exists\ n\text{: } N_{L_i,n,t} = 1 \\ 0 & \text{if } \nexists\ n\text{: } N_{L_i,n,t} = 1 \end{cases} \tag{3.14}$$

$$T_{L_i} = \sum_n T_{L_i,n} \cdot \Delta t_t \tag{3.15}$$

$$T_{L_i,ratio} = \frac{\sum_n C_{L_i,n} \cdot \Delta t_t}{T} \tag{3.16}$$

Average propagation

The propagation through time is another characteristics of the flood curve [55]. As average propagation is translated to average failure of water distribution systems through time [56], it gives a give an absolute number to each dimension as the average failure through time. Equation (3.17) explores the failure dimension F_{L_i} as a average value against total duration of failure. The failure dimensions of service, spatial, and customers is denoted by F_{L_i}, such that $F_{L_i} \in [PS_{L_i}, N_{L_i}, C_{L_i}]$.

$$\overline{F_{L_i}} = \frac{F_{L_i}}{\sum_t T_{L_i,t}} \tag{3.17}$$

Equivalently, to address continuity–related metrics, the average failure per node of Equation (3.18) is applied as the ratio between total duration failure and total number of nodes affected [56].

$$\overline{NT_{L_i}} = \frac{T_{L_i}}{N_{L_i}} \tag{3.18}$$

Severity

In terms of severity of a stress testing event, peak values of the performance loss function derived from the maximum value reached through the simulation duration. The peak value of failure dimension $F_{L_i,n,t}$ is calculated by Equation (3.19) to give the severity F_{peak,L_i}. Severity of service failure refers to the peak volume of degraded water, while the spatial and customer dimension is the maximum number of nodes and customers affected at the same time.

$$F_{peak,L_i} = \max_{t_0:T} \sum_{n=1}^{N} F_{L_i,n,t} \tag{3.19}$$

$$F_{peak,L_i} \in [PS_{peak,L_i}, N_{peak,L_i}, C_{peak,L_i}]$$

Crest factor

While some flood events produce a gradually increasing runoff value, others occur rapidly as a flash flood. The flood analogy can be translated to stress testing events, as the impact of some events is abruptly observed at the source points, while others gradually increase in severity [56]. Peak-To-Average (PAR) is a crest factor [25] addressing the relationship between peak failure and the average propagation of a failure dimension in Equation (3.20). The crest factor communicates a constant failure profile as the value approaches one, while larger values may indicate abrupt changes in severity [56].

$$F_{PAR,L_i} = \frac{F_{peak,L_i}}{\overline{F_{L_i}}} \tag{3.20}$$

$$F_{PAR,L_i} \in [PS_{PAR,L_i}, N_{PAR,L_i}, C_{PAR,L_i}]$$

Rapidity

Rapidity is another aspect of the failure. This is measured as the time from the start of a stress testing event to the peak value. As shown presented by the generic loss function (Figure 2.2), the beginning of the event is denoted as time t_e, and the peak value occurs at time t_p. Thus, time from Event-to-Peak (TEP) is calculated with Equation (3.21).

$$F_{TEP,L_i} = t_{F_{peak,L_i}} - t_e \tag{3.21}$$

$$F_{TEP,L_i} \in [PS_{TEP,L_i}, N_{TEP,L_i}, C_{TEP,L_i}]$$

3.1.4 Global failure metrics

Based on the three-step methodology of Moraitis *et al.* [56] to quantify failure, the proposed global failure metrics communicate the resilience water distribution models exposed to a failure mode. As noted by Makropoulos *et al.* [48], following the principle of parsimony implies reserving the term resilience by its intended usage as a dynamic measure of reliability, and thus limiting the global failure metrics to an indication of its definition. Being a reflection of the resilience metrics, global failure metrics address the lack of resistance, absorption, and restoration capacities of a water distribution system exposed to substance intrusions.

In order to preserve the interoperability of the aforementioned failure metrics of Moraitis *et al.* [56], the global failure metrics are calculated as the mean failure metrics of the simulated stress-testing events. Equation (3.22) express the global failure metric F based on the properties of magnitude, average propagation, severity, crest factor, and rapidity of the system loss function for the service dimension of interest. The number of stress testing events is denoted by s and F_i is the quantified failure of a stress testing event, such that $i \in [1, 2, 3.., s]$. Thus, global failure metrics differ from the failure metrics of Moraitis *et al.* [56] as the former includes failure quantification of multiple stress-testing events.

$$F = \frac{\sum_{i=1}^{s} F_i}{s} \tag{3.22}$$

3.2 Topological attribute assessment

Structural attributes are defined as key characteristics of the physical objects described in the input file. This study combines elements of complex graph theory to describe system elements' relation to each other. The following subsections give an introduction to complex graph theory and present relevant formulas to calculate global structural attributes.

3.2.1 Complex network theory

Graph theory is a tool to study mathematical structures and relationships between objects. Like many other infrastructures (e.g., road networks, electrical grids, railway, gas, etc.), water distribution systems consist of multiple interconnected components. As the input file describes these components for simulating hydraulic and quality behavior, it also can be translated to the realm of graphs. A graph consists of nodes connected with edges. As such, the objects inherent properties are excluded, such that its building blocks (e.g., connectivity) can be evaluated at the most basic level [4].

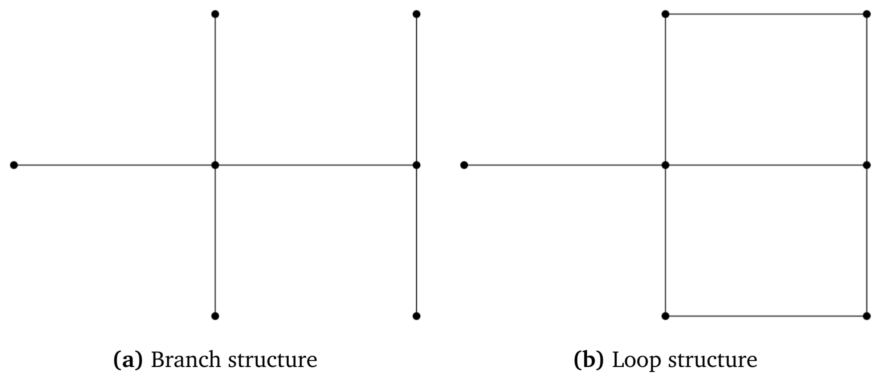

(a) Branch structure (b) Loop structure

Figure 3.1: The two main types of network design structures [19]

At its core, water distribution systems, or at least simplified versions, is often characterised as a branch structure or a loop structure [19]. A looped structure (see Figure 3.1b) is a network where nodes are supplied from multiple directions by connecting the pipes as to form rings. The other type of system is termed branch structure (see Figure 3.1a) as the pipes form branches to distribute water to end nodes. The looped structure is often regarded as preferable, as it provides a higher degree of reliability, robustness, and resilience in terms of pipe failure [33]. Yet, most water distribution forms a complex combination of both types of systems as the network adapts to the changing physical, social, and economic landscape. Thus, analysis of complex networks using graph theory techniques involves quantification of network building blocks to identify characteristics of the water distribution networks. Through identification of vulnerabilities and critical locations, complex network theory can be used to classify water distributions systems by their structural features [40].

3.2.2 Topological attributes

The topological properties of a water distribution system are characterised by statistical indicators in complex network theory [24]. This study uses the five topological attribute indicators of link density, average path length, clustering coefficient, central-point of dominance, and average closeness centrality.

Link density

Link density d is the most basic indicator as it describes the ratio between nodes and links. Link density address the connectivity of a graph, e.i. a water distribution network with an abundance of links is referenced as highly connected. The link density is given by Equation (3.23), where n is the number of nodes and m is the number of links in the graph.

$$d = \frac{2m}{n(n-1)} \tag{3.23}$$

E.g., the branch structure of Figure 3.1a has a link density of 0.29 with seven nodes and six links, while the loop structure (see Figure 3.1b) with an additional two links has a density of 0.38. Thus, the ring system has higher connectivity than the tree system in respect to the link density.

Average shortest path length

The average shortest path length estimates the average number of links that need to be traversed in order to reach from one node to another. V of Equation (3.24) is a set of nodes in G, and $d(s,t)$ is the shortest distance between the nodes s and t.

$$a = \frac{1}{n(n-1)} \sum_{s,t \in V} d(s,t) \tag{3.24}$$

The average shortest path length is a measure of the efficiency of water transport in a network. An efficient network is distinguished from a complicated and inefficient one, with a shorter average path length being more desirable. E.g., the looped structure has a lower value ($a = 1.8$) than the branched structure ($a = 2$), since the terminal nodes of the branched network are connected in the looped network. It must be noted that the average shortest path length used in the correlation analysis is independent of the link lengths or any other physical property of the system.

Clustering coefficient

The clustering coefficient C_c is based on triplets of nodes to measure the degree of nodes tend to cluster together. Equation (3.25) quantifies the density of triangular loops of triplets and the degree to which junctions in a graph tend to be linked. A triplet consists of three nodes connected to each other. The triplet is defined as open if the three nodes are connected by two links or closed if the nodes are connected by three links. The clustering coefficient is the number of closed triplets ($3N_\Delta$) over the total number of triplets (N_3), which includes both open and closed loops.

$$C_c = \frac{3N_\Delta}{N_3} \tag{3.25}$$

The clustering coefficient is usually found to be a small number in systems with fewer structural loops that differ from a simple triangle [16]. E.g., as the loop structure of Figure 3.1b forms a grid-like system without any triangle loops, its clustering coefficient equals zero.

Central-point of Dominance

The betweenness centrality of a node i is denoted as B_i and is the number of times a node is traversed by the shortest path length $d(s,t)$ such that $s,t \in V$. The most central node is denoted B_{max} and is defined as the node that is traversed the most times. The Central-point of Dominance CB indicator is calculated with Equation (3.26) as the average difference in betweenness centrality of the most central node and all other nodes.

$$C_B = \frac{1}{n-1}\sum_{i=1}^{n}(B_{max} - B_i) \tag{3.26}$$

The indicator is a measure of the node concentration around a central location in respect to other locations in the network. The value of CB is limited to two extremes of 0 for a perfectly distributed network (e.g. all nodes connected to each other) and 1 for a star-like structures (e.g. all nodes connected to a central node). Using Figure 3.1 as an example again, the loop structure have a lower central-point of dominance than the branched structure.

Average closeness centrality

As betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes, closeness centrality can be regarded as a measure of how long it will take to spread water from one node to all other nodes following the shortest path $(d(s,t))$. Thus, the average closeness centrality is a measure of the averaged shortest distance between all nodes in the network independent of the total nodes in the system.

$$C_u = \frac{n-1}{n}\sum_{j=1}^{n}\sum_{i=1}^{n-1}\frac{1}{d(s_j,t_i)} \tag{3.27}$$

3.3 Correlation as a statistical method

In order to limit noise from operational and environmental variables, network variants are generated by adding and removing pipes from existing networks. These networks are referred to as network variants being the networks generated based on a benchmark model. The network variants are a set of water distribution models with the with fixed nodes node and non-physical objects (e.i., curves, patterns, and controls), along with a identical topological backbone. However, the generated networks can be characterised by different topological attributes as the peripheral node connections vary from network variant to variant. The network variants exposed to a random failure sequence of a substance intrusion and their resilience are evaluated with the global failure indicators. Thus, two sets of independent sets of variables addressing resilience and topological attributes may be established for each network variant.

The correlation analysis is a statistical method applied to evaluate the strength of the relationship between produced global failure indicators and the topological attribute indicators of the network variants.

3.3.1 Pearson correlation coefficient

The Pearson correlation coefficient was developed by Pearson (1896) [22] and is a measure of the degree of linearity between two independent variables. The strength and direction of the relationship between two sets vectors of variables are measured with Equation (3.28), where i is a network variant associated with topological attribute indicator x_i and global failure indicator y_i.

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{2}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{2}(y_i - \overline{y})^2} \qquad (3.28)$$

The Pearson correlation coefficient yields a value between -1 and 0 for a decreasing relationship, or a value between 0 and 1 for an increasing relationship. As such, the coefficient describes a scenario in which the size of one variable increases as the other variables also increases or where the size of one variable increases as the other variable also decreases. A coefficient close to 0 means there is no linear relation between the two variables. Whereas a coefficient close to 1 or -1 indicates a strong positive or negative correlation, respectively. Rather than using oversimplified rules and ranges to translate the coefficient into descriptions (e.i. weak, moderate, or strong correlations), it is suggested by Schober *et al.* [14] to interpreted the strength of the relationship in the context of posed research question.

3.3.2 Scatter plots

It is also advised by Asuero *et al.* [22] to conduct visual inspections of the produced data. In addition to the monotonic requirement of the data sequences, the Pearson correlation coefficient is based on a linear relationship between the independent variables. Thus, any observed relationship deviating from this linearity is not captured by the Pearson correlation coefficient. As a scatter plot matrix gives a visual summary of linearity and non-linearity of the data, it is often more useful than the correlation coefficient [22].

Additionally, the correlation coefficient is sensitive to points that substantially deviate from the main clustering. As a scatter plot presents network variants as individual points in the graph based on performance and topology, it makes it possible to detect outliers and evaluate their effect on the correlation coefficient. The scatter plots applied in this study are supported by linear regression of the data. However, the regression line itself provides no information about how strongly the variables are related [22]. Linear regression has only one independent variable and one dependent variable as it fits a line through the data points of the scatter plots. The line is used as an indicator to evaluate the direction and spread of the data.

3.3.3 P-values

The Pearson correlation coefficient of the independent variables is coupled with a p-value. As such, the p-value is a measure of the probable error of a correlation coefficient. The approach for determining the p-value is derived from bootstrapping random permutations from the pool of x_i, y_i variables [15]. In order to obtain a normal distribution of correlation coefficients, it is repeatedly calculated for the pool of bootstrapped variables (x_i', y_i'). The probability of the observed result is derived from the probability distribution, such that a small p-value indicates that the observed result has a small probability of being random.

Chapter 4

Data acquisition and processing

In this chapter, all data which are used in this study are presented. The first part presents the network characteristics of the benchmark models are, and the second part describes the generation of network variants.

4.1 Benchmark models

Three benchmark models of various sizes and characteristics are used for the correlation analysis. The overall network characteristics of the models are summarised in Table 4.1. The Net3 model is an example network included in the installation package of the Epanet software. This model is made to demonstrate the percentage of Lake water in a dual-source system over time in a relatively small system. CTown is selected as a medium-sized model as it is widely applied in other academical studies [55, 65]. In contrast to the small and medium-sized network, Trondheim is a real water distribution network from the region Trondelag region in Norway. The city has a population of approximately 200 000 people with a rapidly growing technology-oriented industry Statistics Norway [43]. The water distribution network is operated by the municipality, yet it is often a subject to research projects due to its approximate to the Norwegian University of Science and Technology (NTNU). Trondheim is a relatively large water distribution network with redundant loops in the urban center and branched configurations in the city outskirts.

Table 4.1: Characteristics of the benchmark models.

Model	Junction	Reservoirs	Tanks	Pipes	Pumps	Valves	Costumers
Net3	92	2	3	117	2	0	1000
Ctown	388	1	7	429	11	4	10000
Trondheim	10743	4	14	11903	83	124	200000

4.1.1 Ctown configuration

The Trondheim model was initially exported from *MIKE+*, a commercial software used by Trondheim municipality, to the format of an Epanet network file (.inp). As the modeling environment of MIKE + has added functionalities and a more flexible labeling acceptance than the modeling environment in WNTR, the network file must be adjusted to meet the following issues:

- Non ASCII characters within the file
- Duplicated and long (>32 characters) IDs
- IDs interpreted as a integer or a float value
- Errors reported by the WNTR package

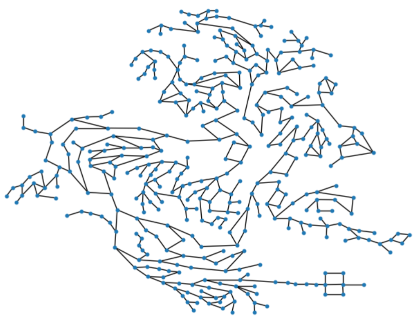
Python scripted was developed to format and relabel the input file as the network IDs of the model partially consist of Norwegian characters and numbering interpreted as integers. This script generates a new labeling system similar to the labeling system used in the CTown model. First by through integrating the network element in the system and relabel the element with a character describing the type and a number uniquely defined for the element (e.g. junction *124* is relabeled *J124*, tank *Storvannet* is relabeled *T1* etc.). Then the network is written back to a new network input file (.inp) with the new labeling and valid formatting (see Code listing C.1).

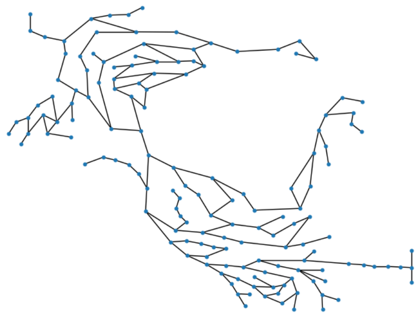
Furthermore, the network models in WNTR do not support the variable speed pump, real-time control, and ignorance of disconnected nodes functionalities of MIKE +. Hence, manual modifications were therefore manually made to obtain an accurate representation of the physical behavior of the system in the WNTR environment.

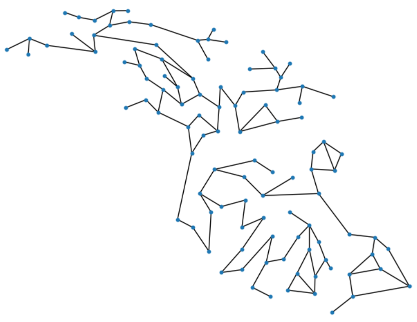
4.2 Service areas

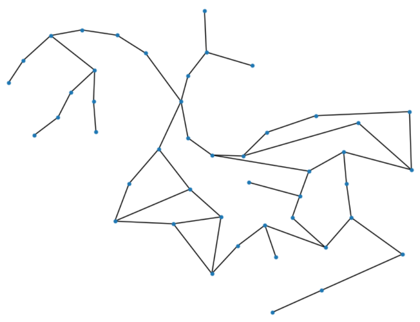
District Metered Areas (DMAs), also referred to as service areas, are defined as discrete areas of a water distribution system. It is usually created by closing boundary valves to break down the network into zones and sub-zones [19]. A zone is connected to the main system with a pumping station which regulates the pressure within the boundary of the service area.

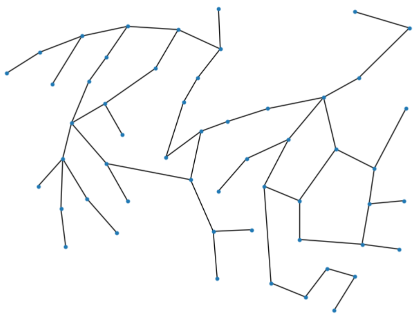
The service areas of the benchmark models are mapped by removing pumping stations, valves, and closed pipes before converting the water distribution model into a graph. The resulting graph of the benchmark models contains clusters of nodes connected by pipes. These node clusters are equivalent to the aforementioned service areas. As such, nodes are grouped with all other nodes within a cluster which are reachable with a remaining pipe connections. Using Ctown as an example, the result of the grouping are presented as individual graphs in Figure 4.1.


(a) CTown


(b) DMA 1


(c) DMA 2


(d) DMA 3


(e) DMA 4

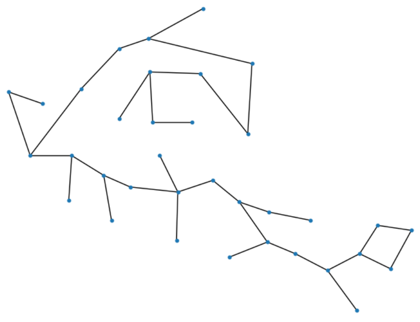

(f) DMA 5

Figure 4.1: Ctown nodes and links grouped by service areas.

4.3 Generating network variants

A water distribution system generation algorithm is developed to produce different topological variants based on the benchmark models (see Code listing C.2). Similarly, Moederl *et al.* [42] automated the generation of 2280 virtual water distribution systems of topological and size variations with a stochastic approach. In their work, the network set were based on a constant elevation of junctions and a fixed elevation of the reservoir. The study of Sitzenfrei *et al.* [41] presents a similar method using geographic information system data and boundary conditions to obtain a more comparable network to real world water distribution networks.

The produced networks variants of this study replicates a benchmark model's topological backbone, while peripheral pipe connections are shuffled between nodes. E.i. the network variants share a fixed number of nodes, pumps, and valves with identical properties, service area configurations, and primarily feeders mains, along with non-physical objects. The network generation procedure are summarised with the following four step:

1. **Link classification** prioritise links in a benchmark network based on a set of predefined rules. Links such as pumps, valves, and primary feeders are classified as critical, while all other links are classified as non-critical to the network's main structure.
2. **Link stripping** removes the defined non-critical links from the network. Its output is the topological backbone of the benchmark model with a high degree of disconnected nodes. Disconnected nodes refers to nodes or groups of nodes not connected to the topological backbone.
3. **Link addition** detects disconnected nodes and generates a list of possible connection points to other nodes. Based on statistical properties and design constraints, nodes are randomly connected to each other in a controlled environment to form edges. Its output is a fully connected graph with a new set of topological attributes.
4. **Model calibration** assigns informational values (e.i. roughness, length, and diameter) to the generated connections. The hydraulic behavior of under business-as-usual scenarios are evaluated to calibrate diameters of the added pipes through multiple simulation runs.

Figure 4.2 illustrates the steps of the network generation procedure. All operations involving topological modifications are supported by graph theory, e.i. link removal, addition, and in part the DMA classification.

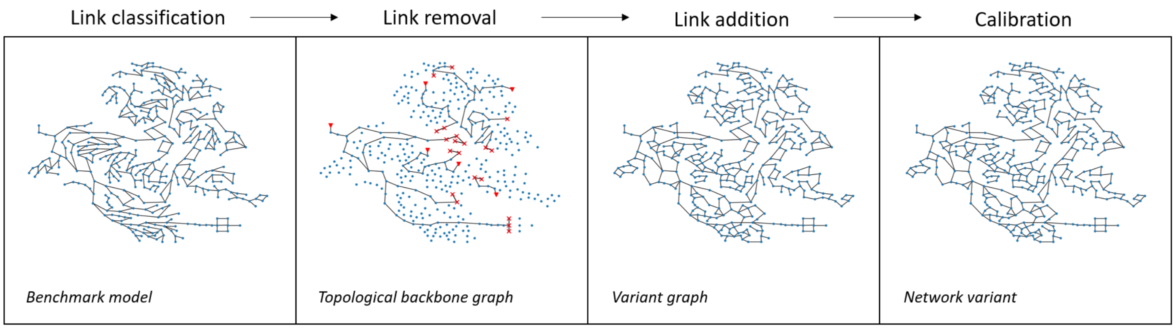


Figure 4.2: Visual representation of the network generation procedure.

4.3.1 Link classification

Generating network variants involves prioritisation of network links in order to reserve links that are important to the network's main structure. Hence, links are classified as either critical or non-critical to the network's operational functionality. Links of critical importance include pumps, valves, and feeder mains, and most not be removed from the network. The links not classified as critical are assigned to the pool of non-critical links and before removed from the network. These links are pipes part of a sub-system within the defined service area with a primary role of connecting supply nodes to the main network structure.

Classification of feeder mains (e.i. larger diameter pipes) involve defining its start and endpoints. From the perspective of complex network theory, the start point t and endpoint v is set of nodes $(v, t) \in V_c$ from a group of nodes in a given service area G_{DMA} such that $V_c \in G_{DMA}$. From the perspective of a water engineering perspective, reservoirs may naturally be defined as the starting point for water transportation. As water in the network is distributed from a source, reservoirs are often connected to a treatment facility and a pumping station for further distribution of water. The service areas solely associated inlet the reservoir, treatment plant, and pumping stations are not included in the network generation procedure, and may be filtered out based on their relatively small size. Drinking water is typically further distributed through feed mains from a pumping station to tanks or other pumping stations within each service areas.Thus, V_c can be defined as all combination of node pairs with the following properties:

- Water tank nodes
- Nodes connected to a pump
- Nodes connected to a valve

A path is a set of links that are traversed in order to reach from one node to another. The Dijkstra algorithm [23] is used to determine the shortest path between all defined start and end points V_c. Dijkstra's closest weighted path is a modified version of the shortest path length $d(t,v)$ from Equation (3.24). Weights are used to determine the path of least resistance. Rather than the lengths of the links, the inverse of the pipe diameters (e.i., $1/diameter$) are used as weights. The inverse diameter approach assigns smaller weights to primary feeder mains as they generally have a higher capacity to transport water. E.i. links with a larger diameter represent a shorter path towards the endpoint than pipes of smaller diameters. All pipes within each service area not included as a critical link is classified as non-critical.

4.3.2 Link removal

The pipes classified as non-critical are removed from the system. The result is the topological backbone graph and are illustrated in Figure 4.2. Start and endpoints are represented by a red cross, and are correspond by feeder mains. As services area of Figure 4.2 is left with multiple floating junctions, e.i. nodes not connected to the main structure of the graph, the link removal generates a none functional water distribution network. As such, the following steps concern the reconfiguration of the graph to generate a fully connected network variant. The output of the link addition procedure is a list of removed edges in the network graph.

4.3.3 Link addition

Link addition step is based on stochastic procedure to establish generate edges with start and end nodes. The iterative procedure are based on the following statistical properties and network design constraints to generate a edge:

1. The probability exponent of node degrees produces a probability distribution of floating nodes. E.i., each floating node of a service area is assigned a probability of being selected as a start node. The probability associated with a node is based on the number of nodes connected to the node of interest. The node degree is raised to an exponent to amplify its probability further. Thus, increasing the exponent value increases the probability of selecting a node with a low node degree as a start node. The probability distribution ensures convergence of the iterative link addition procedure.
2. The probability exponent of proximity assigns a probability distribution of all nodes in a service area based on the Euclidean distance from the selected start node. Nodes geographically adjacent to the selected start node have a higher probability of being selected as an end node. The distances are further amplified with the exponent of proximity.
3. The number of possible connections is regulated to limit possible connection points from a start node. The potential connections from step (2) are ranked in ascending order, and N closed nodes are included in the array of possible end nodes. A random end node is generated from the array to form

a potential edge.

4. Alpha is an angle constraint of a potential edge. This step calculates the minimum angle between the potential edge and all existing edges from both the start and end nodes. If the potential edge does satisfy the alpha constraint in respect to the other edges, it is added to the graph, or else it is disregarded as a network edge.

The step (1) to (4) is repeated until all nodes are connected to the topological backbone of the service area. The procedure are repeated in each of the defined service areas (see Section 4.2). The output of the link addition procedure is a list of added edges to the network graph.

4.3.4 Model calibration

The output of the link removal and addition steps are used to remove and add links to a digital copy of the benchmark model. Pipe length of the added links is determined by the Euclidean distance between the start node and end node, while the roughness value is assigned based on the average roughness of the removed pipes in the service area.

The automatic pipe sizing procedure of Sitzenfrei *et al.* [41] is used to determine appropriate diameters to the added pipes. The implemented approach is based on an iterative calculation of flow velocities to optimise the diameter for a economic flow rate. All pipes are first assign a pipe diameter of 80 centimeters. The flow rate in each of the added pipes are evaluated from a simulation under normal conditions. If the calculated flow velocity in a pipe exceeds the corresponding flow velocity of a diameter, its pipe diameter is incrementally increased to the next available pipe diameter. This process continues until all pipes satisfy the velocity requirement of their pipe diameter.

Chapter 5

Input uncertainty analysis

This chapter outlines the steps of the input uncertainty analysis related to the boundaries of substance intrusions, and its effects on the failure metrics related to magnitude (see Equation (3.22)). Using a service area of Ctown as an example, the following sections include a sensitivity analysis of the control variables and variability analysis of the source points.

5.1 Sensitivity of control variables

This section concerns a sensitivity analysis of control variables and their effect on failure metrics. As water distribution systems are dynamic infrastructures that adapt to changing heads, demand patterns, and operational constraints, its hydraulic landscape may differ substantially from time step to time step. Hence, the impact of a substance intrusion is in part dependent on the temporal and magnitude aspects of a stress-testing event. The timing and the total mass entering the system are characteristics of a stress testing event reflected through the control variables start time, mass flow, and duration.

Through a One-At-a-Time (OAT) approach, one control variable is changed by incremental steps at the time, while the other control variables are kept at a baseline value. As the stress testing event is only changed by one variable each simulation run, noise is reduced from other variables, making the failure metrics ambiguous [31].

Table 5.1 presents the baseline and ranges of values for each of the three control variable included in the sensitivity analysis. The procedure involves moving the control variables with incremental steps from the lower to upper bound. The recorded change of performance is measured through $PS_{L_0,ratio}$, $N_{L_0,n,ratio}$, $T_{L_0,t,ratio}$, an $C_{L_0,n,ratio}$, e.i. magnitude as a failure ratio. This process is repeated such that each junction of DMA 1 of Ctown (see Figure 4.1) is individually tested as a source point for substance intrusion with moving control variables.

Table 5.1: One-At-a-Time (OAT) control variable ranges.

Variable	Baseline	Lower bound	Upper bound	Increment	Unit
Start time	0	0	24	1	hr
Duration	1	1	10	1	hr
Mass flow	25	10	100	10	g/s

Through a One-At-a-Time (OAT) approach, one control variable is changed by incremental steps, while the other control variables remain at a baseline value. As the stress testing event is only changed by one variable each simulation run, noise is reduced from other variables, making the failure metrics ambiguous [31].

Table 5.1 presents the baseline and ranges of values for each of the three control variables included in the sensitivity analysis. The procedure involves moving the control variables with incremental steps from the lower to the upper bound. The recorded change of performance is measured through $PS_{L_0,ratio}$ (see Equation 3.7), $N_{L_0,n,ratio}$ (see Equation 3.10), $T_{L_0,t,ratio}$ (see Equation 3.13), and $C_{L_0,n,ratio}$ (see Equation 3.16). This process is repeated such that each junction in DMA 1 of Ctown (see Figure 4.1) is individually used as a source point for substance intrusion with moving control variables.

5.1.1 Start time of a substance intrusion

The start time input is defined as the number of hours from the beginning of the simulation to the substance intrusion at a source point. The start times input range from 0 hours to 23 hours, with an incremental step of 1 hour for stress-testing event. The mass flow input is 25 g/s, and the duration input is 1 hour. E.i., the level of stress is equivalent to 90 kg of a substance entering the system.

It is essential to understand the underlying mechanisms of a water distribution model. The control and rules of Ctown regulate the pump operation based on sensor data on tank levels. Tank levels are in part regulated by the inflow (e.i. pumped water) or water outflow to cover demand in the service area. However, the operation is highly dependent on the environmental variables enforced by demand patterns. As such, demand patterns are used to evaluate the continuously changing behavior of the system. Figure 5.1 presents the demand multipliers as a function of time. It forms a periodic function repeating with a period of 24 hours. The highest demand multipliers are observed between the time steps of 15 and 21, while the lowest demand value is observed at the time steps between 0 and 7.

The maximum $N_{L_0,n,ratio}$ value is occurring at the start time input of 0 with a value of 0.278, while the minimum $N_{L_0,n,ratio}$ value of 0.138 occurs at the start time input of 16. One could observe that the propagation of $N_{L_0,n,ratio}$ values tends to decrease with increasing start time inputs of 0 to 16 and increase for the remaining inputs.

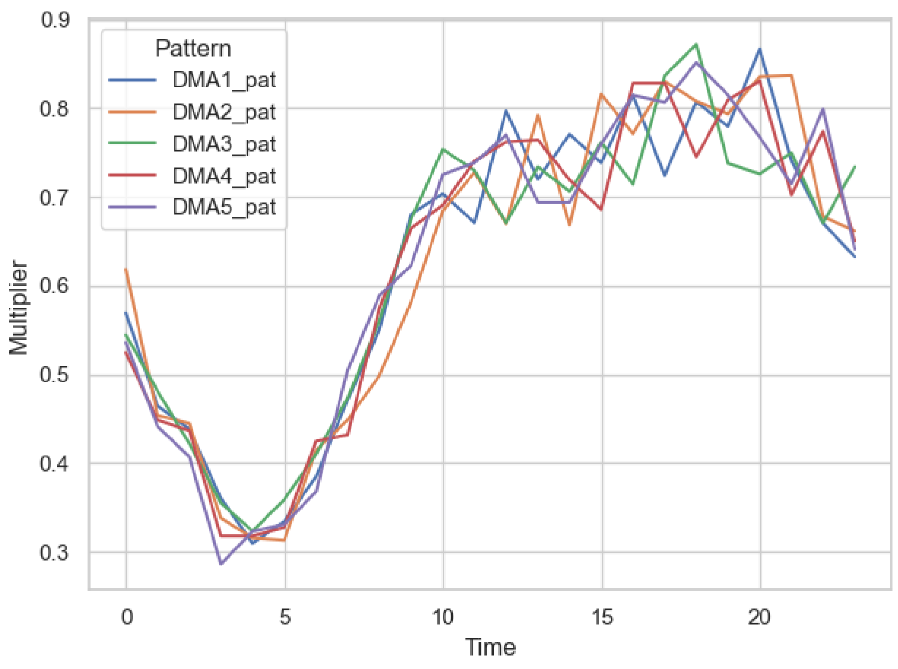


Figure 5.1: Demand multipliers of Ctown as a function of time.

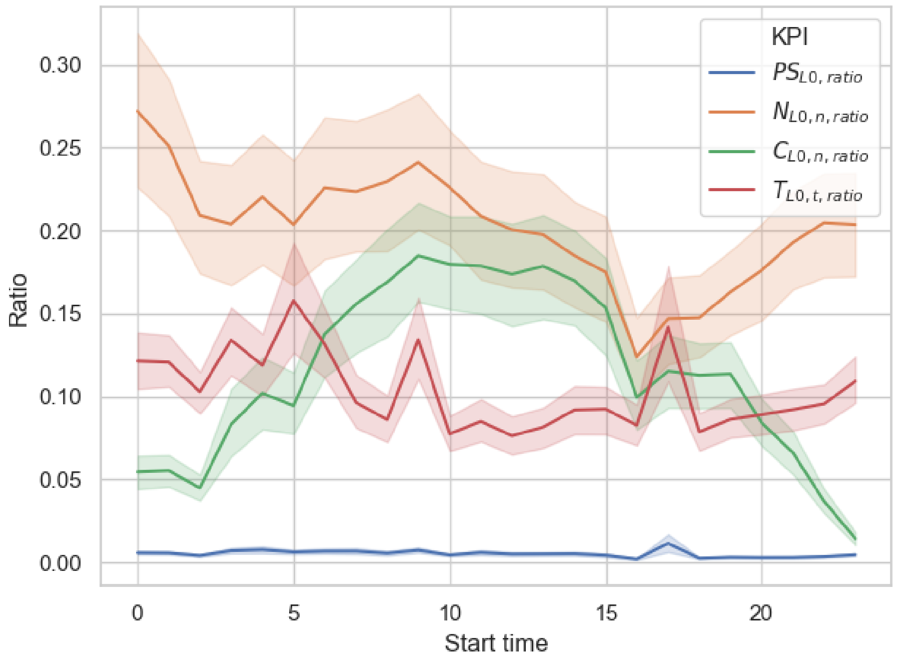


Figure 5.2: Ctown magnitude of failure as a function of start time input.

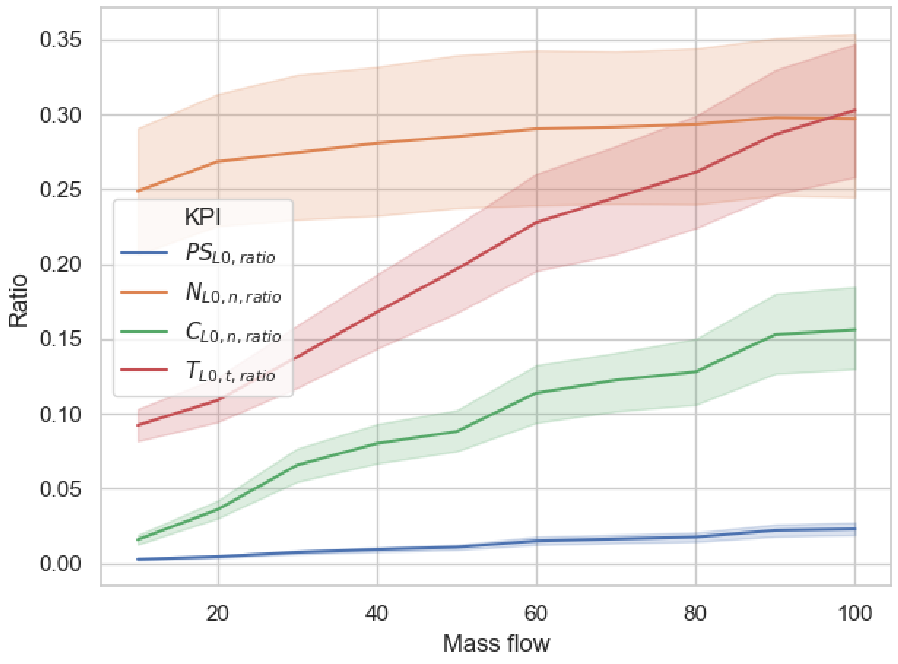


Figure 5.3: Ctown magnitude of failure as a function of mass flow input.

The customer dimension is a function of demand pattern and baseline demand values (see Figure 5.1). As such, the maximum $C_{L_0,n,ratio}$ value occurs with a start time input of 9 hours, while the minimum number of customers affected occurs at the lower and upper input boundaries. E.i. an apparent trend is observed between demand patterns and the failure of the customer dimension. Although there are no prominent trends concerning the spatial and temporal dimensions, the $PS_{L_0,n}$ and $T_{L_0,t}$ values are to some extent affected by the input uncertainty of the start time.

5.1.2 Mass flow of a substance intrusion

Mass flow input (g/s) is a fixed mass value of a substance introduced to the system per unit of time. E.g., with a fixed duration of 1 hour, a mass flow input of 10 g/s accounts for a total of 36 kg substance injected into network, while a mass flow of 20 g/s accounts for 72 kg substance. The mass value mixes with the water flow of in going pipes to form a substance concentration associated with the outflow from the source node [61]. Thus, a node with high flow rates at its inlet accounts for lower substance concentrations than a source node with the same mass flow value and lower in-going water flow rates.

The start time and duration inputs are kept at their baseline value, while the mass flow is increased by incremental steps from 10 to 100 (see Table 5.1). The steps

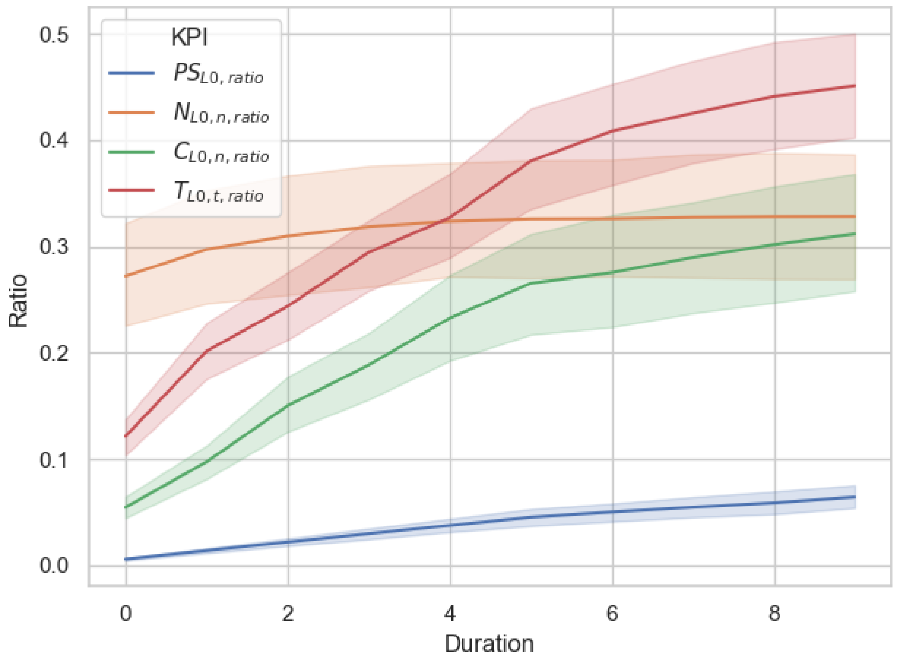


Figure 5.4: Ctown magnitude of failure as a function of duration input.

are repeated such that all junctions in DMA 1 of Ctown are applied as a source node. The resulting failure can be observed in Figure 5.3 as a function of stress testing events with various mass flows.

It is observed that failure metrics are strictly increasing with a progressively increasing input value of mass flow until a certain threshold is reached at a system scale (see Equation (3.22)). The progressively increasing disturbance is communicated as a monotonic increasing relationship between mass rate input and the global failure indicators (see Figure 5.3) at the scale of the DMA1. The global failure metrics of $PS_{L0,ratio}$, $C_{L0,n,ratio}$, and $T_{L0,t,ratio}$ presents a positive linear increasing relationship to the input configurations, while $N_{L0,n,ratio}$ converge towards a fixed failure value.

5.1.3 Duration of a substance intrusion

Duration input is the time period in which a mass booster is active. The duration of a substance intrusion inherent properties of both temporal and magnitude aspects of a substance intrusion. As plain as it sounds, the duration affects the total mass entering the system and the timing of a substance intrusion.

The input values of mass flow and start times remain fixed, while duration is incrementally increased from 1 to 9 according to the OAT procedure (see Table 5.1).

E.g., at the duration input of 4 hours, a total of 360 kg substance is injected into the network. This is equivalent to a mass flow of 100 g/s in the previous section (see Section 5.1.2). The upper bound for the duration input, a total of 8100 kg is injected into the system over a period of 9 hours. The simulation of substance intrusion is repeated with incremental steps such that all junction in DMA1 of Ctown is tested individually as a source node.

Figure 5.4 presents the results from the simulation procedure for the four failure dimensions with respect to magnitude. Similarly to mass flow, the duration forms a strain-strain relationship with the failure metrics. The ratio of $N_{L0,n,ratio}$ converges towards a maximum value of 0.33. As a result of increasing the time span of substance intrusion, an increase of 0.03 is observed from the convergence value of the mass flow function. The tendency of convergence is also observed by $C_{L_0,n,ratio}$ and $T_{L_0,t,ratio}$ as they reach their maximum value of 0.31 and 0.45 at the upper duration boundaries.

5.2 Variability of source nodes

The aforementioned OAT procedure involves a comprehensive sensitivity analysis of the control variables for a range of different source nodes. As a result, large quantities of failure data are produced describing the impact of substance intrusion at different nodes within DMA 1 in Ctown. The failure metrics data is grouped by its source point, and the global failure metrics are calculated with respect to each junction as a source point. The resulting matrices provide information on the impact of substance intrusion at the system nodes with respect to service, temporal, spatial, and customer aspects of the operation.

Figure 5.5, Figure 5.6, Figure 5.7, Figure 5.8 maps the global failure metrics associated with substance intrusion at different source points in DMA1 of Ctown. The four graphs reflect the magnitude of failure for the service (see Figure 5.5), the spatial (see Figure 5.6), the customer (see Figure 5.7), and the temporal (see Figure 5.8) dimension. The scales indicate the observed global failure metrics as a result of a substance intrusion at the various junctions in the sub-graph.

The extent of the spatial and customer dimension is mapped in figure 5.6 and 5.7 respectively. Substance intrusion close to the system inlet (e.i. lower right of the graph plot) have the highest $N_{L0,n,ratio}$ and $C_{L0,n,ratio}$ impact. The impact reflected by the global failure metrics decrease as the source point is shifted further away from the reservoir. It is also noted that the placement of the tank (upper left corner of the graph) has an influence on customer dimension compared to the spatial dimension.

Figure 5.8 displays the extent of the temporal dimension at the aforementioned junctions. The maximum $T_{L0,t,ratio}$ is observed at junction J12 as a source point. Junction J12 is a part of a cluster of high-impact source points close to the outlet

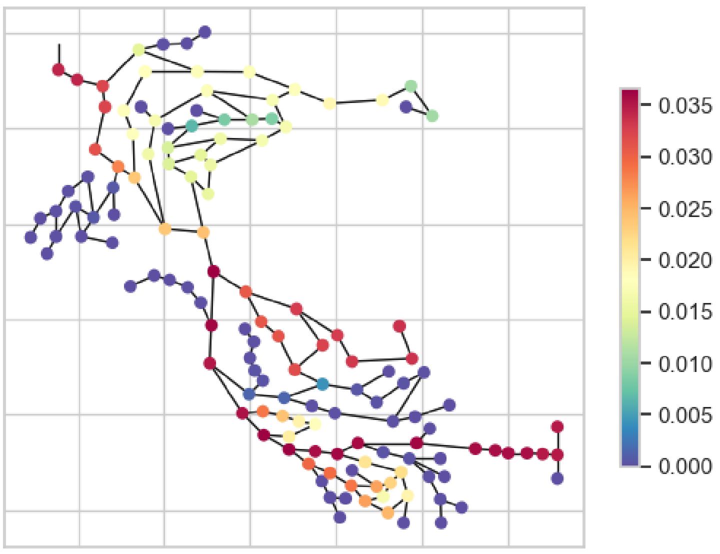


Figure 5.5: Consequence map scaled by $PS_{L0,ratio}$ at various source points.

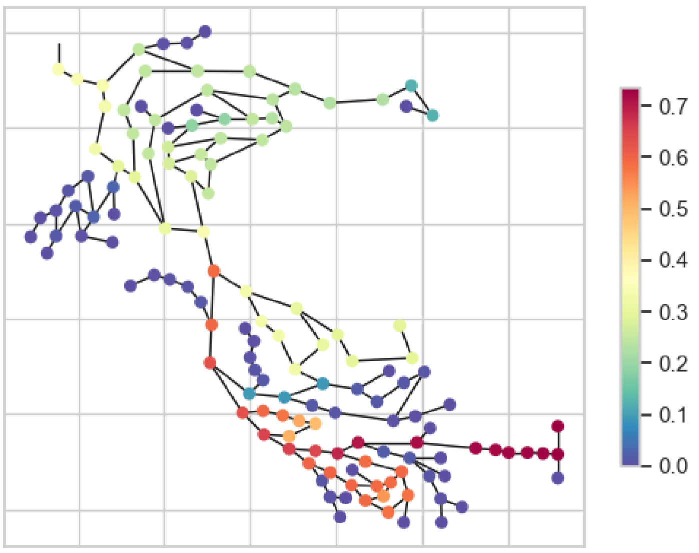


Figure 5.6: Consequence map scaled by $N_{L0,n,ratio}$ at various source points.

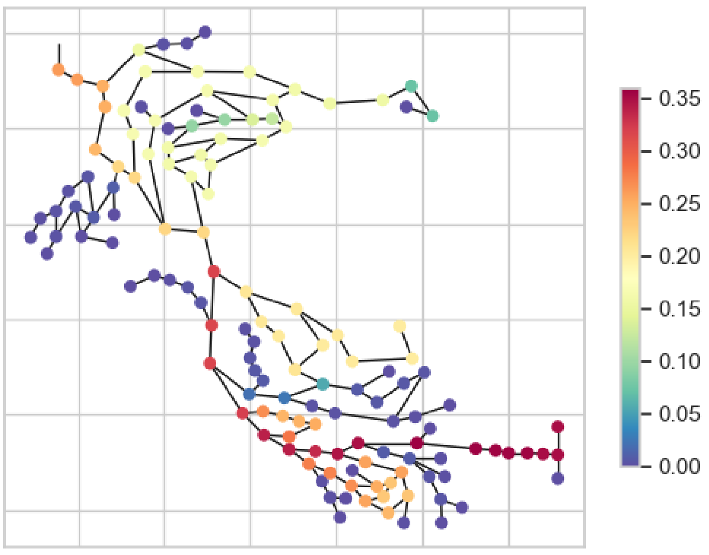


Figure 5.7: Consequence map scaled by $C_{L0,n,ratio}$ at various source points.

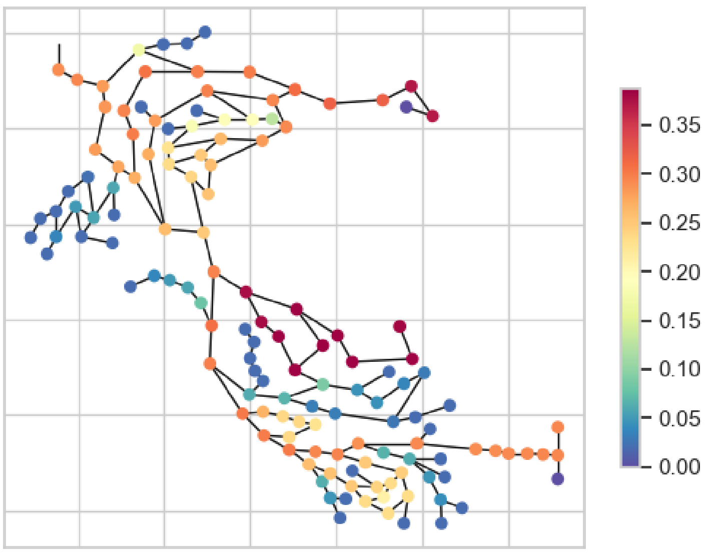


Figure 5.8: Consequence map scaled by $T_{L0,t,ratio}$ at various source points.

to DMA3 and DMA4 (lower right outlet). However, junction J12 is not connected to any of the primary feeders. High impact nodes to the temporal dimension are also observed by the inlet to DMA2 and DMA5.

It is observed that the highly looped cluster of nodes (e.g., the bottom cluster and cluster at the upper right corner) produced higher global failure metrics compared to the branched subsystems (e.g. left clusters of nodes). In terms of topological attributes, the results may indicate preferable network designs of substructures as some part of the system tends to have a higher impact on the system performance.

5.3 Discussion

A quality-related stress testing event is defined by the control variables at source nodes and the geographical distribution of source nodes. The input uncertainty analysis characterises the randomness from each source of uncertainty based on its manifestation to the magnitude, temporal, and spatial aspects of a substance intrusion.

- The magnitude aspect of a substance intrusion relates to the total mass of the substance entering the network during the simulation period. The mass flow input is a direct translation of the magnitude aspect, as the temporal and spatial aspects of the stress testing event are unaffected by any changes of its input value. In addition to mass flow, another required property of magnitude is the duration of substance intrusion. Yet, it is observed that duration is also dependent on uncertainties related to temporal aspects. The magnitude aspect of the failure form a ranked relationship to the performance of the systems. This implies that a positive direction between the ordered values of mass flow and failure to form a stress-strain relationship. Performance failure is recorded as the mass flow input reaches a certain threshold. Increments of the mass flow input above these thresholds result in increased failure up to a maximum failure value.
- The temporal aspect of a substance intrusion relates to the timing of the appearance of a substance in relation to operational and environmental parameters of the water distribution system. The start time input reflects the first appearance of a substance intrusion, while duration input addresses the intrusion period. The temporal variables and their effect on system performance are influenced by a high degree of randomness due to the complexity of water distribution systems. In contrast to the magnitude input variables, temporal input variables present a seemingly unpredictable relationship with the loss of performance.
- The spatial aspect of a substance intrusion relates to variability of system failure in regard to the characteristics of a source point. Source points close to the inlet, or source points connected to feeder mains tend to yield greater failures compared to source points close to or at system end nodes.

The input uncertainty analysis demonstrates that temporal aspects of a substance intrusion affect the system performance. Its degree of variability may differ between source points and failure dimensions, and the randomness of its output is partially dependent on the complexity of the system.

In terms of validity of the global performance metrics (see Equation (3.22)), a stochastic modeling procedure should address magnitude, temporal, and spatial aspects in order to capture the full extent of a substance intrusion and all possible failure states of a water distribution system. This will ensure to include spatial variability and temporal uncertainty while capturing different stress levels with an increasing magnitude of the failure mode.

Chapter 6

Convergence of random failure sequences

This chapter shows the steps of the convergence analysis of the random failure sequences, along with the confidence interval of the produced global failure indicators of the Ctown benchmark model.

6.1 Random failure sequences

A random failure sequence can be applied to any type of system malfunction (including substance intrusion) for any specified period in the simulation [32, 35, 65]. A key strength of the random failure sequence is the shift in objective from the threats themselves to explicit consideration of system performance when exposed to a large set of stress testing events [29]. Thus, its result can be used to evaluate the impact of substance intrusion on various stress levels irrespective of their occurrence probability.

Random failure sequences involve a random cumulative selection of nodes configured as the source point for substance intrusions. As the number of nodes selected as a source point reflects the magnitude aspect of a substance entering the system, it may be used to indicate the level of stress inflicted on the water distribution model. The main steps involved in the simulation procedure include [32]:

1. A simulation to determine the performance of the network under normal conditions.
2. A random selection of i number of nodes and configured as a source point for substance intrusion.
3. The behavior of the network configured with i source points is simulated using the *Epanet Simulator* of WNTR, and the failure metrics are quantified based on the resulting behaviour (see Section 3.1.3).

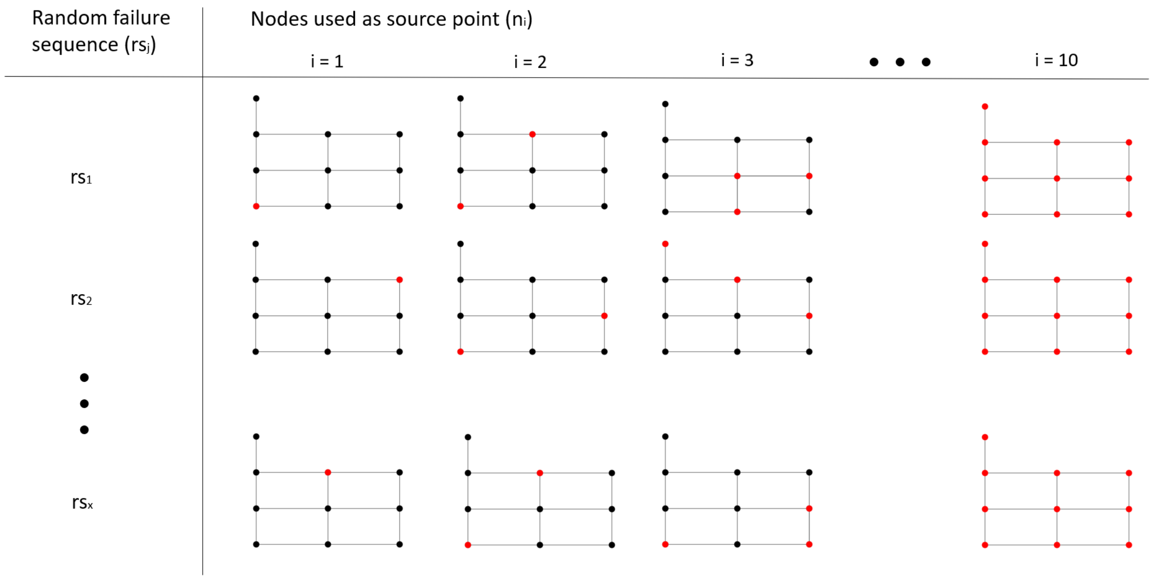

Random failure sequence (rsj)
Nodes used as source point (ni)
i = 1
i = 2
i = 3
i = 10
rs1
rs2
rsx

Figure 6.1: Random failure sequence based on illustrations of Diao *et al.* [35].

4. The procedure in (2)-(3) is repeated for $n_i : i = 1, 2, 3..., N$ number of source points.
5. The procedure (2)–(4) is repeated x number of times in order to determine the global failure indicators (see Equation (3.22)).

Figure 6.1 presents the described modeling framework for cumulative random failure sequence for substance intrusion of a simplified water distribution network (e.i. 10 nodes and 13 links). A random failure sequence rs_j includes simulated stress testing events with an increasing number of source points, as shown in Figure 6.1. The number of source points is denoted as n_i and derived as the percentage of total number of nodes in the network N, such that $i \in [1, 2, 3, ..., 10]$ and $n_i = i/10 \cdot N$. This study adapts the random failure sequence of Mugume *et al.* [32] as a multivariate approach in order to address both temporal and spatial variability of the substance intrusion. As such, start time inputs and source point input are randomly selected. The process is repeated for x number of random failure sequences.

6.2 Control variables

As for the control variables, duration and mass flow input is configured as a fixed value of 1 hour and 1 g/s, respectively. As such, each of the selected source points contributes with a maximum total mass of 3.6 kg during the simulation period. In order to capture the operational and environmental variability of the water distribution system during a day, the start time input is randomly selected between the range of 0 to 23 hours from the simulation start. The start time input range is defined by an incremental step of 1 hour such that each source node can in total have 24 different control variable configurations.

A random failure sequence includes a set of scenarios with an increasing number of source points (n_i). The number of source points translates to the total mass of substance injected into the system. As the input uncertainty analysis demonstrates that tends global failure indicators converge at higher mass inputs, the maximum number of source points is limited to a maximum of 300 kg of substance. Exposing Ctown to a total mass of 300 kg substance requires a total of 83 source points with the given set of control variables. In order to level the size difference of the benchmark models and the potential impact of a random failure sequence, the maximum number of nodes is converted to a percentage of the total number of nodes in the network. Consequently, the range of potential source points is defined as $n_i : i \in [0.01N, 0.02N, 0.03N, ..., 0.2N]$, such that the maximum substance mass of Net3 and Trondheim is limited to 43.2 and 7732.8 kg, respectively.

As observed by the input uncertainty analysis for mass flow, the steepest slope occurs at lower stress levels (e.i., the total mass of substance entering the system). By the limited number of source points, the random failure sequence captures the most sensitive range of mass input while minimising the total number of required simulations.

6.3 Maximum number of stress-testing events

In order to capture the full extent of a failure mode, every possible combination of source node should, in principle, be considered [30]. As such, a random failure sequence of a network with a total of nodes (N) includes simulation with $n_i : i \in [1, 2, 3, ..., N]$ source nodes. By considering that each system node is either a source point or not a source point, the number of unique combinations of source nodes (n_i) can be calculated by Equation (6.1) [32].

$$C(N, n_i) = \frac{N!}{(N - n_i)!n_i!} \tag{6.1}$$

Using the benchmark model of Ctown with 388 junctions as an example, the total number of possible combinations involving substance intrusion of a single node ($n_i = 1$) is 388. The total number of combinations involving two ($n_i = 2$), three ($n_i = 3$), four ($n_i = 4$) source nodes would be 75078, 9660036 and 929778465 respectively. The highest number of possible combinations occurs at the mid-point ($n_i = N/2$), such that the analysis of the distribution of the source point combinations at each stress level indicates a normal distribution [32]. The total number of simulations required to include all possible combinations in the entire solution space is calculated as the sum of $C(N, n_i) : i = [1, 2, 3, ..., N]$, and accounts for $6.5E+116$ failure scenarios. Simulating such a large number of failure scenarios is outside the scope of this study as it would require huge computational resources.

6.4 Minimum number of stress-testing events

The minimum number of stress-testing events is a limited number of random failure sequences. A bootstrapping procedure is applied to determine the minimum number of random failure sequences (rs_x) that should be analysed in order to achieve consistent resilience results while covering as many failure states as possible. In doing so, a convergence analysis [32] is carried out with the following steps:

1. A simulation of 500 random failure sequences is carried out for Ctown (e.i., 10000 stress-testing events).
2. A general bootstrap technique [5] is applied as a resampling approach of the random failure sequences. The technique involves the sampling of residuals from the original dataset produced in step (1) with the following steps:
   a. Data pairs (F_i, n_i) are sampled from the original dataset to form x number of random failure sequences $rs_j = [F_{0.01N}, F_{0.02N}, F_{0.03N}, ...F_{0.2N}]$ that corresponds to the defined stress levels of $n_i : i = [0.01N, 0.02N, 0.03N, ..., 0.2N]$.
   b. Step (a) is repeated for $x \in [10, 25, 50, 75, 100]$ and the mean rs is calculated for stress level and in within group x.
   c. Step (b) is repeated G times to produce groups categorised by x such that the variance of rs of each group can be determined.
3. The global failure indicators (see Equation (3.22)) is calculated from the residual sampling of rs. The selected number of random failure sequences rs_x, and are used to determine the confidential interval of the global failure indicators.

The performance indicator $F_{x,i,j}$ is a part of the multidimensional matrix generated by the bootstrapping procedure of the convergence analysis. The number of bootstrapped sequences of rs_x is indicated by N such that $n \in [1, 2, 3, ..., 100]$, while i refers to the stress level, and x is the number of rs_j used. As such, the variance within each group of rs_x can be calculated with Equation (6.2).

$$Var(x, n_i) = \frac{\sum_{n=1}^{N} (F_{x,n_i,n} - \overline{F_{x,n_i}})^2}{N} \tag{6.2}$$

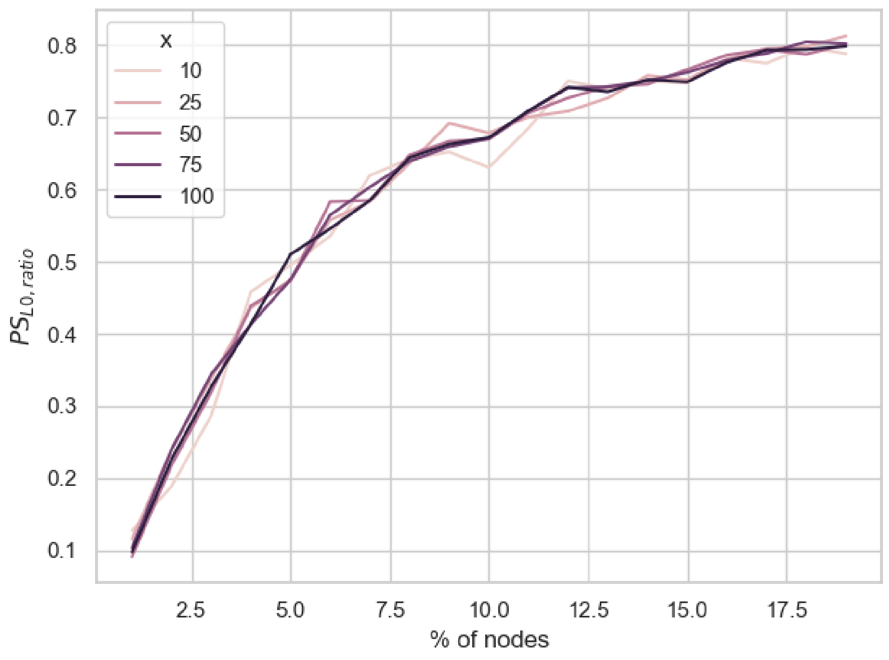


Figure 6.2: Global failure indicator derived from random failure sequences rs_x.

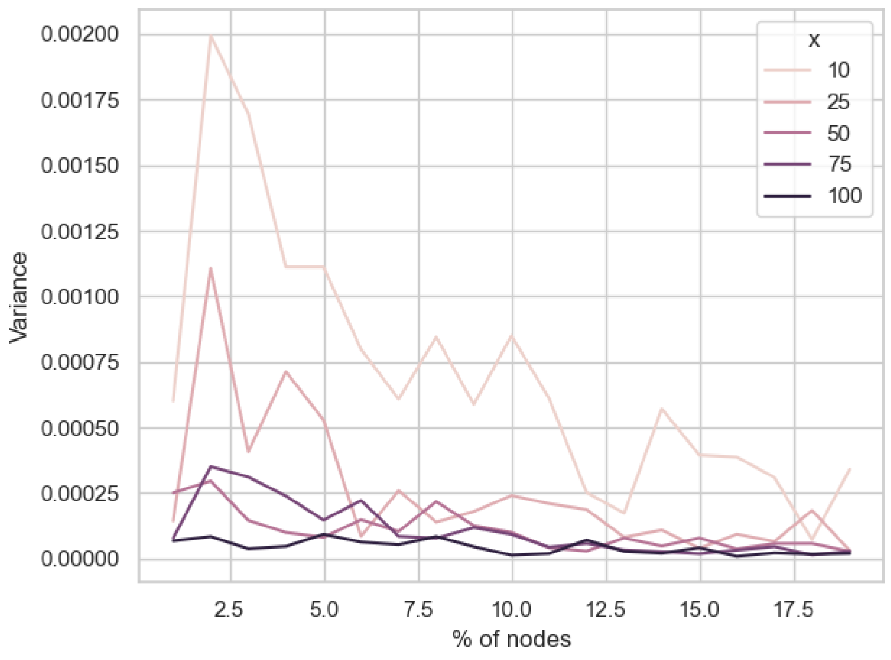


Figure 6.3: Variance of $PS_{L0,ratio}$ derived from random failure sequences rs_x.

Table 6.1: Convergence result of global failure metrics.

KPI	Sample mean	Maximum	Minimum	Confidence interval (95%)
$PS_{L0,ratio}$	0.6085	0.6144	0.6040	$\pm 3.19E-04$
$N_{L0,n,ratio}$	0.6476	0.6517	0.6425	$\pm 3.91E-04$
$C_{L0,n,ratio}$	0.6661	0.6706	0.6617	$\pm 3.61E-04$
$PS_{L0,PAR}$	0.0150	0.0151	0.0148	$\pm 9.37E-06$
$C_{L0,PAR}$	0.9216	0.9243	0.9182	$\pm 2.32E-04$
$N_{L0,PAR}$	0.9655	0.9678	0.9628	$\pm 1.71E-04$
$PS_{L0,peak}$	0.1457	0.1471	0.1447	$\pm 9.03E-05$
$N_{L0,peak}$	243.6535	245.2400	241.4795	$\pm 1.45E-01$
$C_{L0,peak}$	222.9990	225.2489	221.1574	$\pm 1.25E-01$
$PS_{L0,TEP}$	54.1565	55.3089	53.0932	$\pm 8.05E-02$
$N_{L0,TEP}$	46.5191	47.8389	45.4432	$\pm 9.39E-02$
$C_{L0,TEP}$	19.5193	19.5600	19.4737	$\pm 4.50E-03$

The results presented in Figure 6.2 and Figure 6.3 indicate that the increase in the number of random failure sequences results in convergence of the performance indicators. Using $PS_{L0,ratio}$ as an example, Figure 6.2 present the stress-strain relationship formed by a random failure sequence with an increasing number of source points along the x-axis. The lines are the output of step (2a) and are categorised by $x \in [10, 25, 50, 75, 100]$.

Figure 6.3 presents the variance of each stress level for 10, 25, 50, 75, and 100 random failure sequences. The maximum variance is observed at the stress level of 2% with a value of 0.002 for 10 random failure sequences. The maximum value is further reduced to 0.0012, 0.0003, and 0.0028 by considering 25, 50, and 75 random failure sequences, respectively. It is also observed that the variance decreases with an increasing stress level.

6.5 Reliability of global failure indicators

Based on the variance evaluations, a minimum of 100 random failure sequences is adapted to measure the global failure metrics. Equation (3.22) is used to calculate the global failure from the dataset produced by the bootstrapping procedure. This provides a dataset ($N = 100$) of global failure metrics. Table 6.1 presents a summary of the global failure results with mean, maximum, and minimum values from the dataset.

Table 6.1 includes the sample mean, the maximum, and the minimum values, along with the confidential interval of 95% ($t = 0.95$) of the global performance indicators. E.g., there is a 95% chance that the true sample means of $PS_{L0,ratio}$ is within the range of $\pm 3.19E-04$ from the estimated sample mean.

6.6 Discussion

It must be noted that the random failure sequences differ from the ones applied in the study [32, 35, 65], as they do not include the full spectrum of possible link failures. The maximum number of nodes applied as a source point is limited to 20% of the total number of nodes in the system. Rather than include all possible levels of stress, the adapted random failure sequences capture the most sensitive area of the stress-strain curve. This limits the required computational resources while capturing a critical range of stress-testing events. Furthermore, the level of stress may also be regulated by the configured mass flows and duration of the substance intrusions. As the magnitude aspect of the substance intrusion is addressed by the number of source points, input values of mass flow and duration remain fixed (e.i. mass flow and duration are configured to 1 g/s and 1 hour, respectively).

Although efforts to capture different aspects of substance intrusions, the simulation period of 96 hours does not capture the full extent of the loss function of the continuity dimension (fig. 2.2); consequently, the random failure sequences yield a constant value of 1 for the global failure metric of $T_{L0,t,ratio}$ at all levels of stress. As the failure metrics of average propagation address the relationship of loss function properties to the temporal dimension, the result presents a scaled version of original failure metrics. Thus, average propagation failure metrics are excluded from the correlation analysis, along with the magnitude of the continuity dimension.

The temporal variability is another aspect of a substance intrusion included in the random failure sequences. The input uncertainty analysis reflects the periodic changes of environmental variables and their effect on certain failure metrics on the scale of a service area. Certain features of the hydraulic behavior may present hidden quality-related vulnerabilities in the system in order to address the uncertainty of the temporal aspect. It is included as a part of the random failure sequences by a random selection of the start time inputs of the source points.

A key feature of the random failure sequences is the random selection of source points in order to address spatial variability. As demonstrated in the input uncertainty analysis, the impact of a substance intrusion is primarily dependent on the selected source point. Generally, source points with a large number of downstream nodes present a higher failure than nodes close to or at a system endpoint. Yet, this may be dependent on the magnitude of the failure, as nodes connected to large diameter pipes may dilute the substance concentration below critical thresholds. The random failure sequences is an ideal stochastic simulation procedure, as regardless of failure probability, presents a selection of source point or combination of source points that address representative samples of the resulting impact population.

In terms of convergence of random failure sequences, bootstrapping presents a simple and straightforward way to estimate variance and confidence intervals for complex estimators of the distribution [18]. Additionally, bootstrapping is a convenient method that avoids the unnecessarily computational cost of repeating the experiment to get other groups of sample data. As such, a total of 10000 stress-testing events was simulated (e.i. 500 random failure sequences) to sample groups of 10, 25, 50, 75, and 100 random failure sequences. Although bootstrapping is asymptotically consistent, it does not provide general finite-sample guarantees [18]. The result of a higher number of random failure sequences may be influenced by the representative sample. However, the convergence of magnitude through random failure sequences of increasing sizes indicates a decrease in variance. The presented result corresponds with the finding of Mugume *et al.* [32], evaluating convergence of an accumulative number of pipe failures in an urban drainage system.

A total of 100 random failure sequences is selected as the desired number of samples in the correlation analysis. This implies that global failure metrics of each benchmark models and the presented network variants are derived from a total of 2000 stress-testing events. The result in Table 6.1 presents the convergence result of global failure metrics and is used to evaluate the reliability of the following global failure results in the correlation analysis.

Chapter 7

Resilience of topological attributes

This chapter presents a characterisation of water distribution models and variants through topological attributes and their performance results exposed to the aforementioned random failure sequences. Following this, correlation analysis and clustering of the network variants are conducted in order to evaluate the relationship between resilience and topological attributes.

7.1 Topological attributes and global failure of benchmark models

The benchmark models include the networks of Net3, Ctown, and Trondheim. The first part is a characterisation of the network design through topological attributes (see Section 3.2.2), followed by the performance result by exposing the networks to random failure sequences (see Chapter 6) measured with global failure metrics (see Equation (3.22)).

7.1.1 Topological attributes

As presented in chapter 4, the models differ substantially from each other as they are of various network sizes and are comprised of different technologies and network designs. The topological attributes presented in chapter 3.2.2 further highlights the structural differences in the network design of the aforementioned benchmark models.

Net3 is the network with the highest number of links compared to the number of nodes in the network ($d = 0.0128$), along with the highest average shortest path lengths ($a = 2.448$), clustering coefficient ($C_c = 0.0429$), and average closeness centrality ($C_u = 0.0319$). Compared to the two other benchmark models, Net3 is a compact network with a uniform distribution of pipes compressed into only one service area.

Networks influenced by a high concentration of terminal nodes centralised around the core of the network generally present high values for the central-point of dominance indicator. Ctown characterised by both looped and branched parts of the network, along with a dominant topological backbone in terms of centrality. Due to the central influence of junction J411 (e.i. source point with highest produced failure with respect to magnitude), Ctown yields the highest a central-point of dominance among the benchmark models ($C_B = 0.5422$).

The network of Trondheim is based on a real water distribution system. The model is characterised by large parts dedicated to the transportation of water and highly detailed treatment and pumping station (e.i. clusters of nodes). Yet, the topological attributes reflect a relative low connectivity with the lowest link density ($d = 0.0001$) and average closeness centrality ($C_u = 0.0002$). The central-point of dominance in Trondheim ($C_B = 0.3848$) is significantly higher than tin Ctown ($C_B = 0.2666$), while Trondheim have a the lowest average shortest path length ($a = 0.0065$) and clustering coefficient ($C_c = 0.01$) of the benchmark models.

7.1.2 Global failure indicators

The benchmark models are exposed to 100 random failure sequences with control variables configured according to the findings of the convergence analysis (see Chapter 6). The resulting behavior of the benchmark models is evaluated with the global failure metrics (see Equation (3.22)) based on the presented three-step methodology of Moraitis *et al.* [56].

The largest spread of the benchmark models is observed through the lens of the service dimension (PS_{ratio,L_0}), where Net3 yields a value of 0.12, Ctown yields a value of 0.61, and Trondheim yields a value of 0.77. The differences between the benchmark models are less distinct in respect to $N_{L0,n,ratio}$, where Net3, Ctown, and Trondheim yield 0.42, 0.65, and 0.85, respectively. The $C_{L0,n,ratio}$ relates to the ratio between the customers affect by low-quality water and the total number of customers in the network and yields a value of 0.20 for Net3, 0.67 for Ctown, and 0.81 for Trondheim as a result of the defined random failure sequence.

The mean peak value of the service dimension (PS_{peak}) is reduced with 51.5% from the moderate to critical service threshold in Net3, while Ctown and Trondheim observe a reduction of 77.8% and 82.5% respectively. A similar change is observed of the global failure indicators of N_{peak} (44.6%, 79.9%, and 87.9%) and C_{peak} (56.3%, 75.0%, and 82.7%). The results indicate that the peak value is most sensitive in the larger Trondheim in respect to changes in service thresholds and/or mass flow changes.

Peak-to-average describes the relationship between the average performance during the simulation period and the peak loss of performance. PAR values close to 1, as observed in Trondheim ($PS_{PAR} = 1$), generally exhibit a more uniform failure propagation profile. While the larger values of 1.7 for PS_{peak,L_0} and 2.4 for PS_{peak,L_2} is observed in Net3 implies a more spike-like failure [56].

The rapidity of a random failure sequence is a measure of the average time from the start of the substance intrusion to the time of the peak failure. With a C_{TEP,L_0} value of 15.4, 19.5, and 16.2 for the benchmark models Net3, Ctown, and Trondheim, respectively, the customer dimension research the peak failure rapidly in comparison to the other dimensions. Derived from the same random failure sequences, Net3 presented an average of 40.6 hours to the peak number of nodes affected, while Ctown and Trondheim presented an average of 46.6 and 88.1 hours, respectively. In respect to the peak value of lower quality water supplied, Net3 presents an average of 36.6 hours, Ctown presents an average of 54.1 hours, and Trondheim presented an average of 86.8 hours.

7.2 Topological attributes and global failure of network variants

A detailed description of the network generation procedure is presented in Section 4.3. The produced network variants include 100 network models generated based on Net3 and 100 network models generated based on the Ctown. As the network generation procedure only removes and adds pipes of low criticality, the main backbone remains fixed in all variants of a benchmark model. The following two parts present a characterisation and the result of the resilience assessment of the aforementioned network variants.

7.2.1 Topological attributes

The link density of the generated Net3 variants forms a normal distribution with a mean value of 0.0125, e.i. the network generation procedure tends to complete with approximately the number of links as the benchmark model of Net3 ($d = 0.0128$). A normal distribution is also observed for the link density of Ctown variants with a mean value of 0.003. It is noted that 98 of 100 Ctown variants are generated with more pipes than the benchmark model. One outlier is observed in the pool of Net3 variants with a link density of 0.0153 (network Net3_62).

The average shortest path length for network variants is normal distributed around the mean value of 0.993 and 0.069 for the Net3 and Ctown models, respectively. The average shortest path length of the benchmark models is substantially larger than the maximum value observed by the variants. The difference between mean a of the Net3 variants and the benchmark model is most extreme, with a factor of approximately 2.5 is observed, while the difference between the Ctown networks is

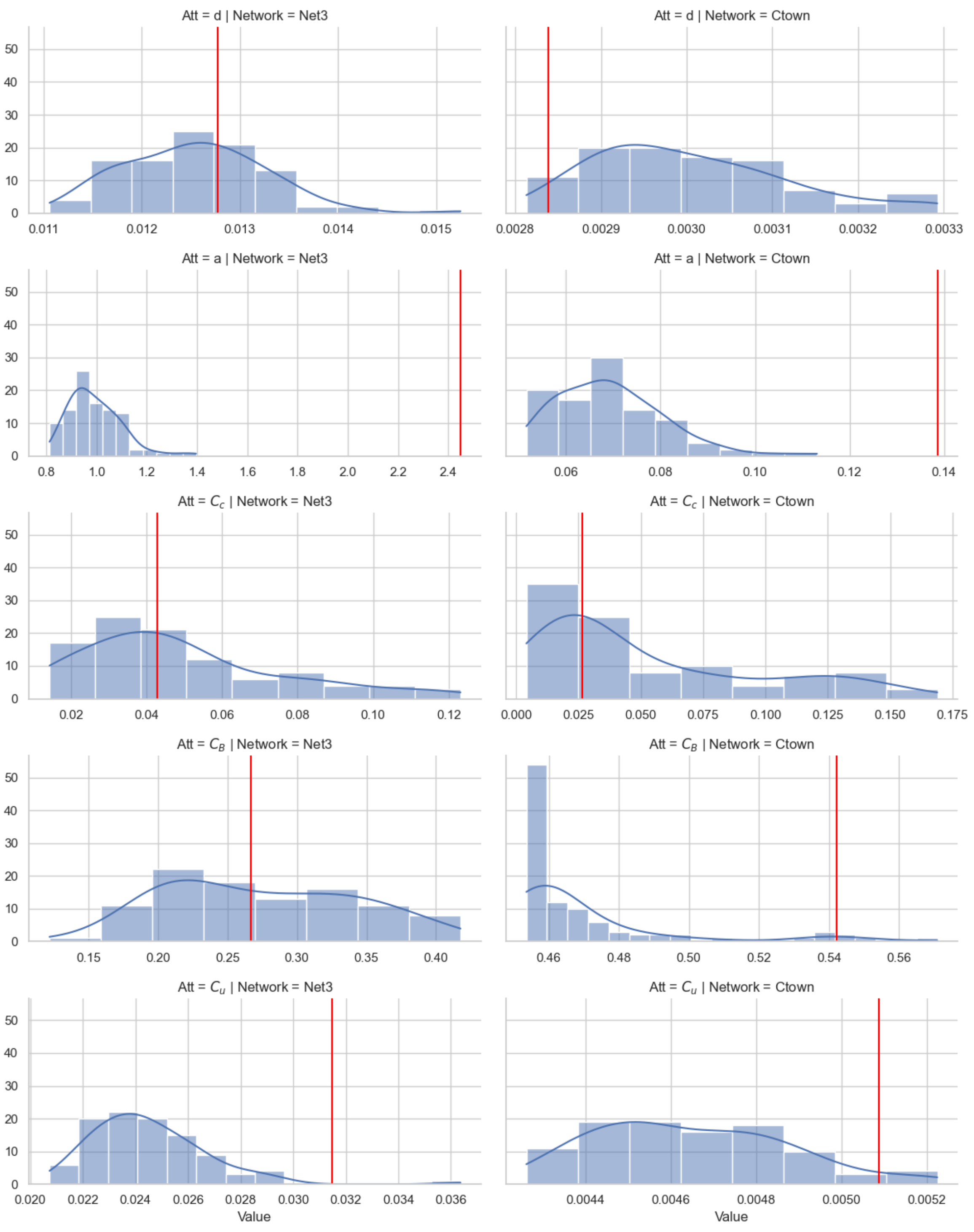


Figure 7.1: Distribution of topological attributes in network variants.

approximated to 2.0. The results may indicate that the generated network models are less efficient in regard to transportation of water than the benchmark models.

The clustering coefficient of Ctown variants includes the two extremes C_c values of 0.004 and 0.169. Yet, a relatively high concentration of variants is observed below the threshold of 0.05, resulting in a mean C_c value of 0.052. A similar shape of the distribution curve of the clustering coefficient is observed for the Net3 variants.

The number of Net3 variants covers a wide range of C_B values with a minimum of 0.122 and a maximum of 0.418. The distribution of observations is apparently evenly distributed between the two extremes. The Ctown variants, on the other hand, present a much narrower range of C_B, limited to the two extremes of 0.454 and 0.570. Out of the 100 network variants of Ctown, 57 networks yields a C_B value below 0.46. Additionally, a group of eight outliers is observed with a C_B value above 0.53 for the Ctown variants.

Average closeness centrality forms a normal distribution for the Net3 variants with a mean value of 0.024. One outlier characterised by a redundancy of pipes ($d = 0.015$) and is deviating from the normal distribution with a C_u value of 0.036. The Ctown variants form a uniform distribution of C_u values within the range from 0.004 to 0.005 in contrast to the distribution of the Net3 variants. The Net3 variant with the highest link density (network Net3_62) also produces an outlier in regard to the average closeness centrality.

7.2.2 Global failure indicators

The 100 network variants of Ctown and 100 network variants of Net3 are all exposed to the random failure sequences, and the global failure is calculated according to the equation for global failure matrices (see Equation (3.22)). Figure 7.2 and Figure 7.3 presents a distribution of the global failure metrics as the number of network variants that yields a performance within the discrete bins. The red vertical line represents the value of the associated benchmark model.

The mean global failure metrics of the Ctown variants in terms of magnitude (service, spatial, and customer dimension), peak (service and spatial dimension), rapidity (spatial dimension) are greater than the benchmark model. However, a significant portion of the network variants yields global failure values below the benchmark model. E.i. some of the networks variants produced by the network generation procedure is more resilient than the benchmark model in terms of quality-related failures. The the global magnitude failure values in terms of the service dimension range from a minimum value of 0.5927 to 0.6592, while the spatial dimension range from 0.6268 to 0.6592 and the customer dimension range from 0.6284 to 0.7060. The confidence interval for the produced failure metrics is presented in Table 6.1. In terms of the magnitude, the confidence interval refers to a 95% probability that the benchmark produces a global failure within the range ±0.0003 of the observed value.

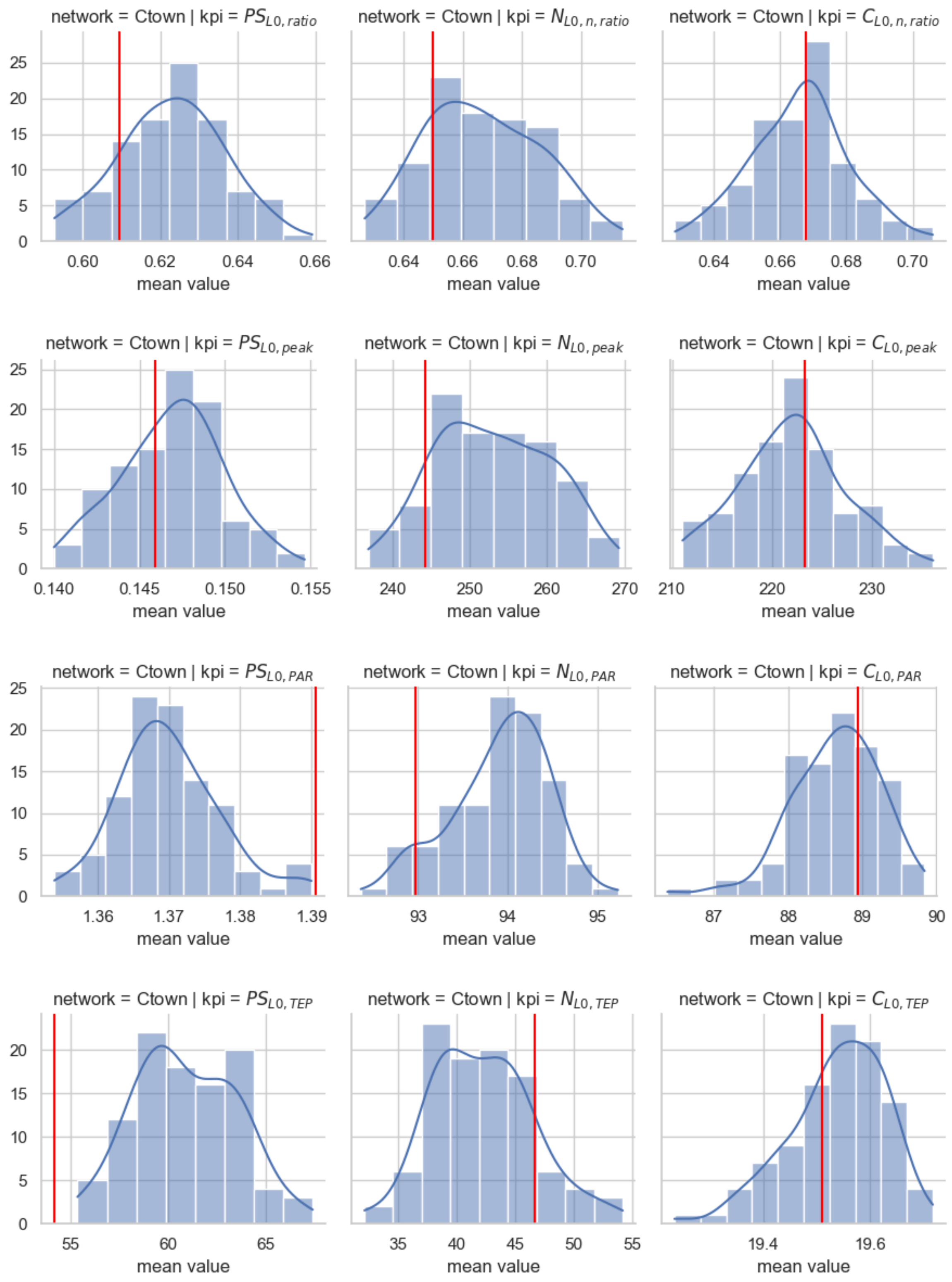


Figure 7.2: Distribution of global performance indicators in Ctown variants.

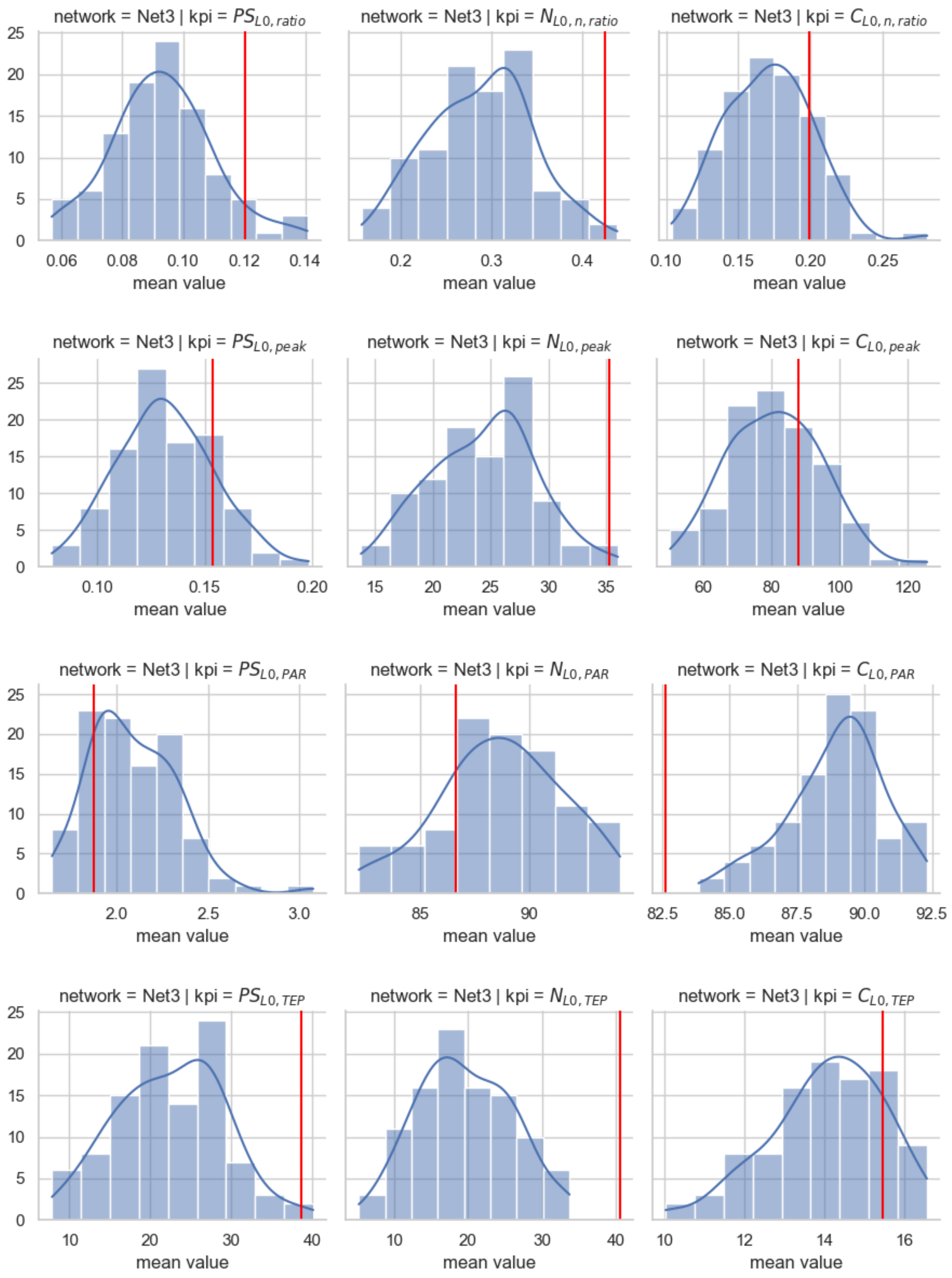


Figure 7.3: Distribution of global performance indicators in Net3 variants.

It is observed that the sample means of global failure values of Net3 variants are below the produced failure values of the benchmark model, with the expectation of crest factor (e.i. PS_{PAR}, N_{PAR}, and C_{PAR}). As a matter of fact, 96 of 100 Net3 variants are more resilient than the benchmark models in terms of $N_{L_0,n,ratio}$. E.i. the result indicate that the performance of certain network variants surpass the benchmark model.

7.3 Correlation analysis of network variants

This section presents scatter plots of global failure metrics to topological attributes for Net3 variants and Ctown variants, respectively, along with the associated Pearson correlation coefficient and p-value (see Section 7.3). The correlation analysis is separated by the networks benchmark affliction due to distinct ranges of observed global failure metrics (see Figure 7.3 and Figure 7.2) and calculated topological attributes (see Figure 7.1).

The figures of this section are related to the magnitude, crest factor, severity, or rapidity of the global failure, where each failure aspect presents a grid of scatter plots. The columns of the figures relate to the five topological attributes addressing connectivity, efficiency, redundancy, robustness, and centrality aspects of the network design. The rows of the figures is a manifestation to service, spatial, and customer dimension. Thus, a scatter plot displays the relationship between a global failure indicator (y-axis) and topological attribute (x-axis) of the presented network variants. The scatter plots are associated with a Pearson correlation coefficient coupled with a p-value that is presented above each grid cell.

7.3.1 Ctown variants

The scatter plots of Figure 7.4, Figure 7.8, Figure 7.6, Figure 7.10 presents scatter plots between global performance and topological attributes vectors of Net3 variants. Link density and average closeness centrality form a moderate positive correlation with the observed vectors of the magnitude of failure (see Figure 7.4) and severity of failure (see Figure 7.8). A weak correlation is observed between average shortest path length and the vectors of the magnitude of failure (see Figure 7.4) and severity of failure (see Figure 7.8), while rapidity of failure (see Figure 7.10) forms a weak positive correlation with link density and average shortest path length. A weak negative correlation is presented between crest factors (see Figure 7.6) and the vectors of link density and average closeness centrality.

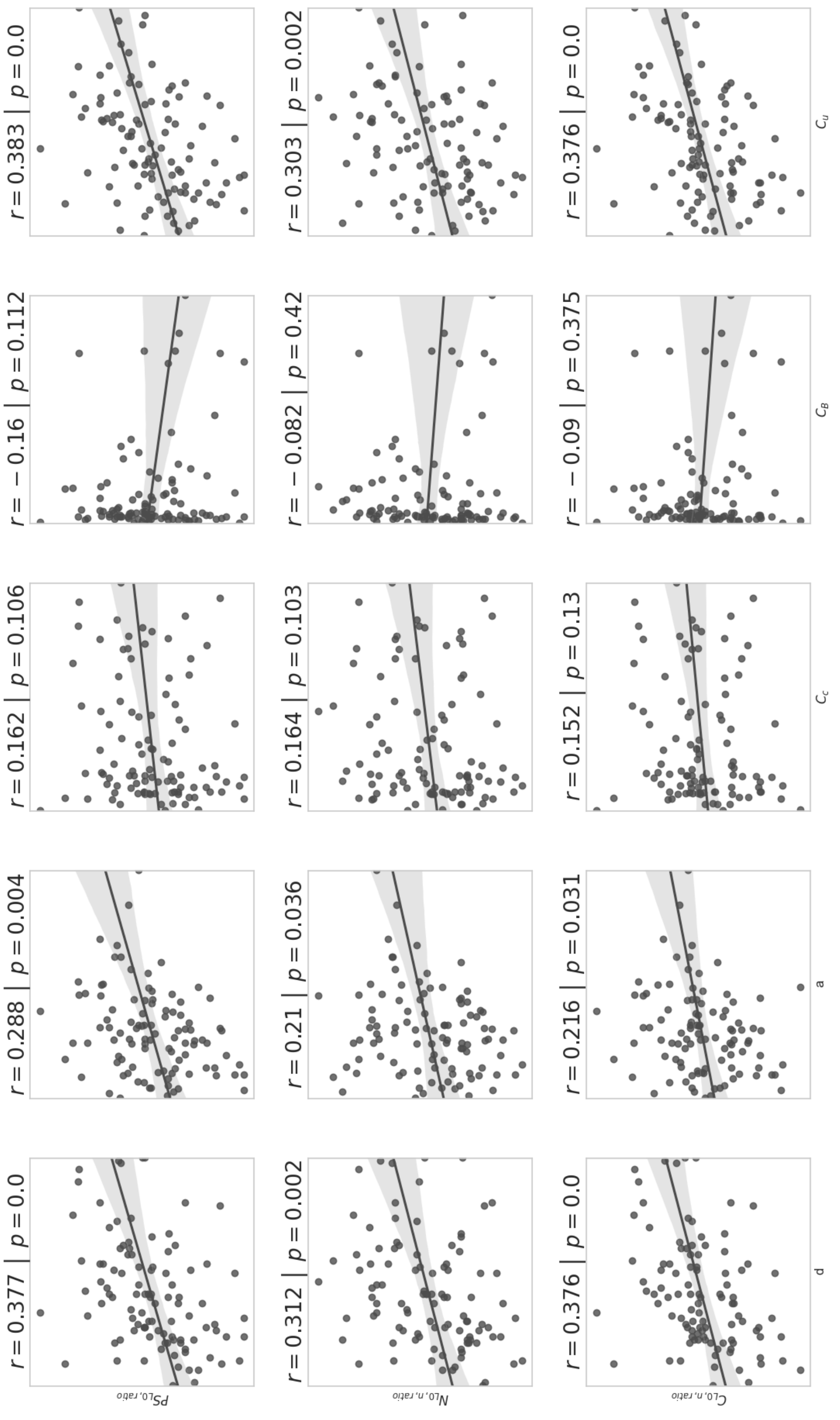


Figure 7.4: Ctown variants global magnitude of failure to topological attributes.

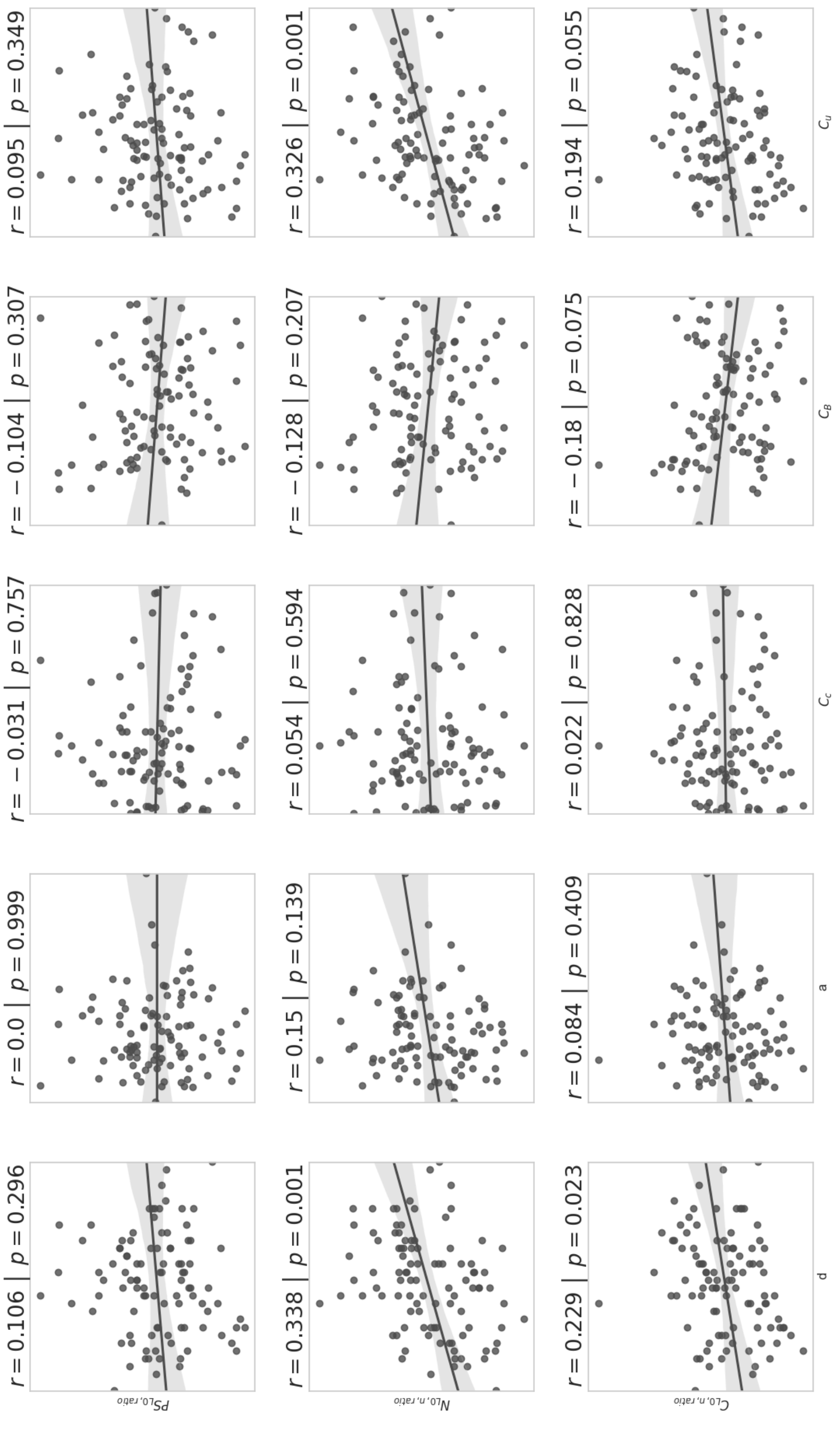


Figure 7.5: Net3 variants global magnitude of failure to topological attributes

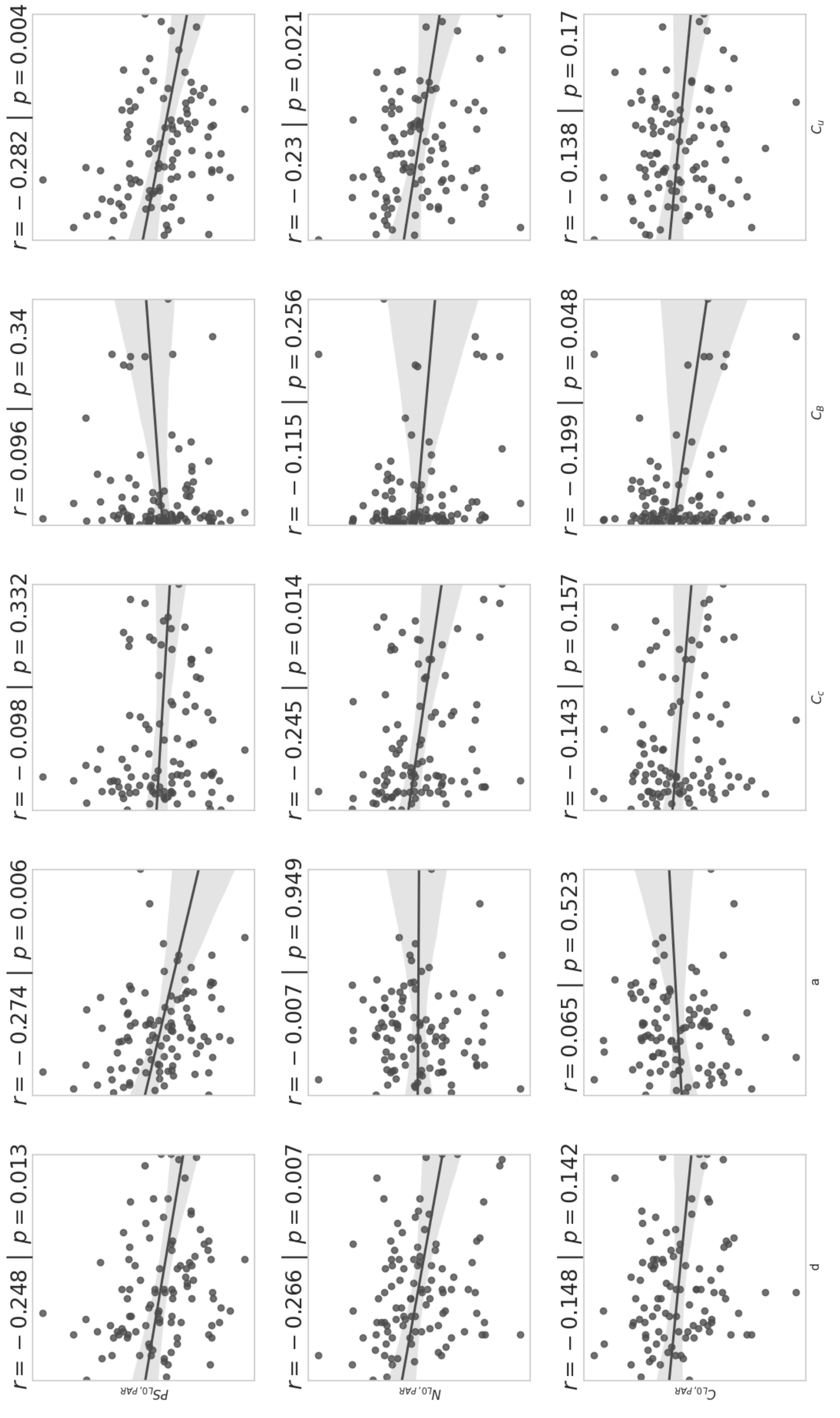


Figure 7.6: Ctown variants global crest factor of failure to topological attributes.

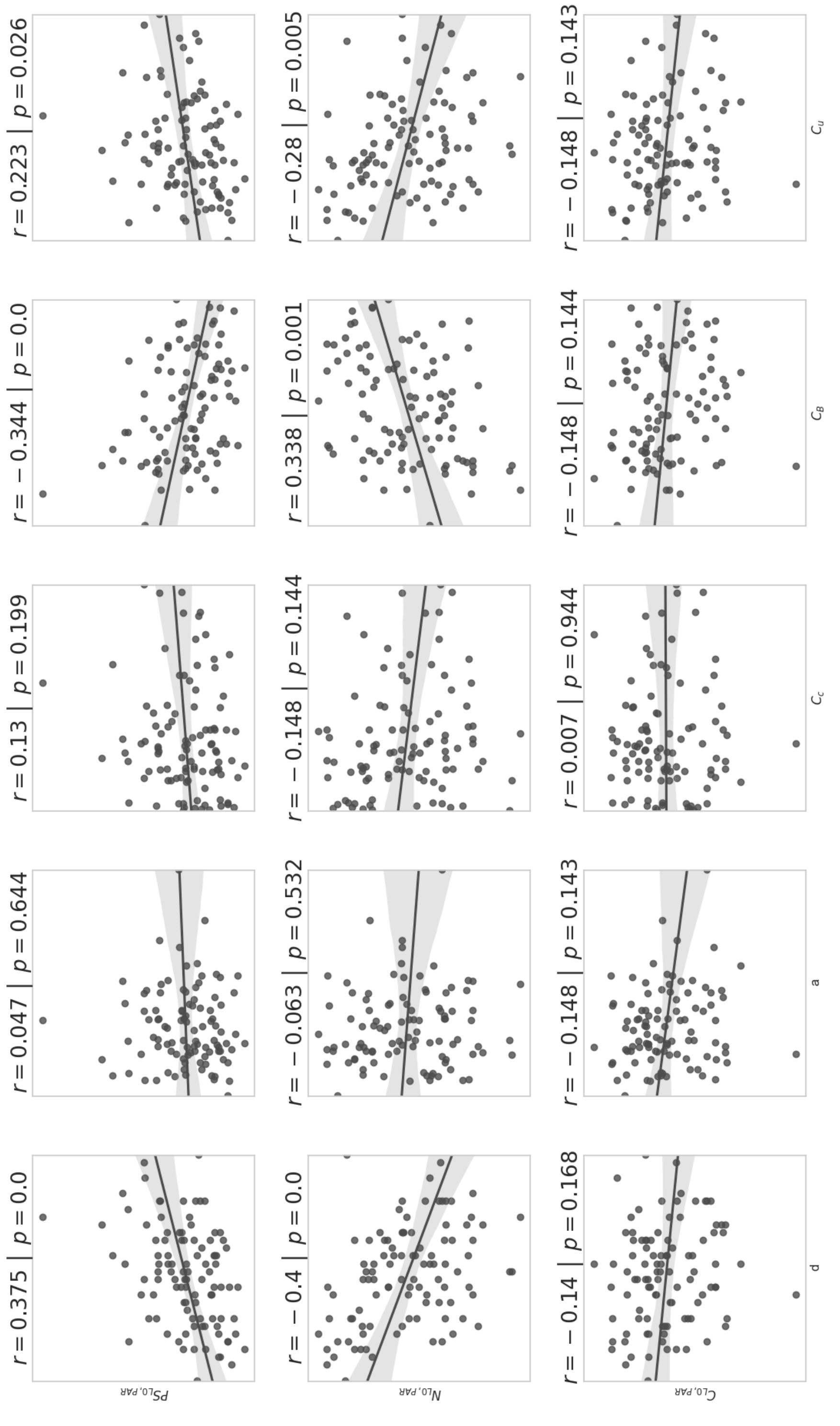


Figure 7.7: Net3 variants global crest factor of failure to topological attributes

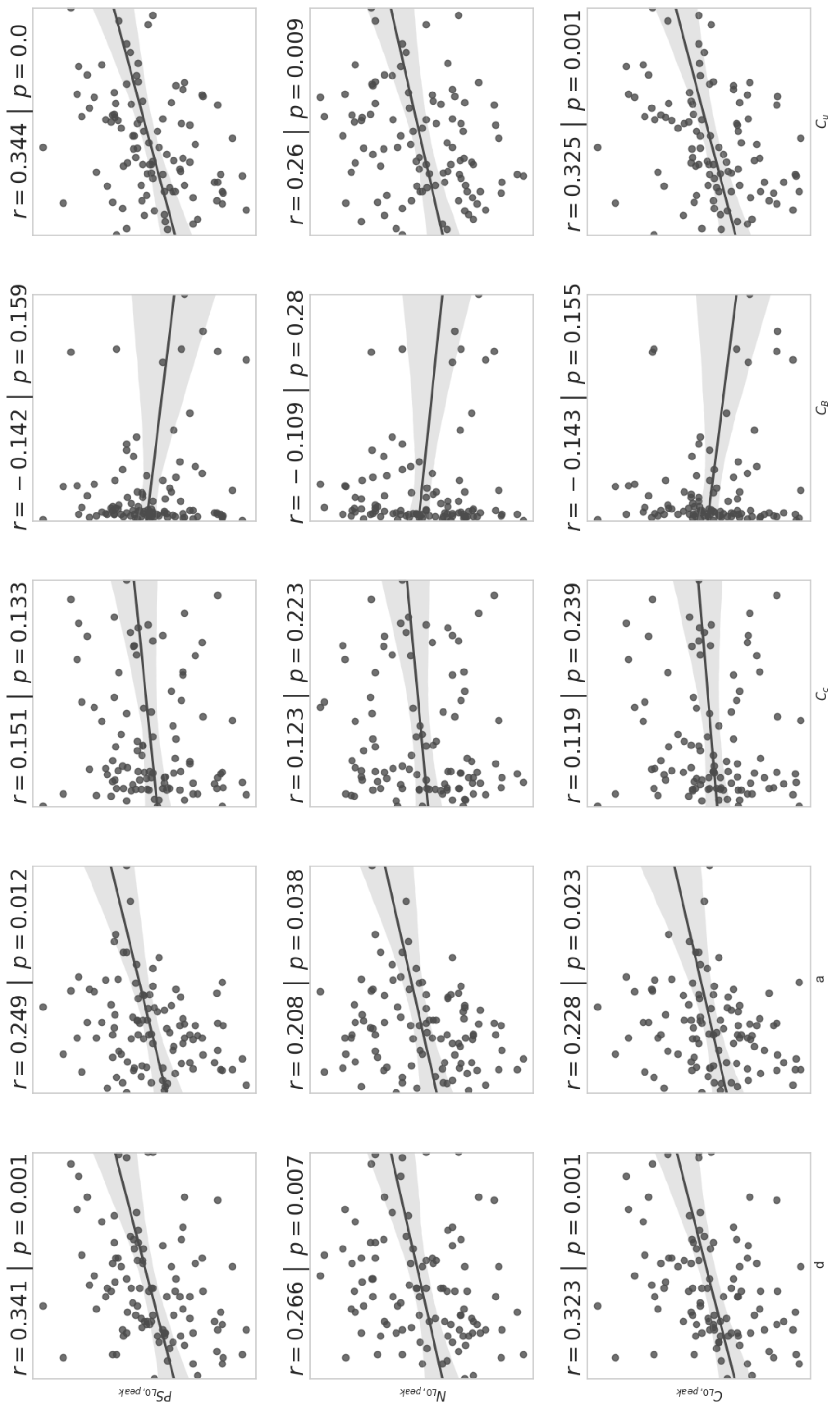


Figure 7.8: Ctown variants global severity of failure to topological attributes.

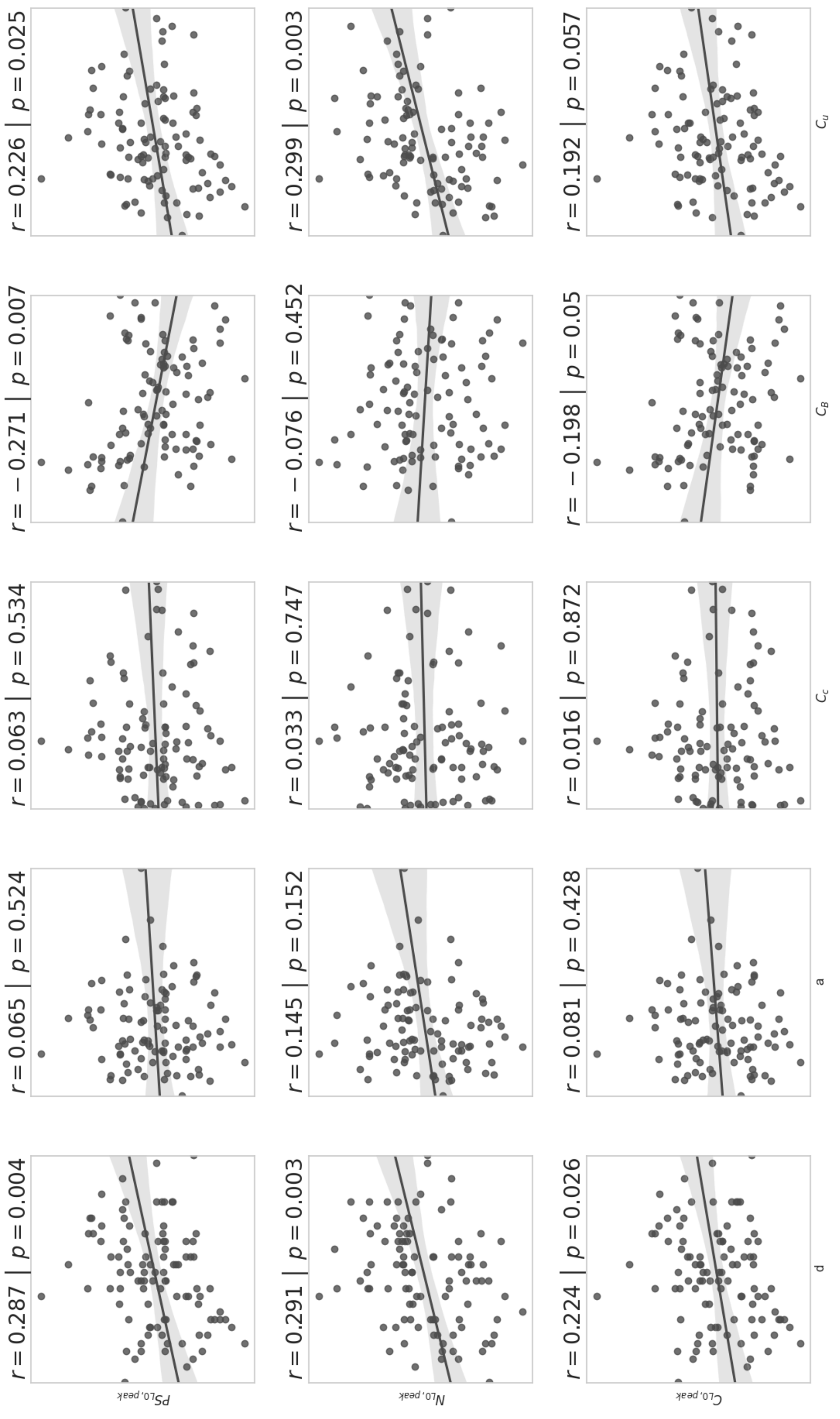


Figure 7.9: Net3 variants global severity of failure to topological attributes.

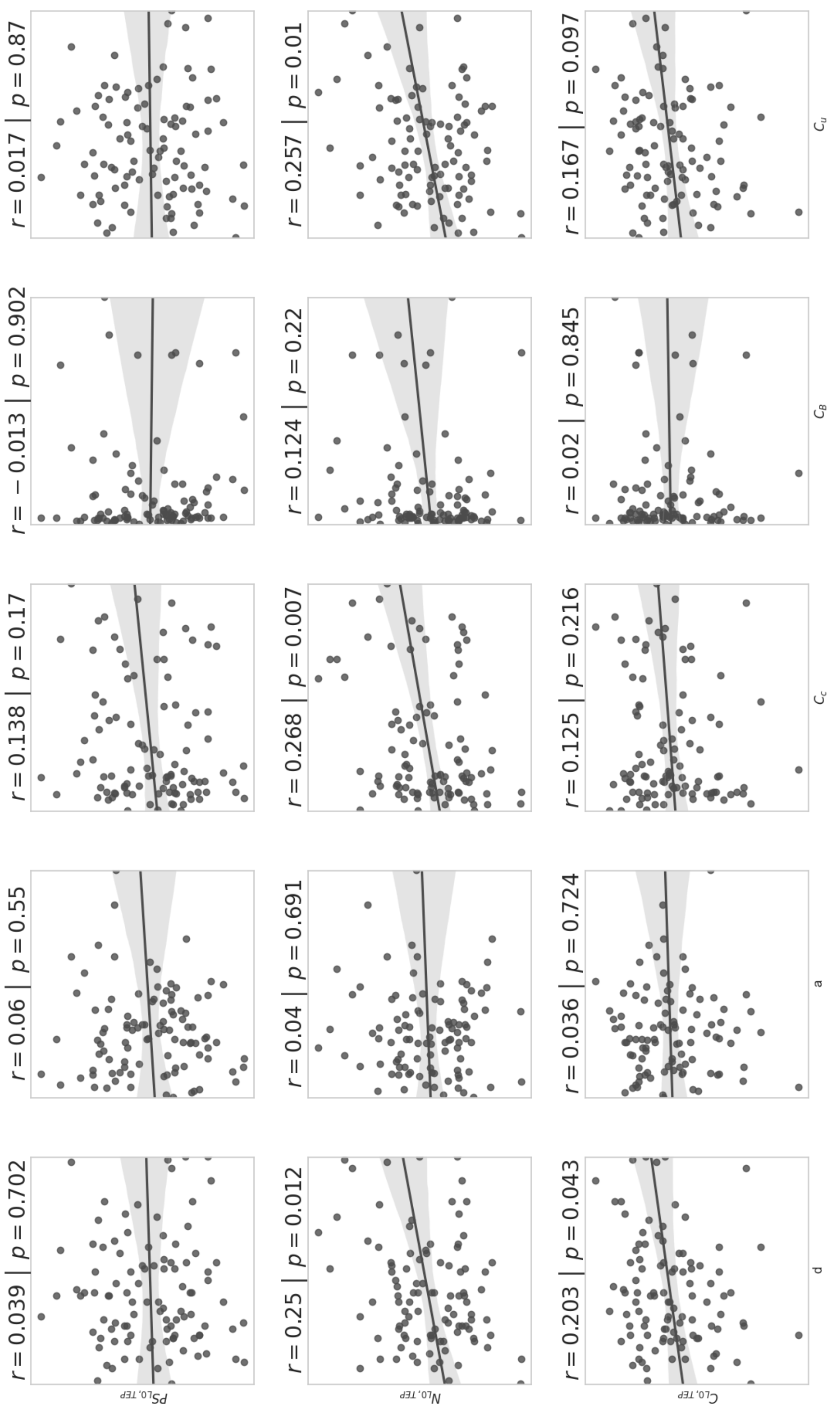


Figure 7.10: Ctown variants global rapidity of failure to topological attributes.

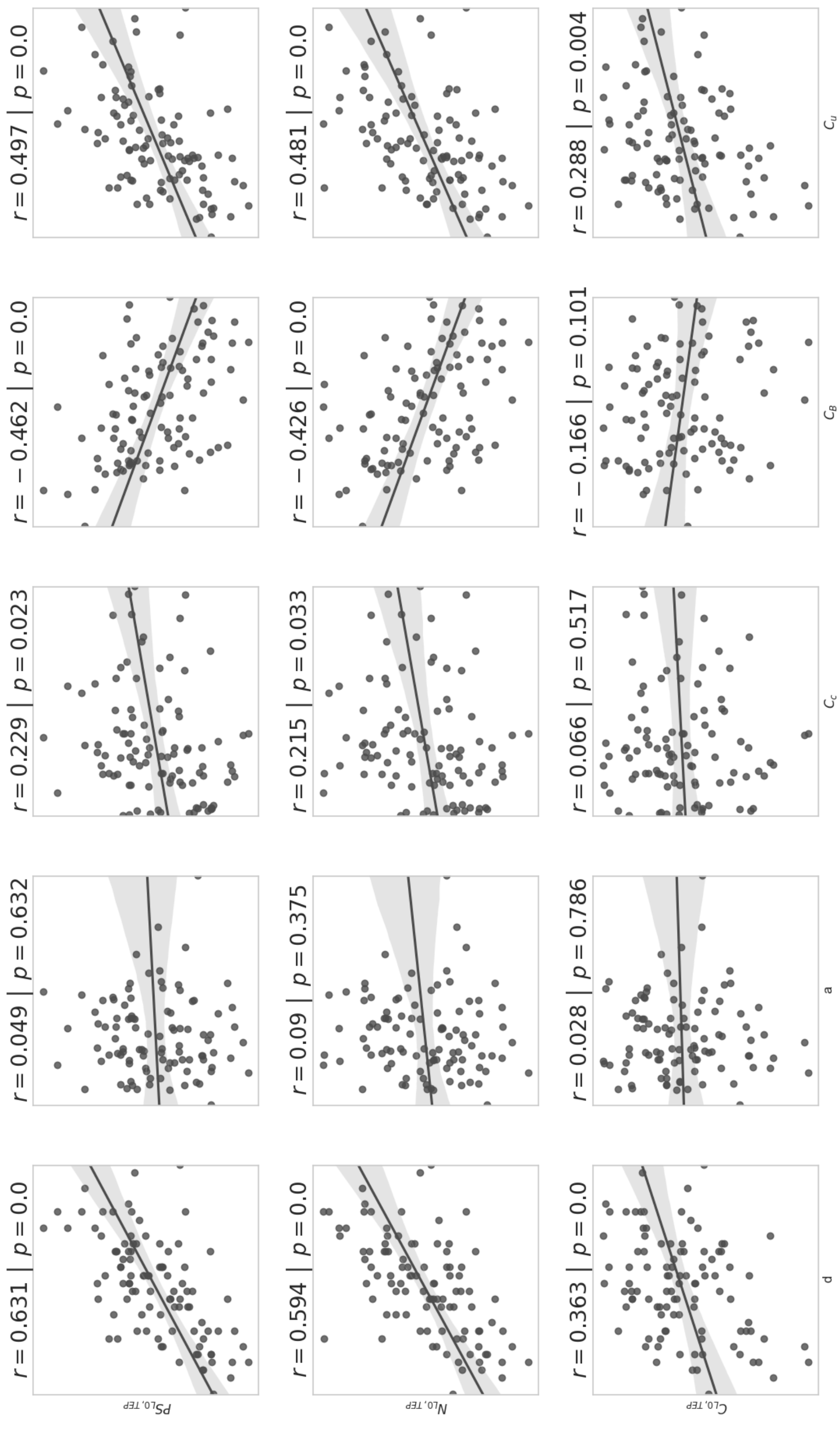


Figure 7.11: Net3 variants global rapidity of failure to topological attributes.

7.3.2 Net3 variants

The scatter plots of Figure 7.5, Figure 7.9, Figure 7.7, Figure 7.11 presents scatter plots between global performance and topological attributes vectors of Ctown variants. Moderate positive correlations are observed between magnitude (see Figure 7.5 spatial and customer dimension) and severity (see Figure 7.9) of failure to the vectors of link density and average closeness centrality. Moderate negative correlations are observed between the global crest factors in terms of spatial manifestation (see Figure 7.5) to the vectors of link density and average closeness centrality. A moderate correlation is also observed between the central point of dominance and the failure aspects of crest factor (see Figure 7.5) and severity (see Figure 7.9). Finally, the Net3 variants form a strong positive correlation to the topological attributes of link density and average closeness centrality.

7.4 Resilience assessment of network variants

This section presents a clustering of the most and least resilient network variants. The presented results of this section is further supported by the figures presented in Appendix B. The global performance indicator of the rapidity of failure (service and spatial dimension for low-quality supply), the magnitude of failure (service and customer dimension for low-quality supply), and the severity of failure (service and spatial dimension for low-quality supply) are evaluated for networks variants.

7.4.1 Ranking network variants

The Ctown variants of 80, 87, 44, and 19 are the most resilient networks, while the Ctown variants of 40, 79, 26, and 72 are the least resilient network in respect to magnitude of failure (see Figure B.1). The Ctown variants of 26, 32, 61, and 16 are the most resilient networks, while the Ctown variants of 18, 13, 27, and 87 are the least resilient network in respect to severity of failure (see Figure B.5. The Ctown variants of 72, 26, 31, and 79 are the most resilient networks, while the Ctown variants of 80, 44, 19, and 82 are the least resilient network in respect to rapidity of failure (see Figure B.6). Ctown variant 80 is among the most resilient network variants in respect to magnitude and severity of failure, while it's rapidity of failure is among the group of worst performing networks. The Ctown variant 87 is among the least resilient network in respect to both severity and rapidity of failure. At the same time, Ctown variant 87 is the second most resilient network in respect to the magnitude of failure.

The Net3 variants of 92, 89, 45, and 49 are the most resilient networks, while the Net3 variants of 47, 99, 41, and 70 are the least resilient network in respect to magnitude of failure (see Figure B.4). The Net3 variants of 14, 90, 45, and

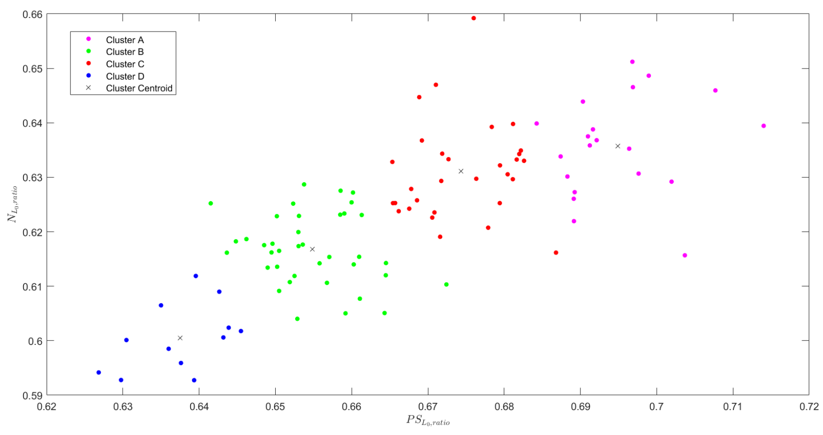


Figure 7.12: Clustering Ctown variants based on magnitude of failure.

16 are the most resilient networks, while the Net3 variants 91, 43, 81, and 99 are the least resilient network in respect to severity of failure (see Figure B.5). The Net3 variants of 49, 56, 45, and 74 are the most resilient networks, while the Net3 variants of 1, 62, 41, and 47 are the least resilient network in respect to rapidity of failure (see Figure B.6). Net3 variant 92 ranks among the most resilient network variants in respect to both magnitude and severity of failure. However, Net3 variant 92 also ranks among the least resilient network variants in respect to rapidity of failure. Net3 variant 14 is the most resilient variant with respect to rapidity, and among the group of average resilient network in respect to severity and rapidity of failure. Net3 variant 47 is the most resilient variant with respect to rapidity, among the mean group of variants in respect to severity of failure, and among the group of worst performing network in respect to magnitude of failure.

7.4.2 Fuzzy C-Means Clustering

Using the magnitude of failure in Ctown as an example, Figure 7.12 presents the results of the Fuzzy C-Means clustering of network variants performance[1]. As observed in Figure 7.12, the network variants are divided into four clusters, each network variant belonging to either of the four clusters. *Cluster D* is the best performing network variants, while *Cluster A* contains the worst performing network variants in respect to the magnitude of failure. The Fuzzy C-Means clustering is conducted for the remaining two dimensions of interest (e.i., rapidity and severity of the failures). Network variants which is a part of *Cluster A* are registered for each of the failure aspects.

The Ctown variants are further divided into four groups based on the Fuzzy C-

[1]Georgios Moraitis, Civil Engineer and Ph.D. Candidate, Dept. of Water Resources and Environmental Engineering, School of Civil Engineering, National Technical Univ. of Athens.

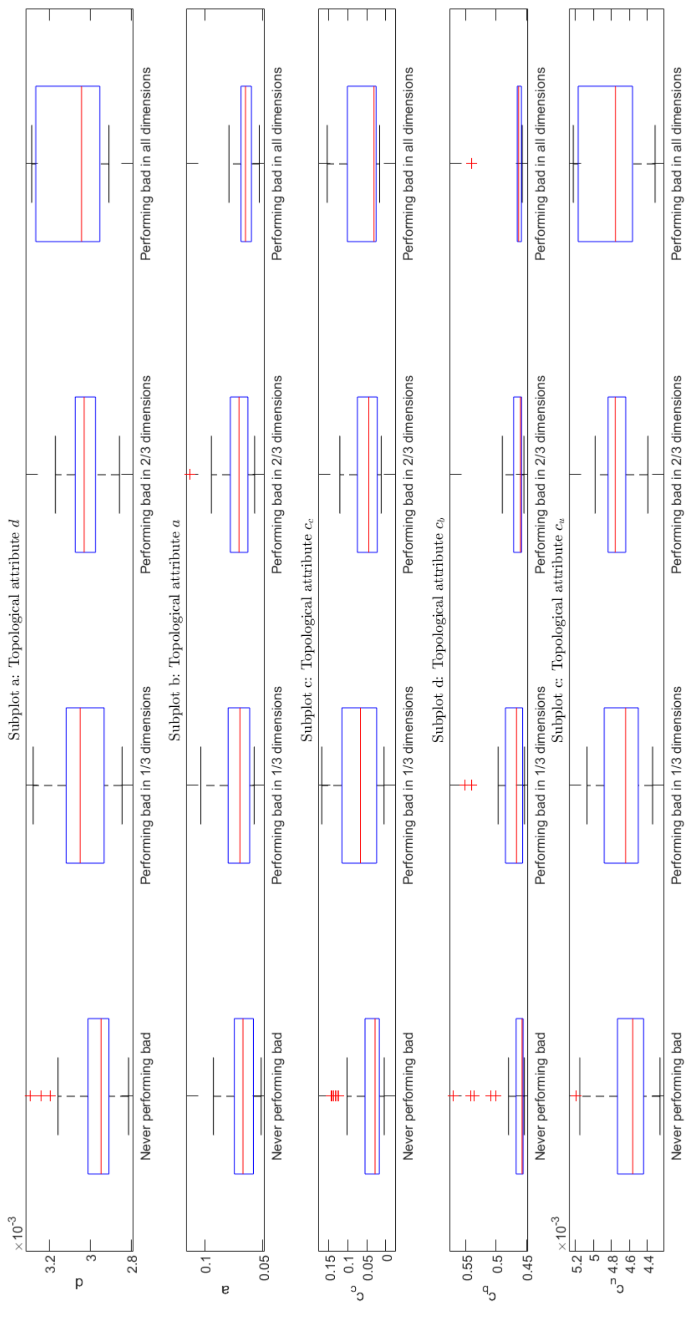


Figure 7.13: Resilience grouping to topological attributes of Ctown variants.

Means clustering result; The network variants that are a part of *Cluster A* in all of the three dimensions (e.i., magnitude, severity, and rapidity failure) are defined as the least resilient networks. The network variants that are a part of *Cluster A* in two out of the three dimensions (e.i., magnitude and severity failure, magnitude and rapidity failure, or rapidity and severity failure) are defined as the second least resilient networks. The network variants that are a part of *Cluster A* in one out of the three dimensions (e.i., magnitude of failure, severity of failure, or rapidity of failure) are defined as the second most resilient networks. The last group of network variants is defined as the most resilient networks as they never appear in *Cluster A*.

Figure 7.13 displays box plots of the network variants distribution of topological attributes grouped by performance (e.i., *Cluster A* affiliation). The rows of Figure 7.13 present the topological attribute indicators of link density, average shortest path length, clustering coefficient, central-point of dominance, and average closeness centrality, respectively (see Figure 7.1). A box plot summarises the topological attributes of a group of network variants based on a five-number summary. The sample mean of the dataset is indicated with a red horizontal line inside the box. The box presents the boundaries of the first quarterlies (e.i., 25th percentile) and third quarterlies (e.i., 75th percentile) of the dataset, while the lines extending outside the box indicate the variability outside the upper and lower quarterlies. Network variant outliers are, for certain groups, presented as individual points.

7.5 Discussion

The global performance indicators reflect the differences in the applied technologies and network designs of the benchmark models. The Net3 model is the most resilient benchmark model in magnitude, severity, and average propagation. At the same time, the large-sized structure of Trondheim model is evaluated as the most resilient network in terms of rapidity. A rapid event implies that the water utility has less time to react to the failure. As the Net3 is the smallest network, low-quality water is rapidly distributed within the network structures compared to the more extensive travel distances observed in the Ctown and the Trondheim model.

In contrast to the Net3 model, both the Ctown and the Trondheim model have a distinct pipe hierarchy indicated by a dominant topological backbone structure. E.i. pipes are ranked according to their relative status and water distribution purpose reflected by the central point of dominance. Furthermore, the flushing mechanisms of Net3 facilitate absorption and recovery rates of the failure through a unidirectional flow pattern within looped parts of the network.

The proposed network generation procedure in *Data Acquisition and Processing* (see Section 4.3) presents a tool to limit the number of uncertainties associated with the system to the variability of link connections. The distribution of topological attributes in network variants (see Figure 7.1) reflects the connection constraints enforced by the classification of links (e.i. service areas and critical links) and the link addition step (e.i. alpha and spatial proximity). As the skeletonized network of Ctown consists of more floating nodes than skeletonized Net3, it has a higher probability of connecting multiple links to an already connected node. Hence, Ctown generally required more pipes added pipes in order to complete the network generation procedure.

In terms of Ctown variants, the value ranges of the produced global failure indicators (see Figure 7.2) are of statistical significance in respect to the result of the convergence analysis. The 95% confidence interval (see Table 6.1) of the convergence analysis are smaller than the 95% confidence interval of the global failure indicators of the Ctown variants. The result indicates that a relationship between quality-related resilience and the network design exists.

Strong positive correlations were only observed between global failure metrics of the rapidity of failure (see Figure 7.11) and the topological attributes of link density and average closeness centrality in Net3 variants. However, weak correlations were observed between the same Ctown vectors. A weak to moderate correlation was observed for other global failure indicators and specific topological attributes (link density, central point of dominance, and average closeness centrality).

Nevertheless, the relationship between resilience and network design is ambiguous. As resilience is a term that encompasses multiple aspects of system performance, it is too complex to be represented by a single global performance indicator [48, 65]. Figure 7.2 and Figure 7.3 illustrates that a system can be more resilient in terms of one global failure indicator, and less resilient in terms of another global failure indicator. The result is supported by Figure 7.13, as the grouping based on performance illustrates that network variants may not exclusively perform bad in all dimensions (e.i., magnitude, severity, and rapidity of failures).

The clustering of Ctown variants based on the performance categories (see Figure 7.13) presents a weak trend in increasing median of link density and average closeness centrality based on the threshold for low-quality water supplied. However, it is noted that the shape of the percentiles is inconsistent with the observed sample mean of the consecutive group. Evaluations of polluted water supplied threshold show a more prominent trend, as the less resilient networks generally display a higher link density, clustering coefficient, and average closeness centrality values.

As stated before, this study is only concerned with quality-related water concerns. Conversely, the study of Meng *et al.* [65] demonstrated a strong negative correlation between link density and specific temporal dimensions in terms of water quantity-related water concerns. The results of the water quality-related failures indicate that higher link density promotes the spread of contaminants and increases the rapidity of the stress-testing event (see Figure 7.11). The findings suggest that there may be a trade-off between quality-related and quantity-related resilience in terms of temporal aspects of the failure. Similarly, the study of Diao *et al.* [35] reveals that increased resilience to one failure mode may decrease resilience to another.

It is noted that the topological attributes applied in this study are concerned with a graphical representation of the water distribution networks. Specific global failure indicators may be more dependent on the hydraulic behavior of the network variants. These node connections may significantly influence the hydraulic in the system than multiple other node connections. However, as the number of pipes in the network generation procedure increases, it also increases the probability of establishing a high influence connection or a series of high influence connections in terms of hydraulic behavior. The result of the correlation analysis (see 7.3) suggests that the topological backbone of the Ctown variants strongly affects the temporal behavior of the system, and changes to the peripheral subsystems have less influence on the overall rapidity of the event. As such, the results of the source point variability analysis (see Section 5.2) show that the peripheral source points have a more substantial influence on the system performance in terms of the temporal dimension.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this study, three benchmark models are exposed to substance intrusion in order to evaluate the behavior and the resilience of the systems. An automatic network generation procedure is proposed to generate network variants with a unique set of topological attributes. The generation procedure uses statistical properties of the system to classify, remove, and add node connections, while the calibration of the added pipe diameters is based on economic flow rates. Topological attributes are applied to characterise the network design of the network variants. The models are exposed to random failure sequences of substance intrusions. The system response is measured with the proposed global failure indicators to evaluate the resilience of topological attributes. Based on the obtained results of this study, the following conclusions have been drawn:

- The proposed global failure metrics are derived from the novel framework of Moraitis *et al.* [56] to address various aspects of the failure (e.i. magnitude, peak, average propagation, crest factor, and rapidity). A global failure indicator is the mean key performance indicator of stress-testing events with increasing disturbance. The global failure metrics combined address the system's lack of ability to resist, absorb, and restore quality-related failures.
- The network generation procedure presents a framework to generate network variants based on design principles automatically. The input, defined by system constraints and statistical parameters, may be modified to promote desirable topological network features on a network scale (e.i. design of the system). Furthermore, the economic flow rates may be modified to alter the system's hydraulic landscape and flow patterns (e.i. technologies).
- The system behavior exposed to a substance intrusion is characterised by the spatial distribution of source nodes and the input boundaries defined by the control variables. The magnitude of the mass entering the system (e.i. re-

stricted by mass flow, duration, and the number of nodes) may be used as a predictable measure of stress inflicted on the system. Spatial (e.i. node or nodes used as a source point) and temporal (e.i. start time and duration) variability address hydraulic uncertainties of the system. The random failure sequences are adapted to communicate stress through the magnitude of mass (e.i. number of source points selected) while addressing spatial and temporal variability in a multivariate stochastic approach (e.i. random selection of source points and start time inputs).

- The confidence intervals presented in the convergence analysis (see Table 6.1) in light of the performance distribution of network variants implies that resilience is dependent on the structural changes enforced by the network generation procedure.
- Strong correlations were only observed between global failure metrics of the rapidity of failure (service, spatial, and customer dimension) and topological attributes for Net3 variants. The result suggests that the topological backbone of Ctown strongly affects the temporal scales of failure. In contrast, topological changes to the peripheral subsystem of Ctown have less influence on the overall rapidity of the event.
- Trade-offs are observed between global failure indicators a benchmark model and the the mean performance of the associated network variants. E.i., one network variant may be more resilient in terms of one global failure indicator and less so in light of other global failure indicators.
- Correlation trends between topological attributes and global failure indicators are non-consistent comparing the results of Ctown variants to the results of Net3 variants.
- The results of this study suggest that it can be misleading to use topological attribute metrics, or at least without hydraulic considerations, as a surrogate in evaluating resilience enhancing strategies in terms of substance intrusions in water distribution systems.

This study suggests that resilience, as an integrated concept that allows multiple risks and stress to be measured with global failure indicators, to be evaluated in the context of a portfolio of technologies and network designs comprised within the boundaries of water distribution models. As such, resilience assessment may be used to evaluate the effectiveness of alternative system designs in strategic planning of water distribution systems.

Nonetheless, a systematic approach to address multiple stress-testing events of alternative network designs has its own drawbacks. Practitioners may want to fall back on more familiar concepts with which they have practical experience. Risk and risk management procedures provide such familiarity and allow for cross-disciplinary communication [55]. Thus, combining elements of the resilience assessment and risk management procedures will likely be the most practical option.

8.2 Future Work

Being in its infancy, several aspects of resilience remains to be explored. While resilience clearly has attractions as a unifying concept and a vision for sustainable water distribution systems in uncertain times, achieving positive outcomes will require intensive research of its underlying components. Because of this study, a comprehensive toolkit for modifying and generating network variants is developed, along with an algorithm for stress-testing water distribution systems with substance intrusions (see Appendix C). Therefore, different studies can be done to extend and improve this work by using the developed algorithms as a starting point. Some of the studies may be suggested here:

- There is a dependency between network design and loss of quality-related objectives of the models evaluated in this study. Since the dependencies vary between variants derived from different benchmark models, one study might extend the pool of benchmark models and generated network variants to evaluate the installed technologies, their relationship to the network design, and the consistency of the produced global failure indicators.
- The results of the Fuzzy C-Means clustering indicate that the critical concentration threshold yields a more prominent trend of the least resilient network groups and the topological attribute indicators than any concentration threshold. Due to dilution, lower concentration values are observed in the topological backbone, making the global failure indicators more influenced by substance propagation in the peripheral subsystems. Furthermore, most stress-testing events produced undesirable conditions at all time steps of the simulation period. As a result, the temporal scales of the failure became trivial with the input configuration of the random failure sequences used in this study. Thus, this study suggests that future studies use a lower mass flow input to highlight the importance of network design in subsystems.
- A total of five topological attribute indicators are included in this study. The indicators are derived from graphical representations of the water distribution networks, and thus, without consideration of technical characteristics. One study might incorporate structural properties or hydraulic relations (e.g., as weights of the Dijkstra shortest path) to further highlight the importance of certain connections as a part of the topological attribute indicators. Similarly, future studies may evaluate the relationship between the input of the generation procedure and the resilience of the produced network variant.
- In order to fully grasp the complexity of resilience, one study should incorporate several failure modes related to both quantity and quality concerns of the water distribution system (e.g., substance intrusion, pipe failure, and shifting demand patterns). In light of the findings in the previous studies of Diao *et al.* [35] and Meng *et al.* [65], the results of this study indicate there is a trade-off between quality and quantity failures. As such, one study might

evaluate the resilience of the water distribution system through a multi-objective optimisation procedure.

Bibliography

[1] J. F. Claerbout, 'A scrutiny of the introduction,' *The Leading Edge*, vol. 10, no. 1, pp. 39–41, 1991.

[2] K. K. Landes, 'A scrutiny of the abstract,' *Bulletin of the American Association of Petroleum Geologists*, vol. 35, no. 7, p. 1660, 1951.

[3] Directive of the European Parliament and of the Council, 'Eu 2020/2184 on the quality of water intended for human consumption,' *(EU) 2020/2184*, 2020.

[4] A. Gibbons, *Algorithmic graph theory*. Cambridge university press, 1985.

[5] B. Efron, 'Bootstrap methods: Another look at the jackknife,' in *Breakthroughs in statistics*, Springer, 1992, pp. 569–593.

[6] S. Gössling, 'New performance indicators for water management in tourism,' *Tourism Management*, vol. 46, pp. 233–244, 2015.

[7] C. S. Holling and M. A. Goldberg, 'Ecology and planning,' *Journal of the american Institute of Planners*, vol. 37, no. 4, pp. 221–230, 1971.

[8] W. D. Callister and D. G. Rethwisch, *Materials science and engineering*. John wiley and sons NY, 2011, vol. 5.

[9] S. H. Strogatz, 'Exploring complex networks,' *nature*, vol. 410, no. 6825, pp. 268–276, 2001.

[10] R. Nazempour, M. A. S. Monfared and E. Zio, 'A complex network theory approach for optimizing contamination warning sensor location in water distribution networks,' *International Journal of Disaster Risk Reduction*, vol. 30, pp. 225–234, 2018.

[11] W. Ge, F. Chen, J. Gao, S. Gao, J. Huang, X. Liu, Y. Ren, Q. Sun, L. Wang, W. Wang *et al.*, 'Analytical multi-scale method for multi-phase complex systems in process engineering—bridging reductionism and holism,' *Chemical Engineering Science*, vol. 62, no. 13, pp. 3346–3377, 2007.

[12] O. Giustolisi, A. Simone and L. Ridolfi, 'Network structure classification and features of water distribution systems,' *Water Resources Research*, vol. 53, no. 4, pp. 3407–3423, 2017.

[13] S. Atkinson, R. Farmani, F. A. Memon and D. Butler, 'Reliability indicators for water distribution system design: Comparison,' *Journal of Water Resources Planning and Management*, vol. 140, no. 2, pp. 160–168, 2014.

[14] P. M. Schober, C. Boer and L. A. Schwarte, 'Correlation coefficients: Appropriate use and interpretation,' *Anesthesia and Analgesia*, 2018.

[15] C. J. Kowalski, 'On the effects of non-normality on the distribution of the sample product-moment correlation coefficient,' *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 21, no. 1, pp. 1–12, 1972.

[16] A. Yazdani, R. A. Otoo and P. Jeffrey, 'Resilience enhancing expansion strategies for water distribution systems: A network theory approach,' *Environmental Modelling and Software*, vol. 26, no. 12, pp. 1574–1582, 2011.

[17] A. Hazra, 'Using the confidence interval confidently,' *Journal of thoracic disease*, vol. 9, no. 10, p. 4125, 2017.

[18] T. J. DiCiccio, B. Efron *et al.*, 'Bootstrap confidence intervals,' *Statistical science*, vol. 11, no. 3, pp. 189–228, 1996.

[19] H. Ødegaard, *Vann- og avløpsteknikk*, N. Vann, Ed. 2014, vol. 2.

[20] A. Di Nardo, C. Giudicianni, R. Greco, M. Herrera, G. F. Santonastaso and A. Scala, 'Sensor placement in water distribution networks based on spectral algorithms,' vol. 7, 2018.

[21] S. Abney, 'Bootstrapping,' pp. 360–367, 2002.

[22] A. G. Asuero, A. Sayago and A. Gonzalez, 'The correlation coefficient: An overview,' *Critical reviews in analytical chemistry*, vol. 36, no. 1, pp. 41–59, 2006.

[23] A. Goyal, P. Mogha, R. Luthra and N. Sangwan, 'Path finding: A* or dijkstra's,' *International Journal in IT and Engineering*, vol. 2, no. 1, pp. 1–15, 2014.

[24] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez and D.-U. Hwang, 'Complex networks: Structure and dynamics,' *Physics Reports*, vol. 424, no. 4, pp. 175–308, 2006, ISSN: 0370-1573. DOI: `https://doi.org/10.1016/j.physrep.2005.10.009`.

[25] J. A. Ramirez *et al.*, 'Prediction and modeling of flood hydrology and hydraulics,' *Inland flood hazards: Human, riparian and aquatic communities*, p. 498, 2000.

[26] A. G. Seyoum, T. T. Tanyimboh and C. Siew, 'Assessment of water quality modelling capabilities of epanet multiple species and pressure-dependent extension models.,' *Water Supply 13 (4): 1161–1166*, 2013.

[27] S. Mohapatra, A. Sargaonkar and P. Labhasetwar, 'Distribution network assessment using epanet for intermittent and continuous water supply,' *Water Resour Manage 28, 3745–3759 (2014).*, 2014.

[28] X. Yang and D. L. Boccelli, 'The impacts of demand variability on distribution system hydraulics and transport,' *World Environmental and Water Resources Congress*, 2009.

[29] J. Johansson, 'Risk and vulnerability analysis of interdependent technical infrastructures: Addressing socio-technical systems,' eng, 2010.

[30] R. Kellagher, Y. Cesses, M. Di Mauro and B. Gouldby, 'An urban drainage flood risk procedure - a comprehensive approach.,' *WaPUG Annual Conference 2009, 11-13 November 2009, Hilton Hotel, Blackpool.*, 2009.

[31] X. Sun, L. Newham, B. Croke and J. Norton, 'Three complementary methods for sensitivity analysis of a water quality model,' *Environmental Modelling and Software*, vol. 37, pp. 19–29, 2012, ISSN: 1364-8152. DOI: https://doi.org/10.1016/j.envsoft.2012.04.010.

[32] S. N. Mugume, D. E. Gomez, G. Fu, R. Farmani and D. Butler, 'A global analysis approach for investigating structural resilience in urban drainage systems,' *Water Research*, vol. 81, pp. 15–26, 2015, ISSN: 0043-1354. DOI: https://doi.org/10.1016/j.watres.2015.05.030.

[33] M. Pizzol, 'Life cycle assessment and the resilience of product systems,' *Industrial Ecologyy*, 2015. DOI: https://doi.org/10.1111/jiec.12254.

[34] R. Francis and B. Bekera, 'A metric and frameworks for resilience analysis of engineered and infrastructure systems,' *Reliability Engineering and System Safety*, vol. 121, pp. 90–103, 2014, ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2013.07.004.

[35] K. Diao, C. Sweetapple, R. Farmani, G. Fu, S. Ward and D. Butler, 'Global resilience analysis of water distribution systems,' *Water Research*, vol. 106, pp. 383–393, 2016, ISSN: 0043-1354. DOI: https://doi.org/10.1016/j.watres.2016.10.011.

[36] L. Rossman, H. Woo, M. Tryby, F. Shang, R. Janke and T. Haxton, 'Epanet 2.2 user manual. u.s.,' *Environmental Protection Agency, Washington*, 2020.

[37] K. Klise, R. Murray and T. Haxton, 'An overview of the water network tool for resilience (wntr),' *In Proceedings of the 1st International WDSA/CCWI Joint Conference, Kingston*, 2018.

[38] R. Janke, T. N. Taxon and M. J. Davis, 'Mass imbalances in epanet water-quality simulations,' *Drinking Water Engineering and Science*, 2018.

[39] R. Wéber, T. Huzsvár and C. Hős, 'Vulnerability analysis of water distribution networks to accidental pipe burst,' *Water Research*, vol. 184, p. 116 178, 2020, ISSN: 0043-1354. DOI: https://doi.org/10.1016/j.watres.2020.116178.

[40] R. Albert, H. Jeong and A.-L. Barabási, 'Error and attack tolerance of complex networks,' *Nature volume*, vol. 406, pp. 378–382, 2000. DOI: https://doi.org/10.1038/35019019.

[41] R. Sitzenfrei, M. Möderl and W. Rauch, 'Automatic generation of water distribution systems based on gis data,' *Environmental Modelling and Software*, vol. 47, pp. 138–147, 2013, ISSN: 1364-8152. DOI: https://doi.org/10.1016/j.envsoft.2013.05.006.

[42] M. Moederl, R. Sitzenfrei, T. Fetz, E. Fleischhacker and W. Rauch, 'Systematic generation of virtual networks for water supply,' *Advancing Earth and Space Science*, 2011. DOI: https://doi.org/10.1029/2009WR008951.

[43] Statistics Norway. (2021). 'Trondheim municipality,' [Online]. Available: https://www.ssb.no/kommunefakta/trondheim.

[44] A. Nicholson, S. Webber, S. Dyer, T. Patel and H. Janicke, 'Scada security in the light of cyber-warfare,' *Computers and Security*, vol. 31, no. 4, pp. 418–436, 2012, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2012.02.009.

[45] E. A. Lee, 'Cyber physical systems: Design challenges,' pp. 363–369, 2008. DOI: 10.1109/ISORC.2008.25.

[46] J. Li, X. Yang and R. Sitxenfrei, 'Rethinking the framework of smart water system: A review.,' *MDPI*, 2020. DOI: 10.3390/w12020412.

[47] T. M. Walski, 'A history of water distribution,' *Journal AWWA*, vol. 98, no. 3, pp. 110–121, 2006. DOI: 10.1002/j.1551-8833.2006.tb07611.x.

[48] C. Makropoulos, D. Nikolopoulos, L. Palmen, S. Kools, A. Segrave, D. Vries, S. Koop, H. J. van Alphen, E. Vonk, P. van Thienen, E. Rozos and G. Medema, 'A resilience assessment method for urban water systems,' *Urban Water Journal*, vol. 15, no. 4, pp. 316–328, 2018. DOI: 10.1080/1573062X.2018.1457166.

[49] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, A. Ostfeld, D. G. Eliades, M. Aghashahi, R. Sundararajan, M. Pourahmadi, M. K. Banks, B. M. Brentan, E. Campbell, G. Lima, D. Manzi, D. Ayala-Cabrera, M. Herrera, I. Montalvo, J. Izquierdo, E. Luvizotto, S. E. Chandy, A. Rasekh, Z. A. Barker, B. Campbell, M. E. Shafiee, M. Giacomoni, N. Gatsis, A. Taha, A. A. Abokifa, K. Haddad, C. S. Lo, P. Biswas, M. F. K. Pasha, B. Kc, S. L. Somasundaram, M. Housh and Z. Ohar, 'Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks,' *Journal of Water Resources Planning and Management*, vol. 144, no. 8, 2018. DOI: 10.1061/(ASCE)WR.1943-5452.0000969.

[50] A. Hassanzadeh, A. Rasekh, S. Galelli, M. Aghashahi, R. Taormina, A. Ostfeld and M. K. Banks, 'A review of cybersecurity incidents in the water sector,' *Journal of Environmental Engineering*, vol. 146, no. 5, p. 03 120 003, 2020. DOI: 10.1061/(ASCE)EE.1943-7870.0001686.

[51] R. Taormina, S. Galelli, H. Douglas, N. Tippenhauer, E. Salomons and A. Ostfeld, 'A toolbox for assessing the impacts of cyber-physical attacks on water distribution systems,' *Environmental Modelling and Software*, vol. 112, pp. 46–51, 2019, ISSN: 1364-8152. DOI: `https://doi.org/10.1016/j.envsoft.2018.11.008`.

[52] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons and A. Ostfeld, 'Characterizing cyber-physical attacks on water distribution systems,' *Journal of Water Resources Planning and Management*, vol. 143, no. 5, p. 04 017 009, 2017. DOI: `10.1061/(ASCE)WR.1943-5452.0000749`.

[53] D. Nikolopoulos, G. Moraitis, D. Bouziotas, A. Lykou, G. Karavokiros and C. Makropoulos, 'Cyber-physical stress-testing platform for water distribution networks,' *Journal of Environmental Engineering*, vol. 146, no. 7, p. 04 020 061, 2020. DOI: `10.1061/(ASCE)EE.1943-7870.0001722`.

[54] D. Nikolopoulos, C. Makropoulos, D. Kalogeras, K. Monokrousou and I. Tsoukalas, 'Developing a stress-testing platform for cyber-physical water infrastructure,' in *2018 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, 2018, pp. 9–11. DOI: `10.1109/CySWater.2018.00009`.

[55] C. Makropoulos, G. Moraitis, D. Nikolopoulos, G. Karavokiros, A. Lykou, I. Tsoukalas, M. Moreley, M. C. Gama, E. Okstad and J. Vatn, *Risk analysis and evaluation toolkit wp4 manual*, 2019.

[56] G. Moraitis, D. Nikolopoulos, D. Bouziotas, A. Lykou, G. Karavokiros and C. Makropoulos, 'Quantifying failure for critical water infrastructures under cyber-physical threats,' *Journal of Environmental Engineering*, vol. 146, no. 9, p. 04 020 108, 2020. DOI: `10.1061/(ASCE)EE.1943-7870.0001765`.

[57] 'Risk management - Guidlines,' International Organization for Standardization, Geneva, CH, Standard, Mar. 2009.

[58] N. S. Grigg, 'Water utility security: Multiple hazards and multiple barriers,' *Journal of Infrastructure Systems*, vol. 9, no. 2, pp. 81–88, 2003.

[59] J. F. Hughes and K. Healy, *Measuring the resilience of transport infrastructure*, 546. 2014.

[60] L. A. Grossman, 'Epanet 2 - users manual,' *Water Supply and Water Resources Division*, 2000. DOI: `https://www.epa.gov/water-research/epanet`.

[61] K. A. Klise, D. B. Hart and D. Moriarty, 'Water network tool for resilience (wntr) user manual,' 2017. DOI: `https://cfpub.epa.gov/si/si_public_record_report.cfm?Lab=NHSRC&dirEntryId=337793`.

[62] T. Tanyimboh, B. Tahar and A. Templeman, 'Pressure-driven modelling of water distribution systems,' *Water Supply*, 2003. DOI: `https://doi.org/10.2166/ws.2003.0112`.

[63] L. Reis, I. Carrijo and A. Soares, 'Head-driven simulation model (hdsm) for water distribution system calibration,' 2003.

[64] E. v. Loucks Daniel P. Beek, J. R. Stedinger, D. J. P.M. and M. Villars, *Water resources systems planning and management: an introduction to methods, models and applications*. 2005, ch. 9. Model Sensetivity and Uncertanty Analysis, p. 255.

[65] F. Meng, G. Fu, R. Farmani, C. Sweetapple and D. Butler, 'Topological attributes of network resilience: A study in water distribution systems,' *Water Research*, vol. 143, pp. 376–386, 2018, ISSN: 0043-1354. DOI: `https://doi.org/10.1016/j.watres.2018.06.048`.

Appendix A

Critical quality threshold to topological attribute indicators

Appendix A presents the scatter plots and the Pearson correlation coefficients of failure based on the polluted water-quality threshold and topological attributes vectors. The topological attributes of link density, average shortest path length, clustering coefficient, central-point of dominance, and average closeness centrality is presented along the columns in the figures (e.i., x-axis), while the service, spatial, and customer dimension is presented along the rows of the figures (e.i., y-axis). Figure A.1 and Figure A.2 present the magnitude of each failure dimension. Figure A.3 and Figure A.4 present the crest factor of each failure dimension. Figure A.5 and Figure A.6 present the severity of each failure dimension. Figure A.7 and Figure A.8 present the rapidity of each failure dimension. Network variants are represented as points in the figures.

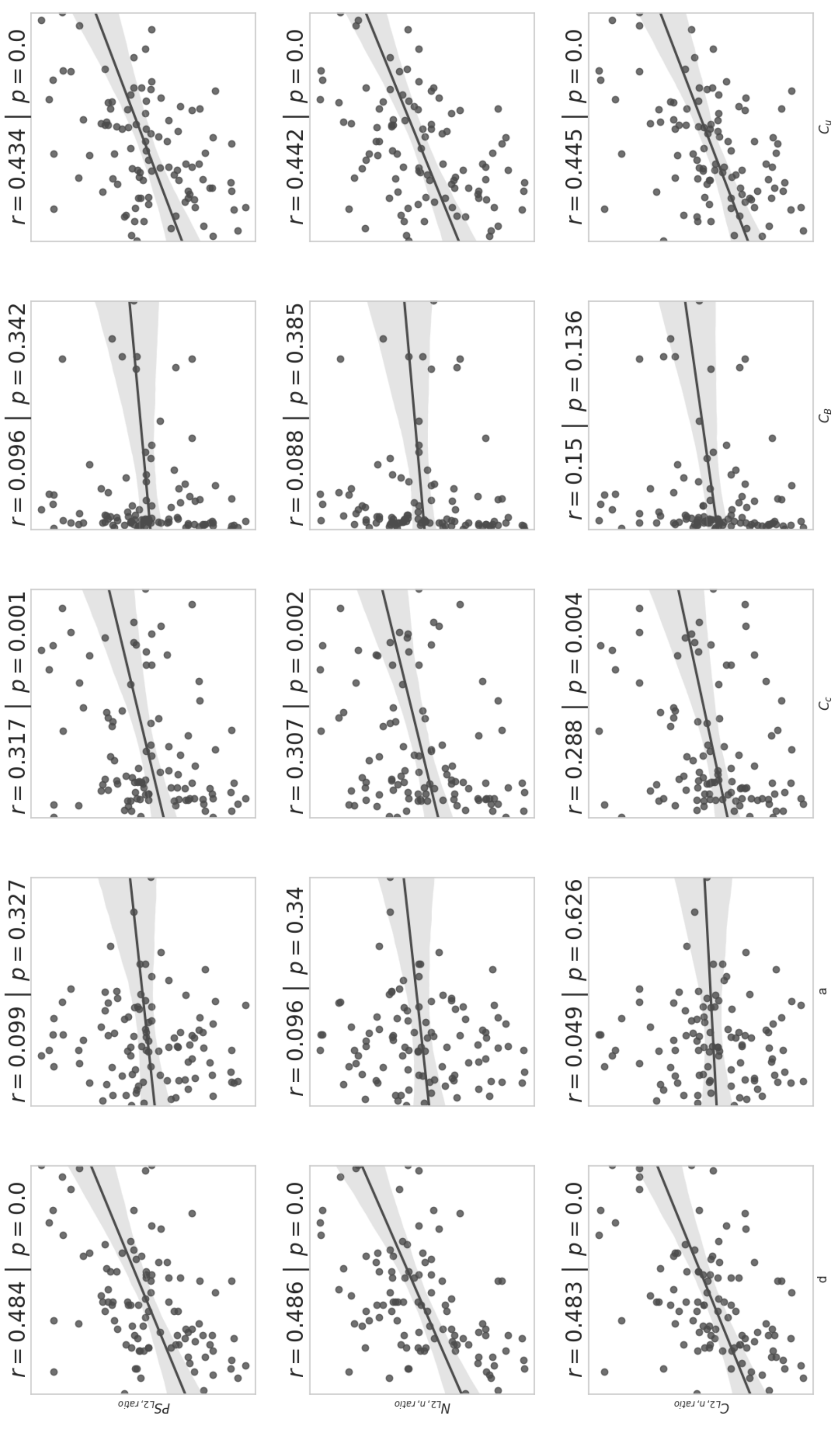


Figure A.1: Ctown variants global magnitude of failure to topological attributes.

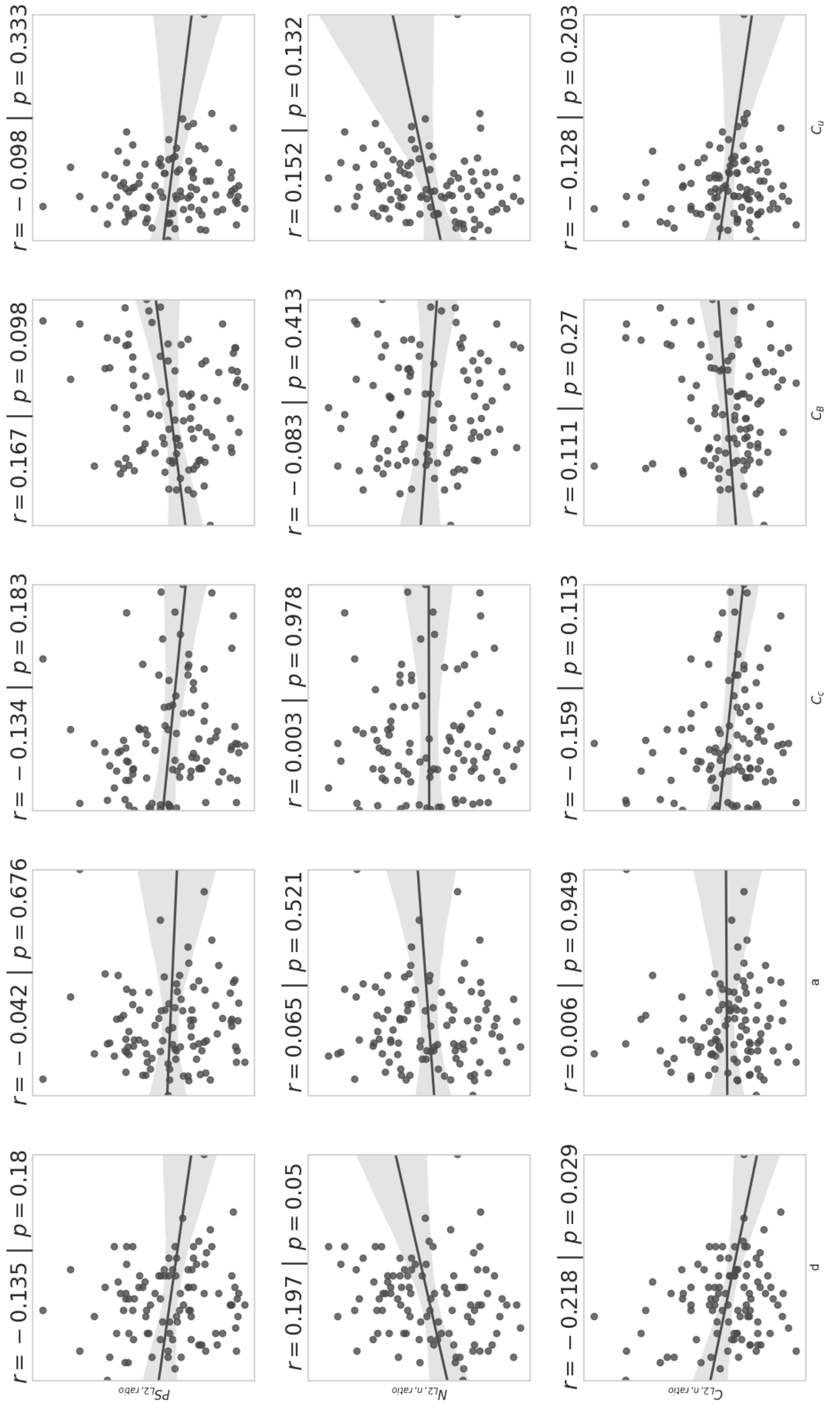


Figure A.2: Net3 variants global magnitude of failure to topological attributes

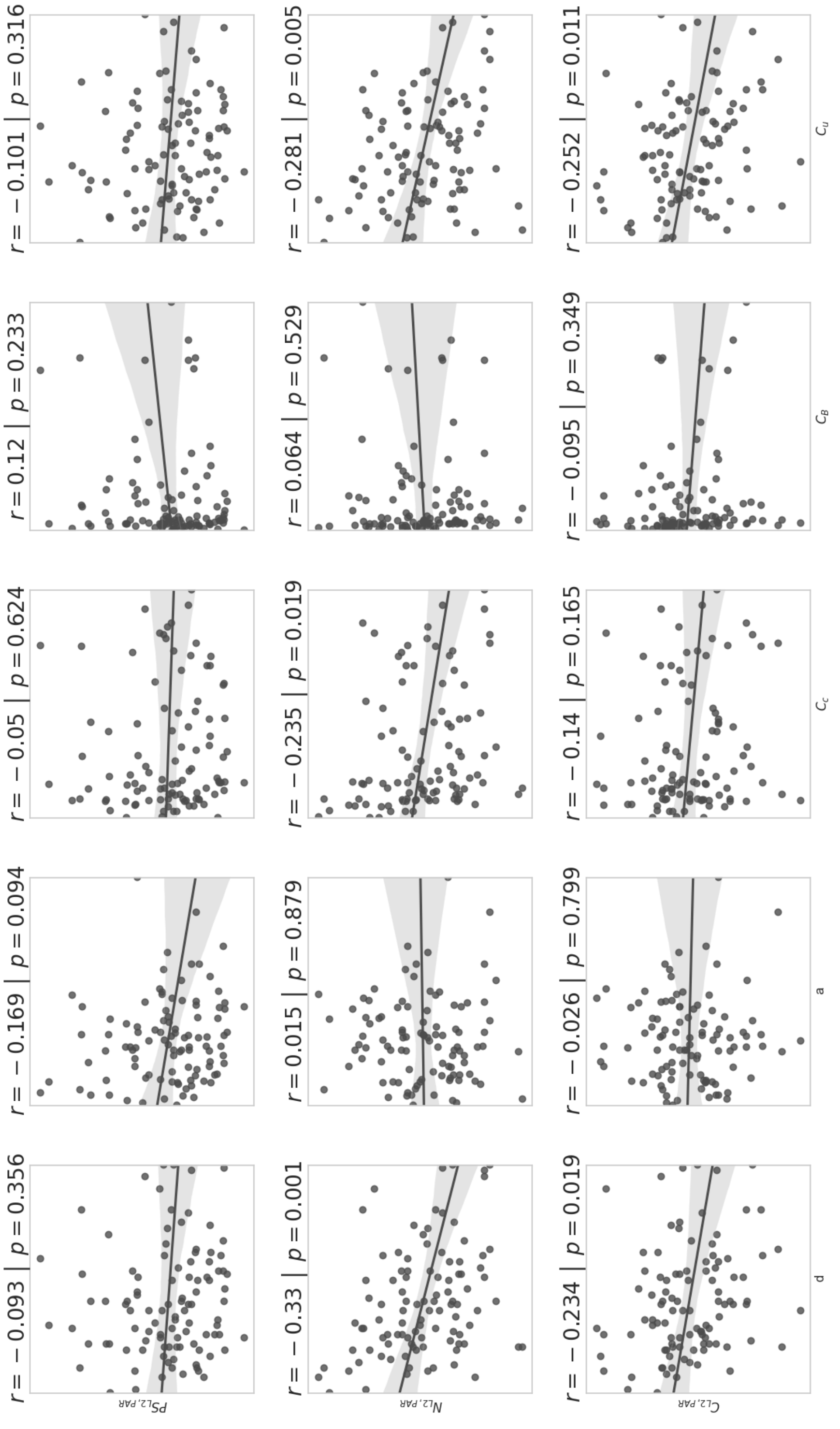


Figure A.3: Ctown variants global crest factor of failure to topological attributes.

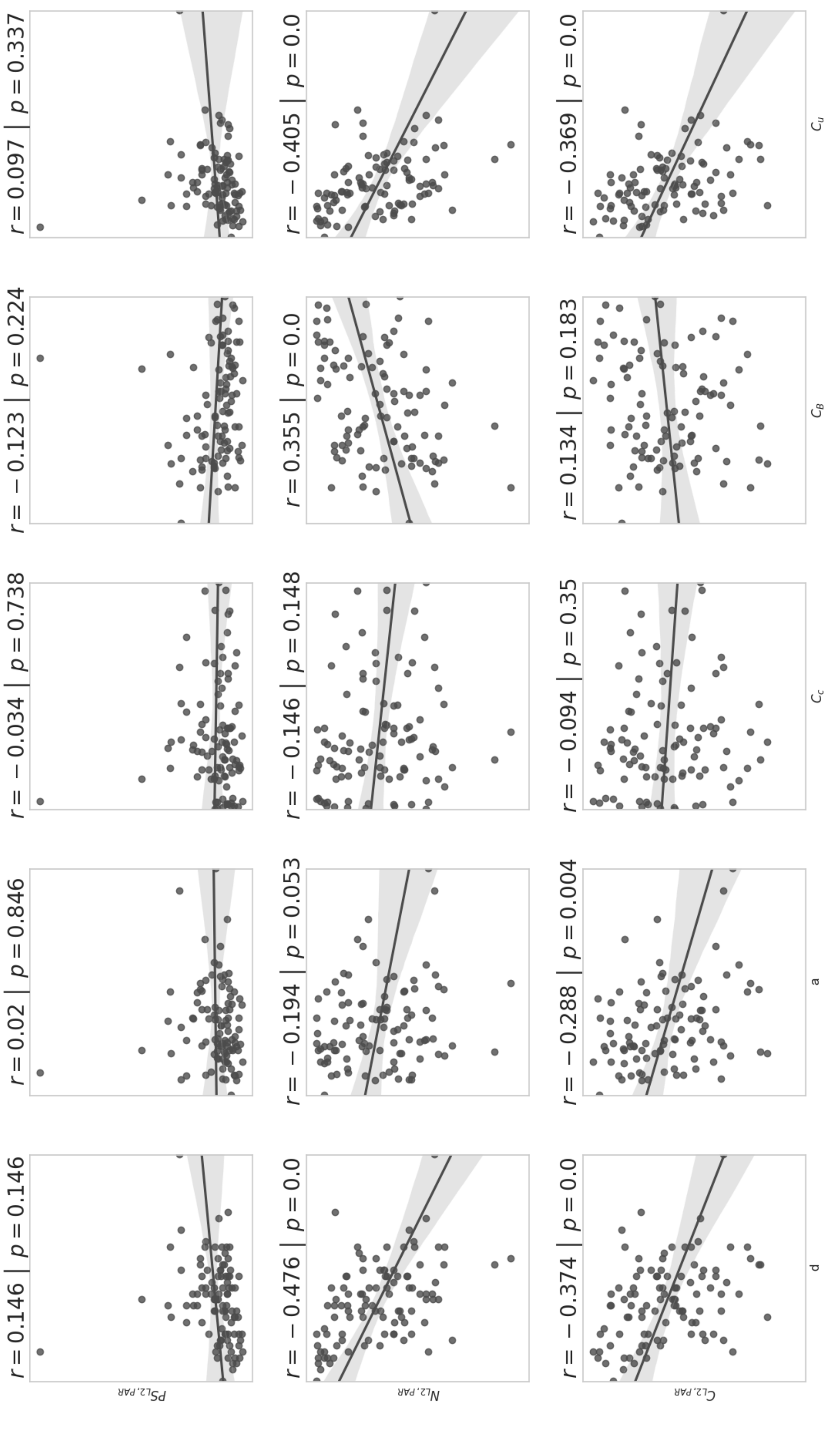


Figure A.4: Net3 variants global crest factor of failure to topological attributes

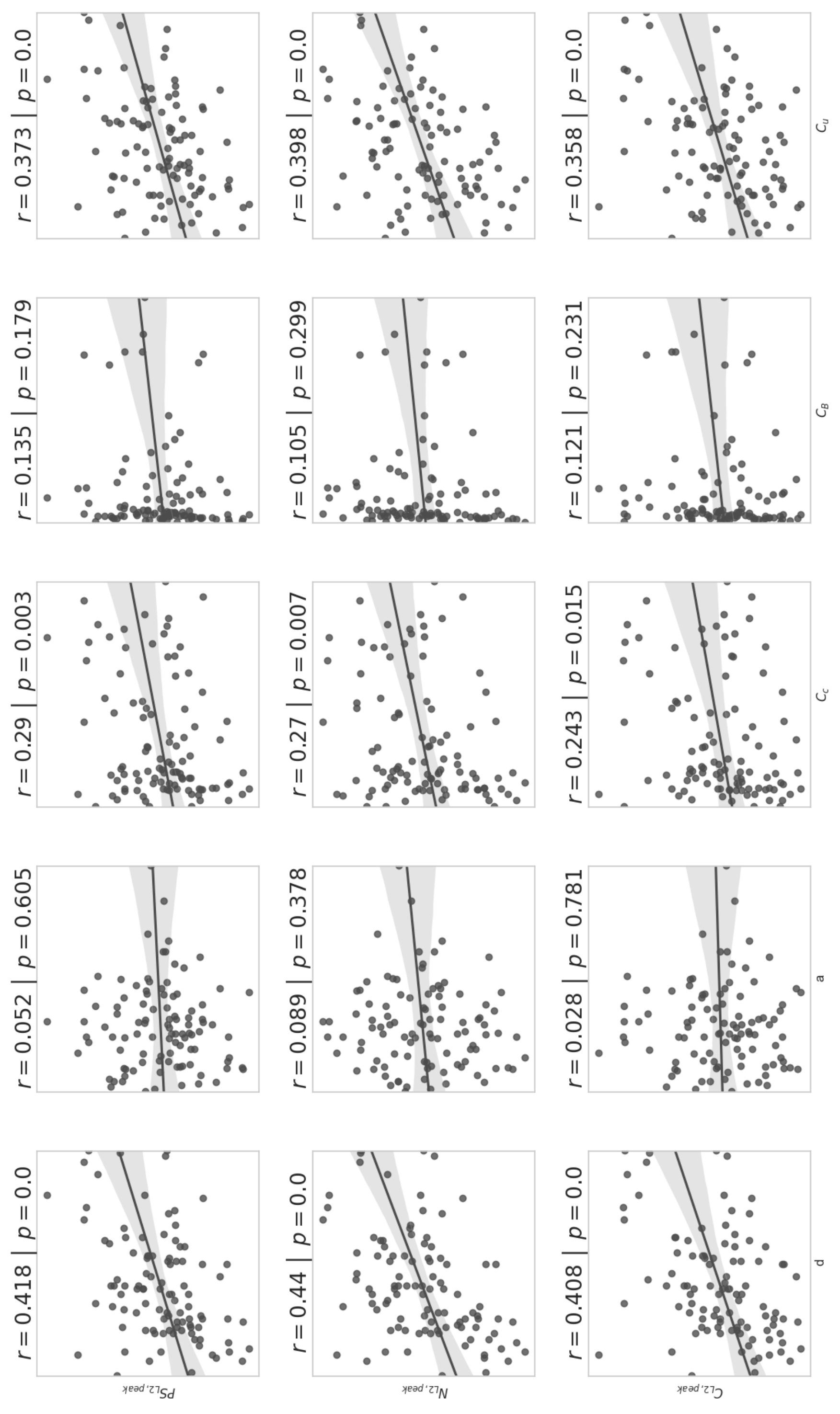


Figure A.5: Ctown variants global severity of failure to topological attributes.

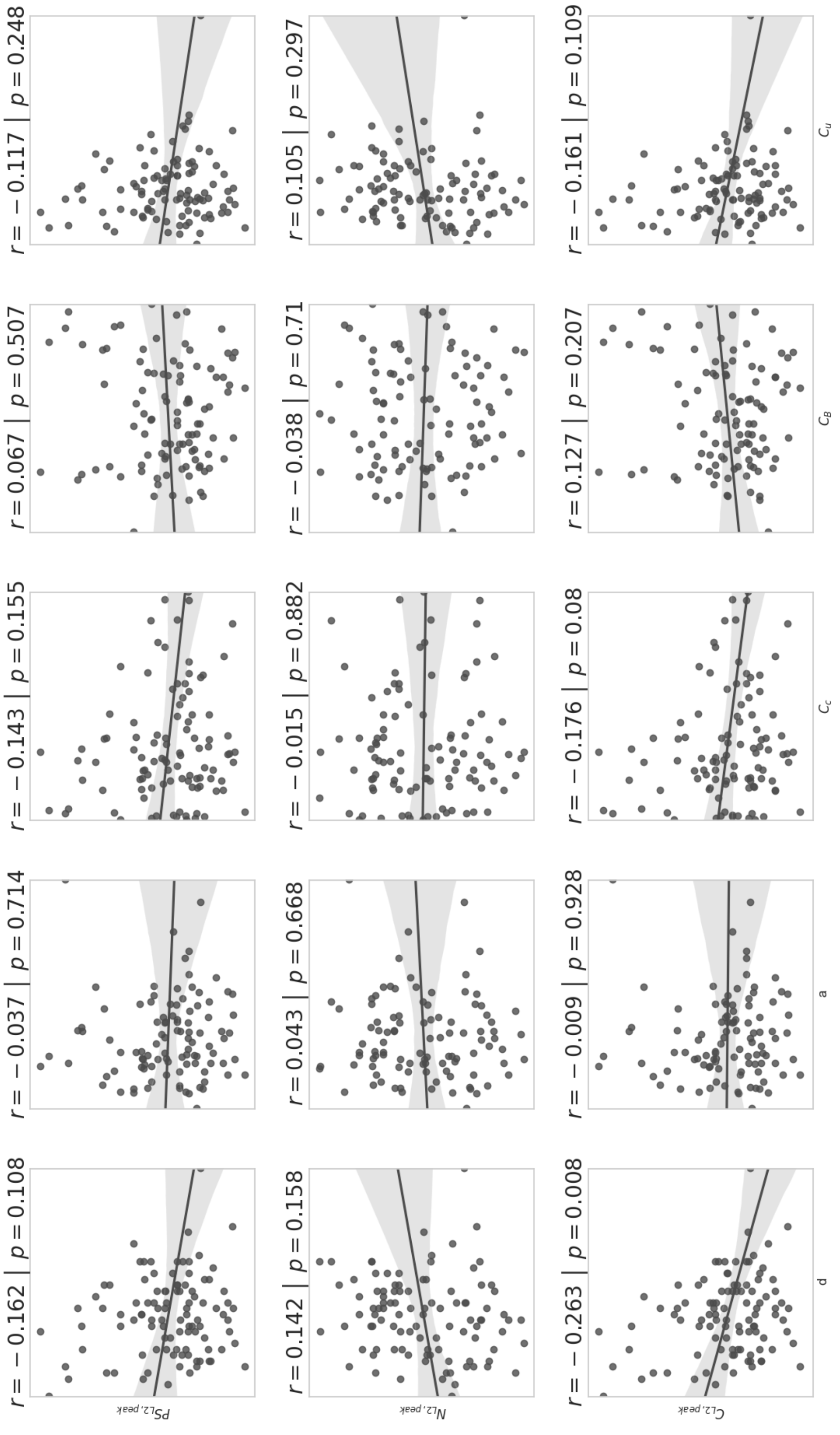


Figure A.6: Net3 variants global severity of failure to topological attributes.

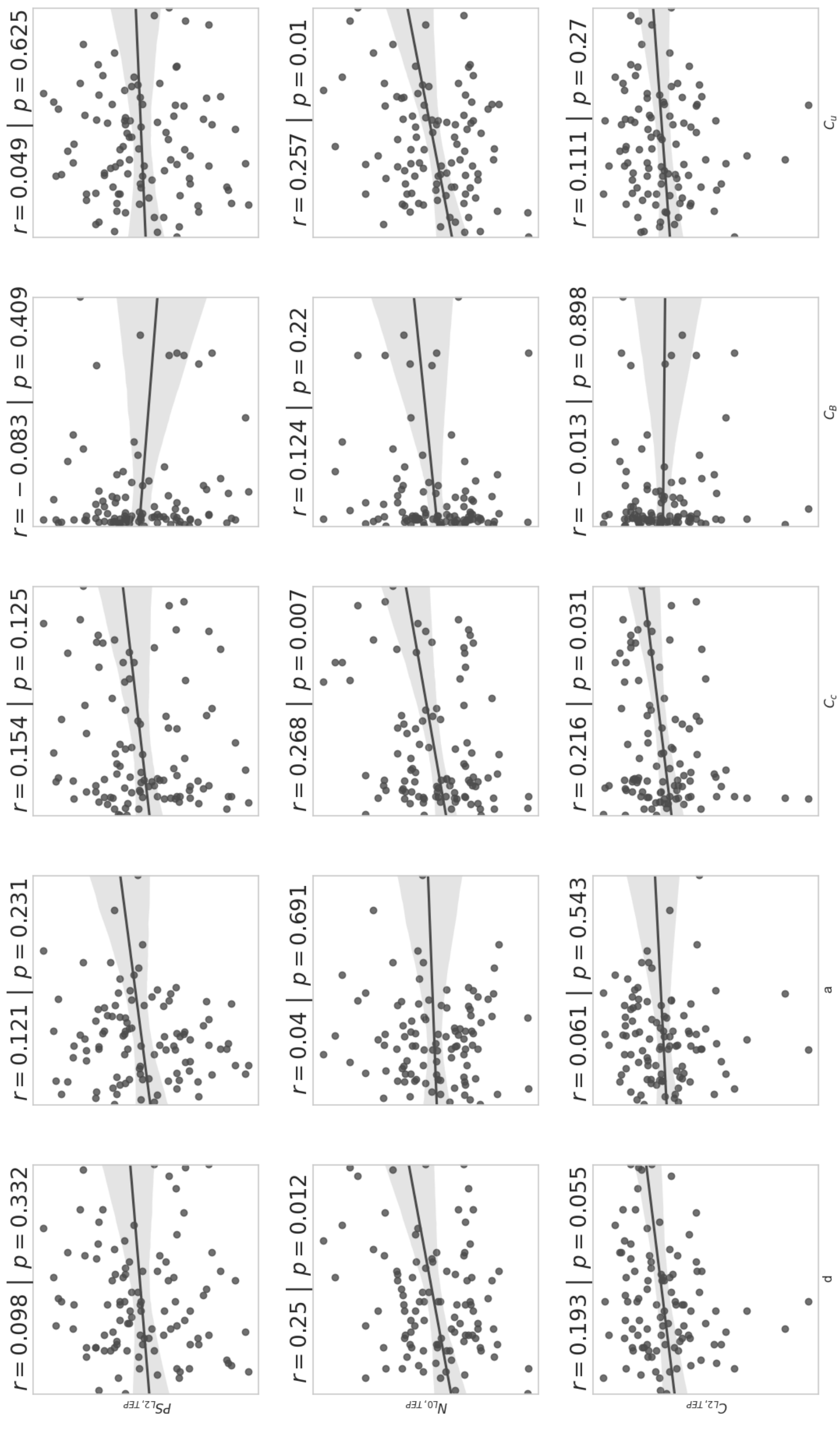


Figure A.7: Ctown variants global rapidity of failure to topological attributes.

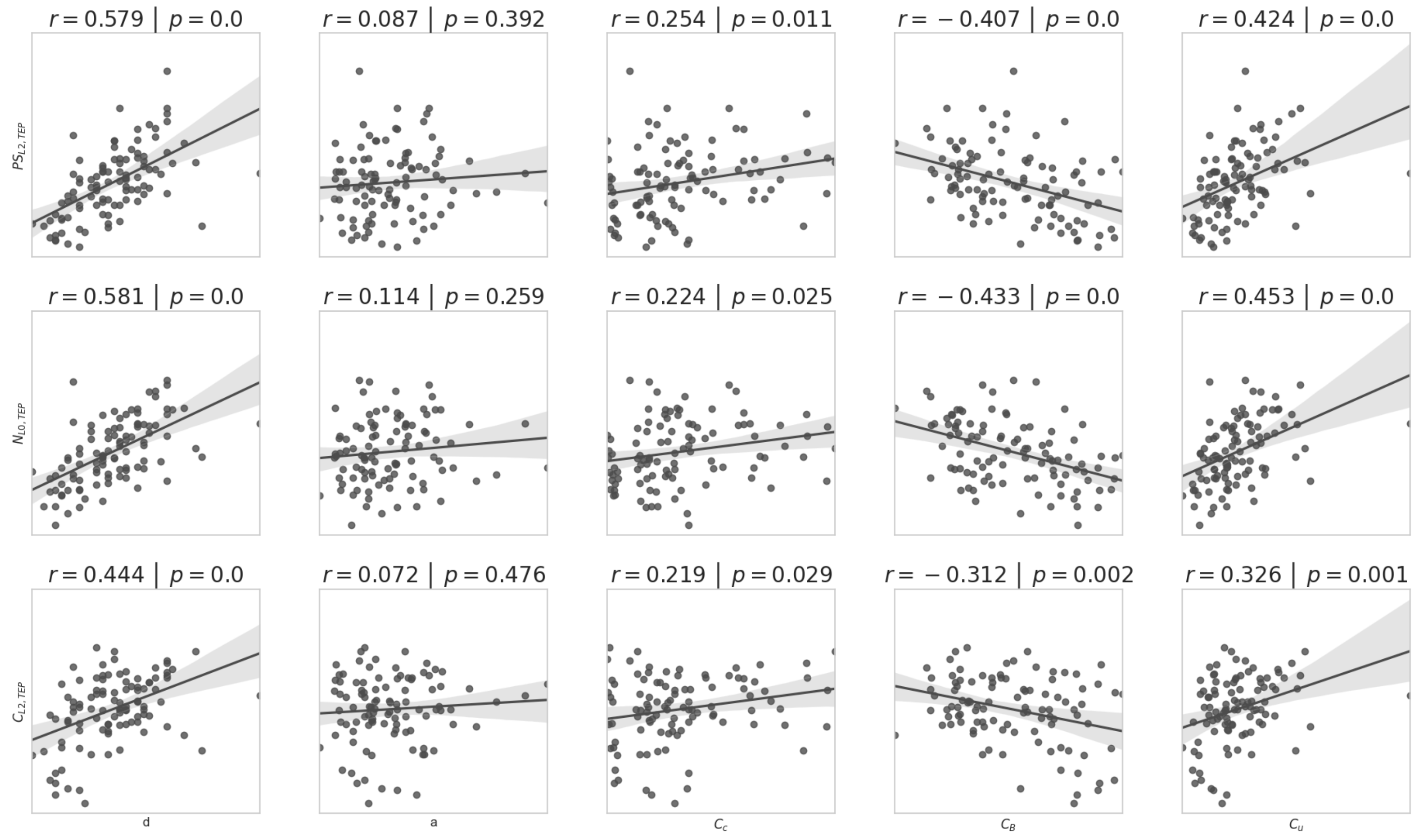


Figure A.8: Net3 variants global rapidity of failure to topological attributes.

Appendix B

Ranking of network variants

Appendix B presents the network variants ranked by performance. Figure B.2 presents the Ctown variants ranked by magnitude of failure in ascending order (e.i., the service and the spatial dimension). Figure B.2 presents the Ctown variants ranked by severity of failure in ascending order (e.i., the service and the spatial dimension). Figure B.3 presents the Ctown variants ranked by rapidity of failure in descending order (e.i., the service and the spatial dimension). Identical to the ranking of Ctown variants, Figure B.4, Figure B.5, and Figure B.6 presents Net3 variants ranked by performance.

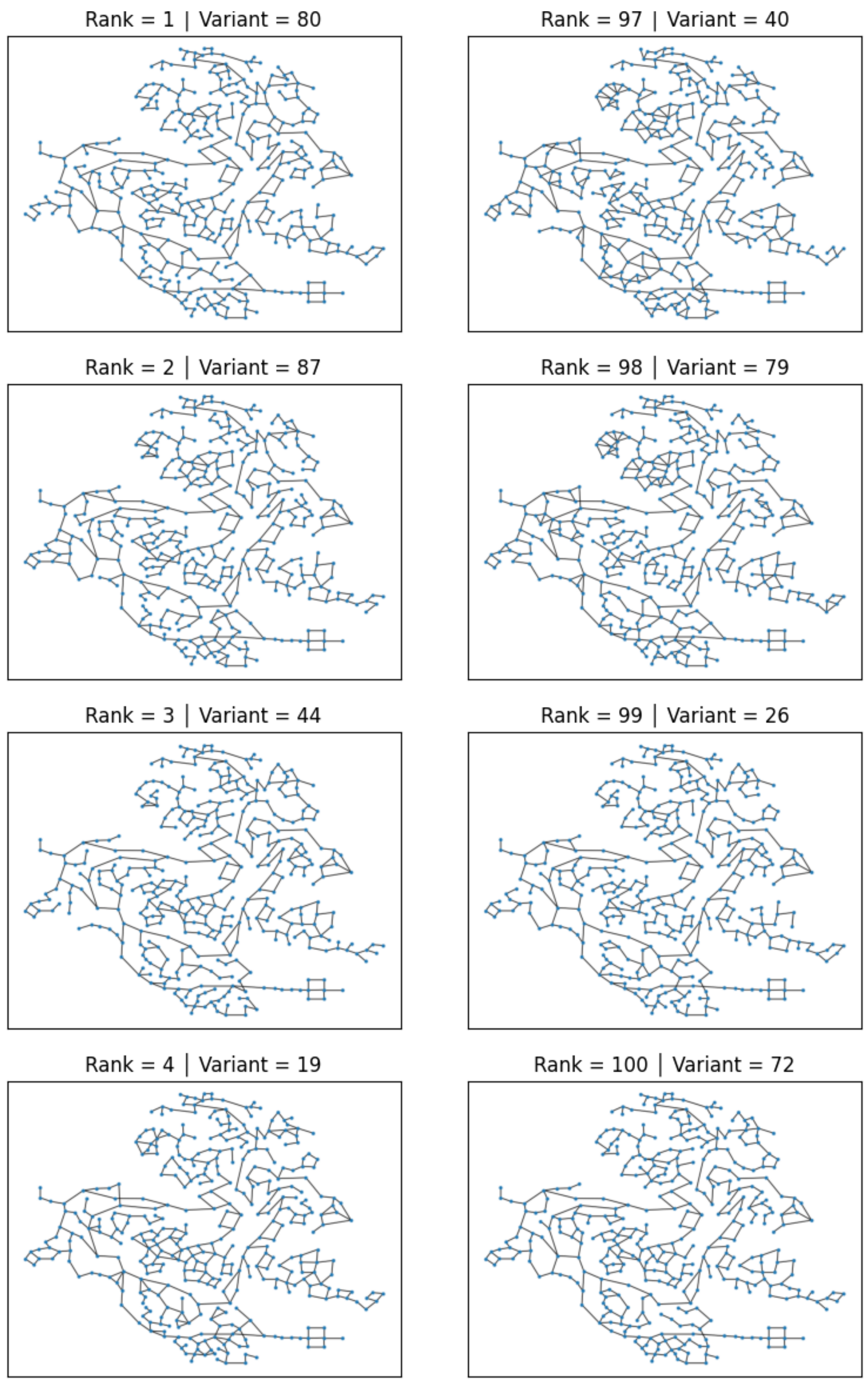


Figure B.1: Ctown variants ranked by ascending $PS_{L0,ratio}$ and $N_{L0,n,ratio}$.

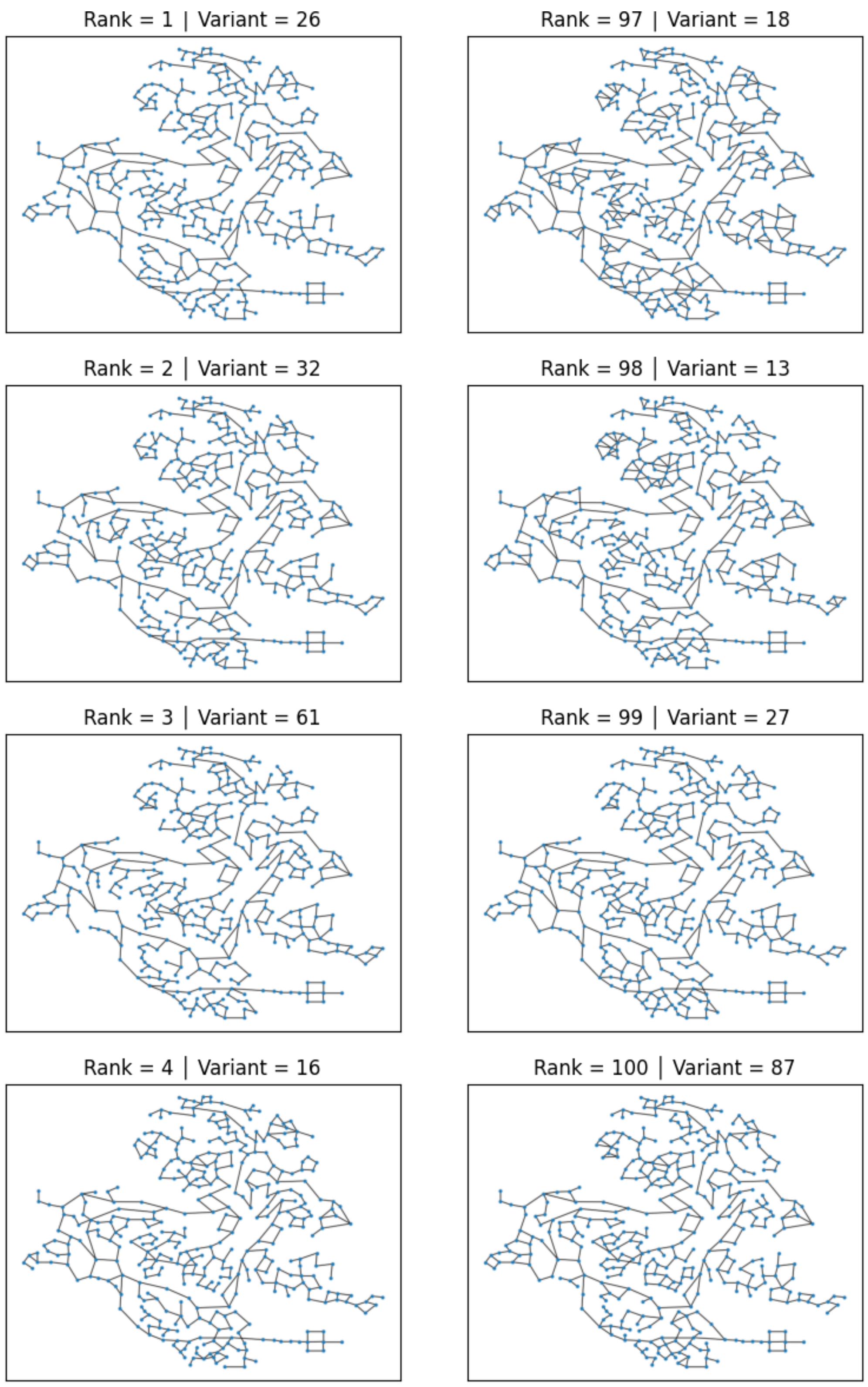


Figure B.2: Ctown variants ranked by ascending $PS_{L0,peak}$ and $N_{L0,peak}$.

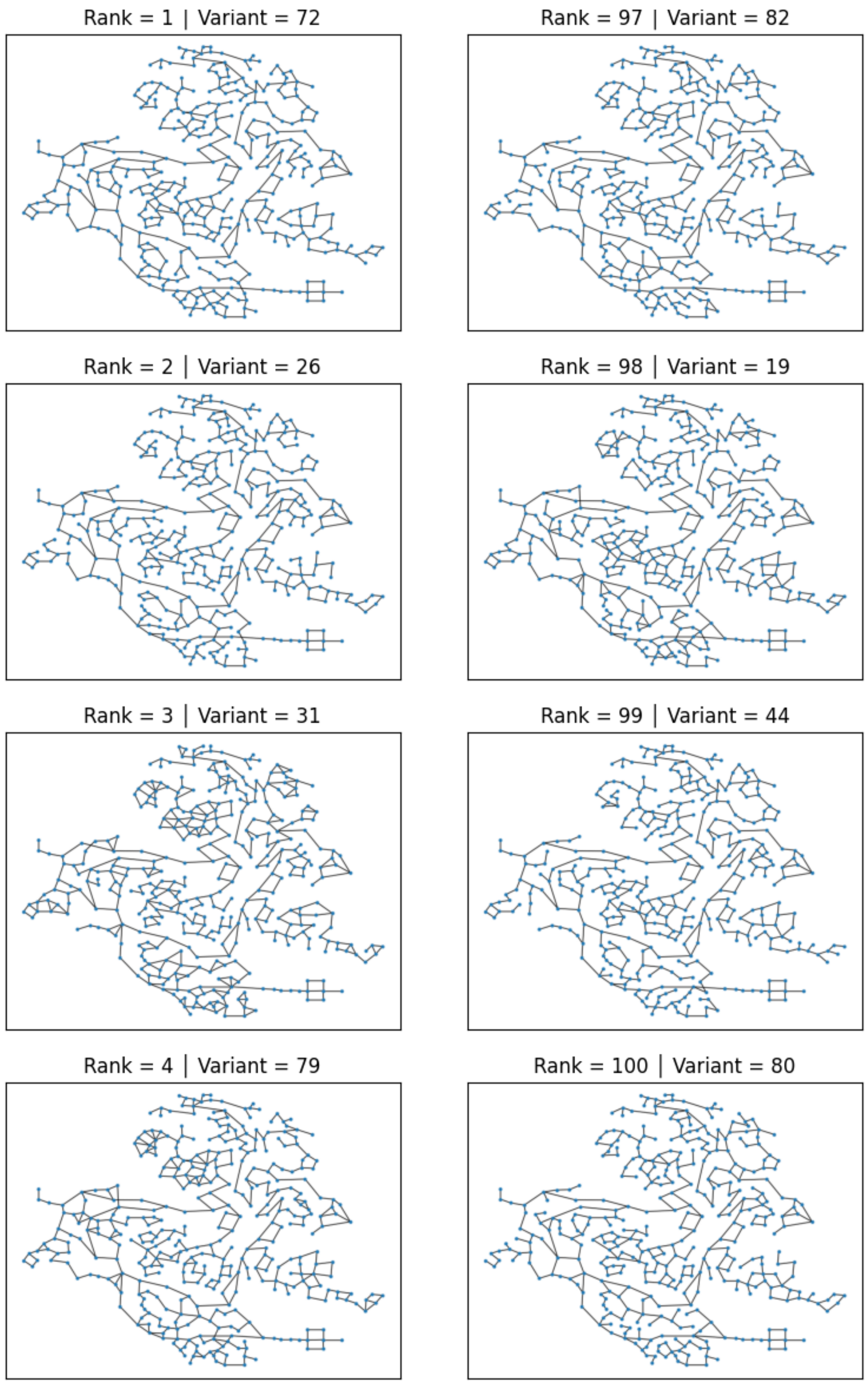


Figure B.3: Ctown variants ranked by descending $PS_{L0,TEP}$ and $N_{L0,TEP}$.

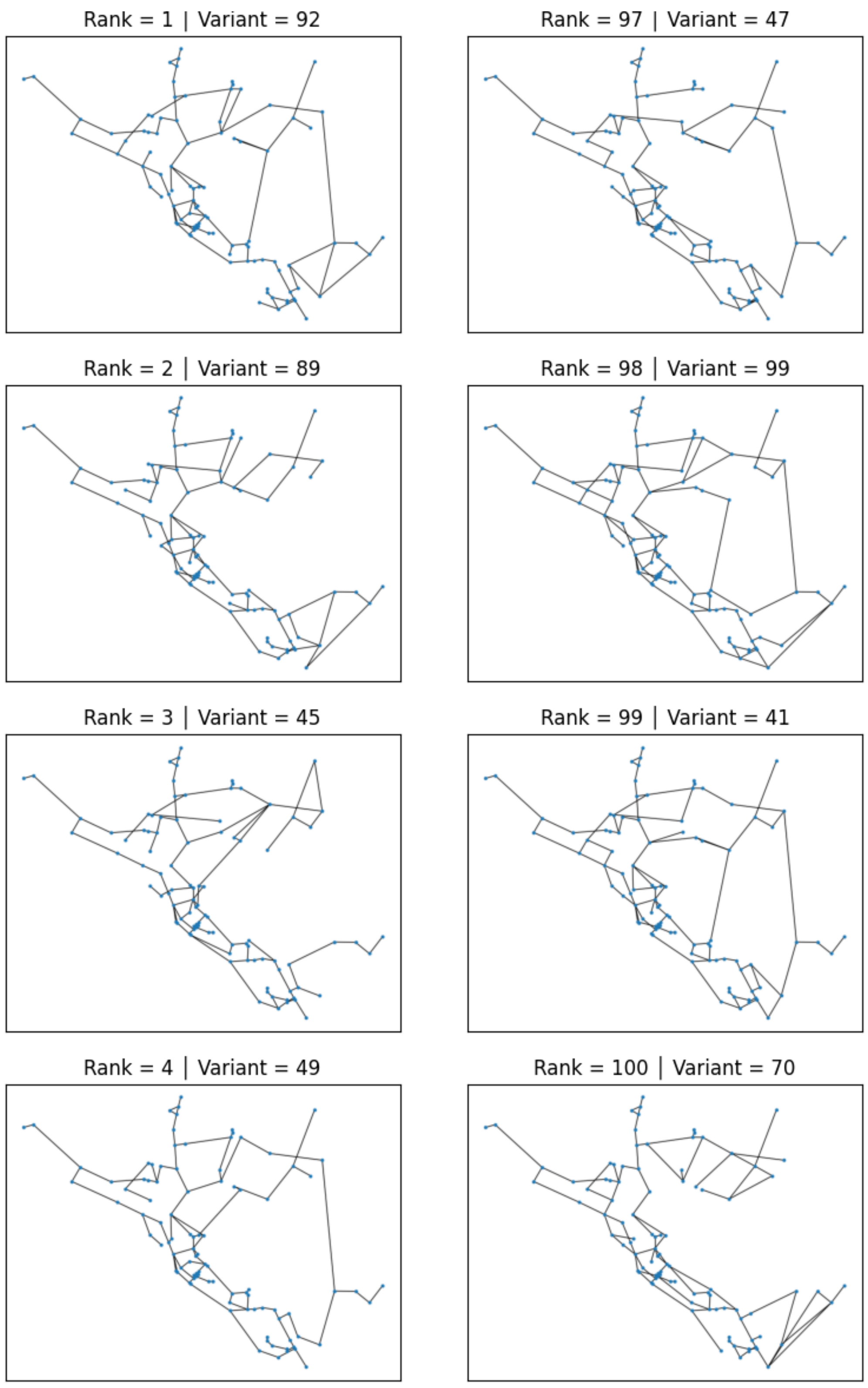


Figure B.4: Net3 variants ranked by ascending $PS_{L0,ratio}$ and $N_{L0,n,ratio}$.

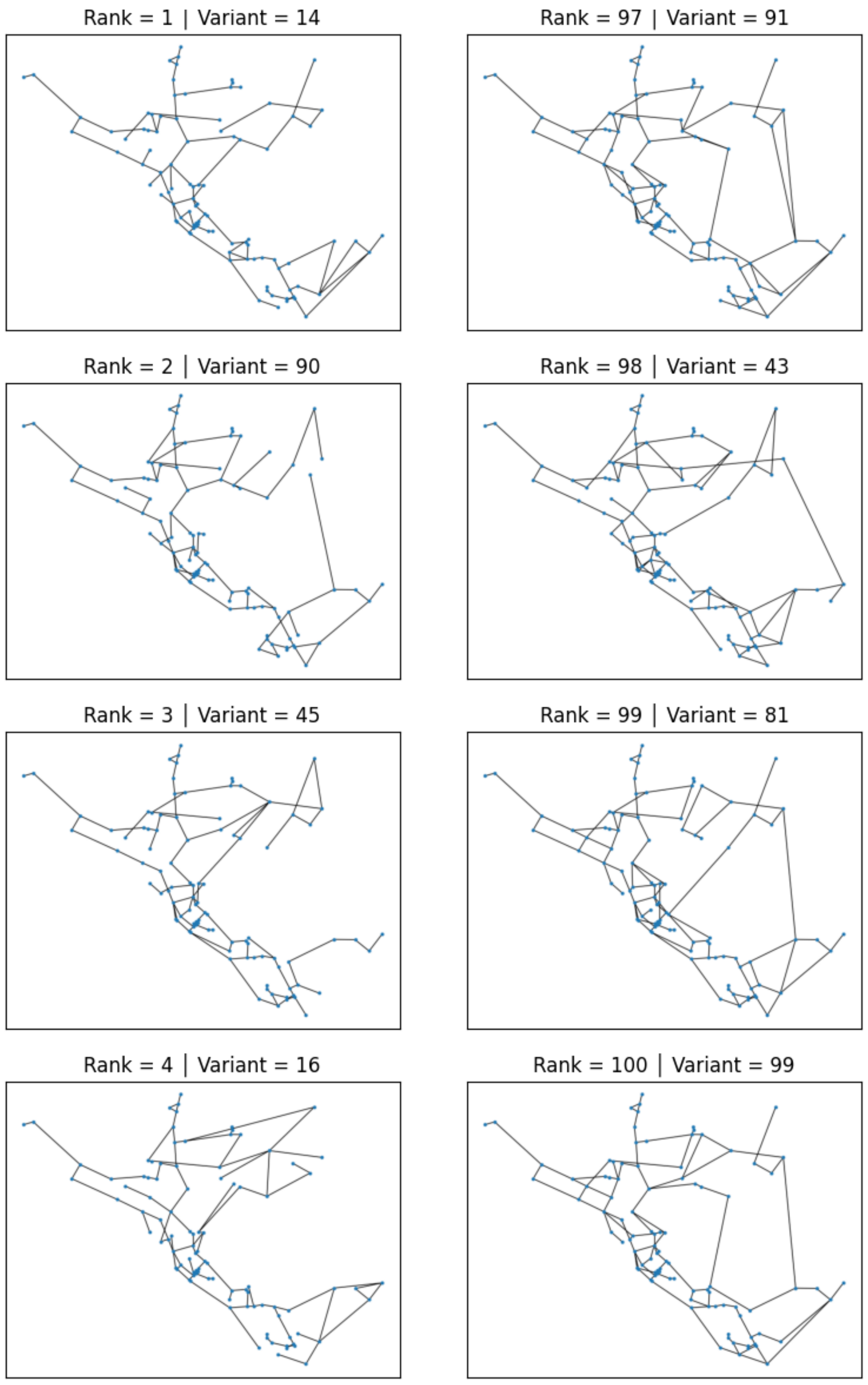


Figure B.5: Net3 variants ranked by ascending $PS_{L0,peak}$ and $N_{L0,peak}$.

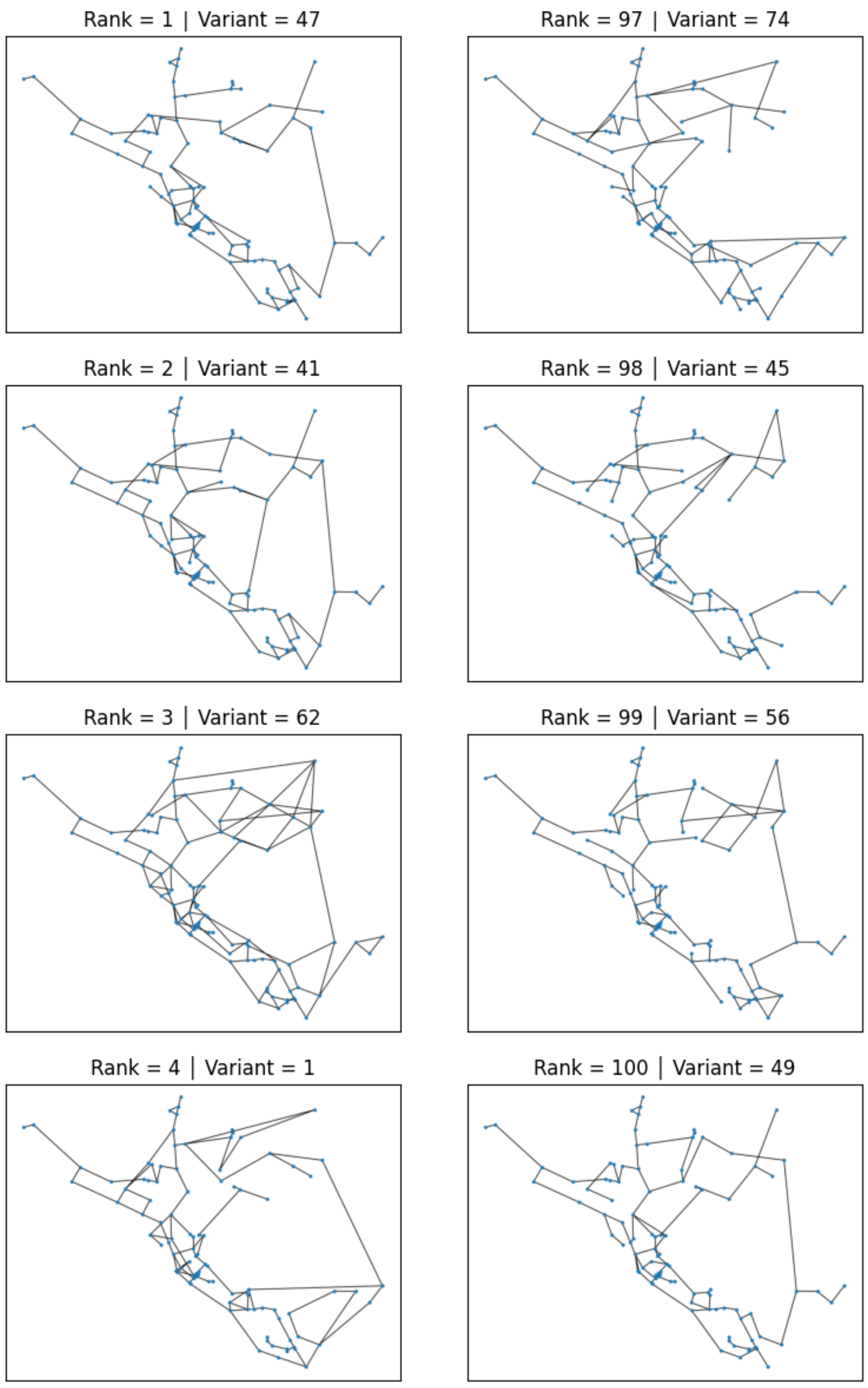


Figure B.6: Net3 variants ranked by descending $PS_{L0,TEP}$ and $N_{L0,TEP}$.

Appendix C

Computer and code listings

C.1 Computer and Python packages

The tools used in this study is a Lenovo ThinkPad X1 Carbon Generation 8 Intel Core i7 2.00 GHz processor, 16.0 GB RAM, and SSD storage. Python 3.7 qith a Anaconda interpreter is used for all simulations, calculations, and data visualisation. Table C.1 present a overview the file name, description, chapter and listing of the codes developed under the course of this thesis. The following list presents a short description of the Python packages extensively used in all scripts.

- *WNTR v0.3.1* is based on the EPANET solver used to simulate hydraulic and water quality behaviour.
- *NetworkX v2.5.0* is used to generate and study topological attributes of water distribution networks.
- *NumPy v1.20.0* is a part of the library PyLab, and is applied to solve mathematical operations.
- *Pandas v1.2.4* is applied to structure, manage, and store produced data.
- *SeaBorn v1.20.0* and *Matplotlib v3.4.1* functionalities is used to represent and visualise data.
- *SciPy v1.20.0* is used to calculate the Spearman correlation strength and direction.

Table C.1: Summary of attached algorithms.

File	Description	Chapter	Listings
config.py	Configuration of input file	Chapter 4	Code listing C.1
generation.py	Groups nodes by service areas	Chapter 4	Code listing C.2
generation.py	Network generation procedure	Chapter 4	Code listing C.2
stress.py	Random failure sequences/KPI's	Chapter 7	Code listing C.3
attributes.py	Topological indicators	Chapter 7	Code listing C.4

C.2 Listings

Code listing C.1: Code listing of config.py script.

```
import re
import io
import os, sys
from collections import OrderedDict
from datetime import datetime

INP_SECTIONS = ['[OPTIONS]', '[TITLE]', '[JUNCTIONS]', '[RESERVOIRS]',
                 '[TANKS]', '[PIPES]', '[PUMPS]', '[VALVES]', '[EMITTERS]',
                 '[CURVES]', '[PATTERNS]', '[ENERGY]', '[STATUS]',
                 '[CONTROLS]', '[RULES]', '[DEMANDS]', '[QUALITY]',
                 '[REACTIONS]', '[SOURCES]', '[MIXING]',
                 '[TIMES]', '[REPORT]', '[COORDINATES]', '[VERTICES]',
                 '[LABELS]', '[BACKDROP]', '[TAGS]']




class InpFile(object):
    """
    EPANET INP file reader and writer class for ID convert based on WNTR package
    This class provides read and write functionality for EPANET INP files.
    The EPANET Users Manual provides full documentation for the INP file format.
    """
    def __init__(self, input, output):
        self.sections = OrderedDict()
        self.nodes = OrderedDict()
        self.links = OrderedDict()
        self.patterns = OrderedDict()
        self.curves = OrderedDict()

        for sec in INP_SECTIONS:
            self.sections[sec] = []
        self.read(input)
        self.write((output))

    def read(self, input):
        section = None
        with io.open(input, 'r', encoding='utf-8') as f:
            for lnum, line in enumerate(f):
                line = line.strip()
                nwords = len(line.split())
                if len(line) == 0 or nwords == 0:
                    # Blank line
                    continue
                elif line.startswith('['):
                    vals = line.split(None, 1)
                    sec = vals[0].upper()
                    if sec in INP_SECTIONS:
                        section = sec
                        self.sections[section] = []
                        continue
                    elif sec == '[END]':
                        section = None
                        break
                    else:
```

```
                    #print('Section {} not included'.format(sec))
                    section = None
            elif section is None and line[0] == ';':
                # Top comment
                continue


            if section is not None:
                self.sections[section].append(line)

        self.format_patterns()
        self.format_junctions()
        self.format_reservoirs()
        self.format_tanks()
        self.format_curves()
        self.format_pipes()
        self.format_pumps()
        self.format_valves()
        self.format_demands()
        self.format_status()
        self.format_controls()
        self.format_rules()
        self.format_mixing()
        self.format_coordinates()

    def format_patterns(self):
        lnum = 1
        patterns = OrderedDict()
        for line in self.sections['[PATTERNS]']:
            line = line.split(';')[0]
            current = line.split()
            if current == []:
                continue
            pattern_name = current[0]
            if pattern_name not in self.patterns.keys():
                self.patterns[pattern_name] = 'PAT{}'.format(lnum)
                patterns[self.patterns[pattern_name]] = []
                lnum += 1
                for i in current[1:]:
                    patterns[self.patterns[pattern_name]].append(i)
            else:
                for i in current[1:]:
                    patterns[self.patterns[pattern_name]].append(i)

        format = list()
        num_columns = 6
        format.append('{:10s}␣{:10s}'.format(';ID', 'Multipliers'))
        for pattern_name in patterns.keys():
            string = ''
            for lnum, multiplier in enumerate(patterns[pattern_name]):
                if lnum % num_columns == 0:
                    string += '\n{:10s}␣{:10s}'.format(
                        pattern_name, multiplier
                    )
                else:
                    string += '{:10s}'.format(multiplier)
            format.append('\n;')
            format.append(string)
        self.sections['[PATTERNS]'] = format

    def format_junctions(self):
        count = 1
        junctions = ['{:10s}␣{:10s}\n'.format(';ID', 'Elevation')]
        for line in self.sections['[JUNCTIONS]']:
            line = line.split(';')[0]
            current = line.split()
            if current == []:
                continue
            if current[0] not in self.nodes:
                id = 'J{}'.format(count)
                self.nodes[current[0]] = id
                junctions.append('{:10s}␣{:10s}\t;\n'.format(id, current[1]))
                count += 1

        self.sections['[JUNCTIONS]'] = junctions

    def format_reservoirs(self):
        count = 1
        reservoir = ['{:10s}␣{:15s}␣{:15s}\n'.format(';ID', 'Elevation', 'Pattern')]
        for line in self.sections['[RESERVOIRS]']:
            line = line.split(';')[0]
            current = line.split()
            if current == []:
                continue
            if current[0] not in self.nodes:
                id = 'R{}'.format(count)
                count += 1
                self.nodes[current[0]] = id
                if len(current) == 2:
                    reservoir.append('{:10s}␣{:15s}␣{:15s};\n'.format(
                        id, current[1], ''
                    ))
                elif len(current) == 3:
                    reservoir.append('{:10s}␣{:15s}␣{:15s};\n'.format(
                        id, current[1], self.patterns[current[2]]
                    ))
                else:
                    print('Check␣Reservoir:␣', line)

        self.sections['[RESERVOIRS]'] = reservoir

    def format_tanks(self):
        count = 1
        tank_format = '{:10s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}\n'
        tanks = [tank_format.format(
            ';ID', 'Elevation', 'MinLevel', 'MaxLevel',
            'Diameter', 'MinVol', 'VolCurve'
        )]

        tank_format = '{:10s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}\t;\n'
        for line in self.sections['[TANKS]']:
            line = line.split(';')[0]
            current = line.split()
            if current == []:
                continue
            id = 'T{}'.format(count)
            count += 1
            self.nodes[current[0]] = id
            tanks.append(tank_format.format(
                id, current[1], current[2], current[3], current[4], current[5], '')
            )
        self.sections['[TANKS]'] = tanks

    def format_curves(self):
        count = 1
        curves = ['{:10s}␣{:15s}␣{:15s}␣\n'.format(';ID', 'X-Value', 'Y-Value')]

        for line in self.sections['[CURVES]']:
            line = line.split(';')[0]
            current = line.split()
            if current == []:
                continue
            if current[0] not in self.curves.keys():
                id = 'CURVE_{}'.format(count)
                self.curves[current[0]] = id
                curves.append(
                    '{:10s}␣{:15s}␣{:15s}␣\n'.format(id, current[1], current[2])
                )
                count += 1
            else:
                curves.append('{:10s}␣{:15s}␣{:15s}␣\n'.format(
                    self.curves[current[0]], current[1], current[2]
                ))

        self.sections['[CURVES]'] = curves

    def format_pipes(self):
        count = 1
        pipe_format = '{:10s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}\n'
        pipes = [pipe_format.format(
            ';ID', 'Node1', 'Node2', 'Length',
            'Diameter', 'Roughness', 'MinorLoss', 'CV'
        )]

        pipe_format = '{:10s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}\t;\n'
        for line in self.sections['[PIPES]']:
            line = line.split(';')[0]
            current = line.split()
            if current == []:
                continue

            id = 'P{}'.format(count)
            count += 1
            node1 = self.nodes[current[1]]
            node2 = self.nodes[current[2]]
            self.links[current[0]] = id

            if len(current) == 7:
                pipes.append(pipe_format.format(
                    id, node1, node2, current[3], current[4],
                    current[5], current[6], ''
                ))
            if len(current) == 8:
                pipes.append(pipe_format.format(
                    id, node1, node2, current[3], current[4],
                    current[5], current[6], current[7]
            ))

        self.sections['[PIPES]'] = pipes

    def format_pumps(self):
        count = 1
        pumps = ['{:10s}␣{:15s}␣{:15s}␣{:15s}␣\n'.format(
            ';ID', 'Node1', 'Node2', 'Parameters'
        )]

        for line in self.sections['[PUMPS]']:
            line = line.split(';')[0]
            current = line.split()
            if current == []:
                continue

            id = 'PU{}'.format(count)
            count += 1
            node1 = self.nodes[current[1]]
            node2 = self.nodes[current[2]]
            self.links[current[0]] = id


            pumps.append('{:10s}␣{:15s}␣{:15s}␣{:5s}{:15s}\t;\n'.format(
                id, node1, node2, current[3], self.curves[current[4]]
            ))

        self.sections['[PUMPS]'] = pumps

    def format_valves(self):
        count = 1
        valves = ['{:10s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣\n'.format(
            ';ID', 'Node1', 'Node2', 'Diameter', 'Type', 'Setting', 'MinorLoss'
        )]

        for line in self.sections['[VALVES]']:
            line = line.split(';')[0]
            current = line.split()
            if current == []:
                continue

            id = 'V{}'.format(count)
            self.links[current[0]] = id
            count += 1
            node1 = self.nodes[current[1]]
            node2 = self.nodes[current[2]]
            valves.append(
                '{:10s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣{:15s}␣;\n'.format(
                id, node1, node2, current[3], current[4], current[5], '0'
            ))

        self.sections['[VALVES]'] = valves

    def format_demands(self):
        count = 1
        demands = ['{:10s}␣{:15s}␣{:15s}␣{:15s}\n'.format(
            ';ID', 'Demand', 'Pattern', 'Category'
        )]
        for line in self.sections['[DEMANDS]']:
            line = line.split(';')
            current = line[0].split()
            if current == []:
                continue
            if current[0] in self.nodes.keys():
                id = self.nodes[current[0]]
                if len(current) == 3:
                    pattern = self.patterns[current[2]]
                else:
                    pattern = ’’

                if len(line) > 1:
                    if line[1].upper() == ’LEKKASJE’:
                        demands.append(’{:10s}␣{:15s}␣{:15s};␣{:15s}\n’.format(
                            id, current[1], pattern, ’leakage’
                        ))
                    elif line[1].upper() == ’FASTBOENDE’:
                        demands.append(’{:10s}␣{:15s}␣{:15s};␣{:15s}\n’.format(
                            id, current[1], pattern, ’residential’
                        ))
                    elif line[1].upper() == ’VIRKSOMHET’:
                        demands.append(’{:10s}␣{:15s}␣{:15s};␣{:15s}\n’.format(
                            id, current[1], pattern, ’enterprise’
                        ))
                    else:
                        demands.append(’{:10s}␣{:15s}␣{:15s};␣{:15s}\n’.format(
                            id, current[1], pattern, ’other’
                        ))
                else:
                    demands.append(’{:10s}␣{:15s}␣{:15s};␣\n’.format(
                        id, current[1], pattern
                    ))


                count += 1

        self.sections[’[DEMANDS]’] = demands

    def format_status(self):
        count = 1
        status = [’{:10s}␣{:10s}\n’.format(’;ID’, ’Status/Setting’)]
        for line in self.sections[’[STATUS]’]:
            line = line.split(’;’)[0]
            current = line.split()
            if current == []:
                continue
            if current[0] not in self.nodes:
                id = self.links[current[0]]

                status.append(’{:10s}␣{:10s}\n’.format(id, current[1]))
                count += 1

        self.sections[’[STATUS]’] = status

    def format_controls(self):
        count = 1
        control = []
        controls = list()
        for line in self.sections[’[CONTROLS]’]:
            line = line.split(’;’)[0]
            current = line.split()
            if current == []:
                continue
            elif len(current) == 7: # Clocktime
                link = self.links[current[1]]
                if link not in control:
                    controls.append(
                        ';␣------------␣Clocktime␣{}␣---------------␣\n'.format(
                            link
                        ))
                    control.append(link)
                controls.append(
                    '{:1s}\t{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}␣\n'.format(
                        current[0], link, current[2], current[3],
                        current[4], current[5], current[6]
                ))
            elif len(current) == 8:
                link = self.links[current[1]]
                node = self.nodes[current[5]]
                if node not in control:
                    controls.append(
                        ';␣----------␣Tank␣{}␣---------------␣\n'.format(node[1:]))
                    control.append(node)
                controls.append(
                    '{:1s}\t{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}\n'.format(
                    current[0], link, current[2], current[3],
                    current[4], node, current[6], current[7]
                ))
            else:
                print('Check␣control:␣', line)
        self.sections['[CONTROLS]'] = controls

    def format_rules(self):
        def string(words):
            if words[1].upper() == 'TANK':
                return '{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}\n'.format(
                    words[0], words[1], self.nodes[words[2]],
                    words[3], words[4], words[5]
                )

            elif words[1].upper() == 'SYSTEM':
                return '{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}\n'.format(
                    words[0], words[1], words[2],
                    words[3], words[4]
                )

            elif words[1].upper() == 'LINK':
                return '{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}\n'.format(
                    words[0], words[1], self.links[words[2]],
                    words[3], words[4], words[5]
                )
            elif words[1].upper() == 'VALVE':
                return '{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}\n'.format(
                    words[0], words[1], self.links[words[2]],
                    words[3], words[4], words[5]
                )

            elif words[1].upper() == 'PUMP':
                return '{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}␣{:1s}\n'.format(
                    words[0], words[1], self.links[words[2]],
                    words[3], words[4], words[5]
                )
            else:
                print(words[1])

        rules = []
        if len(self.sections['[RULES]']) > 0:
            rule = 1
            for line in self.sections['[RULES]']:
                line = line.split(';')[0]
                words = line.split()

                if words[0].upper() == 'RULE':
                    rules.append('\nRULE␣{}␣\t␣;\n'.format(words[1]))

                elif words[0].upper() == 'PRIORITY':
                    rules.append('{}␣{}\n'.format(words[0], words[1]))
                else:
                    rules.append(string(words))


        self.sections['[RULES]'] = rules

    def format_mixing(self):

        count = 1
        mixing = ['{:10s}␣{:15s}\n'.format(';Tank', 'Model')]

        for line in self.sections['[MIXING]']:
            line = line.split(';')
            current = line[0].split()
            if current == []:
                continue

            mixing.append('{:10s}␣{:15s}\n'.format(
                self.nodes[current[0]], current[1]
            ))

        self.sections['[MIXING]'] = mixing

    def format_coordinates(self):
        count = 1
        coordinates = ['{:10s}␣{:15s}␣{:15s}\n'.format(
            ';ID', 'X-Coord', 'Y-Coord'
        )]

        for line in self.sections['[COORDINATES]']:
            line = line.split(';')
            current = line[0].split()
            if current == []:
                continue

            coordinates.append('{:10s}␣{:15s}␣{:15s}\n'.format(
                self.nodes[current[0]], current[1], current[2]
            ))

        self.sections['[COORDINATES]'] = coordinates

    def write(self, output):

        with io.open(output, 'wb') as f:
```

```
now = datetime.now().strftime("%d/%m/%Y␣%H:%M:%S")
f.write(';␣Generated:␣{}\n'.format(now).encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[TITLE]\n'.encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[JUNCTIONS]\n'.encode('ascii'))
for line in self.sections['[JUNCTIONS]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[RESERVOIRS]\n'.encode('ascii'))
for line in self.sections['[RESERVOIRS]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[TANKS]\n'.encode('ascii'))
for line in self.sections['[TANKS]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[PIPES]\n'.encode('ascii'))
for line in self.sections['[PIPES]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[PUMPS]\n'.encode('ascii'))
for line in self.sections['[PUMPS]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[VALVES]\n'.encode('ascii'))
for line in self.sections['[VALVES]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[TAGS]\n'.encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[DEMANDS]\n'.encode('ascii'))
for line in self.sections['[DEMANDS]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[STATUS]\n'.encode('ascii'))
for line in self.sections['[STATUS]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[PATTERNS]\n'.encode('ascii'))
for line in self.sections['[PATTERNS]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))
f.write('\n'.encode('ascii'))

f.write('[CURVES]\n'.encode('ascii'))
for line in self.sections['[CURVES]']:
    f.write('{}'.format(line).encode('ascii'))
f.write('\n'.encode('ascii'))
```

```
        f.write('[CONTROLS]\n'.encode('ascii'))
        for line in self.sections['[CONTROLS]']:
            f.write('{}'.format(line).encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[RULES]\n'.encode('ascii'))
        for line in self.sections['[RULES]']:
            f.write('{}'.format(line).encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[ENERGY]\n'.encode('ascii'))
        for line in self.sections['[ENERGY]']:
            f.write('{}'.format(line).encode('ascii'))
            f.write('\n'.encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[EMITTERS]\n'.encode('ascii'))
        f.write('{:10s}␣{:10s}\n'.format(
            ';Junction', 'Coefficient').encode('ascii')
                )
        f.write('\n'.encode('ascii'))

        f.write('[QUALITY]\n'.encode('ascii'))
        f.write('{:10s}␣{:10s}\n'.format(';Node', 'InitQual').encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[SOURCES]\n'.encode('ascii'))
        f.write('{:10s}␣{:10s}␣{:10s}␣{:10s}\n'.format(
            ';Node', 'Type', 'Quality', 'Pattern').encode('ascii')
                )
        f.write('\n'.encode('ascii'))

        f.write('[REACTIONS]\n'.encode('ascii'))
        for line in self.sections['[REACTIONS]']:
            f.write('{}'.format(line).encode('ascii'))
            f.write('\n'.encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[MIXING]\n'.encode('ascii'))
        for line in self.sections['[MIXING]']:
            f.write('{}'.format(line).encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[TIMES]\n'.encode('ascii'))
        for line in self.sections['[TIMES]']:
            f.write('{}'.format(line).encode('ascii'))
            f.write('\n'.encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[REPORT]\n'.encode('ascii'))
        for line in self.sections['[REPORT]']:
            f.write('{}'.format(line).encode('ascii'))
            f.write('\n'.encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[OPTIONS]\n'.encode('ascii'))
        for line in self.sections['[OPTIONS]']:
            f.write('{}'.format(line).encode('ascii'))
            f.write('\n'.encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[COORDINATES]\n'.encode('ascii'))
        for line in self.sections['[COORDINATES]']:
            f.write('{}'.format(line).encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[VERTICES]\n'.encode('ascii'))
        f.write('{:10s}␣{:10s}␣{:10s}\n'.format(
            ';ID', 'X-Coord', 'Y-Coord').encode('ascii')
                )
        f.write('\n'.encode('ascii'))

        f.write('[LABELS]\n'.encode('ascii'))
        f.write('{:10s}␣{:10s}␣{:10s}\n'.format(
            ';X-Coord', 'Y-Coord', 'Label').encode('ascii')
                )
        f.write('\n'.encode('ascii'))

        f.write('[VERTICES]\n'.encode('ascii'))
        f.write('{:10s}␣{:10s}␣{:10s}\n'.format(
            ';ID', 'X-Coord', 'Y-Coord').encode('ascii')
                )
        f.write('\n'.encode('ascii'))

        f.write('[BACKDROP]\n'.encode('ascii'))
        f.write('\n'.encode('ascii'))

        f.write('[END]\n'.encode('ascii'))
```

Code listing C.2: Code listing of generation.py script.

```
import matplotlib.pyplot as plt
from scipy.spatial import distance
import copy
import networkx as nx
import numpy as np
import pandas as pd
import wntr
import pickle

DIAMETER = [80, 100, 125, 150, 200, 250, 300, 400, 500, 600]
FLOW = [1, 1, 1, 1.5, 1.5, 1.75, 1.75, 2, 2, 2, 2]
class GenerationProcedure(object):
    """
    GenerationProcedure is a network generation procedure to produce network variants
    based on a benchmark model. The procedure is initialized with a input file (.inp)
    of a working water distribution network model.
    """
    def __init__(self, input):
        self.wn = wntr.network.WaterNetworkModel(input)
        self.roughness = list()
        self.links = list()
        for name in self.wn.pipe_name_list:
            link = self.wn.get_link(name)
            self.links.append(
                [name, (link.end_node.name, link.start_node.name),
                 link.diameter, link.roughness]
            )
        self.links = pd.DataFrame(self.links, columns=['ID', 'edge', 'd', 'r'])
        self.pos = dict()
        for name in self.wn.node_name_list:
            node = self.wn.get_node(name)
            self.pos[name] = node.coordinates

        self.remove_links = list()
        self.critical_nodes = self.wn.tank_name_list + self.wn.reservoir_name_list
        self.graph = self.wn.get_graph().to_undirected()
        self.service_areas(min_size=15)
        self.classification()

    def plot_graph(self, graph):
        nx.draw(graph, pos=self.pos, node_size=20)
        plt.show()

    def service_areas(self, min_size=15):
        G = self.graph

        self.brigdes = self.wn.pump_name_list + self.wn.valve_name_list
        for name in self.wn.pipe_name_list:
            link = self.wn.get_link(name)
            if str(link.status) == 'Closed':
                self.brigdes.append(name)
            elif link._cv is True:
                self.brigdes.append(name)
            else:
                self.roughness.append(link.roughness)

        for brigde in self.brigdes:
            link = self.wn.get_link(brigde)
            G.remove_edge(link.start_node.name, link.end_node.name)
            if link.start_node.name not in self.critical_nodes:
                self.critical_nodes.append(link.start_node.name)
            if link.end_node.name not in self.critical_nodes:
                self.critical_nodes.append(link.end_node.name)

        components = [G.subgraph(c).copy() for c in nx.connected_components(G)]
        self.dma = dict()
        for idx, g in enumerate(components, start=1):
            if len(g.nodes()) > min_size:
                dma = nx.Graph()
                dma.add_nodes_from(g.nodes(data=True))
                dma.add_edges_from(g.edges(data=True))
                self.dma[idx] = dma

    def classification(self):
        self.dma_critical_nodes = dict()
        self.dma_float_nodes = dict()
        self.dma_links = dict()
        self.backbone = dict()
        for dma_id in self.dma.keys():
            self.dma_float_nodes[dma_id] = list()
            self.dma_links[dma_id] = list()
            self.backbone[dma_id] = nx.Graph()
            self.backbone[dma_id].add_nodes_from(self.dma[dma_id].nodes())
            self.dma_critical_nodes[dma_id] = list()
            for n in self.dma[dma_id].nodes:
                if n in self.critical_nodes:
                    self.dma_critical_nodes[dma_id].append(n)
            weights = []
            for edge in self.dma[dma_id].edges():
                u, v = edge[0], edge[1]
                if (u,v) in self.links['edge'].to_list():
                    row = self.links[(self.links['edge'] == (u,v))]
                else:
                    row = self.links[(self.links['edge'] == (v,u))]
                self.dma_links[dma_id].append(
                    (u, v, self.links.loc[row.index[0], 'ID'])
                )
                weights.append((u, v, 1/self.links.loc[row.index[0], 'd']))

            self.dma[dma_id].add_weighted_edges_from(weights)
            for n1 in self.dma_critical_nodes[dma_id]:
                for n2 in self.dma_critical_nodes[dma_id]:
                    if n1 != n2:
                        path = nx.shortest_path(
                            self.dma[dma_id], source=n1, target=n2, weight='weight'
                        )
                        for i in range(len(path)-1):
                            self.backbone[dma_id].add_edge(path[i], path[i+1])

            for node in self.backbone[dma_id].nodes:
                if self.backbone[dma_id].degree[node] == 0:
                    self.dma_float_nodes[dma_id].append(node)

            for link in self.dma_links[dma_id]:
                if link[0] in self.dma_float_nodes[dma_id] \
                        or link[1] in self.dma_float_nodes[dma_id]:
                    self.remove_links.append(link[2])

    def generate_variant(
            self,
            dgr_exp=10.0,
            dst_exp=10.0,
            n_connections=10,
            alpha=45,
            calibrate=True
    ):

        self.add_links = list()
        for dma_id in self.backbone.keys():
            dma = self.backbone[dma_id]
            float_nodes = self.dma_float_nodes[dma_id]
            connections = dict()
            for node_id in float_nodes:
                connections[node_id] = []
                for neighbour_id in dma.nodes():
                    if neighbour_id != node_id:
                        dst = distance.euclidean(
                            self.pos[node_id],
                            self.pos[neighbour_id]
                        )
                        connections[node_id].append([neighbour_id, dst])
                connections[node_id] = \
                    pd.DataFrame(connections[node_id], columns=['ID', 'dst'])
                connections[node_id] = \
                    connections[node_id].sort_values('dst').head(n_connections)
                connections[node_id]['dst'] = \
                    1/np.power(connections[node_id]['dst'], dst_exp)
                connections[node_id]['dst'] = \
                    connections[node_id]['dst']/connections[node_id]['dst'].sum()

            while nx.is_connected(dma) == False:
                float_pool = pd.DataFrame(
                    [[node, dma.degree[node]] for node in float_nodes],
                    columns=['ID', 'dgr']
                    )
                float_pool['dgr'] = 1/np.power((float_pool['dgr'] + 1), dgr_exp)
                float_pool['dgr'] = float_pool['dgr']/float_pool['dgr'].sum()
                u = float_pool.sample(1, weights='dgr')['ID'].tolist()[0]

                if len(connections[u]['ID']) == 0:
                    float_nodes.remove(u)
                    continue

                v_row = connections[u].sample(1, weights='dst')
                v_index = v_row.index.tolist()[0]
                connections[u] = connections[u].drop(v_row.index.tolist()[0])

                v = v_row['ID'].tolist()[0]
                include = True
                for v_ex in dma.neighbors(u):
                    v_1 = [
                        self.pos[u][0] - self.pos[v][0],
                        self.pos[u][1] - self.pos[v][1]
                    ]
                    v_2 = [
                        self.pos[u][0] - self.pos[v_ex][0],
                        self.pos[u][1] - self.pos[v_ex][1]
                    ]
                    radians = self._angle_between(v_1, v_2)
                    degrees = np.degrees(radians)
                    if abs(degrees) < alpha:
                        include = False
                        break

                if not include:
                    continue

                for v_ex in dma.neighbors(v):
                    v_1 = [
                        self.pos[v][0] - self.pos[u][0],
                        self.pos[v][1] - self.pos[u][1]
                    ]
                    v_2 = [
                        self.pos[v][0] - self.pos[v_ex][0],
                        self.pos[v][1] - self.pos[v_ex][1]
                    ]
                    radians = self._angle_between(v_1, v_2)
                    degrees = np.degrees(radians)
                    if abs(degrees) < alpha:
                        include = False
                        break

                if include:
                    self.add_links.append(([u, v]))
                    dma.add_edge(u, v)
        self.variant = copy.deepcopy(self.wn)
        for pipe in self.remove_links:
            self.variant.remove_link(pipe)

        self.pipes_added = list()
        roughness = round(np.array(self.roughness).mean(), 1)
        for n, con in enumerate(self.add_links):
            name = f'NP{n+1}'
            length = distance.euclidean(self.pos[con[0]], self.pos[con[1]])
            diameter = DIAMETER[0]
            self.variant.add_pipe(
                name, start_node_name=con[0], end_node_name=con[1],
                length=length, diameter=diameter, roughness=roughness
            )
            self.pipes_added.append(name)

        if calibrate:
            self.calibrate()

    def calibrate(self, save=None):
        self.variant.options.hydraulic.demand_model = 'PDD'
        for n, flow in enumerate(FLOW[1:]):
            sim = wntr.sim.EpanetSimulator(self.variant)
            results = sim.run_sim()
            for name in self.pipes_added:
                maximum = results.link['velocity'][name].max()
                i = 0
                if maximum > flow:
                    link = self.variant.get_link(name)
                    link.diameter = DIAMETER[n]
                    i += 1
            if i == 0:
                break

    def _angle_between(self, v1, v2):
        v1_u = v1 / np.linalg.norm(v1)
        v2_u = v2 / np.linalg.norm(v2)
        return np.arccos(np.clip(np.dot(v1_u, v2_u), -1.0, 1.0))
```

Code listing C.3: Code listing of quality.py script.

```
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
import pandas as pd
import wntr
import pickle
import sys
import os
import seaborn as sns


class substance_intrusion():
    def __init__(self, inp, pop, mass_value=1000, duration=1, cs_e=50, cs_p=100):
        '''
        :arg
        inp: Epanet .inp file
        pop: Population .pkl file
        duration: [min, max]
        start_time: [min, max]
        mass_value: MASS [mg/s] at booster point
        cs_e: Threshold for low quality water supply [mg/L]
        cs_p: Threshold for polluted water supply [mg/L]
        '''

        self.cs_e = cs_e
        self.cs_p = cs_p
        self.duration = duration
        self.mass_value = mass_value
        self.network_file = 'wn.pickle'
        self.inp = inp
        self.quality_timestep = 60
        self.report_timestep = 3600
        self.hydraulic_timestep = 3600
        self.simulation_duration = 24 * 3600 * 4


        wn = wntr.network.WaterNetworkModel(self.inp)
        wn.options.hydraulic.demand_model = 'DD'
        wn.options.quality.parameter = 'CHEMICAL'
        wn.options.time.duration = self.simulation_duration
        wn.options.time.hydraulic_timestep = self.hydraulic_timestep
        wn.options.time.report_timestep = self.report_timestep
        wn.options.time.quality_timestep = self.quality_timestep


        self.node_name_list = wn.node_name_list
        self.junction_name_list = wn.junction_name_list
        self.demand = wntr.metrics.expected_demand(wn)[self.junction_name_list]
        self.total_demand = self.demand.sum().sum()
        self.total_nodes = len(self.demand.columns)
        self.total_time = self.simulation_duration # hours

        with open(pop, 'rb') as f:
            self.population = pickle.load(f)
            self.total_population = self.population.sum(axis=1).max()

        with open(self.network_file, 'wb') as f:
            pickle.dump(wn, f)

    def random_failure_sequences(
            self, nsim, start_range=np.arange(0, 24, 1),
            n_sourcepoints = np.arange(0.01, 0.2, 0.01), duration=1
    ):
        self.result = pd.DataFrame()
        nodes_injected = np.arange(0.01, 0.2, 0.01)
        for n in range(nsim):
            for p in nodes_injected:
                self.metrices = {}
                self.metrices['%␣of␣nodes'] = p
                pool = self.junction_name_list.copy()
                with open(self.network_file, 'rb') as f:
                    wn = pickle.load(f)

                for s in range(int(p * len(pool))):
                    start = np.random.choice(start_range)
                    source = np.random.choice(pool)
                    pool.remove(source)
                    source_pattern = wntr.network.elements.Pattern.binary_pattern(
                        f'SP_{source}', start_time=start*3600,
                        end_time=(start+self.duration)*3600,
                        duration=wn.options.time.duration,
                        step_size=wn.options.time.pattern_timestep
                    )
                    wn.add_pattern(f'SP_{source}', source_pattern)
                    wn.add_source(
                        f'Source_{source}', source, 'MASS',
                        self.mass_value, f'SP_{source}'
                    )
                try:
                    sim = wntr.sim.EpanetSimulator(wn)
                    results = sim.run_sim()
                except:
                    print('Epanet␣Error')
                    continue
                self.supply = results.node['demand'].loc[:, wn.junction_name_list]
                self.concentration =\
                    results.node['quality'].loc[:, wn.junction_name_list]
                self._quality_performance()
                self.result = self.result.append(self.metrices, ignore_index=True)

        self.result['%␣of␣nodes'] = self.result['%␣of␣nodes'] * 100

    def global_failure(self, indicator):
        sns.set_theme(style="whitegrid", palette="deep")
        sns.lineplot(
            data=self.result, x='%␣of␣nodes', y=indicator, ci=None)
        sns.lineplot(
            data=self.result, x='%␣of␣nodes', y=indicator, estimator='min', ci=None)
        sns.lineplot(
            data=self.result, x='%␣of␣nodes', y=indicator, estimator='max', ci=None)
        plt.show()


    def _quality_performance(self):
        concentration = self.concentration
        supply = self.supply

        self.N = {
            0: concentration > self.cs_e,
            1: (concentration > self.cs_e) & (concentration < self.cs_p),
            2: concentration > self.cs_p
        }

        self.T = self.N
        self.PS, self.C = {}, {}
        for k in self.N.keys():
            self.PS[k] = supply.where(self.N[k], other=0)
            self.C[k] = self.population.where(self.N[k], other=0)

        self._magnitude()
        self._average_propagation()
        self._severity()
        self._crest_factor()
        self._rapidity()

    def _magnitude(self):
        for i in range(3):
            # Service Metrics
            self.metrices[r'$PS_{{L{0}}}$'.format(i)] = self.PS[i].sum().sum()
            self.metrices[r’$PS_{{L{0},␣ratio}}$’.format(i)] \
                = self.metrices[r’$PS_{{L{0}}}$’.format(i)] / self.total_demand

            # Spatial Metrics
            value_count = (self.N[i].sum(axis=0) > 0).value_counts()
            if True in value_count:
                self.metrices[r’$N_{{L{0},␣n}}$’.format(i)] = value_count[True]
                self.metrices[r’$N_{{L{0},␣n,␣ratio}}$’.format(i)] \
                    = self.metrices[r’$N_{{L{0},␣n}}$’.format(i)] / self.total_nodes
            else:
                self.metrices[r’$N_{{L{0},␣n}}$’.format(i)] = 0
                self.metrices[r’$N_{{L{0},␣n,␣ratio}}$’.format(i)] = 0

            # Costumers Metrics
            self.metrices[r’$C_{{L{0},␣n}}$’.format(i)] \
                = np.sum([self.C[i][col].max() for col in self.C[i].columns])
            self.metrices[r’$C_{{L{0},␣n,␣ratio}}$’.format(i)] \
                = self.metrices[r’$C_{{L{0},␣n}}$’.format(i)] / \
                  self.total_population

            # Continuity Metrics
            value_count = (self.N[i].sum(axis=1) > 0).value_counts()
            if True in value_count:
                self.metrices[r’$T_{{L{0},␣t}}$’.format(i)] = value_count[True]
                self.metrices[r’$T_{{L{0},␣t,␣ratio}}$’.format(i)] = \
                    self.metrices[r’$T_{{L{0},␣t}}$’.format(i)] * 3600 / \
                    self.total_time
            else:
                self.metrices[r’$T_{{L{0},␣t}}$’.format(i)] = 0
                self.metrices[r’$T_{{L{0},␣t,␣ratio}}$’.format(i)] = 0

    def _average_propagation(self):
        for i in range(3):
            T = self.T[i].sum().sum()
            if T > 0:
                self.metrices[r’$\overline{{PS_{{L{0}}}}}$’.format(i)] =\
                    self.metrices[r’$PS_{{L{0}}}$’.format(i)] / T
                self.metrices[r’$\overline{{N_{{L{0}}}}}$’.format(i)] = \
                    self.metrices[r’$N_{{L{0},␣n}}$’.format(i)] / T
                self.metrices[r’$\overline{{C_{{L{0}}}}}$’.format(i)] = \
                    self.C[i].sum().sum() / T
                self.metrices[r’$\overline{{NT_{{L{0}}}}}$’.format(i)] = \
                    self.metrices[r’$T_{{L{0},␣t}}$’.format(i)] / T
            else:
                self.metrices[r’$\overline{{PS_{{L{0}}}}}$’.format(i)] = None
                self.metrices[r’$\overline{{N_{{L{0}}}}}$’.format(i)] = None
                self.metrices[r’$\overline{{C_{{L{0}}}}}$’.format(i)] = None
                self.metrices[r’$\overline{{NT_{{L{0}}}}}$’.format(i)] = None

    def _severity(self):
        for i in range(3):
            self.metrices[r’$PS_{{L{0},␣peak}}$’.format(i)] = \
                self.PS[i].sum(axis=1).max()
            self.metrices[r’$N_{{L{0},␣peak}}$’.format(i)] = \
                self.N[i].sum(axis=1).max()
            self.metrices[r’$C_{{L{0},␣peak}}$’.format(i)] = \
                self.C[i].sum(axis=1).max()

    def _crest_factor(self):
        for i in range(3):
            if self.metrices[r’$PS_{{L{0}}}$’.format(i)] > 0:
                self.metrices[r’$PS_{{L{0},␣PAR}}$’.format(i)] =\
                    self.metrices[r’$PS_{{L{0},␣peak}}$’.format(i)] / \
                    self.metrices[r’$PS_{{L{0}}}$’.format(i)]
            else:
                self.metrices[r’$PS_{{L{0},␣PAR}}$’.format(i)] = None

            if self.metrices[r’$N_{{L{0},␣n}}$’.format(i)] > 0:

                self.metrices[r’$N_{{L{0},␣PAR}}$’.format(i)] =\
                    self.metrices[r’$N_{{L{0},␣peak}}$’.format(i)] / \
                    self.metrices[r’$N_{{L{0},␣n}}$’.format(i)]
            else:
                self.metrices[r’$N_{{L{0},␣PAR}}$’.format(i)] = None

            if self.metrices[r’$C_{{L{0},␣n}}$’.format(i)] > 0:
                self.metrices[r’$C_{{L{0},␣PAR}}$’.format(i)] =\
                    self.metrices[r’$C_{{L{0},␣peak}}$’.format(i)] / \
                    self.metrices[r’$C_{{L{0},␣n}}$’.format(i)]
            else:
                self.metrices[r’$C_{{L{0},␣PAR}}$’.format(i)] = None

    def _rapidity(self):
        t_e = 0
        for i in range(3):
            if  self.PS[i].sum(axis=1).idxmax() / self.report_timestep > t_e:
                self.metrices[r’$PS_{{L{0},␣TEP}}$’.format(i)] = \
                    self.PS[i].sum(axis=1).idxmax() / self.report_timestep- t_e
            else:
                self.metrices[r’$PS_{{L{0},␣TEP}}$’.format(i)] = None

            if self.N[i].sum(axis=1).idxmax() / self.report_timestep > t_e:
                self.metrices[r’$N_{{L{0},␣TEP}}$’.format(i)] = \
                    self.N[i].sum(axis=1).idxmax() / self.report_timestep - t_e
            else:
                self.metrices[r’$N_{{L{0},␣TEP}}$’.format(i)] = None

            if self.C[i].sum(axis=1).idxmax() / self.report_timestep > t_e:
                self.metrices[r’$C_{{L{0},␣TEP}}$’.format(i)] = \
                    self.C[i].sum(axis=1).idxmax() / self.report_timestep - t_e
            else:
                self.metrices[r’$C_{{L{0},␣TEP}}$’.format(i)] = None
```

Code listing C.4: Code listing of attributes.py script.

```
import wntr
import networkx as nx

def attributes(G):
    properties = {}
    uG = nx.Graph(G.to_undirected())
    properties[’d’] = [nx.density(G)]
    properties[’a’] = [nx.average_shortest_path_length(G)]
    properties[r’$C_c$’] = [nx.transitivity(sG)]
    properties[r’$C_B$’] = [wntr.metrics.central_point_dominance(G)]
    properties[r’$C_u$’] = \
        [pd.array([nx.closeness_centrality(G)[k]
                   for k in nx.closeness_centrality(G)]).mean()]
    return properties
```

Jupyter notebook - A demonstration

Elias Gulla Rokstad

Resilience of topological attributes

This is a demonstation of the classes developed during the course of my master thesis. The code listings are found in attributes.py, quality.py, and generation.py.

[1]:
```
import matplotlib.pyplot as plt
import networkx as nx
import pandas as pd
import seaborn as sns
import numpy as np

import attributes
import quality
import generation

sns.set_theme(style="whitegrid", palette="deep")
networks = ['Net3', 'Ctown']
```

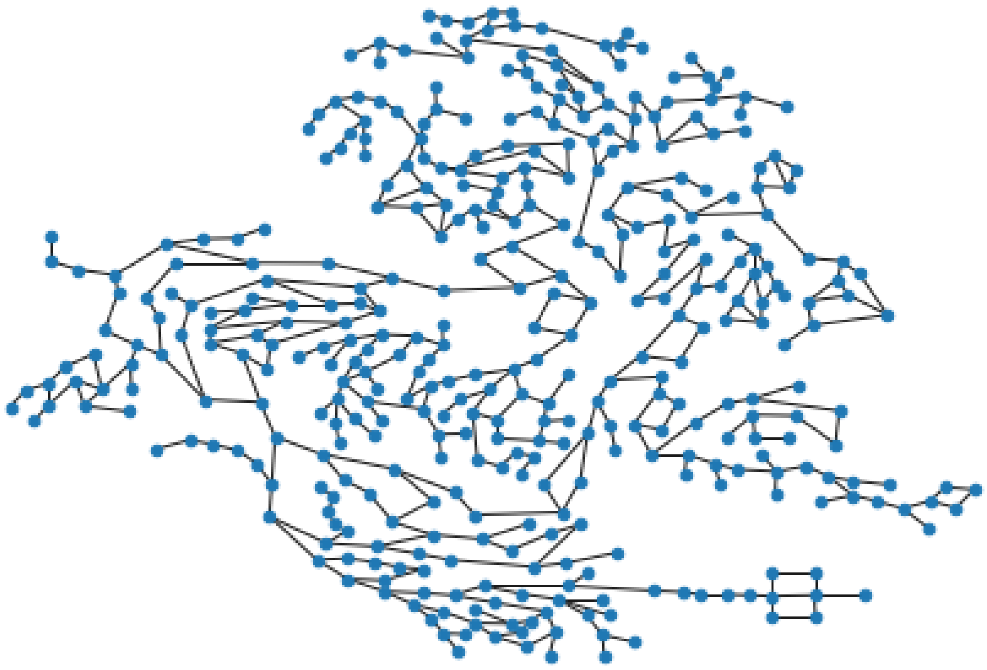
Network generation procedure

The network generation procedure is developed to automatically generate network variants of the benchmark model. It is a four-step procedure with the following steps:

1. Link classification
2. Link removal
3. Link addition
4. Model calibration

[3]:
```
net = networks[1]
gen = generation.GenerationProcedure(f'networks/{net}.inp')

# Benchmark model
G = gen.wn.get_graph().to_undirected()
gen.plot_graph(G)
```

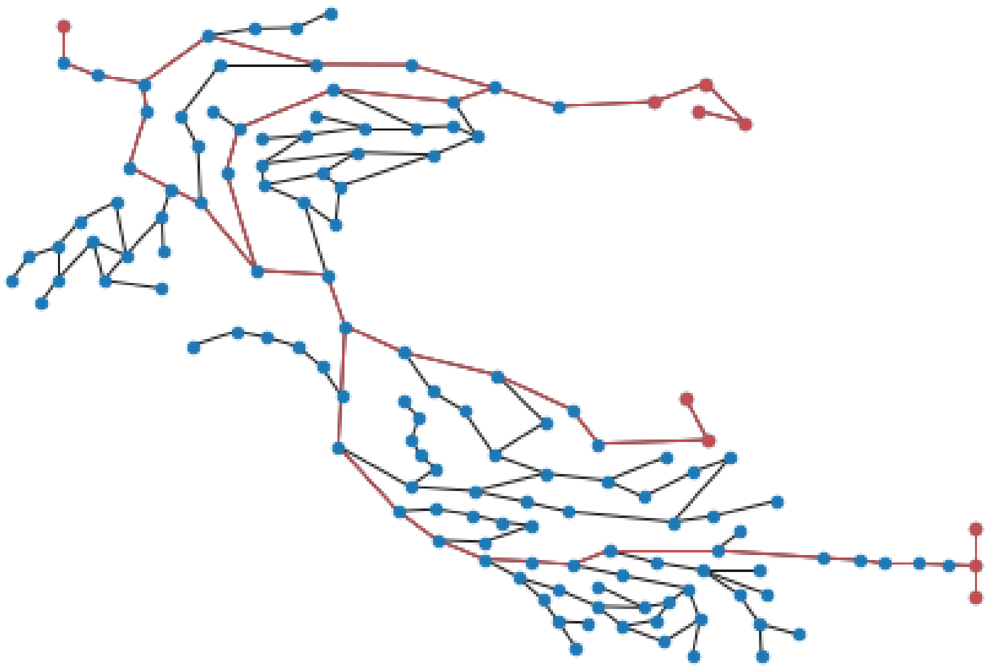


Link classification

The link classification convert the water distribution model into a graph (Graph theory). Links classified as pumps, valves or pipes with CV settings are removed from the graph. The following operations are conducted for each service area:

1. Connected nodes are grouped together as service areas.
2. Junctions connected to the pumps, valves or CV pipes, tanks, and reservoirs are stored as critical nodes
3. Dijkstra's shortest path algorithm are used to map the links connected the critical nodes together. The inverse of the diameter of the links are used as weights, e.i. large diameter pipes presents a shorter traveling distance than smaller pipes. pipes part of the shortest path are stored as critical links.

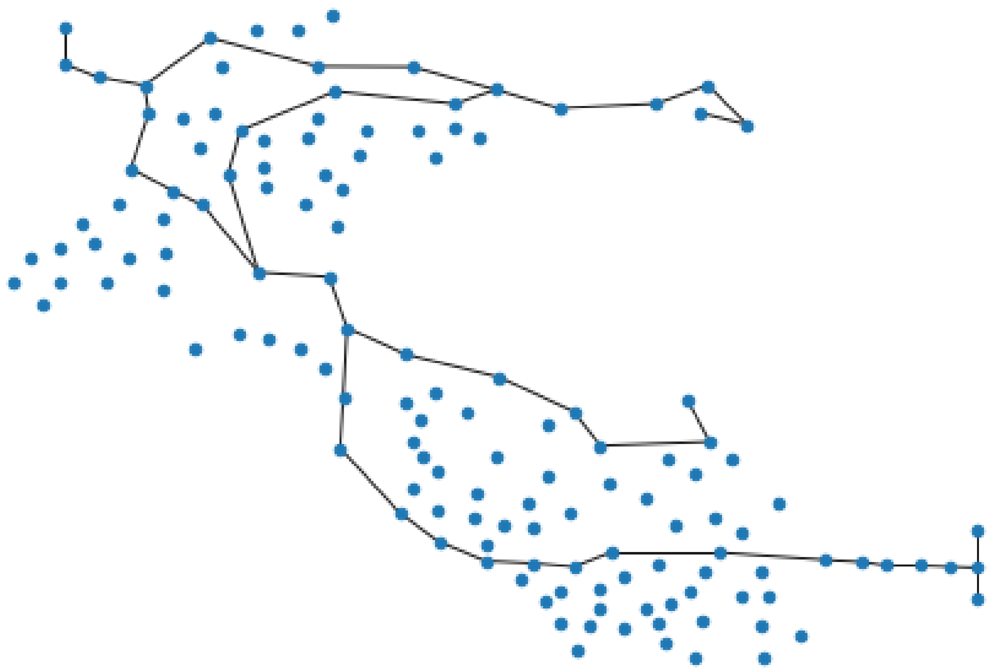
[5]:
```
color_links = gen.backbone[1].edges()
color_nodes = gen.critical_nodes
gen.plot_graph_highlight(G, nodes=color_nodes, links=color_links)
```



Link removal

The links not included by Dijkstra's shortest path algorithm are removed in each service area.

```
[6]: G = gen.backbone[1]
     gen.plot_graph(G)
```



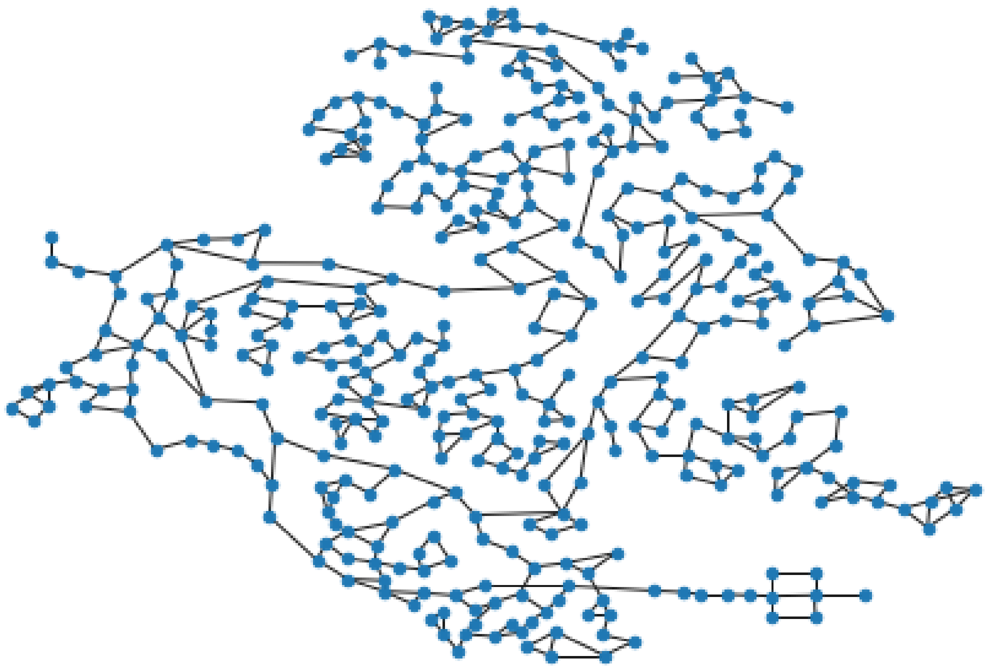
Link addition

Pipes are added to the graph based on following statistical parameters:

1. *dgr_exp*: probability exponent of node degrees (e.i. high *dgr_exp* value increase probability of selecting nodes with low node degree as a start point).
2. *dst_exp*: probability exponent of spatial closeness (e.i. high *dst_exp* value increase probability of selecting nodes close to the start node as a end node).
3. *n_connection*: the number of possible connection from each start node ranked by closed neighbours.
4. *alpha*: the mimimum allowed angle between a possible connections and existing connections at start or end node.

[20]:
```
gen.generate_variant(dgr_exp=10.0, dst_exp=5.0, n_connections=5, alpha=20,␣
 ↪calibrate=False)
G = gen.variant.get_graph()
print(pd.DataFrame.from_dict(attributes.attributes(G)))
uG = G.to_undirected()
gen.plot_graph(uG)
```

```
          d         a     $C_c$     $C_B$    $C_u$
0  0.003101  0.051234  0.162963  0.476203  0.00429
```

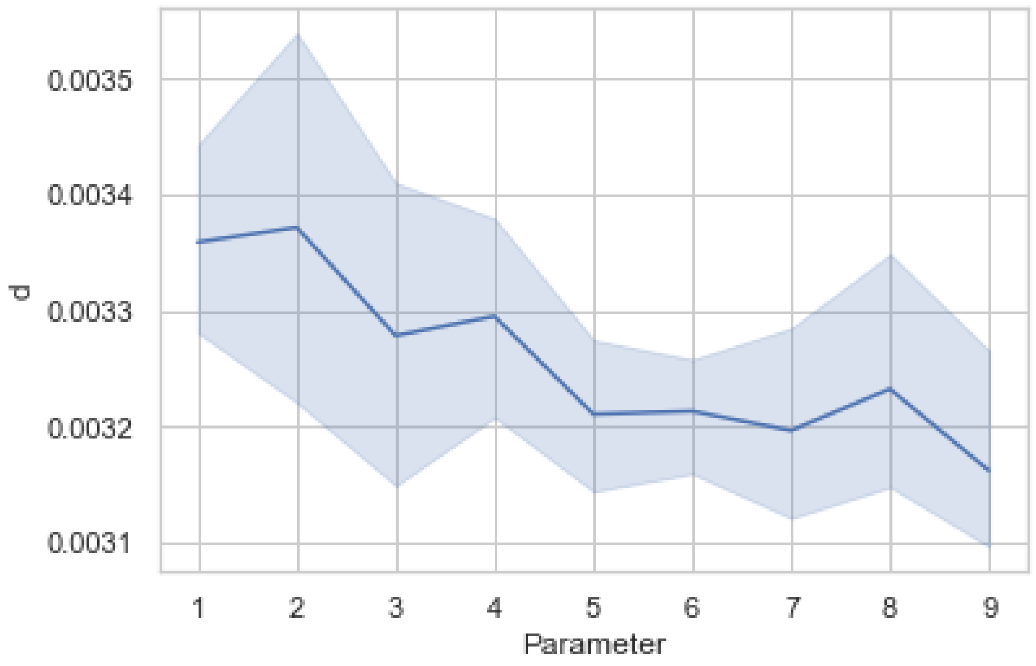


[30]:
```
# Degree exponent
data = []

for dgr in np.arange(1, 10, 1):
    for n in range(5):
        try:
            gen.generate_variant(dgr_exp=dgr, dst_exp=10.0, n_connections=10,␣
 ↪alpha=20, calibrate=False)
            att = attributes.attributes(gen.variant.get_graph())
            temp = [dgr]
            for k in att.keys():
                temp.append(att[k][0])
            data.append(temp)
        except:
            print(f'Generation did not converge: parameter={dgr}')


columns = ['Parameter', 'd', 'a', '$C_c$', '$C_B$', '$C_u$']
data = pd.DataFrame(data, columns=columns)
print(data.groupby('Parameter').mean())
sns.lineplot(data=data, x='Parameter', y='d')
plt.show()
```

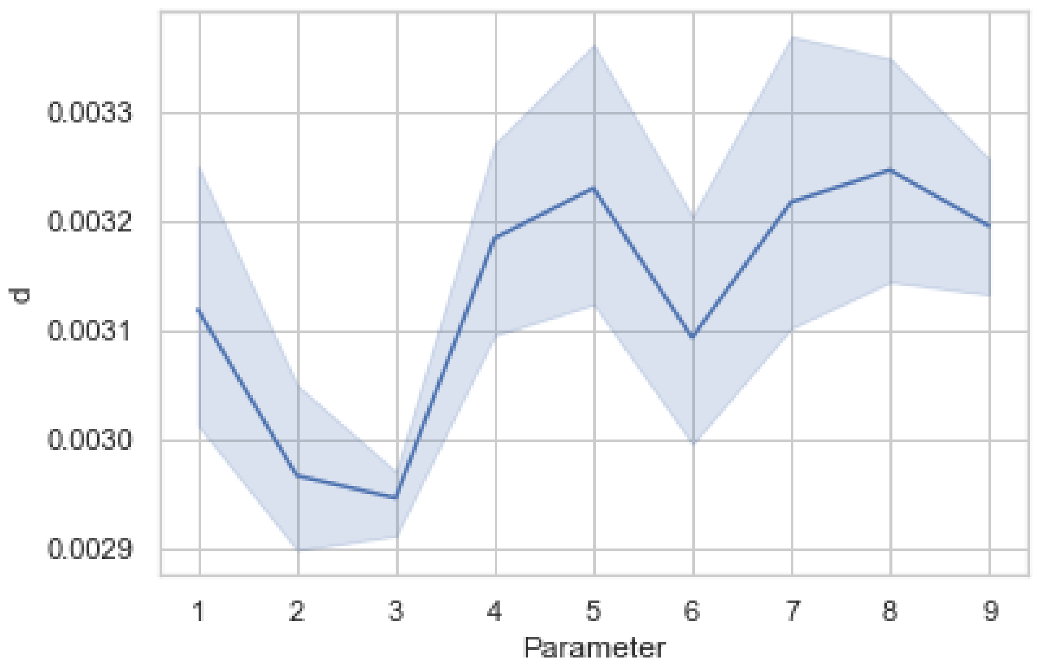
```
                  d         a     $C_c$     $C_B$     $C_u$
Parameter
1          0.003359  0.063179  0.217721  0.467299  0.004890
2          0.003372  0.067628  0.214109  0.464630  0.004934
3          0.003278  0.067658  0.181934  0.460170  0.004750
4          0.003295  0.059747  0.180518  0.459995  0.004716
5          0.003211  0.060876  0.183827  0.475639  0.004557
6          0.003213  0.058660  0.172766  0.460961  0.004605
7          0.003197  0.069192  0.167241  0.458544  0.004645
8          0.003232  0.061102  0.163880  0.466311  0.004667
9          0.003162  0.061488  0.172492  0.483199  0.004506
```



[32]:
```
# Distribution exponent
data = []
for dgr in np.arange(1, 10, 1):
    for n in range(5):
        try:
            gen.generate_variant(dgr_exp=10.0, dst_exp=dgr, n_connections=10,␣
 ↪alpha=20, calibrate=False)
            prop = attributes.attributes(gen.variant.get_graph())
            temp = [dgr]
            for k in prop.keys(): temp.append(prop[k][0])
            data.append(temp)
        except:
            print(f'Generation did not converge: parameter={dgr}')

columns = ['Parameter', 'd', 'a', '$C_c$', '$C_B$', '$C_u$']
data = pd.DataFrame(data, columns=columns)
print(data.groupby('Parameter').mean())
sns.lineplot(data=data, x='Parameter', y='d')
plt.show()
```

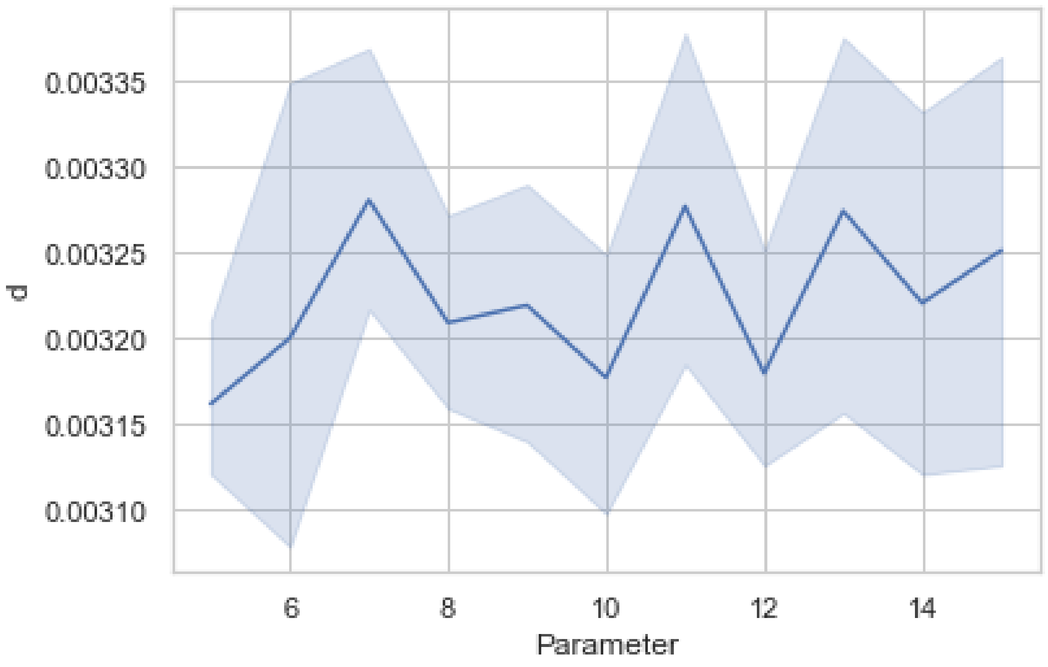
```
                  d         a     $C_c$     $C_B$     $C_u$
Parameter
1          0.003119  0.067675  0.069922  0.489786  0.004832
2          0.002966  0.053652  0.065965  0.505071  0.004373
3          0.002946  0.062386  0.080099  0.467963  0.004328
4          0.003184  0.064021  0.134851  0.489888  0.004767
5          0.003230  0.062787  0.148626  0.464156  0.004738
6          0.003093  0.070542  0.152123  0.463272  0.004471
7          0.003217  0.065164  0.168145  0.466604  0.004695
8          0.003246  0.065099  0.175769  0.467905  0.004713
9          0.003195  0.061849  0.161146  0.461516  0.004624
```



[33]:
```python
# Number of possible end nodes connections pr. start node
data = []
for connections in np.arange(5, 16, 1):
    for n in range(5):
        try:
            gen.generate_variant(dgr_exp=10.0, dst_exp=10.0,␣
 ↪n_connections=connections, alpha=20, calibrate=False)
            prop = attributes.attributes(gen.variant.get_graph())
            temp = [connections]
            for k in prop.keys(): temp.append(prop[k][0])
            data.append(temp)
        except:
            print(f'Generation did not converge: parameter={connections}')

columns = ['Parameter', 'd', 'a', '$C_c$', '$C_B$', '$C_u$']
data = pd.DataFrame(data, columns=columns)
print(data.groupby('Parameter').mean())
sns.lineplot(data=data, x='Parameter', y='d')
plt.show()
```

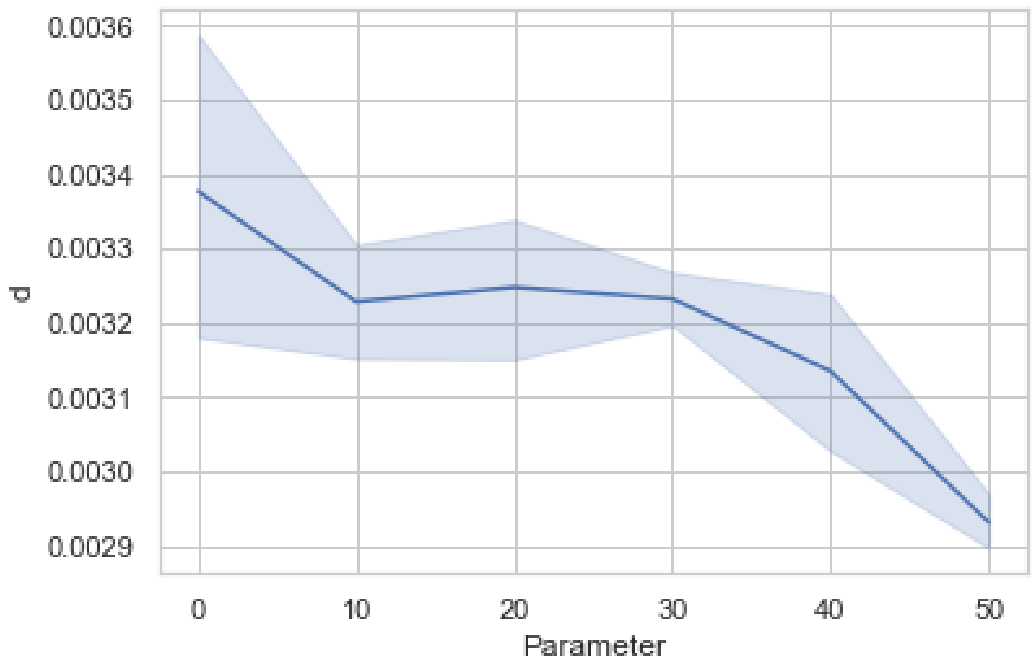
```
                  d         a     $C_c$     $C_B$     $C_u$
Parameter
5          0.003162  0.057748  0.160293  0.457022  0.004480
6          0.003200  0.067271  0.168446  0.482575  0.004623
7          0.003281  0.060292  0.182354  0.459884  0.004682
8          0.003209  0.064154  0.170501  0.475741  0.004647
9          0.003220  0.059715  0.173642  0.475412  0.004599
10         0.003177  0.061211  0.176558  0.458031  0.004566
11         0.003277  0.066789  0.184438  0.458132  0.004766
12         0.003180  0.060428  0.160254  0.458719  0.004554
13         0.003275  0.057096  0.180497  0.470411  0.004712
14         0.003221  0.060370  0.170536  0.481412  0.004624
15         0.003252  0.066187  0.168694  0.459670  0.004717
```



[34]:
```
# Minimum angle constraint between added links
data = []
for  angle in np.arange(0, 60, 10):
    for n in range(5):
        try:
            gen.generate_variant(dgr_exp=10.0, dst_exp=10.0, n_connections=10,␣
 ↪alpha=angle, calibrate=False)
            prop = attributes.attributes(gen.variant.get_graph())
            temp = [angle]
            for k in prop.keys(): temp.append(prop[k][0])
            data.append(temp)
        except:
            print(f'Generation did not converge: parameter={angle}')

columns = ['Parameter', 'd', 'a', '$C_c$', '$C_B$', '$C_u$']
data = pd.DataFrame(data, columns=columns)
print(data.groupby('Parameter').mean())
sns.lineplot(data=data, x='Parameter', y='d')
plt.show()
```

```
                  d         a     $C_c$     $C_B$     $C_u$
Parameter
0          0.003377  0.069097  0.166071  0.516136  0.004913
10         0.003228  0.061961  0.170843  0.476627  0.004727
20         0.003248  0.064281  0.178603  0.460016  0.004668
30         0.003232  0.063656  0.170840  0.482568  0.004669
40         0.003135  0.056098  0.110578  0.463398  0.004499
50         0.002932  0.050924  0.030279  0.456973  0.004188
```



Model calibartion

The network variant may be calibrated to obtain a economic flow regime. All added pipes is configured with a 80 mm diameter. These pipes may lead to high flow rates and pressure loss. Thus, if the simulated flow rate of a pipe is above a predifned velocity, the pipe's diameter are increased before the next simulation run. Pipe roughness is defined as the average of the removed pipes in the area, while the pipe length is calculated as the distance between the start node and the end node.

[36]:
```
gen.calibrate(save=None) # save to .inp format
```

Random failure sequences

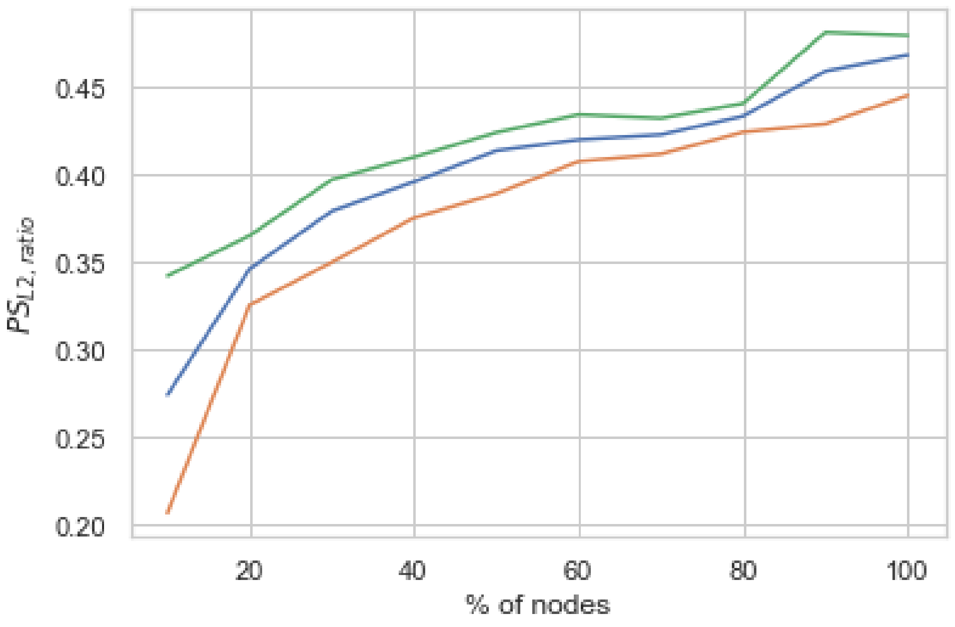
[9]:
```
input_file = f'networks/{net}.inp'
population_file = f'networks/population/{net}.pkl'
sim = quality.substance_intrusion(inp=input_file, pop=population_file,␣
↪mass_value=1000, duration=1, cs_e=50, cs_p=100)
```

[10]:
```
sim.random_failure_sequences(nsim=10, start_range=np.arange(0, 12, 1),␣
↪n_sourcepoints=np.arange(0.1, 1.1, 0.1))
```

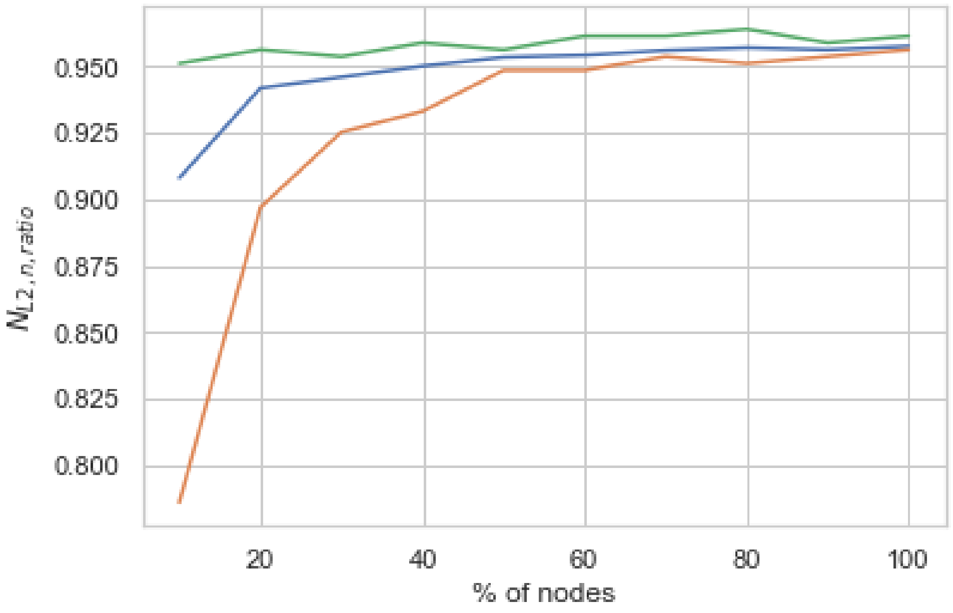
Mean, maximum, and minimum values

The line plot of mean, maximum, and minimum failure values for each level of stress can be expressed with the global_failure(indicator) function.

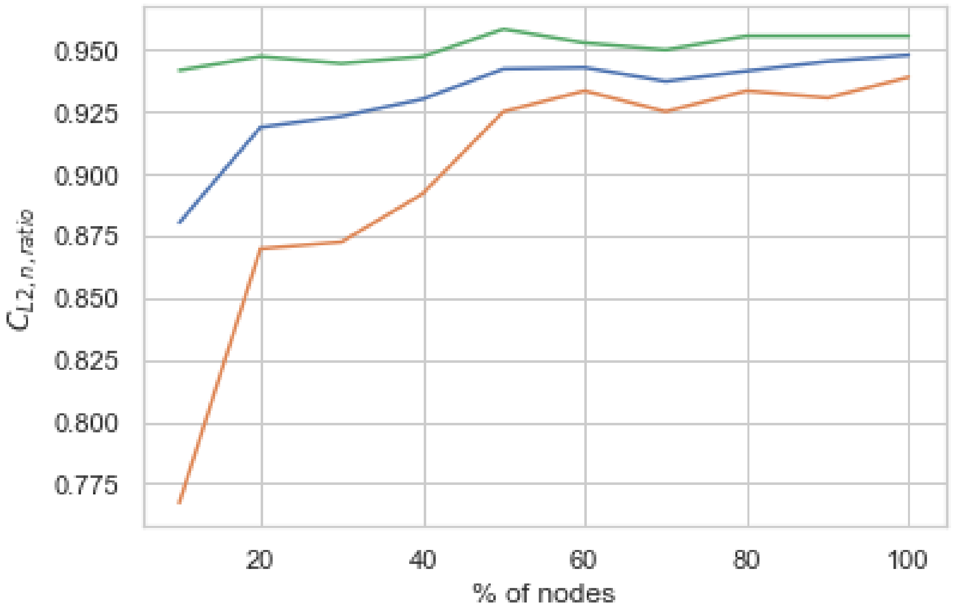
[11]:
```
sim.global_failure(r'$PS_{L2, ratio}$')
```



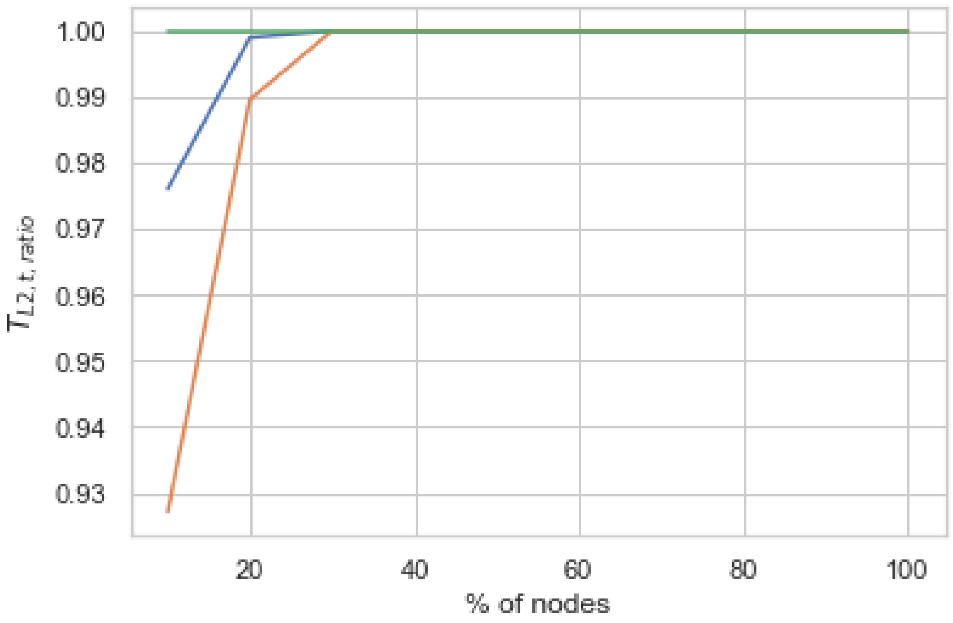
[12]:
```
sim.global_failure(r'$N_{L2, n, ratio}$')
```



[13]:
```
sim.global_failure(r'$C_{L2, n, ratio}$')
```



[14]:
```
sim.global_failure(r'$T_{L2, t, ratio}$')
```



Elias Gulla Rokstad

Resilience of WDS

NTNU
Norwegian University of
Science and Technology