Georg Arbesser-Rastburg, BSc

Serious Sensor Placement

Optimale Sensorplatzierung als Serious Game

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Bauingenieurwissenschaften - Infrastruktur

eingereicht an der

Technischen Universität Graz

Betreuerin:

Assoc.Prof. Dipl.-Ing. Dr.techn. Daniela Fuchs-Hanusch

Institut für Siedlungswasserwirtschaft und Landschaftswasserbau

Graz, Jänner 2019

Kontakt:
Georg Arbesser-Rastburg
georg.arbesser-rastburg@student.tugraz.at

EIDESSTATTLICHE ERKLÄRUNG

AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used anything other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or contextually from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Datum / Date

Unterschrift / Signature

Danksagung

Diese Masterarbeit markiert das Ende einer großartigen Studienzeit und nun habe ich die Ehre all jenen zu danken, die mir im Laufe meines Studiums mit Rat und Tat zur Seite gestanden sind.

Zuallererst möchte ich mich bei derjenigen Person bedanken, die es mir überhaupt erst ermöglicht hat, mich im Bereich der Trinkwasserversorgung zu vertiefen und dank derer ich nun hier meine vollendete Masterarbeit vorlegen kann: Meiner Betreuerin Assoc.Prof. Dipl.-Ing. Dr.techn. Daniela Fuchs-Hanusch.

Weiters gilt mein ausgesprochener Dank Dipl.-Ing. Dr.techn. David B. Steffelbauer, der mich vor rund zwei Jahren unter seine Fittiche nahm und durch dessen buchstäblich ausgezeichnete Forschung im Bereich der optimalen Sensorplatzierung diese Masterarbeit erst möglich gemacht wurde.

Auch ohne David Camhy wäre es mir nicht möglich gewesen, diese Arbeit umzusetzen. Gleichgültig, ob der Code nicht das tut was er soll oder ein Problem unlösbar scheint, David weiß Rat!

Aber auch den Probandinnen und Probanden des Feldversuchs und allen Arbeitskollegen vom Institut für Siedlungswasserwirtschaft und Landschaftswasserbau, insbesondere Dipl.-Ing. Markus Günther, Dipl.-Ing. Michael Pointl sowie Reinhard Klingenberg, gilt mein Dank für die nette Arbeitsatmosphäre und ihre Unterstützung.

Dank gilt des Weiteren meiner ersten „Testperson“, meinem langjährigen Freund Stefan Mazuheli. Bedanken will ich mich auch bei meinen Studienkollegen, vor allem bei Lan Rajh, Philipp Rockenbauer und Valentin Ronchetti. Danke für die schönen Studienjahre!

Ebenfalls danken möchte ich meinen Schwiegereltern, deren Unterstützung nicht nur die letzte Zeit deutlich leichter gemacht hat.

Selbstverständlich darf in dieser Aufzählung meine gesamte Familie nicht fehlen, vor allem meine Eltern, deren unermüdlichen Unterstützung ich mir immer sicher sein durfte. Ebenfalls danken möchte ich meinen Geschwistern, besonders meiner Schwester Eva, die zu jeder Tages- und Nachtzeit ein offenes Ohr für mich hatte. Nicht vergessen werden soll auch Dr. Anton Rohrer für seine immerwährende Unterstützung. Danke für alles!

Schlussendlich bleibt nur noch eine Person übrig, der ich dafür umso mehr danken möchte: Nina, danke nicht nur für deine Unterstützung und dass du mich auch dann noch ertragen hast, wenn der Stress bei mir gerade ganz besonders groß war, sondern allgemein für die Bereicherung, die du in meinem Leben darstellst!

Kurzfassung

In dieser Arbeit wurden zwei Themenbereiche miteinander kombiniert: Die ideale Platzierung von Drucksensoren in Trinkwasserversorgungsnetzen zum Zwecke der Leckagedetektion und -lokalisierung einerseits und Serious Gaming andererseits. Ein Serious Game ist ein Spiel, das die Spielenden nicht nur unterhalten, sondern dabei noch einen zusätzlichen Zweck erfüllen soll, wie beispielsweise das Verständnis für Lerninhalte zu erhöhen.

Ziel war es, eine Weboberfläche als Spiel zu entwickeln, in welcher Sensoren in einer graphischen Darstellung eines Trinkwassernetzes spielerisch platziert werden können, um nach einer Evaluierung der Sensorpositionen in Hinblick auf Spielziele noch bessere Positionen zu finden. Spielziel ist es, eine hohe Netzabdeckung hinsichtlich Detektier- und Lokalisierbarkeit von Lecks bei einer möglichst geringen Sensoranzahl zu erreichen.

Zu diesem Zweck wird für die Bewertung der Sensorpositionen ein geeigneter Algorithmus benötigt, mit dem eine Netzabdeckung ermittelt werden kann. Dafür wurden existierende Algorithmen und Programmcodes zur Sensorplatzierung adaptiert und an die Erfordernisse der webbasierten Spieloberfläche angepasst. Für die sehr rechenintensive Bestimmung von idealen Sensorpositionen wurde eine auf einem leistungsstarken Server existierende, auf diese Berechnungen ausgerichtete Weboberfläche genutzt und um zusätzliche Funktionalitäten erweitert.

Umgesetzt wurde das auf Python, JavaScript und HTML basierende Serious Game anhand zweier hydraulischer Modelle, die der Literatur entnommen wurden.

Die von den Spielenden zu erreichenden Zielwerte (% Netzabdeckung und Sensoranzahl) werden so gesetzt, dass sie einerseits erreichbar und andererseits auch motivierend sind. Die Spielziele leiten sich von idealen Sensorplatzierungsergebnissen ab, die in Voruntersuchungen mit einem Optimierer berechnet wurden.

Das Serious Game wurde von zwölf Probandinnen und Probanden zeitgleich getestet und mit einem Fragebogen bewertet. Dabei zeigte sich, dass die Spielenden innerhalb kürzester Zeit ähnliche Ergebnisse produzieren können wie der verwendete Optimierungsalgorithmus. Außerdem wurde gezeigt, dass die Umsetzung des Problems der idealen Sensorplatzierung als Serious Game motivierend ist und anspornt, immer bessere Ergebnisse zu erzielen.

Die ProbandInnen unterbreiteten in den Fragebögen außerdem Verbesserungsvorschläge, die in das Serious Game einfließen können. Zusätzlich besteht Optimierungspotential in Bezug auf die Rechenzeit der Evaluierung der Sensorpositionen.

Abstract

This thesis combines two different topics: On the one hand the ideal placement of pressure sensors in water distribution networks to detect and localize leaks and Serious Gaming on the other hand. A Serious Game is a game, which is not only meant to entertain, but also to serve another purpose, for instance to increase the understanding of educational contents.

The goal was to create a web interface, through which gamers could ludic place sensors in a graphic representation of a water distribution network, in order to improve these sensor positions after they were evaluated by an algorithm. The goal for the players is to achieve a high net coverage regarding the possibility to detect and localize leaks with as little sensors as possible.

For this purpose, a suitable algorithm for the evaluation of sensor positions is needed to calculate a net coverage. Existing optimal sensor placement algorithms and program codes were adapted to meet the web-based game interface's requirements. For the very CPU-intensive determination of ideal sensor positions, an already existing web interface, that was built for these calculations, was used and enhanced with new functionalities.

The Serious Game web interface, which is based on Python, JavaScript and HTML, was implemented using two hydraulic models, which were picked from existing publications.

The two goals for the players (% net coverage and number of sensors) are set in an achievable and motivating way. These goals are derived from optimal sensor placement results, that were calculated in a preliminary analysis using an optimizer.

The Serious Game was tested by twelve test persons simultaneously and rated using a questionnaire. This showed that human players are able to reach results similar to those of an optimization algorithm within in a very short time. Furthermore, it was shown that the implementation of the ideal sensor placement problem as a Serious Game is motivating for the players to get better and better results.

In addition, the test subjects suggested improvements using the questionnaires, which in the future could be implemented in the Serious Game. Further optimization potential in relation to the required computation time for the evaluation of sensor positions persists.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

DE	Differential Evolution
GA	Genetischer Algorithmus
OSP	Optimal Sensor Placement

1 Motivation und Ziele

Das Institut für Siedlungswasserwirtschaft und Landschaftswasserbau beschäftigt sich bereits seit geraumer Zeit mit der optimalen Platzierung von Sensoren in Trinkwasserversorgungsnetzen zum Zweck der Leckagedetektion und -lokalisierung (OSP – optimal sensor placement). Im Rahmen der Forschung wurde ein Algorithmus entwickelt, der an bereits existierende Algorithmen angelehnt ist, aber noch zusätzlich die Verbraucherunsicherheiten miteinbezieht.

Bei der Anwendung der bisher vorhandenen Algorithmen wurden allerdings einige Grenzen aufgezeigt. Aufgrund der Funktionsweise der OSP-Algorithmen und der dafür nötigen Optimierer (wie beispielsweise genetische Algorithmen), sind Ergebnisse manchmal schwer nachvollziehbar und oft stellen sich die Nutzer die Frage, weshalb ein unbefriedigendes Ergebnis zustande kommt und was an der Vorgehensweise geändert werden muss, um plausiblere oder bessere Ergebnisse zu erzielen.

Durch einen Tagungsbeitrag von Savic et al. (2016) wurde das Institut für Siedlungswasserwirtschaft und Landschaftswasserbau auf eine alternative Art der Darstellung von Optimierungsproblemen aufmerksam: Serious Games.

Savic et al. betteten ein in der Wissenschaft bekanntes Problem (das New-York-Tunnel-Problem) in eine Weboberfläche ein und ermöglichten es den Nutzern eigene Lösungen für dieses Problem zu finden, welches ansonsten mittels eines geeigneten Algorithmus gelöst werden müsste.

Aufgrund dieses Konferenzbeitrages entstand daraufhin die Idee, Serious Gaming mit OSP zu kombinieren. Ziel war es, eine Weboberfläche zu entwickeln, in welcher Spielende selbst Sensoren platzieren können und diese Positionen auf Basis eines OSP-Algorithmus bewertet werden. Durch die visuelle Unterstützung durch die Weboberfläche sollte dabei ein besseres Verständnis für die optimale Platzierung von Sensoren erzielt werden.

Im Rahmen dieser Arbeit sollte eine webbasierte Spieloberfläche entwickelt werden und mittels einer Feldstudie ein Vergleich zwischen Lösungen eines Optimierers und von Testpersonen erarbeiteten Lösungen angestellt werden.

2 Methodische Grundlagen

In den folgenden Unterkapiteln werden die methodischen Grundlagen, die für diese Arbeit benötigt wurden, angeführt. Der erste Teil dieses Kapitels beschäftigt sich mit der idealen Platzierung von Sensoren, während der zweite Teil Serious Games behandelt.

2.1 Optimal Sensor Placement

Leckagen in Trinkwasserversorgungssystemen stellen für Wasserversorgungsunternehmen große Probleme dar. Einerseits muss das Wasser, das durch Lecks in Trinkwasserversorgungsnetzen verloren geht, gefördert und evtl. auch aufbereitet werden und andererseits kann dieses verlorene Wasser nicht an Kunden verkauft werden. In einer Veröffentlichung der Weltbank 2006 wird die weltweit durch Lecks verlorene Wassermenge auf mindestens 48,6 Mrd. Kubikmeter pro Jahr und die daraus resultierenden Kosten auf mindestens 14 Mrd. US-Dollar geschätzt (Liemberger & Marin, 2006).

Um diese Lecks finden zu können, können Drucksensoren in Trinkwassersystemen platziert werden, um damit den durch Leckagen hervorgerufenen Druckverlust zu detektieren und auf Basis dieser Daten das Leck zu lokalisieren. Um die dazu nötigen Sensoren zu platzieren, wurden verschiedenste Algorithmen entwickelt, von denen in diesem Kapitel einige vorgestellt werden.

2.1.1 Graphenbasierte Methoden

Graphenbasierte Methoden verwenden die Topologie des Trinkwasserversorgungsnetzes zur Sensorplatzierung und beinhalten keine Betrachtung der Hydraulik des Netzes. De Schaetzen et al. (2000) entwickelten zwei Algorithmen zur Platzierung von Drucksensoren in einem Trinkwasserversorgungsnetz, allerdings nicht zur Leckagedetektion und -lokalisierung, sondern zum Zweck der Kalibrierung. Beide Algorithmen bedienen sich dabei des Shortest Path Algorithmus und werden deswegen als *Shortest Path 1* und *Shortest Path 2* bezeichnet.

Beiden Algorithmen ist es gemein, dass zu Beginn die kürzesten Wege (*shortest paths*) von der Wassereinspeisung zu allen Knoten berechnet werden. Sind mehrere Einspeisepunkte vorhanden, wird eine virtuelle, übergeordnete Quelle (*super source*) zum Netz hinzugefügt und mittels Leitungen (bzw. im Graph mit Kanten) mit der Länge 0 mit allen Einspeisepunkten verbunden.

Beim Algorithmus Shortest Path 1 wird ein Sensor auf jenem Knoten platziert, der den längsten Shortest Path zum Einspeisepunkt (bzw. zur super source) aufweist. Anschließend wird eine neue Leitung mit der Länge 0 zwischen Einspeisepunkt und Sensorposition zum Netz hinzugefügt. Nun werden wieder die kürzesten Wege vom Einspeisepunkt zu allen Knoten bestimmt und wiederum ein

Sensor auf jenem Knoten platziert, der den längsten Shortest Path zum Einspeisepunkt aufweist. Dieses Prozedere wird so lange wiederholt, bis die gewünschte Anzahl an Sensoren platziert ist.

Der Algorithmus Shortest Path 2 funktioniert ähnlich wie der Algorithmus Shortest Path 1, allerdings wird hier keine neue Leitung mit der Länge 0 zum Netz hinzugefügt, sondern die Länge aller Leitungen entlang des Shortest Path zwischen Einspeisepunkt und Sensor auf 0 gesetzt.

Shortest Path 1 platziert Sensoren eher im Inneren des Netzes, während Shortest Path 2 die Sensoren eher am Rand des Netzes positioniert (de Schaetzen et al., 2000).

2.1.2 Sensitivitätsbasierte Methoden

Sensitivitätsbasierte Methoden berücksichtigen im Gegensatz zu graphenbasierten Methoden die Hydraulik des Netzes. Alle folgenden Algorithmen nutzen dazu die Sensitivitätsmatrix, verwenden sie jedoch auf unterschiedliche Art und Weise, weswegen sich die jeweils errechneten Sensorpositionen voneinander unterscheiden.

Im Folgenden werden zuerst die Sensitivitäts- und die Residuenmatrix vorgestellt, gefolgt von zwei sensitivitätsbasierten Algorithmen: Casillas und SPUDU.

Es soll an dieser Stelle angemerkt werden, dass die hier verwendete Notation (insbesondere die Indices) nicht immer mit den Quellen übereinstimmen. Dies liegt daran, dass versucht wurde, die unterschiedlichen Schreibweisen der verschiedenen Autoren zu vereinheitlichen, um die Vergleichbarkeit zu verbessern.

2.1.2.1 Sensitivitäts- und Residuenmatrix

Die Sensitivitätsmatrix (*leak sensitivity matrix*) beschreibt die Sensitivität von Knoten (i.e. Sensorpositionen) auf Leckagen in anderen Knoten (i.e. Leckageszenarien) (Pudar & Liggett, 1992). Bei n möglichen Sensorpositionen und m Leckageszenarien hat die Sensitivitätsmatrix S die folgende Form:

$$S = \begin{pmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{m1} & \cdots & s_{mn} \end{pmatrix} \qquad \text{Gleichung 2-1}$$

Die einzelnen Werte s_{ij} der Matrix S sind dabei die Druckdifferenz im Knoten j zufolge einer Leckage der Größe f im Knoten i, wobei diese Druckdifferenz auf die Leckagegröße normiert wird:

$$s_{ij} = \frac{\delta p_j}{\delta f_i} = \frac{p_j^{f_i} - \hat{p}_j}{f_i}$$ Gleichung 2-2

mit:

s_{ij} *… Sensitivität des Knoten j auf Leckagen im Knoten i*
$p_j^{f_i}$ *… Druck im Knoten j bei einer Leckage der Größe* f *im Knoten i*
$\hat{p}_j$ *… Druck im Knoten j im ungestörten System*
f_i *…Größe der Leckage in Knoten i*

Um die Sensitivität eines Knotens darstellen zu können, kann das arithmetische Mittel über alle Leckageszenarien gebildet werden:

$$\bar{s}_j = \frac{1}{m}\sum_{i=1}^{m} s_{ij}$$ Gleichung 2-3

mit:
$\bar{s}_j$ *… mittlere Sensitivität des Knoten j*
s_{ij} *… Sensitivität des Knoten j auf Leckagen im Knoten i*
m *… Anzahl der Leckageszenarien*

Die Residuenmatrix (*residual matrix*) R, die für manche OSP-Algorithmen ebenfalls benötigt wird, soll die Differenzen zwischen (simulierten) Messdaten und Simulationsergebnissen beschreiben. Sie ist dementsprechend bei n Sensoren und m Leckageszenarien analog zur Sensitivitätsmatrix S aufgebaut (Casillas et al., 2013):

$$R = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{m1} & \cdots & r_{mn} \end{pmatrix}$$ Gleichung 2-4

Die Werte r_{ij} beschreiben wie schon s_{ij} die Druckdifferenz im Knoten j zufolge einer Leckage der Größe f, wobei die Leckagegröße bei der Berechnung der Residuenmatrix eine andere Leckagegröße sein kann, als bei der Berechnung der Sensitivitätsmatrix, um die Robustheit der Ergebnisse zu erhöhen (Casillas et al., 2013). Außerdem wird auf eine Normierung verzichtet. Die Elemente r_{ij} ergeben sich demnach zu

$$r_{ij} = \delta p_j = p_j^{f_i} - \hat{p}_j$$ Gleichung 2-5

mit:
r_{ij} *Druckabweichung im Knoten j zufolge einer Leckage in Knoten i*
$p_j^{f_i}$ *… Druck im Knoten j bei einer Leckage der Größe* f *in Knoten i*

2.1.2.2 Casillas

Der Algorithmus von Casillas et al. basiert auf der Übereinstimmung von simulierten Messwerten (in Form der Residuenmatrix) und Simulationsergebnissen (in Form der Sensitivitätsmatrix). Die Übereinstimmung wird dabei mittels Vektorprojektion bestimmt (Casillas et al., 2013).

Um Sensoren auszuwählen, wird ein binärer Vektor q mit der Länge n erzeugt:

$$q = [q_1, \dots, q_n] \qquad \text{Gleichung 2-6}$$

Dabei gilt $q_i = 0$, falls am Knoten i kein Sensor platziert wird und umgekehrt $q_i = 1$, falls an diesem Knoten ein Sensor situiert wird. Aufbauend auf diesem Vektor wird dann eine Diagonalmatrix $Q(q)$ erstellt:

$$Q(q) = diag(q_1, \dots, q_m) \qquad \text{Gleichung 2-7}$$

Um die Übereinstimmung zwischen Residuen- und Sensitivitätsmatrix zu bestimmen, wird die Projektionsmatrix ψ (*projection matrix*) berechnet, wobei die einzelnen Elemente dieser Matrix mittels der folgenden Gleichung berechnet werden:

$$\psi_{ij} = \frac{r_i^T Q(q) s_j}{\left|r_i^T Q(q)^T\right| \left|Q(q) s_j\right|} \qquad \text{Gleichung 2-8}$$

mit:
ψ_{ij} … *Projektion des Residuenvektors i auf den Sensitivitätsvektor j*
r_i^T … *transponierter Residuenvektor i*
s_j … *Sensitivitätsvektor j*

In Worten ausgedrückt bedeutet diese Gleichung, dass die Übereinstimmung einer Zeile der Sensitivitätsmatrix mit einer Zeile der Residuenmatrix in Form einer Projektion berechnet wird. Es wird also überprüft, wie gut sich (simulierte) Messdaten mit Druckunterschieden für jedes Leckageszenario mit berechneten Sensitivitäten für jedes Leckageszenario decken.

Um ein Set an Sensorpositionen bewerten zu können, wird überprüft, ob eine Leckage durch die Sensoren q korrekt lokalisiert wird. Dazu wird für jedes Leckageszenario überprüft, ob die größte Projektion in der Projektionsmatrix am richtigen Knoten vorliegt. Aufgrund der Aufbauweise der Projektionsmatrix, muss die größte Projektion auf der Hauptdiagonale liegen, andernfalls liegt bei einem anderen (falschen) Knoten eine bessere Übereinstimmung zwischen Sensitivitäts- und Residuenvektor vor. Um dies abzubilden, wird ein Fehlerindex ε eingeführt und für jedes Leckageszenario berechnet. Dabei gilt:

$$\varepsilon_i(q) = \begin{cases} 0 \dots falls\ \psi_{ii(q)} = \max(\psi_{i1}(q), \dots, \psi_{im}(q)) \\ 1 \dots sonst \end{cases} \qquad \text{Gleichung 2-9}$$

mit:
ε_i … *Fehlerindex für das Leckageszenario i*

Um nun ein Set an Sensorpositionen als Ganzes bewerten zu können, wird das Mittel über alle Werte ε gebildet. Dies stellt gleichzeigt die Zielfunktion dar, da dieser Mittelwert $\bar{\varepsilon}$ minimiert werden soll:

$$\bar{\varepsilon}(q) = \sum_{i=1}^{m} \frac{\varepsilon_i(q)}{m} \rightarrow \min_q \bar{\varepsilon}(q) \qquad \text{Gleichung 2-10}$$

mit:
$\bar{\varepsilon}$ *… mittlerer Fehlerindex*
ε_i *… Fehlerindex des Leckageszenarios* i
m *… Anzahl der Leckageszenarien*

$\bar{\varepsilon}$ gibt demnach jenen Anteil an Knoten an, der nicht korrekt lokalisiert werden kann. Werden alle simulierten Leckagen dem korrekten Knoten zugeordnet, gilt $\bar{\varepsilon}(q) = 0$, wird jedoch keine simulierte Leckage korrekt zugeordnet, gilt $\bar{\varepsilon}(q) = 1$.

2.1.2.3 SPUDU

SPUDU (*Sensor Placement Under Demand Uncertainties*) wurde vom Institut für Siedlungswasserwirtschaft und Landschaftswasserbau entwickelt (Steffelbauer & Fuchs-Hanusch, 2016). Er stellt eine Erweiterung des Algorithmus von Casillas dar, inkorporiert allerdings die Tatsache, dass Knoten, deren Verbräuche schwanken, Druckschwankungen erfahren. Knoten mit großen Druckschwankungen infolge stark schwankenden Verbrauchs sollten demnach nicht als Sensorpositionen gewählt werden.

SPUDU verwendet wie der Algorithmus von Casillas et al. eine Projektionsmatrix, allerdings werden hier im Vorhinein die Standardabweichung der Drücke zufolge von Verbrauchsschwankung ermittelt (beispielsweise mittels Monte-Carlo-Simulationen). Die Standardabweichung des Druckes im Knoten i σ_{pi} fließt anschließend gewichtet mittels eines Faktors ω in die Berechnung der einzelnen Werte ψ_{ij} ein, indem die Werte des Residuenvektors abgemindert werden:

$$\psi_{ij} = \frac{\left(r_i^T - \omega\sigma_{pi}\right)Q(q)s_j}{\left|\left(r_i^T - \omega\sigma_{pi}\right)Q(q)^T\right|\left|Q(q)s_j\right|} \qquad \text{Gleichung 2-11}$$

mit:
r_i^T *… transponierter Residuenvektor* i
s_j *… Sensitivitätsvektor* j
σ_{pi} *… Standardabweichung der Druckschwankungen aufgrund der Verbrauchsunsicherheit im Knoten* i
ω *… Gewichtungsfaktor für die Druckschwankungen aufgrund der Verbrauchsunsicherheit*

Dies führt dazu, dass Knoten mit einer höheren Verbraucherunsicherheit bestraft werden, indem die zugehörige Projektion von Sensitivitäts- und Residuenvektor verkleinert wird.

Daran anschließend wird analog zum Algorithmus von Casillas et al. je Leckageszenario ein Fehlerindex ε gebildet. Der daraus berechenbare mittlere Fehlerindex $\bar{\varepsilon}$ stellt abschließend wieder die Zielfunktion des Algorithmus dar (siehe Gleichung 2-9 und Gleichung 2-10).

2.1.3 Kostenfunktion

Von Steffelbauer et al. (2016) wurde der Zusammenhang zwischen Sensoranzahl und Fitness untersucht. Dazu wurden mit SPUDU mehrere OSP-Berechnungen mit jeweils unterschiedlicher Sensoranzahl und variierendem Gewichtungsfaktor ω durchgeführt und anschließend eine passende Funktion für den Zusammenhang zwischen Sensoranzahl und Fitness gesucht.

Evaluiert wurden diese Funktionen mittels mehrerer Anpassungsstatistiken:

- Chi-Quadrat-Test
- Reduzierter Chi-Quadrat-Test
- Akaikes Informationskriterium
- Bayessches Informationskriterium

Zwei der untersuchten Funktionen lieferten dabei besonders gute Ergebnisse:

$$f_1(N) = (a + b * N)^{-c} \qquad \text{Gleichung 2-12}$$

$$f_2(N) = a * N^{-b} \qquad \text{Gleichung 2-13}$$

mit:
N *… Sensoranzahl*
a, b, c *… zu bestimmende Parameter*

Beide Funktionen folgen einem Potenzgesetz, was dazu führt, dass die Platzierung eines weiteren Sensors zu einer immer geringeren Verbesserung der Fitness führt.

$f_1(N)$ weist dabei bessere Werte in Bezug auf Chi-Quadrat-Test und reduzierten Chi-Quadrat-Test auf, während $f_2(N)$ bessere Ergebnisse bei den Informationskriterien liefert. Unter Betrachtung der Standardfehler der Parameter a, b und c fällt schließlich die Wahl auf $f_2(N)$ als geeignetste Funktion. Abbildung 2-1 zeigt Kostenfunktionen für verschiedene Gewichtungsfaktoren ω bei der Verwendung von SPUDU.

Die Funktion wird dabei allerdings nicht mittels mehrerer Leckagegrößen errechnet, sondern nur für eine einzige, da die optimierten Sensorpositionen nicht sensitiv auf die Leckagegröße sind (Blesa et al., 2015).

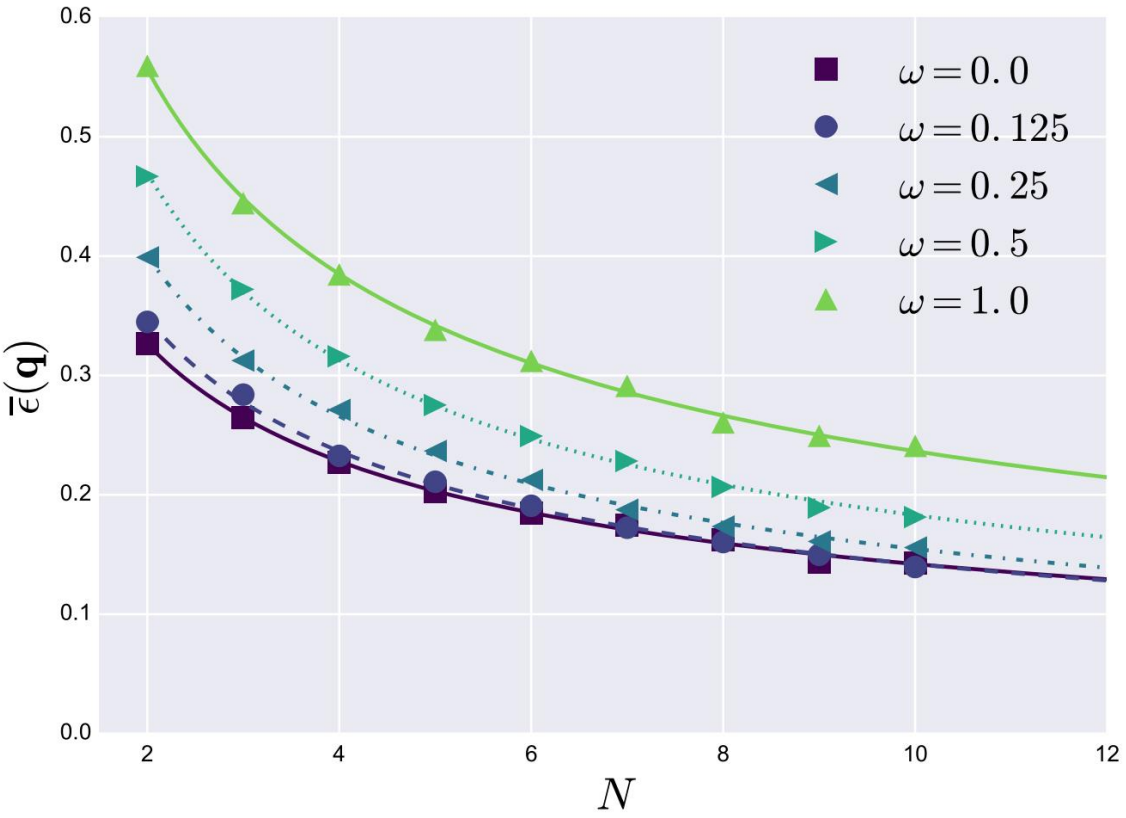

Abbildung 2-1: Kostenfunktionen für unterschiedliche Gewichtungsfaktoren ω für SPUDU (Steffelbauer & Fuchs-Hanusch, 2016)

2.1.4 Optimierungsmethoden

Die in Kapitel 2.1.2 erläuterten sensitivitätsbasierten Algorithmen benötigen einen Optimierer, der versucht, die Zielfunktion eines Algorithmus zu optimieren, d.h. die beste Lösung für das Problem zu finden. Es sollen damit jene Parameter gefunden werden, deren Kombination die optimale Lösung findet (beispielsweise jene Sensorpositionen, welche die beste Lösung für die jeweilige Zielfunktion ergeben).

Es existieren verschiedenste Arten von Optimierern, von denen zwei im Folgenden dargestellt werden sollen: Genetische Algorithmen (Goldberg, 1989) und Differential Evolution (Storn & Price, 1997). Beide gehören zur Gruppe der stochastischen Algorithmen. Im Gegensatz zu anderen Algorithmen können Derartige aus einem lokalen Optimum ausbrechen um ein (besseres) globales Optimum zu finden (Steffelbauer, 2018).

2.1.4.1 Genetische Algorithmen

Genetische Algorithmen basieren auf zufällig generierten Lösungen, die über mehrere Iterationsschritte hinweg optimiert werden. Gegenüber anderen Opti-

miermethoden besitzen Genetische Algorithmen den Vorteil, dass eine große Anzahl an Lösungen getestet und verbessert wird, und nicht nur eine einzige Lösung immer weiter optimiert wird.

Nicht nur die Bezeichnung der Algorithmen ist der Biologie entnommen, auch die Herangehensweise ist an die Biologie angelehnt ebenso wie die Bezeichnungen innerhalb des Algorithmus. Eine Lösung ist hier ein Individuum, das Teil einer Population ist, und das Gene besitzt. Diese Gene bilden dabei die Parameter der Lösung ab (z.B. die Sensorplatzierung). Die Gene werden über mehrere Generationen hinweg vererbt, rekombiniert und mutiert. Von Generation zu Generation werden dabei nicht alle Individuen übernommen, sondern nur ein Teil der Lösungen wird basierend auf deren Fitness in der nächsten Generation zugelassen. Abbildung 2-2 zeigt dieses Schema graphisch.

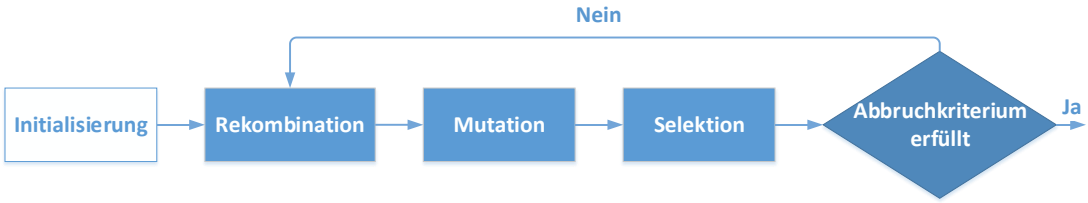

Abbildung 2-2: Genetische Algorithmen: Schema (Steffelbauer, 2018, mod.)

Jede Generation durchläuft demnach die folgenden Schritte:

1. Rekombination
2. Mutation
3. Selektion

Rekombination beschreibt dabei die Kombination der Gene zweier Individuen, aus denen wiederum zwei neue Individuen entstehen. Eine Variante ist dabei, mit einer vorgegebenen Wahrscheinlichkeit die Gene der Eltern ab einer bestimmten Stelle c des Genoms miteinander zu vertauschen. Abbildung 2-3 zeigt diesen Vorgang schematisch.

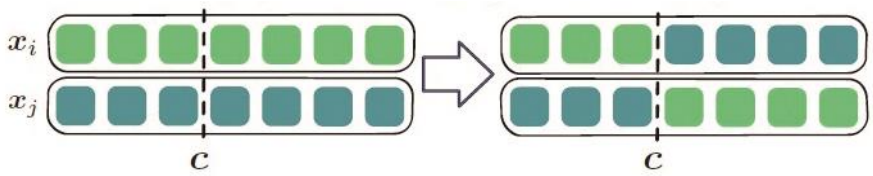

Abbildung 2-3: Genetische Algorithmen: Rekombination (Steffelbauer, 2018, mod.)

Unter Mutation wird jener Vorgang verstanden, bei dem ein Chromosom eines Individuums mit einer vordefinierten Wahrscheinlichkeit gegen einen zufällig generierten Wert aus dem möglichen Parameterraum ersetzt wird. In Abbildung 2-4 wird die Mutation graphisch dargestellt.

Abbildung 2-4: Genetische Algorithmen: Mutation (Steffelbauer, 2018, mod.)

Die Auswahl jener Individuen, die nun in die nächste Generation übernommen wird, kann unterschiedlich vonstattengehen. So können jene Individuen herangezogen werden, die in Bezug auf ihre Fitness die besten Ergebnisse liefern, oder es wird zugelassen, dass auch schlechtere Lösungen in die nächste Generation gelangen. Eine Variante dafür ist beispielsweise, je zwei Lösungen zu vergleichen und die schlechtere der beiden auszuscheiden. Dieser Wettkampf wird so lange durchgeführt, bis die richtige Anzahl an Individuen erreicht wird. Die verbliebenen Individuen bilden die neue Generation und der Kreislauf beginnt von neuem.

Abgebrochen wird dieser Prozess beim Eintreten eines Abbruchkriteriums, wie beispielsweise beim Erreichen einer maximalen Generationenanzahl.

2.1.4.2 Differential Evolution

Differential Evolution ähnelt in der Funktionsweise genetischen Algorithmen. Auch diese Art von Optimierern verbessert Populationen über mehrere Generationen hinweg und bedient sich der Rekombination, Mutation und Selektion, allerdings auf eine unterschiedliche Art und Weise. Abbildung 2-5 zeigt wiederum das Schema der Differential Evolution.

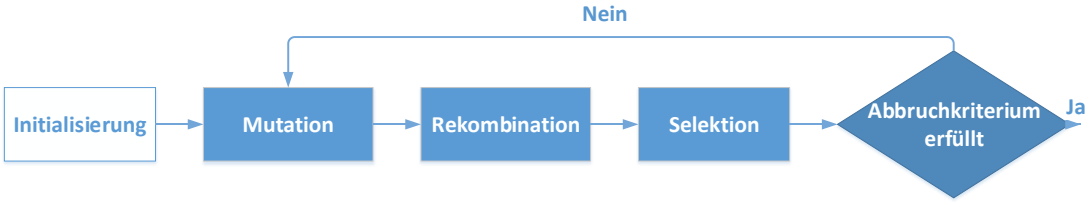

Abbildung 2-5: Differential Evolution: Schema (Steffelbauer, 2018, mod.)

Wie in der Abbildung ersichtlich ist, sind Mutation und Rekombination in der Reihenfolge vertauscht. Mutation und Rekombination unterschieden sich aber zusätzlich auch in der Funktionsweise von Genetischen Algorithmen.

Bei der Mutation wird ein Differenzvektor zwischen (mindestens) zwei zufällig gewählten Individuen x_{r1} und x_{r2} gebildet und mittels eines Faktors F_i gewichtet zu einem dritten Individuum x_{r3} addiert. Es existieren verschiedene Varianten (Mutationsstrategien), wie die Auswahl des dritten Individuums erfolgen kann. So kann das dritte Individuum entweder zufällig aus den Individuen ausgewählt werden, oder es kann das beste Individuum herangezogen werden oder es wird jenes Individuum ausgewählt, das auch zur anschließenden Rekombination verwendet wird.

Abbildung 2-6 zeigt beispielsweise die Mutationsstrategie *DE/best/1*, bei der der Differenzvektor zum besten Individuum addiert wird. Das resultierende Individuum v berechnet sich dabei über (Storn, 1996):

$$v = x_{best} + F * (x_{r1} - x_{r2}) \qquad \text{Gleichung 2-14}$$

mit:
v *… erzeugtes Individuum*
x_{best} *… bestes Individuum*
x_{r1}, x_{r2} *… zufällig gewählte Individuen*
F *… Gewichtungsfaktor*

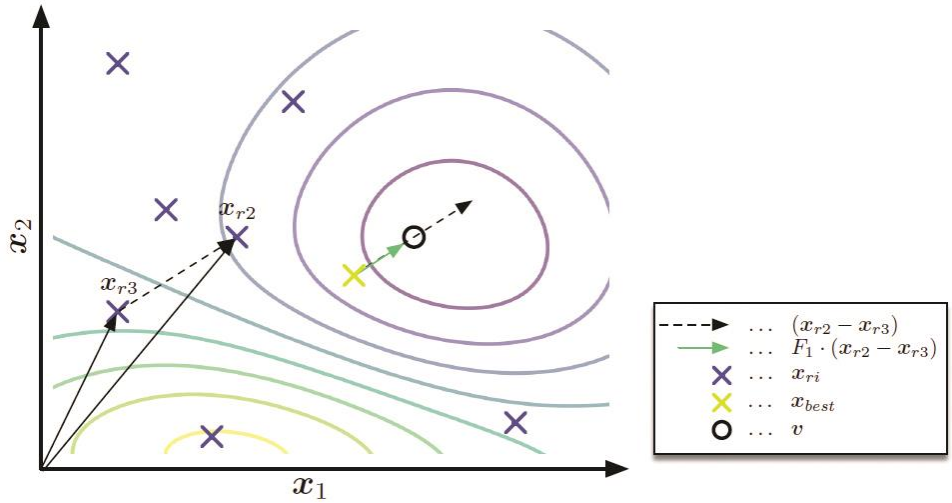

Abbildung 2-6: Differential Evolution: Mutation mit DE/best/1 (Steffelbauer, 2018)

Eine andere Strategie ist *DE/rand/1*, bei dem der Differenzvektor zu einem zufällig gewählten Individuum addiert wird:

$$v = x_{r1} + F * (x_{r2} - x_{r3}) \qquad \text{Gleichung 2-15}$$

mit:
v *… erzeugtes Individuum*
x_{best} *… bestes Individuum*
x_{r1}, x_{r2}, x_{r3} *… zufällig gewählte Individuen*
F *… Gewichtungsfaktor*

Bei der Rekombination wird ein neues Testindividuum erzeugt. Dabei wird iterativ für jedes Chromosom des Testindividuums eine Zahl $rand_{i,j}$ aus dem Intervall $[0\,,1]$ gezogen. Ist diese Zahl größer als eine vorher definierte Crossover-Konstante CR, wird dieses Chromosom von einem Zielindividuum x_j übernommen, ansonsten wird das Chromosom vom Spenderindividuum v_j gewählt. Zusätzlich wird eine weitere Zahl j_{rand} zufällig aus dem Intervall $[1, n]$ gezogen, wobei n der Länge des Gens entspricht. Diese Zahl verweist auf ein Chromosom des Gens, das nicht ersetzt wird, ungeachtet der Konstanten CR. Dadurch wird verhindert,

dass das erzeugte Individuum ident mit dem Zielvektor ist. Das aus der Kombination der Vektoren resultierende Testindividuum u_j errechnet sich demnach wie folgt:

$$u_{i,j} = \begin{cases} v_{i,j} \dots falls\ rand_{i,j} \leq CR \ \vee \ i = j_{rand} \\ x_{i,j} \ \dots \ sonst \end{cases}$$ Gleichung 2-16

mit:
$u_{i,j}$ *… Parameter des Testindividuums* u *an der Stelle* i
$v_{i,j}$ *… Parameter des Spenderindividuums* v_j *an der Stelle* i
$x_{i,j}$ *… Parameter des Zielindividuums* x_j *an der Stelle* i
$rand_{i,j}$ *… aus dem Intervall* $[0\ ,1]$ *zufällig gezogene Zahl*
j_{rand} *… aus dem Intervall* $[1, n]$ *zufällig gezogene Zahl*
CR *… Crossover-Konstante*

Abbildung 2-7 zeigt schematisch diesen Ablauf.

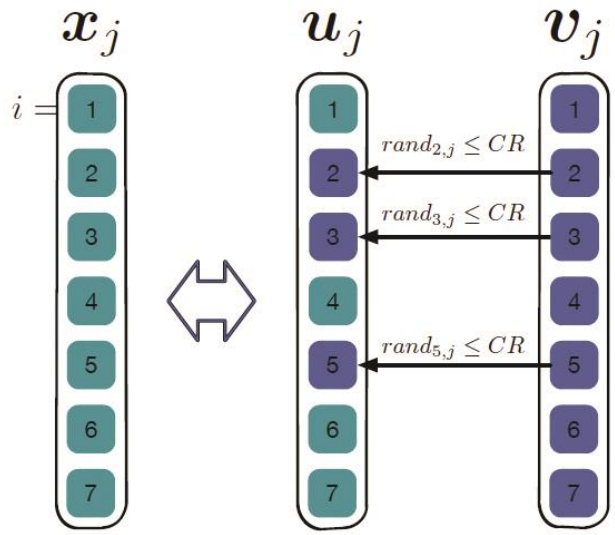

Abbildung 2-7: Differential Evolution: Rekombination (Storn & Price, 1997; Steffelbauer, 2018)

Auf die Rekombination folgt wiederum die Selektion, wozu die Fitness-Werte der einzelnen Individuen der Generation errechnet werden. Die Fitness von x_j und u_j wird jeweils verglichen und das bessere Individuum in die nächste Generation übernommen.

Dieser Vorgang wird, wie schon bei den genetischen Algorithmen, sooft iteriert, bis ein Abbruchkriterium erfüllt ist.

2.1.5 Pythonbasierte Sensorplatzierung

Um Sensorpositionen mittels eines Algorithmus bestimmen zu können, wurden am Insitut für Siedlungswasserwirtschaft und Landschaftswasserbau bereits mehrere Pythonklassen entwickelt.

Grundlage für die OSP-Bestimmung ist dabei OOPNET (*object-oriented EPANET*), eine pythonbasierte Programmierschnittstelle zwischen EPANET und Python. Über diese Schnittstelle können verschiedenste Analysen in EPANET-Modellen mittels einfacher Pythoncodes vorgenommen werden.

Die Pythoncodes für die OSP-Algorithmen und Optimierer sind nicht Teil von OOPNET, allerdings ist OOPNET Voraussetzung für die Verwendung dieser Codes. OSP-Algorithmen und Optimierer sind im Code getrennt, können also unabhängig voneinander konfiguriert und anschließend zusammengefügt werden.

Für die OSP-Algorithmen existiert eine gemeinsame Überklasse *OSP*, die sich wiederum in zwei Klassen *GOSP* (*Graph Theory based OSP Algorithms*) und *SOSP* (*Sensitivity Matrix based OSP Algorithms*) aufteilt. Diesen beiden Klassen sind wiederum die verschiedenen Algorithmen als Unterklassen zugeordnet (siehe Abbildung 2-8). Alle oben angeführten OSP-Algorithmen wurden bereits als Klasse implementiert.

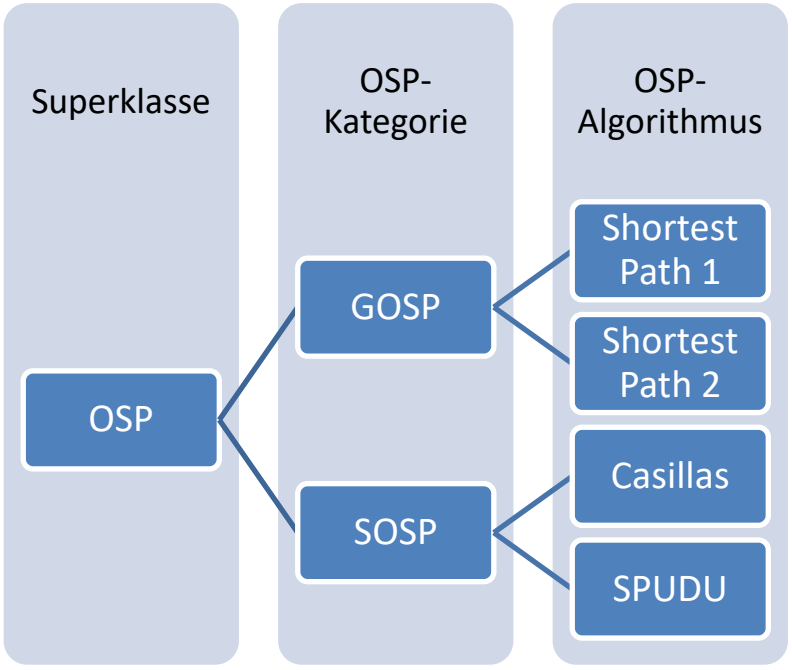

Abbildung 2-8: OSP-Pythoncodestruktur

Bei den graphenbasierten Algorithmen können folgende Einstellungen vorgenommen werden:

- Anzahl der zu platzierenden Sensoren
- Mögliche Sensorpositionen in Form der Knoten-IDs im hydraulischen Modell
- Bereits platzierte Sensoren in Form der zugehörigen Knoten-IDs

Die sensitivitätsbasierten Algorithmen erweitern diese Liste mit zusätzlichen Einstellungsmöglichkeiten, die dem jeweiligen Algorithmus entsprechen (z.B. Gewichtungsfaktor ω bei SPUDU).

Zur Berechnung der Sensitivitäts- und Residuenmatrizen existieren wiederum eigene Pythonklassen, bei deren Instanziierung angegeben werden kann, welcher Parameter zur Leckagesimulation herangezogen werden soll (*demand* zur Simulation einer fixen Leckagegröße bzw. *emittercoefficient* zur Simulation druckabhängiger Leckagen) und um welchen Wert der Parameter variiert werden soll (i.e. die Leckagegröße).

Zur Optimierung stehen ein genetischer Algorithmus und Differential Evolution zur Verfügung. Beide verwenden DEAP (*Distributed Evolutionary Algorithms in Python*) zur Optimierung (Fortin et al., 2012).

Um nun einen OSP-Algorithmus zu optimieren, wird einer Instanz der jeweiligen OSP-Klasse eine Instanz des gewählten Optimierers zugewiesen und anschließend die Berechnung gestartet.

Ergebnis der OSP-Berechnung sind die errechneten Sensorpositionen, sowie bei sensitivitätsbasierten Algorithmen der zugehörige Fitness-Wert.

Alternativ ist aufgrund des Aufbaus des Codes bei den sensitivitätsbasierten Methoden die Fitness für bestimme Sensorpositionen zu berechnen.

2.2 Serious Gaming

Dieses Kapitel beschäftigt sich mit dem Thema der Serious Games. Dazu folgt zuerst ein allgemeiner Teil über Serious Games sowie allgemeine Beispiele, bevor drei existierende Serious Games aus dem Bereich der Trinkwasserversorgung vorgestellt werden. Abschließend wird eine Klassifizierungsmethode von Serious Games im Bereich der Wasserwirtschaft dargelegt.

2.2.1 Allgemeines

Der Begriff *Serious Game* wurde von Clark C. Abt 1970 in seinem *Buch Serious Games* geprägt. Er unterschied zwischen Spielen, die ernsthaft *(serious*) oder locker (*casual*) gespielt werden. Ein Serious Game war für ihn dabei ein Spiel, das in erster Linie der (Aus-)Bildung dient und nicht hauptsächlich der Unterhaltung (Abt, 1987).

Mittlerweile existieren andere Definitionen, wobei keine davon als allgemein anerkannt gilt (Dörner et al., 2016).

Eine modernere Definition wäre beispielsweise die Folgende:

> *„A serious game is a digital game created with the intention to entertain and to achieve at least one additional goal (e.g., learning or health). These additional goals are named characterizing goals."* (Dörner et al., 2016)

Diese Definition beinhaltet demnach, dass ein Serious Game ein digitales Spiel sein muss, das einerseits Spaß machen muss, aber andererseits auch noch ein zusätzliches Ziel verfolgt. Hier erfolgt keine Fixierung mehr auf den Bildungssektor, woraufhin völlig andere Bereiche umfasst werden können. Außerdem ist in dieser Definition nicht umfasst, ob das verfolgte Ziel überhaupt erreicht wird, da schon die Intention der Entwickler für die Klassifizierung als Serious Game ausreicht (Dörner et al., 2016).

Beispiele für Bereiche zusätzlich zum Bildungssektor sind der Gesundheitsbereich, Gesellschaft und öffentliche Wahrnehmung, aber auch Werbung.

2.2.2 Beispiele für Serious Games

In den folgenden Unterkapiteln werden einige Beispiele für die Umsetzung von Serious Games präsentiert.

2.2.2.1 80Days

Bei 80Days handelt es sich um ein von der Europäischen Kommission gefördertes Forschungsprojekt. Zielgruppe des Spiels sind 14- bis 15-Jährige, die in die Rolle eines 14 Jahre alten Kindes schlüpfen. Im Spiel wird die Welt gemeinsam mit einem Außerirdischen erforscht, der Informationen über die Erde sammeln soll. Als sich herausstellt, dass das Ziel der Außerirdischen die Eroberung der Erde ist, sollen die Spielenden die Erde vor den Außerirdischen retten, indem sie das vorher gesammelte Wissen nutzen. Den Spielerinnen und Spielern wird dadurch dabei spielerisch Geographie näher gebracht (Kickmeier-Rust et al., 2009).

Abbildung 2-9 zeigt Screenshots von 80Days. Eine Demoversion des Spiels kann unter http://www.eightydays.eu/index.html bezogen werden.

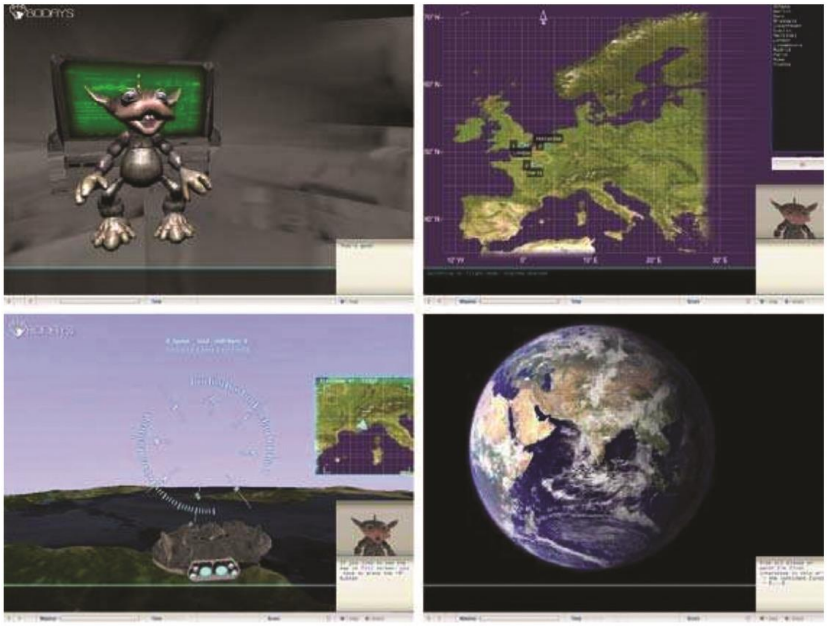

Abbildung 2-9: Screenshots von 80Days (Kickmeier-Rust et al., 2009)

2.2.2.2 SchaVIS – SchadensVISualisierung

SchaVIS dient der Bewusstseinsbildung in Bezug auf Hochwasser. Die Spielenden müssen in einem Haus mit einem vorgegebenen Zeitbudget Handlungen setzen, um durch ein bevorstehendes Hochwasser verursachte Schäden möglichst gering zu halten. Das Spiel ist vorbei, sobald das vorgegebene Zeitbudget verbraucht ist oder der Spielende sich in eine gefährliche Situation gebracht hat. Darauf folgt eine Simulation, in welcher der Wasserspiegel im Haus immer höher steigt und die durch das Hochwasser verursachten Schäden berechnet werden.

Beim Spielen soll vermittelt werden, welche Handlungen im Falle eines Hochwassers gesetzt werden sollen und welche Situationen gefährlich sein können (Dörner et al., 2016).

Abbildung 2-10 zeigt Screenshots von SchaVIS. Das Spiel kann unter https://hochwassermanagement.rlp-umwelt.de/servlet/is/174900/ heruntergeladen werden.

Abbildung 2-10: Screenshots von SchaVIS (Dörner et al., 2016)

2.2.2.3 Aqua Republica

Aqua Republica ist ein von DHI, UNEP (*United Nations Environment Programme*) und DHI-UNEP (einer Partnerschaft von DHI und UNEP) entwickeltes Serious Game. Es kombiniert eine graphische Benutzeroberfläche mit MIKE BASIN, einem Simulationsprogramm für die Planung und das Management von Flusseinzugsgebieten.

In diesem Spiel muss unter Berücksichtigung von Umwelt und Wirtschaft ein Flusseinzugsgebiet geplant werden. Maßnahmen zum Schutze der Umwelt kosten Geld, das wiederum verdient werden muss. Das sich daraus entwickelnde Wechselspiel stellt die Spielenden auch mit Ereignissen, wie beispielsweise Streiks, vor Herausforderungen. Die Bewertung der Spielerinnen und Spieler basiert auf deren getroffenen Entscheidungen. Spielende, die auf die Umwelt Bedacht genommen haben, erreichen eine höhere Punktezahl als jene, die weniger auf Ressourcenschonung und Umweltschutz gesetzt haben (Chew et al., 2014).

Abbildung 2-11 zeigt einen Screenshot von Aqua Republica. Das Spiel kann unter http://aquarepublica.com/ gespielt werden.

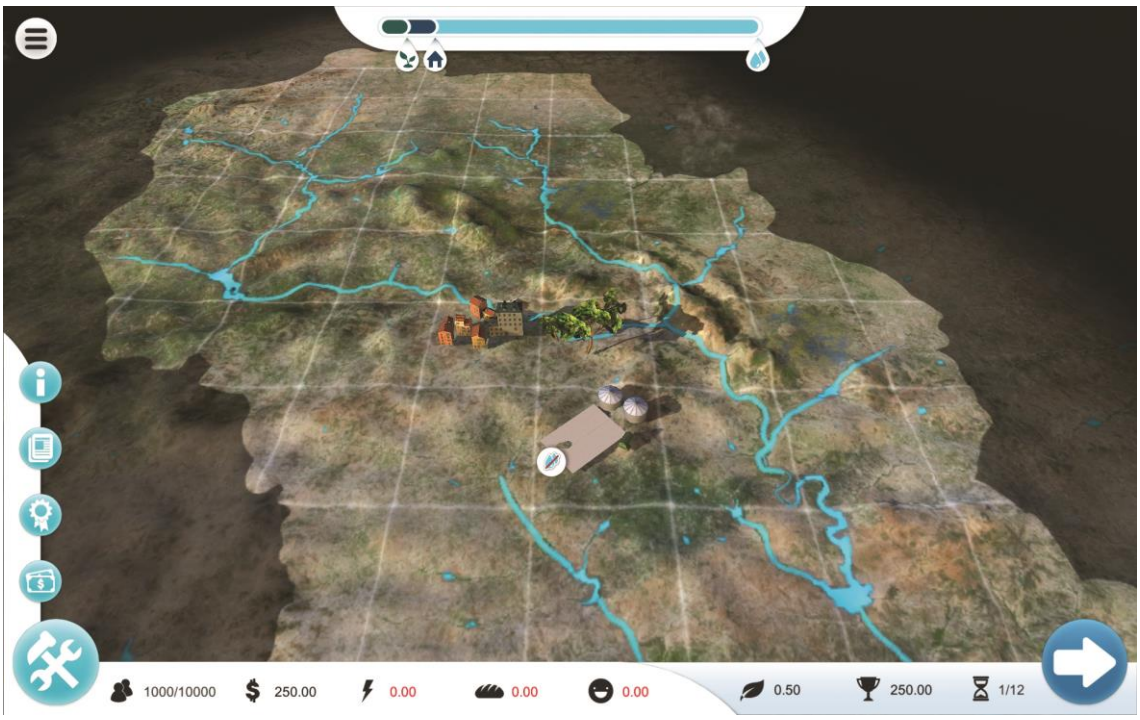

Abbildung 2-11: Screenshot von Aqua Republica

2.2.2.4 Aqualibrium

Nach diesen allgemeinen Beispielen sollen nun drei Serious Games gezeigt werden, die sich mit der Thematik der Trinkwasserversorgung beschäftigen.

Den Anfang macht die digitale Version des Wettbewerbs *Aqualibrium*. Beim Aqualibrium-Wettbewerb müssen in einem Raster 0,28 m lange Leitungen mit einem Durchmesser von je 3 oder 6 mm in einem Raster so angeordnet werden, dass drei Behälter gleichmäßig von einem vorgegebenen Einspeisepunkt aus mit je einem Liter Wasser befüllt werden. Die Positionen der Behälter und des Einspeisepunkts (ein höher gelegener Behälter) im Raster sind vorgegeben. Es müssen mindestens zehn Leitungen verwendet werden und das entstehende Netz darf keine Stichleitungen beinhalten. Verbunden werden die Leitungen mit verschiedenen Formstücken, wie 90°-Bögen und T-Kreuzungen (Meniconi et al., 2017).

Dieses Wettbewerbskonzept wurde mit einer Weboberfläche als Serious Game umgesetzt. Es existiert zwar keine wissenschaftliche Veröffentlichung über dieses Spiel, es kann jedoch (so wie das folgende Beispiel) über http://waterserious-games.org/ aufgerufen werden.

In der Weboberfläche können analog zum Wettbewerb Leitungen platziert werden. Mit einem Button kann das Netzwerk evaluiert werden, woraufhin die Füllstände der drei zu füllenden Behälter angezeigt und in einen Punktestand umgewandelt werden. Je gleichmäßiger die Behälter befüllt werden, desto geringer wird der Punktestand. Ziel für die Spielenden ist es demnach, einen möglichst

geringen Punktestand zu erreichen. Die unterschiedlichen Leitungsdurchmesser werden durch unterschiedliche Strichstärken visualisiert.

Evaluierte Lösungen können auch gespeichert werden, um sie später wieder aufrufen zu können.

Abbildung 2-12 zeigt einen Screenshot von Aqualibrium.

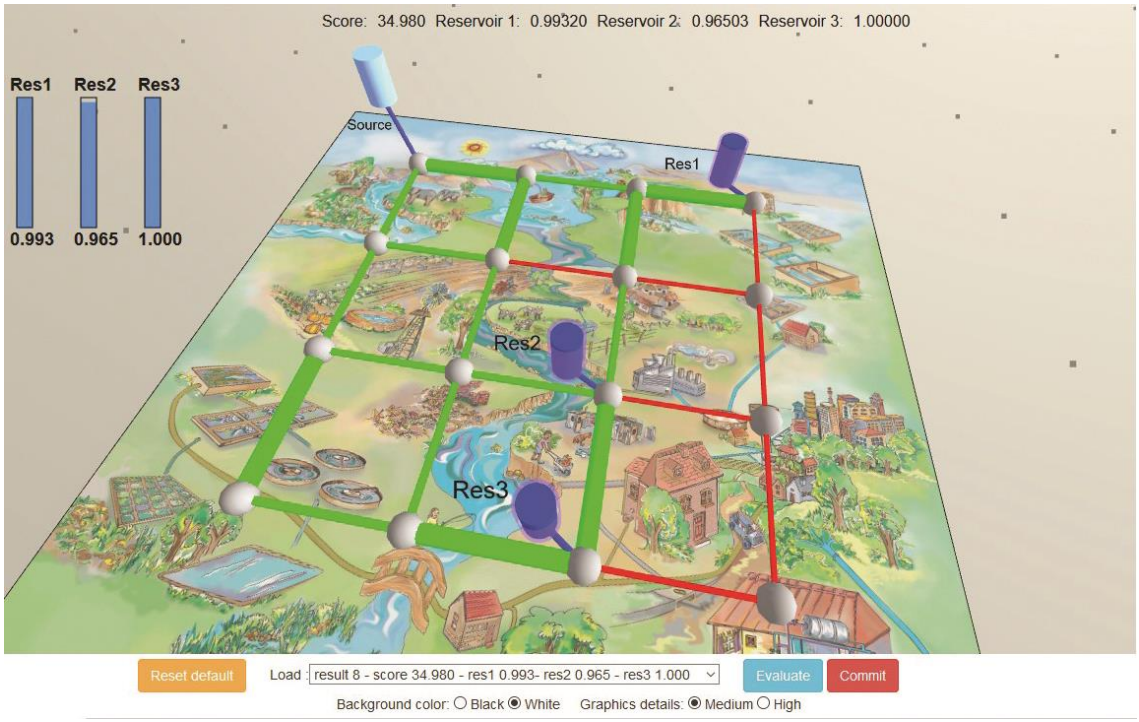

Abbildung 2-12: Screenshot von Aqualibrium

2.2.2.5 SeGWADE

SeGWADE (*Serious Game for Water Distribution System Analysis, Design and Evaluation*) beschäftigt sich mit dem New-York-Tunnel-Problem. Ziel ist es in diesem Spiel, in einem bestehenden Trinkwasserversorgungssystem parallele Leitungen mit einem Durchmesser von 1,5 bis 5,2 m so hinzuzufügen, dass die im Anfangszustand nicht erfüllten Mindestdrücke an jedem Knoten eingehalten werden. Je nachdem, wo eine parallele Leitung verlegt werden soll, weisen die Leitungen unterschiedliche Längen auf. Über eine Kombination aus gewähltem Durchmesser und Länge der Leitung werden Kosten berechnet. Die Summe der entstehenden Kosten soll unter Einhaltung der Mindestdrücke minimiert werden. Die Zielfunktion dieses Problems definiert sich wie folgt (Savic et al., 2016):

$$C_{inf} = f(D_1, \dots, D_{N_l}) = \sum_{j=1}^{N_l} C(D_j, L_j) \rightarrow \min(C_{inf}) \qquad \text{Gleichung 2-17}$$

$$C(D_j, L_j) = 1{,}1 * D_j^{1{,}24} * L_j \qquad \text{Gleichung 2-18}$$

Unter den Nebenbedingungen:

$$H_i \geq H_{i,min}, i = 1, \dots N_n$$

$$D_j \in D(j = 1, \dots N_d)$$

mit:

C_{inf} … *Gesamtkosten*

D_j … *Durchmesser der Leitung j aus einem Set verfügbarer Durchmesser*

L_j… *Länge der Leitung j*

C … *Kosten einzelner Leitungen in Abhängigkeit von Durchmesser und Längen*

H_i … *Druckhöhe im Knoten i*

$H_{i,min}$ … *Mindestdruckhöhe im Knoten i*

N_n … *Anzahl Knoten*

N_d … *Anzahl verfügbarer Leitungsparameter*

Es handelt sich demnach um ein Minimierungsproblem, für das Lösungen auch mittels eines Optimierers generiert werden können.

Die Weboberfläche ist ähnlich der von Aqualibrium aufgebaut. Auch hier werden Leitungen mit unterschiedlichen Durchmessern durch die Spielenden zum Netz hinzugefügt. Ebenso wird auch in diesem Spiel eine Evaluierung des Netzes nach einem Klick auf einen Button durchgeführt und es können Lösungen gespeichert werden. Die Einhaltung der Mindestdrücke in den einzelnen Knoten wird über eine Farbskala dargestellt. Die Leitungsdurchmesser werden auch hier durch unterschiedliche Strichstärken visualisiert.

Abbildung 2-13 zeigt einen Screenshot von SeGWADE. Das Spiel kann unter http://waterseriousgames.org/ gespielt werden.

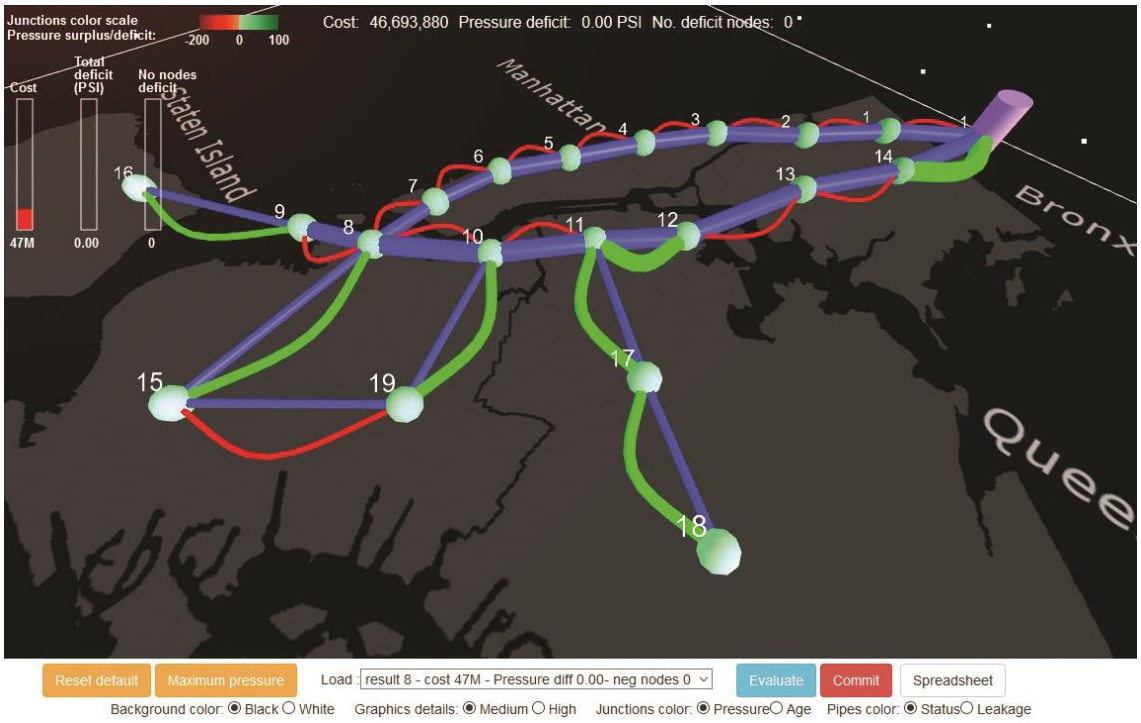

Abbildung 2-13: Screenshot von SeGWADE

2.2.2.6 Network Pipe Sizing

Network Pipe Sizing ist ein weiteres Serious Game aus dem Bereich der Trinkwasserversorgung. Bei diesem Spiel müssen die Spielenden ein Trinkwassernetz redimensionieren. Ähnlich zu SeGWADE führen ersetzte Leitungen zu Kosten, die minimal gehalten werden sollen. Randbedingung ist auch hier, dass ein Mindestdruck eingehalten werden muss. Im Spiel stehen momentan fünf verschiedene Level zur Verfügung, die nacheinander gespielt werden können (Laucelli et al., 2018).

Abbildung 2-14 zeigt einen Screenshot von Network Pipe Sizing. Das Spiel ist nicht öffentlich zugänglich.

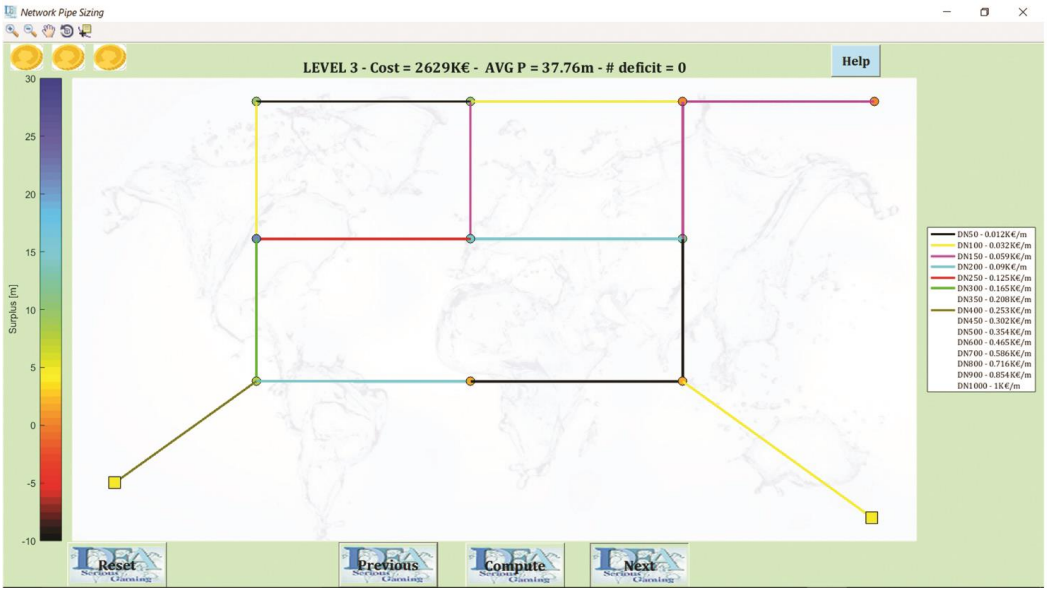

Abbildung 2-14: Screenshot von Network Pipe Sizing (Laucelli et al., 2018)

2.2.3 Klassifizierung von Serious Games aus der Wasserwirtschaft

Savic et al. (2016) haben neben SeGWADE auch ein Klassifizierungssystem für Serious Games aus dem Bereich der Wasserwirtschaft entwickelt. Die folgenden Kriterien werden zur Klassifizierung herangezogen:

- Anwendungsgebiet
- Spielziel
- Initialisierung des Spiels
- Anzahl und Art der Spielenden
- Bedienoberfläche
- Verwendetes Simulationsmodell
- Realismus
- Leistungsfeedback
- Fortschrittsüberwachung
- Portabilität

Beim **Anwendungsgebiet** wird grob zwischen zwei verschiedenen Bereichen unterschieden: Flussgebietsmanagement und Siedlungswasserwirtschaft.

Das **Spielziel** kann entweder ein für das Spiel spezifisches Ziel (wie beispielsweise die Minimierung der Kosten bei SeGWADE), oder bei Spielen ohne explizites Ziel auch Bewusstseinsbildung o.ä. sein.

Initialisierung beschreibt die Art und Weise, wie die Spielenden an das Spiel herangeführt werden. Bei manchen Spielen wird keine Unterstützung beim Spielstart benötigt, während bei anderen Spielen Workshops abgehalten werden.

Betreffend **Anzahl** wird zwischen Einzel- und Mehrpersonenspielen unterschieden. Mehrpersonenspiele haben den Vorteil, dass von den verschiedenen Spielenden unterschiedliche Rollen eingenommen werden können. Mit der **Art** der Spielenden ist die Zielgruppe des Spiels gemeint. Diese können verschiedenste Gruppen, beispielsweise Studierende, Grundstückseigentümer oder Experten aus verschiedensten Bereichen sein.

Die **Bedienoberfläche** kann entweder computerbasiert, eine physische Oberfläche wie bei klassischen Brettspielen oder eine Kombination aus beidem sein.

Beim verwendeten **Simulationsmodell** kann es sich je nach Anwendung um ein sehr simples Modell, oder aber um ein Modell basierend auf professioneller Simulationssoftware handeln.

Der **Realismus** des Spiels bestimmt sich über die im Spiel getroffenen Vereinfachungen bei der Darstellung realer Probleme.

Ein Spiel kann die Leistung der Spielenden kontinuierlich oder im Anschluss an eine bestimmte Handlung, wie beispielsweise der Betätigung eines Buttons, bewerten. Diese Unterscheidung beschreibt der Begriff **Leistungsfeedback**.

Fortschrittsüberwachung bedeutet, dass Zwischenergebnisse gespeichert werden können.

Die **Portabilität** beschreibt, ob es sich beim Spiel um ein Online- oder Offline-Spiel handelt.

3 Methodik

Die Erstellung eines Serious Games, das sich mit der Problematik der optimalen Sensorplatzierung beschäftigt, wurde in drei Schritte geteilt:

- Adaptierung bestehender OSP-Algorithmen und -Pythoncodes
- Erstellung einer webbasierten Spieloberfläche
- Durchführung eines Feldversuchs.

3.1 Adaptierung bestehender OSP-Algorithmen und -Pythoncodes

Der bisher vorhandene Code zielte darauf ab, mit vorab gewählten Eingangsparametern (wie beispielsweise der gewünschten Sensoranzahl) Sensorpositionen in einem Trinkwasserversorgungssystem zu bestimmen. Aufgrund der Strukturierung dieses Codes war es allerdings auch möglich, ein Set von Sensorpositionen mittels der Zielfunktion eines Algorithmus zu evaluieren.

Der bestehende Code wies jedoch einen Nachteil auf: Leckagen wurden nicht auf Leitungen simuliert, sondern auf den bereits bestehenden Knoten im hydraulischen Modell. Dies lässt jedoch keinen Rückschluss darauf zu, welche Leitungen von einem Set an Sensorpositionen abgedeckt sind. Da jedoch genau das in der Weboberfläche visualisiert werden sollte, musste der bestehende Code adaptiert werden. Zusätzlich sind OSP-Berechnung mit sensitivitätsbasierten Methoden sehr rechenintensiv, was derartige Berechnungen auf einem handelsüblichen Computer erschwert. Auch hierfür musste eine Lösung gefunden werden.

Die Bearbeitung dieses Punktes wird in Kapitel 4 geschildert.

3.2 Erstellung einer webbasierten Spieloberfläche

Kernbestandteil dieser Arbeit war die Erstellung einer Spieloberfläche, die es erlaubt, Sensoren in einem Trinkwasserversorgungsnetz zu platzieren, und diese nach einer Evaluierung auf Basis im Hintergrund ausgeführter Pythonscripts weiter zu optimieren.

Die Erstellung und der Aufbau dieser Weboberfläche werden in Kapitel 5 erläutert.

3.3 Durchführung eines Feldversuches

Abschließend musste die erstellte Weboberfläche in einem Feldversuch von einer Testgruppe getestet und bewertet werden.

Durchführung und Ergebnisse dieses Feldversuchs sind in Kapitel 6 angeführt.

4 OSP-Codeadaptionen für Serious Gaming

In diesem Kapitel werden die notwendigen Vorbereitungen in Bezug auf benötigte Pythoncodes für die Erstellung des Serious Games erklärt. Dazu werden zuerst die verwendeten hydraulischen Modelle erläutert, bevor die Auslagerung der OSP-Berechnungen auf einen Automatisierungsserver geschildert wird.

4.1 Verwendete hydraulische Modelle

Basis für jede OSP-Berechnung ist ein Modell jenes Trinkwasserversorgungsnetzes, in dem Sensoren platziert werden sollen. Für das Serious Game wurden zwei Modelle herangezogen, das Netz von Poulakis et al. (2003) und C-Town (Ostfeld et al., 2011).

In den folgenden Abbildungen der hydraulischen Modelle sind reguläre Knoten als Punkt, Hochbehälter und Reservoirs als Quadrate (letztere um 45° gedreht), Pumpen als Fünfecke und Ventile als Dreiecke entlang der zugehörigen Kante dargestellt.

Das Netz von Poulakis besteht aus 50 Leitungen und 31 Knoten, wovon einer ein Reservoir darstellt. Die Leitungen sind rasterförmig angeordnet, wobei die horizontalen Leitungen 1000 m und die vertikalen Leitungen 2000 m lang sind. Die Leitungsdurchmesser befinden sich im Bereich von 300 bis 600 mm, die Gesamtleitungslänge beträgt ca. 73 km. In Abbildung 4-1 ist dieses Netz samt Durchflüssen und Druckhöhen dargestellt.

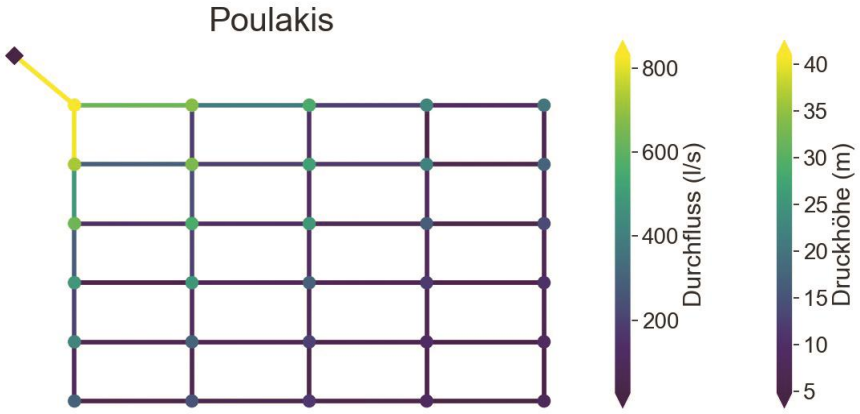

Abbildung 4-1: Netz von Poulakis mit Durchflüssen und Druckhöhen

C-Town wurde für die *Battle of Water Calibration Networks* verwendet. Hier handelt es sich um ein deutlich komplexeres Modell, bestehend aus einem Reservoir, sieben Hochbehältern, fünf Pumpstationen, einem Drossel- und einem Rückschlagventil, sowie 388 weiteren Knoten und 432 Leitungen mit einer Gesamtlänge von rund 57 km. Die Durchmesser variieren zwischen 51 mm und 610 mm. Abbildung 4-2 zeigt C-Town samt Durchflüssen und Druckhöhen.

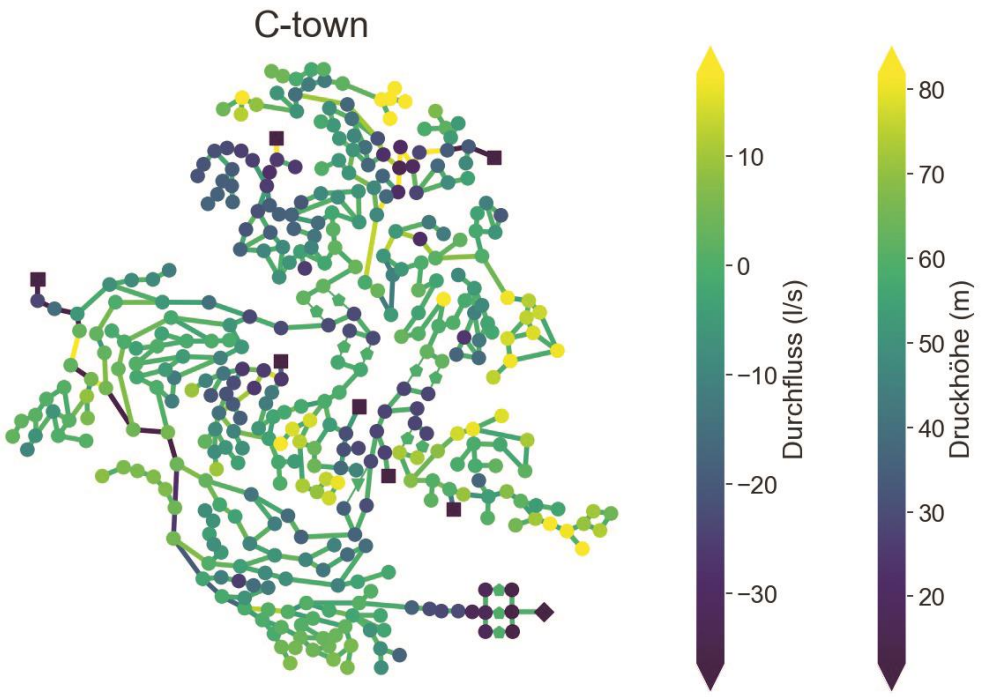

Abbildung 4-2: C-Town mit Durchflüssen und Druckhöhen

Das Netz von Poulakis dient einerseits dazu, den hier erläuterten Code einfach darzustellen. Andererseits können Spielende ein einfaches Testmodell verwenden, um sich mit der Spielmechanik und der Oberfläche vertraut zu machen. Außerdem wurde das Netz von Poulakis im Feldversuch zur Demonstration der Funktionsweise des Spiels herangezogen. Der eigentliche Spielinhalt ist hingegen auf C-Town abgestimmt.

4.2 OSP mittels Jenkins

Aufgrund der mitunter langen Rechenzeiten, die sich bei der Verwendung eines sensitivitätsbasierten OSP-Algorithmus ergeben, wurde dieser Teil der Projektbearbeitung auf einen Server am Institut für Siedlungswasserwirtschaft und Landschaftswasserbau ausgelagert.

Dazu wurde Jenkins (2018) verwendet. Bei Jenkins handelt es sich um einen (open-source) Automatisierungsserver, der es ermöglicht, Skripts über eine Weboberfläche zu bedienen. Die dazu verwendete Oberfläche ist einfach gestaltet und ermöglicht es damit auch Personen, die keine oder nur wenig Programmiererfahrung besitzen, vorab erstellte Scripts zu verwenden. Dazu können verschiedene Formen von Parametern verwendet werden, um die Funktionsweise des Scripts anzupassen. So können beispielsweise Dateien hochgeladen, Einstellungsmöglichkeiten über Dropdown-Menüs auswählt oder Eingaben über Textfelder getätigt werden. Der damit gesteuerte Code wird dabei über ein Git-Repository aktuell gehalten. Von den bedienenden Personen wird je Berechnung ein

build in einem Projekt erstellt. Die Berechnung dieses builds wird dann entweder vom Server selbst durchgeführt oder auf andere Server weiter ausgelagert.

Am Institut für Siedlungswasserwirtschaft und Landschaftswasserbau wird Jenkins bereits benutzt und es wurde auch schon ein Jenkins-Projekt für OSP-Berechnungen erstellt. Dieses musste allerdings abgewandelt werden, um Leckagen auf Leitungen simulieren zu können. Abbildung 4-3 zeigt die in Jenkins erstellte Bedienoberfläche zur OSP-Berechnung.

Um Rechenzeit zu sparen wurde ein Jenkins-Projekt erstellt, mit dem es nicht nur möglich ist, einen OSP-Algorithmus für eine bestimme Anzahl an Sensoren zu nutzen, sondern auch eine mehrmalige Berechnung der OSP-Berechnung sowie eine Veränderung der gewünschten Sensorzahl durchgeführt wird. Dadurch kann aus den erzielten Ergebnissen eine Kostenfunktion berechnet werden.

Die Skripts und die Bedienoberfläche wurden dabei so aufgebaut, dass sie universell einsetzbar sind und nicht nur die Erfordernisse des Serious Games erfüllen. Dadurch können OSP-Berechnungen mit verschiedenen Algorithmen für beliebige Netze erstellt werden.

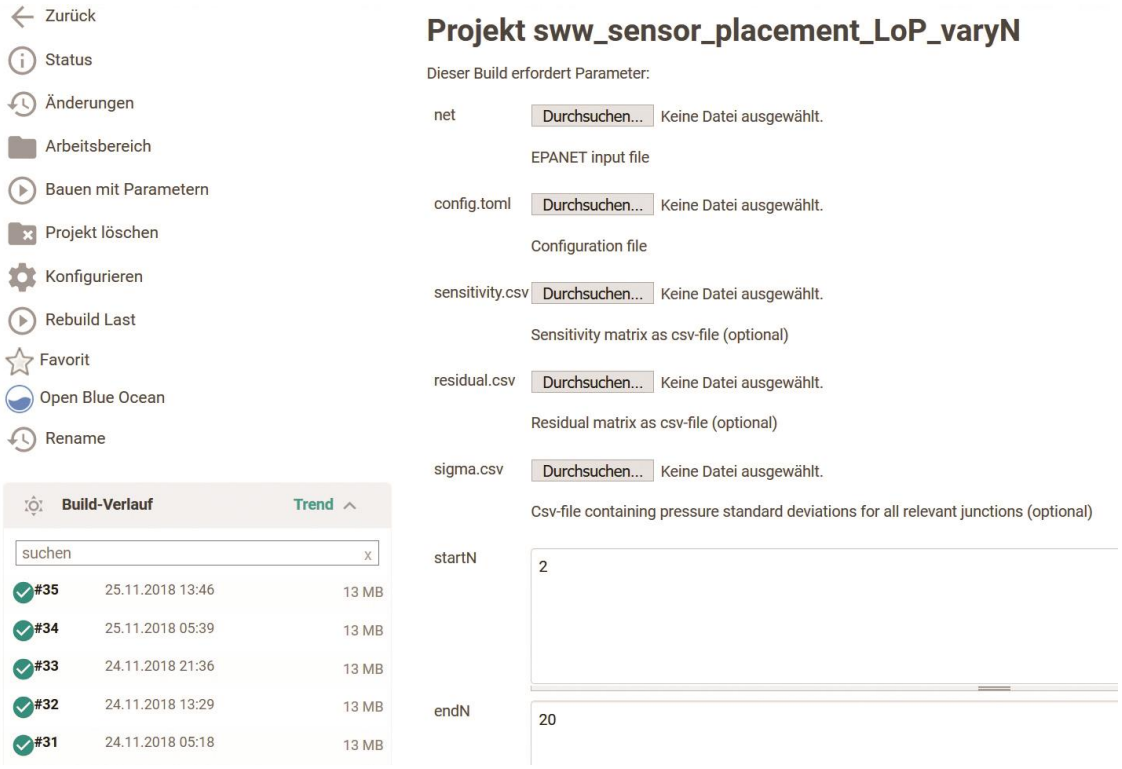

Abbildung 4-3: Bedienoberfläche für die OSP-Berechnung mit Jenkins

Im den folgenden Kapiteln werden die wählbaren Eingabeparameter, die Funktionsweise und die Ausgaben des Jenkins-Projekts erläutert.

4.2.1 Eingabeparameter

Das hier dargestellte Jenkins-Projekt unterstützt die folgenden Algorithmen:

- Shortest Path 1
- Shortest Path 2
- Casillas
- SPUDU

Je nach verwendetem Algorithmus können unterschiedliche Parameter gewählt werden. Diese Parameter werden größtenteils über eine Konfigurationsdatei (*.toml) an Jenkins übergeben. Dies ermöglicht es, ähnliche Berechnungen sehr einfach zu starten, ohne alle Einstellungen einzeln neu setzen zu müssen. Ein Beispiel für eine solche Konfigurationsdatei ist dem Anhang zu entnehmen.

Wie oben bereits erwähnt wurde, verwenden die unterschiedlichen OSP-Algorithmen verschiedene Parameter, alle teilen sich jedoch die folgenden Einstellungen:

- EPANET-Modell (*.inp)
- Sensoranzahl als Intervall
- Bereits platzierte Sensoren in Form von Knoten-IDs (optional)
- Erlaubte Sensorpositionen (optional)

Die erlaubten Sensorpositionen sind jene Knoten, auf denen der Algorithmus einen Sensor platzieren darf. Zusätzlich zu dedizierten Knoten-IDs können mittels des Jenkins-Projekts noch andere Filterkriterien angewandt werden. So können Knoten als potentielle Sensorposition vorgemerkt werden, die einen bestimmen Verbrauch, eine bestimme Beschreibung oder einen bestimmten *tag* im EPANET-Modell aufweisen. Dieser optionale Schritt ermöglicht es beispielsweise, Sensoren nur auf Knoten mit dem tag *Hydrant* zu platzieren. Wird kein Kriterium angegeben, wird jeder Knoten als mögliche Sensorposition in Betracht gezogen.

Wird ein Sensorplacement mit SPUDU oder dem Algorithmus nach Casillas bestimmt, muss zusätzlich ein Optimierer festgelegt werden. Hier stehen zwei Varianten zur Verfügung: Differential Evolution oder ein genetischer Algorithmus.

Bei beiden müssen die Populationsgröße sowie die Anzahl an Generationen, die durchlaufen werden sollen, angegeben werden. Zusätzlich müssen je eine Rekombinations- und Mutationswahrscheinlichkeit beim genetischen Algorithmus bzw. die Faktoren F und CR bei Differential Evolution gewählt werden.

Die Sensitivitäts- und Residuenmatrizen, die SPUDU und Casillas nutzen, können entweder als Datei (*.csv) hochgeladen werden, oder im Rahmen der OSP-Berechnung generiert werden. Dies ermöglicht es einerseits speziell generierte Matrizen zu verwenden als auch Matrizen zu berechnen berechnen. Die Berechnung der Matrizen lässt sich mittels mehrerer Parameter steuern:

- Eingangsparameter (beispielsweise *demand* um eine fixe Leckagegröße oder *emittercoefficient* um eine druckabhängige Leckage zu simulieren)
- Ausgabeparameter (*pressure* oder *head*)
- Leckagegröße für Evaluierung der Sensorpositionen

Alle drei Einstellungen können für die Residuen- und Sensitivitätsmatrixberechnung getrennt konfiguriert werden, dies ermöglicht es, jeweils unterschiedliche Leckagegrößen anzusetzen.

Für SPUDU kann wahlweise eine Datei mit den Standardabweichungen der Druckschwankung je Knoten hochgeladen werden (*.csv), oder diese können mittels Jenkins berechnet werden. Dazu werden Monte-Carlo-Simulationen verwendet, welche die an den Knoten angesetzten Verbräuche normalverteilt streuen lassen und anschließend die Standardabweichung der in den jeweiligen Knoten hervorgerufenen Druckabweichungen bestimmen. Dieser Vorgang wird mehrmals durchlaufen, wobei bei jeder Iteration für jeden Knoten ein neuer Verbrauch aus der Verteilung gezogen wird. Abschließend werden die Druckabweichungen über sämtliche Simulationsdurchläufe je Knoten gemittelt.

Dazu werden zwei Parameter benötigt:

- Die Anzahl an Monte-Carlo-Simulationen, die durchlaufen werden sollen.
- Die Standardabweichung σ der Normalverteilung, aus der die Verbräuche je Simulationsdurchlauf gezogen werden. Der Mittelwert μ dieser Normalverteilung ist dabei immer der ursprünglich angesetzte Verbrauch im Knoten.

Abbildung 4-4 zeigt exemplarisch Normalverteilungen für unterschiedliche Standardabweichungen bei konstantem Mittelwert.

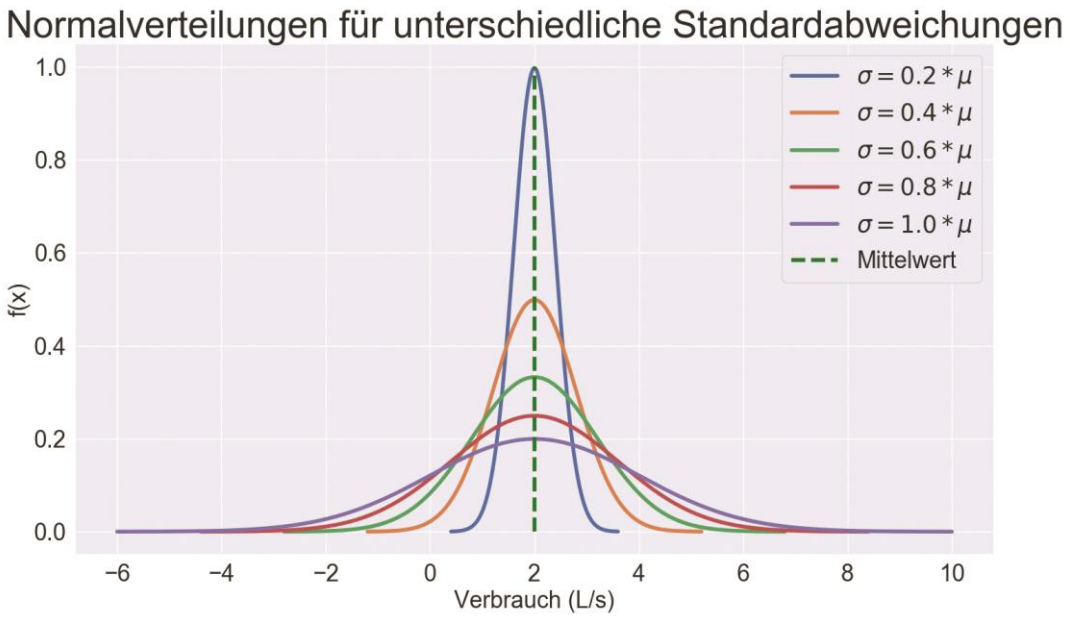

Abbildung 4-4: Normalverteilungen bei unterschiedlichen Standardabweichungen

Für die Darstellung der Ergebnisse der OSP-Berechnung können zwei Einstellungen getroffen werden, die beide die Darstellung der Knoten im Ergebnis betreffen. Im Ergebnisplot werden die Knoten je nach Sensitivität in unterschiedlichen Farben dargestellt. Dabei können die folgenden Einstellungen getroffen werden:

- Das verwendete Farbschema kann gewählt werden.
- Es kann gewählt werden, ob die Farbskala auf Basis der Maximum- und Minimumwerte gebildet wird, oder ob stattdessen die Farbskala auf Basis des zweiten und 98. Perzentils gebildet werden soll. Letzteres führt beim Vorhandensein von Extremwerten zu besseren Ergebnissen.

Abschließend kann noch gewählt werden, ob die Ergebnisse der Berechnung in einer dafür eingerichteten PostgreSQL-Datenbank abgelegt werden sollen.

4.2.2 Aufbau und Funktionsweise

Der erstellte Code weist die folgende Struktur auf:

- Main.py
- Prepare.py
- OSP.py
- Database.py

Main.py stellt dabei das Herzstück dar, von dem aus die von den Nutzern getroffenen Einstellungen eingelesen, das hydraulische Modell vorbereitet, die OSP-Berechnung gestartet, sowie die Ablage der Ergebnisse in der PostgreSQL-Datenbank gesteuert werden.

Der konkrete Ablauf lautet dabei wie folgt:

1. Einlesen der gesetzten Einstellungen
2. Vorbereiten des hydraulischen Modells
3. Hinzufügen von virtuellen Knoten in der Mitte jeder Leitung
4. Berechnung oder Einlesen der Sensitivitäts- und der Residuenmatrix
5. Ermittlung der möglichen Sensorpositionen
6. Erstellen einer Instanz der Klasse des gewählten OSP-Typs
7. Setzen der korrekten Einstellungen
8. Bei sensitivitätsbasierten Methoden: Erstellen einer Instanz des gewählten Optimierers
9. Berechnung der idealen Sensorpositionen
10. Darstellung der Ergebnisse in einem Plot
11. (Optional) Speichern der Simulationsergebnisse in einer Ergebnisdatenbank

Im Folgenden werden nun die oben angeführten Schritte näher erläutert:

4.2.2.1 Einlesen der gesetzten Einstellungen:

Jenkins startet das Skript mittels Bash und übergibt dabei mehrere Parameter:

- Jenkins-Build-Nummer
- EPANET-Modell
- Name des EPANET-Modells
- Konfigurationsdatei
- Sensoranzahl
- (optional) Sensitivitätsmatrix
- (optional) Residuenmatrix
- (optional) Standardabweichungen des Drucks je Knoten

Die Jenkins-Build-Nummer ist eine eindeutige ID des erzeugten Builds. Diese ID wird ebenso wie der Name des EPANET-Modells zur Speicherung der Ergebnisse in der Ergebnisdatenbank an das Skript übergeben.

Die übrigen Parameter wurden bereits in Kapitel 4.2.1 erläutert.

4.2.2.2 Vorbereiten des hydraulischen Modells

Die oben genannten Bash-Parameter werden vom Skript eingelesen. Das EPANET-Modell, der Name des Modells sowie die Konfigurationsdatei werden danach zur Vorbereitung des hydraulischen Modells verwendet.

Die Konfigurationsdatei wird eingelesen und abgelegt.

Das hydraulische Modell wird ebenfalls angepasst: Für den Fall, dass die Einstellungen des hydraulischen Modells eine Langzeitsimulation (*extended period simulation*) vorsehen, wird die Simulationsdauer auf 0 Stunden gesetzt.

Die Anzahl der Nachkommastellen der Ergebnisse wird auf 3 gesetzt und schlussendlich wird festgelegt, dass beim Simulieren des Modells in den Resultaten die Parameter *pressure*, *flow* und *demand* ausgegeben werden sollen.

4.2.2.3 Hinzufügen von virtuellen Knoten in der Mitte jeder Leitung

Bei den verwendeten OSP-Algorithmen werden Leckagen entweder als Verbrauch, oder als druckabhängiger Emitter simuliert. Beides kann jedoch nur in Knoten angegeben werden und nicht bei Leitungen.

Um zu ermöglichen, dass Leckagen auf Leitungen angesetzt werden können, wird in der Mitte jeder Leitung ein neuer Knoten angelegt. Dazu werden die Mittelwerte der Koordinaten der an die jeweilige Leitung angrenzenden Knoten errechnet und mittels dieser errechneten Koordinaten ein neuer Knoten zum Modell hinzugefügt.

Um diesen Knoten mit dem Modell zu verbinden, wird die bestehende Leitung gelöscht und durch zwei neue Leitungen ersetzt. Diese neuen Leitungen weisen

die Rauigkeit und den Durchmesser der vorher bestehenden Leitung auf, jedoch wird die Länge halbiert.

Die ursprünglich bestehenden Knoten werden in weiterer Folge als mögliche Sensorknoten gewertet, die neu erzeugten Knoten werden als mögliche Leckagepositionen angesehen.

Abbildung 4-5 zeigt das Netz von Poulakis, nachdem auf allen Leitungen Knoten zur Leckagesimulation platziert wurden.

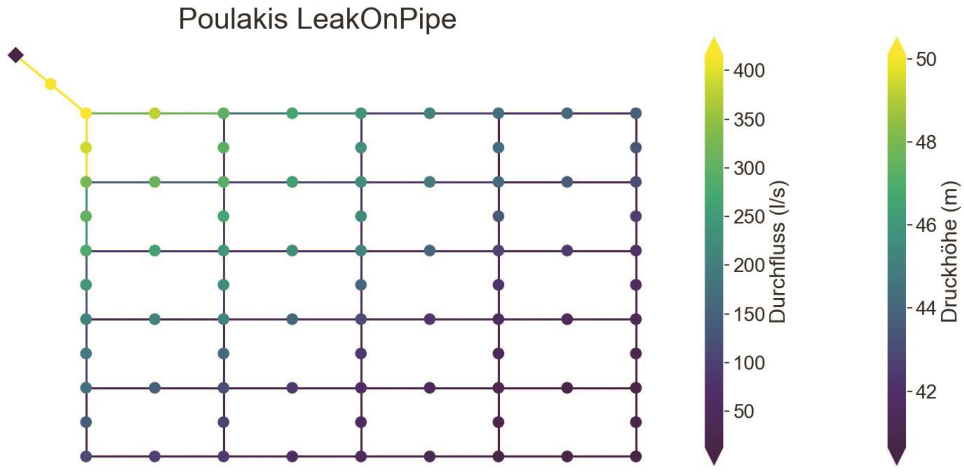

Abbildung 4-5: Netz von Poulakis mit zusätzlichen Knoten zur Leckagesimulation

4.2.2.4 Berechnung oder Einlesen der Sensitivitäts- und Residuenmatrizen

Nutzer können je eine Sensitivitäts- und Residuenmatrix als CSV-Datei über Jenkins als Parameter angeben. Übergebene Matrizen werden dann vom Skript eingelesen und für die OSP-Berechnung verwendet.

Nicht übergebene Matrizen werden vom Skript mit den in der Konfigurationsdatei angegebenen Parametern errechnet. Die so errechneten Matrizen werden anschließend so reduziert, dass in den Spalten der Matrizen nur die ursprünglich vorhandenen Knoten und in den Zeilen nur die generierten Leckageknoten vorhanden sind. Dadurch können die sensitivitätsbasierten Algorithmen einerseits Sensoren nur auf den ursprünglich im Modell vorhandenen Knoten platzieren und andererseits die Evaluierung der Sensorpositionen nur für die neuen, virtuellen Knoten durchführen.

4.2.2.5 Ermitteln der möglichen Sensorpositionen

Wurden in der Konfigurationsdatei Filterkriterien für die möglichen Sensorpositionen angegeben, werden die im Netz bereits bestehenden Knoten nach diesen Kriterien gefiltert.

Mögliche Kriterien sind dabei:

- Knoten-ID
- Knoten-Beschreibung
- Knoten-tag
- Knotenverbrauch

Wurden mehrere Kriterien gewählt, werden diese nicht kombiniert angewandt. Das bedeutet, dass ein Knoten nur eines der Kriterien erfüllen muss, um als mögliche Sensorposition gewertet zu werden.

4.2.2.6 Erstellen einer Instanz der Klasse des gewählten OSP-Typs

Nun kann eine Instanz der Klasse gewählten OSP-Typs (z.B. Casillas oder SPUDU) erstellt werden. Die Auswahl des OSP-Typs erfolgt dabei in der Konfigurationsdatei.

4.2.2.7 Setzen der korrekten Einstellungen

Die vorher ermittelten Parameter für die OSP-Berechnung können nun der OSP-Instanz zugewiesen werden. Je nach OSP-Typ variieren dabei die möglichen Einstellungen.

4.2.2.8 Bei sensitivitätsbasierten Methoden: Erstellen einer Instanz des gewählten Optimierers

Wurde der OSP-Typ Casillas oder SPUDU gewählt, wird nun eine Instanz des in der Konfigurationsdatei definierten Optimierers erstellt. Die derzeit implementierten Optimierer beschränken sich dabei auf einen genetischen Algorithmus (*GA* in der Konfigurationsdatei) und Differential Evolution (*DE*).

Dieser Instanz werden die gewählten Einstellungen zugeordnet:

- Populationsgröße
- Generationenanzahl
- Rekombinations-Wahrscheinlichkeit (GA) bzw. Faktor F (DE)
- Mutationswahrscheinlichkeit (GA) bzw. Konstante CR (DE)

Der OSP-Instanz wird wiederum die soeben erstellte Optimierer-Instanz zugeordnet.

Wurde SPUDU gewählt, werden zusätzlich noch die Druckstandardabweichungen eingelesen oder mit den gesetzten Einstellungen berechnet.

4.2.2.9 Berechnung der idealen Sensorpositionen

Da nun alle nötigen Vorschritte getätigt wurden, kann nun die OSP-Berechnung gestartet werden.

Resultat dieser Berechnung sind die optimierten Sensorpositionen, als auch die zugehörige Fitness sowie die von den Sensorpositionen abgedeckten Leitungen. Bei Shortest Path 1 und 2 wird ebenfalls die Fitness der Sensorpositionen nach Casillas berechnet, um diese vergleichbarer zu machen.

4.2.2.10 Darstellung der Ergebnisse in einem Plot

Um die Ergebnisse der OSP-Berechnung graphisch darzustellen, wird ein Plot des EPANET-Modells erzeugt.

Grundlage für diesen Plot ist allerdings nicht jenes Netz, in dem virtuelle Knoten zur Leckagesimulation erstellt wurden, sondern das ursprüngliche Netz. Es erfolgt dabei eine Zuordnung der neu erzeugten Knoten zu den Leitungen, auf denen sie platziert wurden. Wurde mittels des OSP-Algorithmus festgestellt, dass mit dem berechneten Set an Sensorpositionen eine Leckage in einem (virtuellen) Knoten gefunden werden kann, wird die ursprünglich anstelle des Knotens existierende Leitung im Plot grün dargestellt. Alle nicht von den Sensoren abgedeckten Leitungen werden rot visualisiert.

Abbildung 4-6 zeigt anhand einer Leitung schematisch den Prozess, der nötig ist, um Leckagen auf Leitungen zu simulieren und die Ergebnisse graphisch darzustellen.

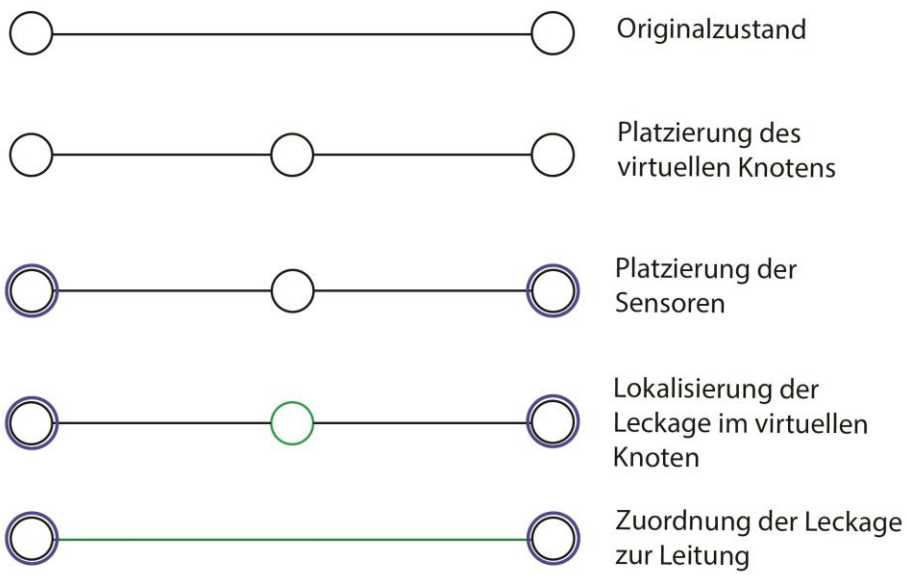

Abbildung 4-6: Schema der Leckagedetektion auf Leitungen

Außerdem wird im Ergebnisplot je Knoten die mittlere in diesem Knoten durch Leckagen der für die Berechnung der Residuenmatrix gewählten Leckagegröße hervorgerufene Druckabweichung über einer Farbskala visualisiert. Diese Druckabweichung wird analog zur mittleren Knotensensitivität (Gleichung 2-3) berechnet und im Plot mit ΔP bezeichnet:

$$\bar{r}_j = \frac{1}{m}\sum_{i=1}^{m} r_{ij} = \Delta P_j$$ Gleichung 4-1

mit:
$\bar{r}_j$ *… mittlere Druckabweichung im Knoten j infolge von Leckagen*
r_{ij} *… Druckabweichung im Knoten j infolge einer Leckage im Knoten i*
m *… Anzahl an Leckageszenarien*
ΔP_j *… Druckdifferenz im Knoten j*

Die Sensorpositionen selbst werden als rote Sechsecke im Plot dargestellt. Abbildung 4-7 zeigt exemplarisch den Ergebnisplot einer OSP-Berechnung mit dem Algorithmus nach Casillas für eine Leckagegröße von 1 L/s für 2 Sensoren im Netz von Poulakis.

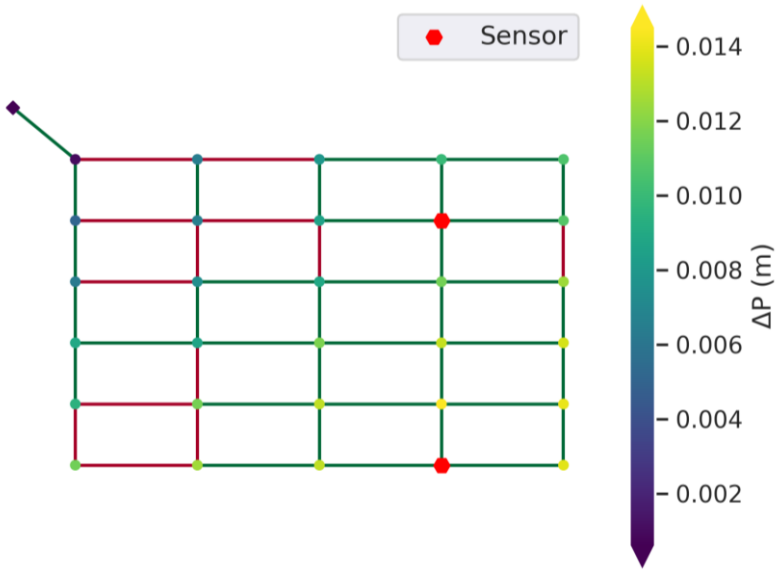

Abbildung 4-7: Sensor Placement mittels Casillas-Algorithmus mit 2 Sensoren für eine Leckagegröße von 1 L/s

4.2.2.11 (Optional) Speichern der Simulationsergebnisse in einer Ergebnisdatenbank

Sowohl die errechneten Ergebnisse, als auch die getätigten Einstellungen können in einer PostgreSQL-Datenbank abgelegt werden. Dies ist der letzte Arbeitsschritt, der vom Skript getätigt wird.

4.3 Sensitivitäts- und OSP-Berechnung

Um den Spielerinnen und Spielern in Bezug auf maximale Sensoranzahl und minimale Netzabdeckung erreichbare Ziele vorzugeben, wurden analog zu Kapitel 2.1.3 mehrere Kostenfunktionen berechnet. Im Gegensatz zu den oben dargestellten Kostenfunktionen wurden diese jedoch auch mit unterschiedlichen Leckagegrößen errechnet. Mithilfe dieser Funktionen lässt sich daraufhin ein Zusammenhang aus Leckagegröße, Sensoranzahl und Fitness herstellen, um darauf aufbauend erreichbare Ziele zu definieren.

Mittels des in Kapitel 4.2 beschriebenen Jenkins-Projekts wurden für das Netz C-Town mehrere OSP-Berechnungen mit der Zielfunktion von Casillas durchgeführt. Für die Berechnung der Sensitivitäts- und Residuenmatrizen wurden dabei fixe Leckagegrößen im Bereich zwischen 1 L/s und 100 L/s gewählt. Die Sensitivitäts- und Residuenmatrizen wurden dabei jeweils mit denselben Leckagegrößen berechnet. Die OSP-Berechnungen wurden jeweils für zwei bis 30 Sensoren durchgeführt.

Als Optimierer wurde Differential Evolution mit einem Faktor $F = 0{,}5$ und einer Konstanten $CR = 0{,}7$ gewählt. Es wurde eine Populationsgröße von 100 Individuen angesetzt, die über 100 Generationen optimiert wurden. Als Mutationsstrategie wurde DE/rand/1 (siehe Gleichung 2-15) verwendet.

Um mittels der so ermittelten Fehlerindices eine Kostenfunktion errechnen zu können, wurde eine Funktion an die generierten Datenpunkte angepasst. Gewählt wurde jene Funktion, die sich in der Literatur als am besten geeignet herausgestellt hat (siehe Kapitel 2.1.3). Diese Funktion weist die folgende Form auf:

$$f(N) = a * N^{-b} \qquad \text{Gleichung 4-2}$$

mit:
a, b … zu bestimmende Parameter
N … Sensoranzahl

In Abbildung 4-8 sind die Kostenfunktionen für sieben verschiedene Leckagegrößen für das Netz C-Town dargestellt. Es fällt auf, dass eine steigende Leckagegröße zu einer Verbesserung des Fehlerindexes führt. Ab einer Leckagegröße von 50 L/s ist allerdings keine Verbesserung mehr festzustellen und bei einer Leckagegröße von 20 L/s wird ab einer Sensoranzahl von 20 Sensoren ebenfalls keine Verbesserung des Fehlerindexes durch Steigerung der Leckagegröße mehr erzielt.

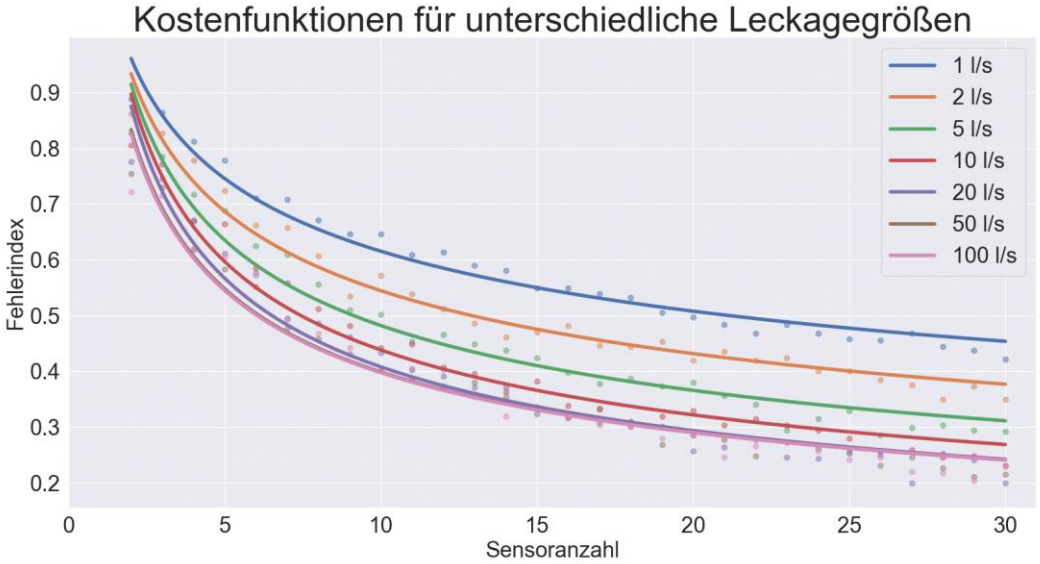

Abbildung 4-8: Kostenfunktionen für unterschiedliche Leckagegrößen

5 Optimal Sensor Placement als Serious Game

In den folgenden Kapiteln werden einerseits die Ziele, die den Spielenden gestellt werden, sowie die Funktionsweise und der Aufbau der Weboberfläche des im Rahmen dieser Masterarbeit entwickelten Serious Games dargestellt. Abschließend wird das vorher beschriebene Serious Game wie in Kapitel 2.2.3 geschildert klassifiziert.

5.1 Softwaregrundlage

Basis für die Weboberfläche ist Flask (2018), ein in Python geschriebenes Webframework mit dem WSGI-Anwendungen (Web Server Gateway Interface) einfach programmiert werden können. WSGI stellt dabei eine Schnittstelle zwischen Webservern (z.B. Apache oder Nginx) und in Python geschriebenen Anwendungen dar (PEP 333 - Python Web Server Gateway Interface v1.0, 2018).

Mittels Flask werden dabei URL-Routen erstellt, so könnte beispielsweise der Login-Bereich auf einer Webseite mittels der Route */login* (z.B. https://example.com/login) erreicht werden. Es können aber auch Routen erstellt werden, die mittels POST- oder GET-Methoden über die Webseite erreicht werden können. POST- und GET-Methoden sind für die Kommunikation zwischen Client (z.B. Browser) und Server zuständig, wobei GET-Anfragen vom Client die Anfrage konkret in der URL beinhalten, während POST-Anfragen im HTTP-Request-Body enthalten sind und demnach nicht über die URL ausgelesen werden können (Bewersdorff, 2014). So könnte beispielsweise eine Berechnung über eine Webseite über die Route */calculate* (z.B. https://example.com/calculate) mittels POST mit Eingangsparametern versehen werden, welche dann in Flask in einem Pythonskript verarbeitet werden. Die Ergebnisse dieser Berechnung können dann wiederum an die Webseite übergeben und auf dieser dargestellt werden.

Als Ergebnisdatenbank wurde PostgreSQL (PostgreSQL, 2018) gewählt, eine open-source objekt-relationale Datenbank, die SQL (Structured Query Language) verwendet und erweitert.

Zur Darstellung der Netze wurde Leaflet (Leaflet, 2018) verwendet. Bei Leaflet handelt es sich um eine open-source JavaScript Bibliothek, mit der interaktive 2D-Karten dargestellt werden können. Ursprünglich gedacht zur Darstellung realer Karten mit realen Koordinatensystemen, ist Leaflet auch in der Lage, beliebige Koordinaten vor einem beliebigen Hintergrund darzustellen.

5.2 Spielziele im Serious Game

Den Spielenden werden im Rahmen des Serious Games zwei Zielgrößen vorgegeben:

- Sensoranzahl
- Netzabdeckung

Die Sensoranzahl soll dabei einen vorgegebenen Wert nicht überschreiten, während die Netzabdeckung eine Mindestabdeckung erreichen soll. Spielziel ist eine optimale Netzabdeckung bei minimalem Einsatz von Sensoren.

Da die Fitness nach Casillas als jener Anteil an Lecks im Netz definiert ist, die mittels der platzierten Sensoren nicht korrekt lokalisiert werden kann, bedeutet dies im Umkehrschluss, dass sich eine Netzabdeckung ableiten lässt.

Die Netzabdeckung nc wird dabei wie folgt definiert:

$$nc = (1 - \bar{\varepsilon}) * 100 \qquad \text{Gleichung 5-1}$$

mit:
nc … Netzabdeckung (%)
$\bar{\varepsilon}$ … mittlerer Fehlerindex nach Casillas

Zugrunde gelegt ist der Netzabdeckung der Algorithmus nach Casillas (siehe Kapitel 2.1.2.2). Dieser Algorithmus wurde gewählt, da mittels dieser Zielfunktion eine eindeutige Einteilung in von den Sensoren abgedeckte Leitungen und nicht abgedeckte Leitungen vorgenommen werden kann.

Auch die Fitness nach SPUDU wurde angedacht. Hier wäre eine graphische Darstellung des für SPUDU relevanten Parameters, der Verbraucherunsicherheit, möglich gewesen. Es war jedoch ein Ziel, das Problem, das die Spielenden lösen müssen, möglichst einfach zu halten. Da zusätzlich bei SPUDU durch die Bestrafung mancher Knoten die erreichbare Fitness und demnach die erreichbare Netzabdeckung sinkt, wurde die sehr ähnliche aufgebaute Fitness nach Casillas als Zielgröße gewählt. In Kapitel 5.2.1 wird anhand eines Beispiels die Berechnung der Netzabdeckung im Detail erklärt.

Analog zu Abbildung 4-8 konnten auch Funktionen für die Netzabdeckung in Abhängigkeit der Sensoranzahl und der Leckagegröße bestimmt werden. Grundlage dafür war wieder die in Gleichung 4-2 dargestellte Funktion, die an die errechneten Datenpunkte angepasst wurde. Die auf diese Weise errechneten Funktionen sind in Abbildung 5-1 dargestellt.

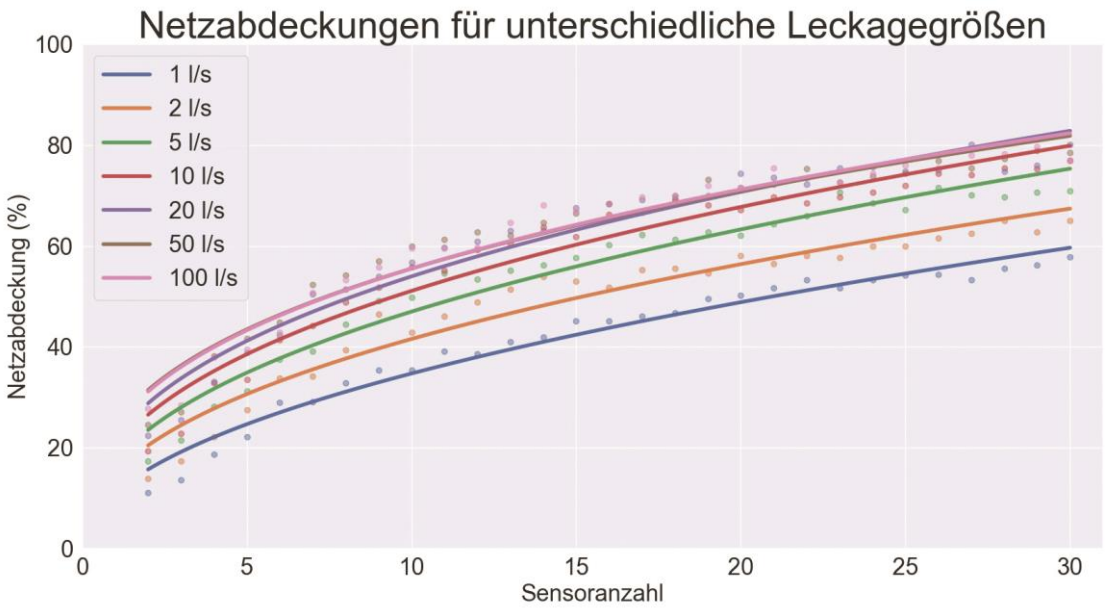

Abbildung 5-1: Netzabdeckungen für unterschiedliche Leckagegröße und Sensoranzahlen

Aufgrund des direkten Zusammenhangs zwischen Fitness und Netzabdeckung ist das Ergebnis analog zu Abbildung 4-8. Auch bei der Netzabdeckung kann ab einer Leckagegröße von 50 L/s keine Verbesserung der Netzabdeckung mehr durch eine Erhöhung der Leckagegröße erreicht werden.

Um es im Spiel zu ermöglichen, möglichst hohe Netzabdeckungen zu erreichen, ohne eine hohe Anzahl an Sensoren zu platzieren, wurde der Berechnung der Netzabdeckung und damit der Berechnung der Fitness nach Casillas eine Leckagegröße von 50 L/s zugrunde gelegt.

Für C-Town wurden basierend auf der zugehörigen Funktion der Netzabdeckung in Abhängig von der Sensoranzahl, bei einer Leckagegröße von 50 L/s, eine Mindestnetzabdeckung und eine maximale Sensoranzahl festgelegt. Beide Werte wurden so gewählt, dass die geforderte Netzabdeckung mit weniger als der maximalen Sensoranzahl erreicht werden kann und vice versa mit der maximalen Sensoranzahl eine höhere Netzabdeckung als gefordert möglich ist. Anvisiert wurde eine Netzabdeckung von 60 bis 70 %, was einer Sensoranzahl von rund 15 Sensoren entspricht. Diese Sensoranzahl wurde als maximale Sensoranzahl gewählt. Die Mindestnetzabdeckung wurde niedriger als die mittels OSP-Algorithmus erreichte Netzabdeckung mit 60 % gewählt. Zum Erreichen einer Netzabdeckung von ca. 80 % wären bereits rund 27 Sensoren nötig, was einen deutlich höheren Aufwand für die Spielenden zur Folge hätte.

Es sei an dieser Stelle angemerkt, dass aufgrund des stochastischen Verhaltens von Differential Evolution evtl. noch bessere Lösungen gefunden werden könnten. Aufgrund der langen Rechenzeiten wurde jedoch auf mehrmaliges Iterieren verzichtet.

5.2.1 Berechnung der Netzabdeckung

Hier soll beispielsweise die Berechnung der Netzabdeckung mittels der Zielfunktion von Casillas dargestellt werden. Verwendet wird dazu eine reduzierte Variante des Netzes von Poulakis bestehend aus neun Knoten, 13 Leitungen und dem ursprünglich vorhandenen Reservoir. Dies hat den Grund, dass Grundlage für diese Berechnung die Residuen- und die Sensitivitätsmatrix sind und diese bei n möglichen Sensorpositionen und m Leckageszenarien die Form $n\ x\ m$ aufweisen. Beim ursprünglichen Netz von Poulakis führt dies zu Matrizen mit 31 Spalten und 50 Zeilen, was es erschwert, diese hier darzustellen. Bei der hier verwendeten Version des Netzes von Poulakis liegen hingegen nur noch Matrizen der Form $9\ x\ 13$ vor. Dieses Netz ist in Abbildung 5-2 abgebildet. Die Leitungs-IDs in diesem Modell werden mit einem vorangestellten *P* bezeichnet, Knoten-IDs mit einem vorangestellten *J*.

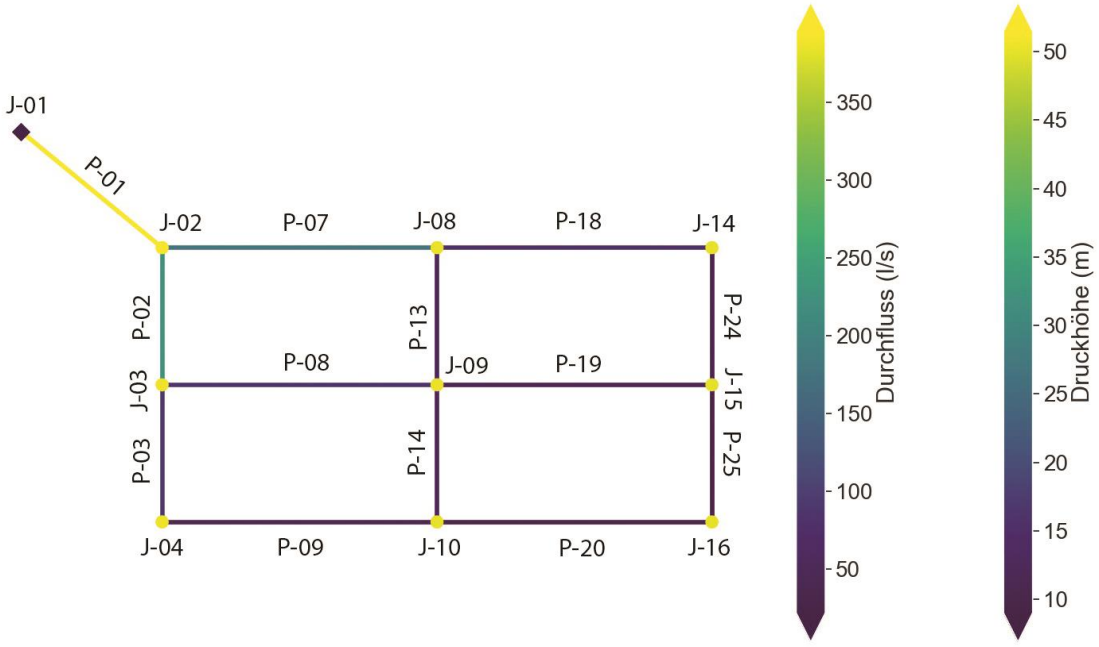

Abbildung 5-2: Darstellung der reduzierten Version des Netzes von Poulakis

Um in diesem Netz die Netzabdeckung bestimmen zu können, muss je Leitung ein virtueller Knoten zur Leckagesimulation erstellt werden. Das daraus resultierende Netz ist in Abbildung 5-3 dargestellt.

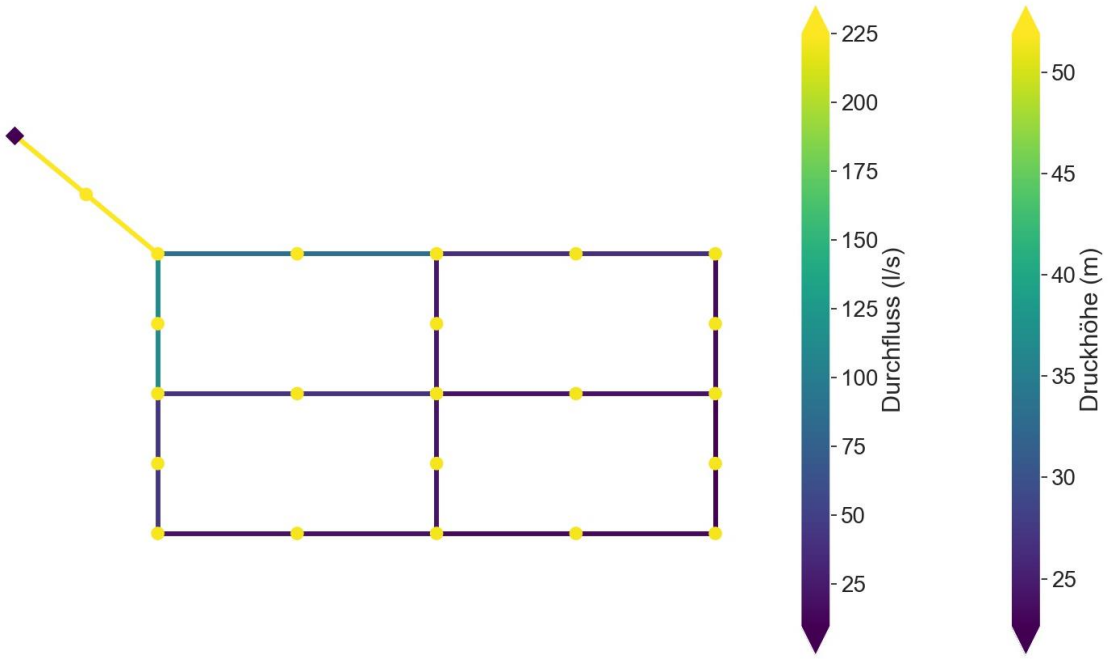

Abbildung 5-3: Darstellung der reduzierten Version des Netzes von Poulakis mit eingefügten Knoten zur Leckagesimulation

Bei einer Leckagegröße von 2 L/s ergibt sich die zugehörige Sensitivitätsmatrix S wie folgt:

$$S = \begin{array}{c} \begin{matrix} \text{J-02} & \text{J-03} & \text{J-04} & \text{J-08} & \text{J-09} & \text{J-10} & \text{J-14} & \text{J-15} & \text{J-16} \end{matrix} \\ \begin{bmatrix} 0 & -0{,}0005 & -0{,}0005 & -0{,}0005 & -0{,}0005 & 0 & -0{,}0005 & -0{,}0005 & -0{,}0005 \\ 0 & -0{,}0015 & -0{,}0010 & -0{,}0010 & -0{,}0010 & -0{,}0010 & -0{,}0010 & -0{,}0010 & -0{,}0010 \\ 0 & -0{,}0020 & -0{,}0025 & -0{,}0015 & -0{,}0015 & -0{,}0015 & -0{,}0015 & -0{,}0020 & -0{,}0020 \\ 0 & -0{,}0010 & -0{,}0010 & -0{,}0015 & -0{,}0015 & -0{,}0010 & -0{,}0015 & -0{,}0015 & -0{,}0015 \\ 0 & -0{,}0020 & -0{,}0020 & -0{,}0020 & -0{,}0020 & -0{,}0015 & -0{,}0020 & -0{,}0020 & -0{,}0020 \\ 0 & -0{,}0020 & -0{,}0025 & -0{,}0020 & -0{,}0020 & -0{,}0025 & -0{,}0020 & -0{,}0020 & -0{,}0025 \\ 0 & -0{,}0015 & -0{,}0015 & -0{,}0025 & -0{,}0020 & -0{,}0020 & -0{,}0020 & -0{,}0025 & -0{,}0025 \\ 0 & -0{,}0015 & -0{,}0020 & -0{,}0020 & -0{,}0025 & -0{,}0025 & -0{,}0020 & -0{,}0025 & -0{,}0025 \\ 0 & -0{,}0015 & -0{,}0015 & -0{,}0025 & -0{,}0020 & -0{,}0020 & -0{,}0025 & -0{,}0025 & -0{,}0025 \\ 0 & -0{,}0015 & -0{,}0015 & -0{,}0020 & -0{,}0025 & -0{,}0020 & -0{,}0025 & -0{,}0030 & -0{,}0030 \\ 0 & -0{,}0015 & -0{,}0020 & -0{,}0020 & -0{,}0020 & -0{,}0025 & -0{,}0025 & -0{,}0030 & -0{,}0035 \\ 0 & -0{,}0015 & -0{,}0015 & -0{,}0025 & -0{,}0020 & -0{,}0020 & -0{,}0030 & -0{,}0030 & -0{,}0030 \\ 0 & -0{,}0015 & -0{,}0020 & -0{,}0020 & -0{,}0020 & -0{,}0025 & -0{,}0025 & -0{,}0035 & -0{,}0035 \end{bmatrix} \end{array} \begin{matrix} \text{P-01} \\ \text{P-02} \\ \text{P-03} \\ \text{P-07} \\ \text{P-08} \\ \text{P-09} \\ \text{P-13} \\ \text{P-14} \\ \text{P-18} \\ \text{P-19} \\ \text{P-20} \\ \text{P-24} \\ \text{P-25} \end{matrix}$$

Gleichung 5-2

mit:

S … *Sensitivitätsmatrix*

Die Residuenmatrix R wird ebenfalls mit einer Leckagegröße von 2 L/s berechnet und führt zu folgenden Ergebnissen:

$$R = \begin{array}{c} \begin{matrix} \text{J-02} & \text{J-03} & \text{J-04} & \text{J-08} & \text{J-09} & \text{J-10} & \text{J-14} & \text{J-15} & \text{J-16} \end{matrix} \\ \begin{bmatrix} 0 & -0{,}0010 & -0{,}0010 & -0{,}0010 & -0{,}0010 & 0 & -0{,}0010 & -0{,}0010 & -0{,}0010 \\ 0 & -0{,}0030 & -0{,}0020 & -0{,}0020 & -0{,}0020 & -0{,}0020 & -0{,}0020 & -0{,}0020 & -0{,}0020 \\ 0 & -0{,}0040 & -0{,}0050 & -0{,}0030 & -0{,}0030 & -0{,}0030 & -0{,}0030 & -0{,}0040 & -0{,}0040 \\ 0 & -0{,}0020 & -0{,}0020 & -0{,}0030 & -0{,}0030 & -0{,}0020 & -0{,}0030 & -0{,}0030 & -0{,}0030 \\ 0 & -0{,}0040 & -0{,}0040 & -0{,}0010 & -0{,}0040 & -0{,}0030 & -0{,}0040 & -0{,}0040 & -0{,}0040 \\ 0 & -0{,}0040 & -0{,}0050 & -0{,}0040 & -0{,}0040 & -0{,}0050 & -0{,}0040 & -0{,}0040 & -0{,}0050 \\ 0 & -0{,}0030 & -0{,}0030 & -0{,}0050 & -0{,}0040 & -0{,}0040 & -0{,}0040 & -0{,}0050 & -0{,}0050 \\ 0 & -0{,}0030 & -0{,}0040 & -0{,}0040 & -0{,}0050 & -0{,}0050 & -0{,}0040 & -0{,}0050 & -0{,}0050 \\ 0 & -0{,}0030 & -0{,}0030 & -0{,}0050 & -0{,}0040 & -0{,}0040 & -0{,}0050 & -0{,}0050 & -0{,}0050 \\ 0 & -0{,}0030 & -0{,}0030 & -0{,}0040 & -0{,}0050 & -0{,}0040 & -0{,}0050 & -0{,}0060 & -0{,}0060 \\ 0 & -0{,}0030 & -0{,}0040 & -0{,}0040 & -0{,}0040 & -0{,}0050 & -0{,}0050 & -0{,}0060 & -0{,}0070 \\ 0 & -0{,}0030 & -0{,}0030 & -0{,}0050 & -0{,}0040 & -0{,}0040 & -0{,}0060 & -0{,}0060 & -0{,}0060 \\ 0 & -0{,}0030 & -0{,}0040 & -0{,}0040 & -0{,}0040 & -0{,}0050 & -0{,}0050 & -0{,}0070 & -0{,}0070 \end{bmatrix} \end{array} \begin{matrix} \text{P-01} \\ \text{P-02} \\ \text{P-03} \\ \text{P-07} \\ \text{P-08} \\ \text{P-09} \\ \text{P-13} \\ \text{P-14} \\ \text{P-18} \\ \text{P-19} \\ \text{P-20} \\ \text{P-24} \\ \text{P-25} \end{matrix}$$

Gleichung 5-3

mit:

R … *Residuenmatrix*

Es soll nun beispielhaft die Netzabdeckung berechnet werden, gesetzt für den Fall, dass auf den Knoten J-04 und J-09 ein Sensor platziert wird. Abbildung 5-4 zeigt die Sensorpositionen im reduzierten Netz von Poulakis.

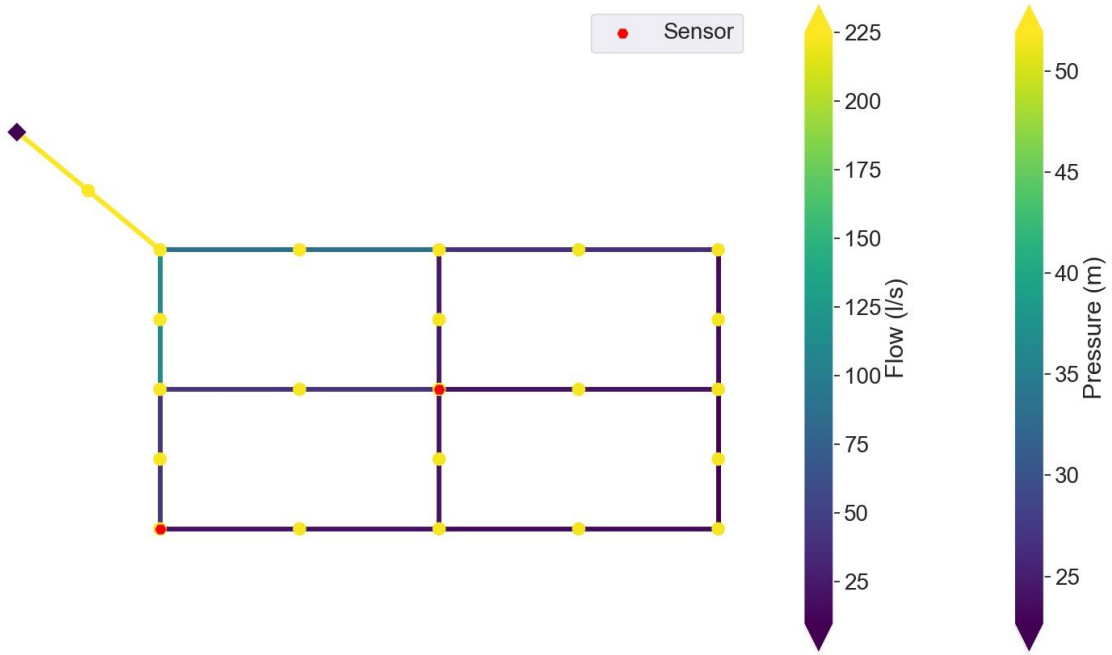

Abbildung 5-4: Darstellung der reduzierten Version des Netzes von Poulakis mit Sensoren auf den Knoten J-04 und J-09

Der dazugehörige Vektor q lautet demnach wie folgt:

$$q = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \text{J-02} \\ \text{J-03} \\ \text{J-04} \\ \text{J-08} \\ \text{J-09} \\ \text{J-10} \\ \text{J-14} \\ \text{J-15} \\ \text{J-16} \end{matrix}$$

Gleichung 5-4

Daraus ergibt sich die Matrix $Q(q)$:

$$Q(q) = \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{J-02} & \text{J-03} & \text{J-04} & \text{J-08} & \text{J-09} & \text{J-10} & \text{J-14} & \text{J-15} & \text{J-16} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \text{J-02} \\ \text{J-03} \\ \text{J-04} \\ \text{J-08} \\ \text{J-09} \\ \text{J-10} \\ \text{J-14} \\ \text{J-15} \\ \text{J-16} \end{matrix}$$

Gleichung 5-5

Nun wird die Projektionsmatrix ψ_{ij} für die gewählten Sensorpositionen berechnet. Dazu werden die Projektionen zwischen je einer Zeile der Sensitivitäts- und der Residuenmatrix bestimmt. Die Berechnung dieser Matrix aus Gleichung 2-8 wird hier in einer leicht modifizierten Version angewandt, da in der oben angeführten Gleichung der Zusammenhang $Q(q)^T Q(q) = Q(q)$ bereits integriert war (Casillas et al., 2013). Zum besseren Verständnis wird hier die vollständige Gleichung dargestellt:

$$\psi_{ij} = \frac{r_i^T Q(q)^T Q(q) s_j}{\left| r_i^T Q(q)^T \right| \left| Q(q) s_j \right|}$$

Gleichung 5-6

mit:
ψ_{ij} *… Projektion des Residuenvektors i auf den Sensitivitätsvektor j*
r_i^T *… transponierter Residuenvektor i*
s_j *… Sensitivitätsvektor j*

Betrachtet man die Projektion des Sensitivitätsvektors $s_{\mathrm{P}-18}$ auf den Residuenvektor r_{P-13}, ergibt sich $r_{P-13}^T Q(q)^T$ wie folgt:

$$r_{P-13}^{T} Q(q)^{T} = \begin{bmatrix} 0 \\ -0{,}0030 \\ -0{,}0030 \\ -0{,}0050 \\ -0{,}0010 \\ -0{,}0040 \\ -0{,}0040 \\ -0{,}0050 \\ -0{,}0050 \end{bmatrix}^{T} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -0{,}0030 \\ 0 \\ -0{,}0040 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^{T}$$

Gleichung 5-7

mit:

r_{P-13} … *Residuenvektor des Leckageszenarios P-13*

$s_{P-18}Q(q)$ ergibt sich für diesen Fall zu:

$$S_{P-18} * Q(q) = \begin{bmatrix} 0 \\ -0{,}0015 \\ -0{,}0015 \\ -0{,}0025 \\ -0{,}0020 \\ -0{,}0020 \\ -0{,}0025 \\ -0{,}0025 \\ -0{,}0025 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -0{,}0015 \\ 0 \\ -0{,}0020 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Gleichung 5-8

mit:

s_{P-18} … *Sensitivitätsvektor des Leckageszenarios P-18*

Die Projektionen der Vektoren aufeinander ergeben:

$$\psi_{P-13,P-18} = \frac{r_{P-13}^{T} Q(q)^{T} Q(q) s_{P-18}}{|r_{P-13}^{T} Q(q)^{T}||Q(q) s_{P-18}|}$$

Gleichung 5-9

mit:

$\psi_{P-13,P-18}$ … *Projektion des Residuenvektors P-13 auf den Sensitivitätsvektor P-18*

r_{P-13}^{T} … *transponierter Residuenvektor P-13*

s_{P-18} … *Sensitivitätsvektor P-18*

Der Zähler dieses Bruches für sich allein ergibt

$$r_{P-13}^{T} Q(q)^{T} Q(q) s_{P-18} = \begin{bmatrix} 0 \\ 0 \\ -0{,}0030 \\ 0 \\ -0{,}0040 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^{T} \begin{bmatrix} 0 \\ 0 \\ -0{,}0015 \\ 0 \\ -0{,}0020 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} =$$

Gleichung 5-10

$$= -0{,}0030 * (-0{,}0015) + (-0{,}0040) * (-0{,}0020)$$
$$= 0{,}0000125$$

mit:
r_{P-13}^T … *transponierter Residuenvektor P-13*
s_{P-18} … *Sensitivitätsvektor P-18*

Der Nenner hingegen berechnet sich wie folgt:

$$|r_{P-13}^T Q(q)^T||Q(q)s_{P-18}| = \sqrt{(-0{,}0030)^2 + (-0{,}0040)^2} * \sqrt{(-0{,}0015)^2 + (-0{,}0020)^2} = 0{,}0000125 \quad \text{Gleichung 5-11}$$

mit:
r_{P-13}^T … *transponierter Residuenvektor P-13*
s_{P-18} … *Sensitivitätsvektor P-18*

Zusammengesetzt führt diese Projektion zu folgender Gleichung:

$$\psi_{P-13,P-18}(q) = \frac{r_{P-13}^T Q(q)^T Q(q) s_{P-18}}{|r_{P-13}^T Q(q)^T||Q(q)s_{P-18}|} = \frac{0{,}0000125}{0{,}0000125} = 1 \quad \text{Gleichung 5-12}$$

mit:
$\psi_{P-13,P-18}$ … *Projektion des Residuenvektors P-13 auf den Sensitivitätsvektor P-18*
r_{P-13}^T … *transponierter Residuenvektor P-13*
s_{P-18} … *Sensitivitätsvektor P-18*

Die Berechnung der vollständigen Projektionsmatrix ergibt Folgendes:

$$\psi(q) = \begin{array}{c} \begin{matrix} \text{P-01} & \text{P-02} & \text{P-03} & \text{P-07} & \text{P-08} & \text{P-09} & \text{P-13} & \text{P-14} & \text{P-18} & \text{P-19} & \text{P-20} & \text{P-24} & \text{P-25} \end{matrix} \\ \begin{bmatrix} 0,\dot{9} & 0,\dot{9} & 0{,}970 & 0{,}981 & 0,\dot{9} & 0{,}994 & 0{,}990 & 0{,}994 & 0{,}990 & 0{,}970 & 0,\dot{9} & 0{,}990 & 0,\dot{9} \\ 0,\dot{9} & 1 & 0{,}970 & 0{,}981 & 0,\dot{9} & 0{,}994 & 0{,}990 & 0{,}994 & 0{,}990 & 0{,}970 & 0,\dot{9} & 0{,}990 & 0,\dot{9} \\ 0{,}970 & 0{,}970 & 1 & 0{,}904 & 0{,}970 & 0{,}991 & 0{,}926 & 0{,}937 & 0{,}926 & 0{,}882 & 0{,}970 & 0{,}926 & 0{,}970 \\ 0{,}981 & 0{,}981 & 0{,}904 & 1 & 0{,}981 & 0{,}953 & 0{,}998 & 0{,}996 & 0{,}998 & 0{,}999 & 0{,}981 & 0{,}998 & 0{,}981 \\ 0,\dot{9} & 10,\dot{9} & 0{,}970 & 0{,}981 & 1 & 0{,}994 & 0{,}990 & 0{,}994 & 0{,}990 & 0{,}970 & 1 & 0{,}990 & 1 \\ 0{,}994 & 0{,}994 & 0{,}991 & 0{,}953 & 0{,}994 & 0,\dot{9} & 0{,}968 & 0{,}976 & 0{,}968 & 0{,}937 & 0{,}994 & 0{,}96 & 0{,}994 \\ 0{,}990 & 0{,}990 & 0{,}926 & 0{,}998 & 0{,}990 & 0{,}968 & 1 & 0{,}999 & 1 & 0{,}995 & 0{,}990 & 1 & 0{,}990 \\ 0{,}994 & 0{,}994 & 0{,}937 & 0{,}996 & 0{,}994 & 0{,}976 & 0{,}999 & 1 & 0{,}999 & 0{,}991 & 0{,}94 & 0{,}999 & 0{,}994 \\ 0{,}990 & 0{,}990 & 0{,}926 & 0{,}998 & 0{,}990 & 0{,}968 & 1 & 0{,}999 & 1 & 0{,}995 & 0{,}990 & 1 & 0{,}990 \\ 0{,}970 & 0{,}970 & 0{,}882 & 0{,}999 & 0{,}970 & 0{,}937 & 0{,}995 & 0{,}991 & 0{,}995 & 1 & 0{,}970 & 0{,}995 & 0{,}970 \\ 0,\dot{9} & 0,\dot{9} & 0{,}970 & 0{,}981 & 1 & 0{,}994 & 0{,}990 & 0{,}994 & 0{,}990 & 0{,}970 & 1 & 0{,}990 & 1 \\ 0{,}990 & 0{,}990 & 0{,}926 & 0{,}998 & 0{,}990 & 0{,}968 & 1 & 0{,}999 & 1 & 0{,}995 & 0{,}990 & 1 & 0{,}990 \\ 0,\dot{9} & 0,\dot{9} & 0{,}970 & 0{,}981 & 1 & 0{,}994 & 0{,}990 & 0{,}994 & 0{,}990 & 0{,}970 & 1 & 0{,}990 & 1 \end{bmatrix} \end{array} \begin{matrix} \text{P-01} \\ \text{P-02} \\ \text{P-03} \\ \text{P-07} \\ \text{P-08} \\ \text{P-09} \\ \text{P-13} \\ \text{P-14} \\ \text{P-18} \\ \text{P-19} \\ \text{P-20} \\ \text{P-24} \\ \text{P-25} \end{matrix}$$

Gleichung 5-13

mit:
$\psi(q)$ … *Projektionsmatrix*

In dieser Matrix wird nun Zeile für Zeile überprüft, ob das Maximum der Zeile auf der Hauptdiagonale liegt. Liegt das Maximum an dieser Stelle, ist die Übereinstimmung der Residuen- und Sensitivitätsvektoren, die beide dasselbe Leckageszenario beschreiben, am höchsten, d.h. die Leckage wird auf der richtigen Leitung lokalisiert. Liegt das Maximum jedoch auf einem anderen Knoten, bedeutet dies, dass die Leckage auf einer falschen Leitung detektiert wird.

Somit errechnet sich je Zeile ein Wert ε_i, der bei korrekt lokalisierter Leckage 0 und ansonsten 1 beträgt.

Betrachtet man die erste Zeile der Projektionsmatrix (Leckage auf Leitung P-01), liegt das Maximum der Zeile zwar auf der Hauptdiagonale, allerdings kommt das

Maximum noch weitere Male in dieser Zeile vor, d.h. die Leckage kann nicht eindeutig auf der richtigen Leitung lokalisiert werden, ε_{P-01} ist demnach 1.

In der dritten Zeile (Leckage auf Leitung P-03) beispielsweise liegt das Maximum auf der Hauptdiagonale und alle anderen Werte der Zeile sind niedriger, demnach ist $\varepsilon_{P-03} = 0$.

Der gesamte Vektor ε ergibt sich demnach wie folgt:

$$\varepsilon(q) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} \text{P-01} \\ \text{P-02} \\ \text{P-03} \\ \text{P-07} \\ \text{P-08} \\ \text{P-09} \\ \text{P-13} \\ \text{P-14} \\ \text{P-18} \\ \text{P-19} \\ \text{P-20} \\ \text{P-24} \\ \text{P-25} \end{matrix} \qquad \text{Gleichung 5-14}$$

mit:
$\varepsilon(q)$ *… Fehlerindex nach Casillas*

Der mittlere Fehlerindex $\bar{\varepsilon}(q)$ ist somit:

$$\bar{\varepsilon}(q) = \frac{1+0+0+0+1+0+1+0+1+0+1+1+1}{13} = 0{,}538 \qquad \text{Gleichung 5-15}$$

mit:
$\bar{\varepsilon}(q)$ *… mittlerer Fehlerindex nach Casillas*

Die Netzabdeckung ergibt sich schlussendlich zu:

$$nc(q) = (1 - 0{,}538) * 100 = 46{,}2\ \% \qquad \text{Gleichung 5-16}$$

mit:
nc *… Netzabdeckung*

Mithilfe des Vektors ε kann auch die Abdeckung des Netzes durch die Sensoren dargestellt werden, da jeder Wert einer Leitung zugeordnet werden kann. In Abbildung 5-5 ist dies graphisch dargestellt. Abgedeckte Leitungen sind grün visualisiert, während nicht abgedeckte Leitungen in Rot dargestellt werden.

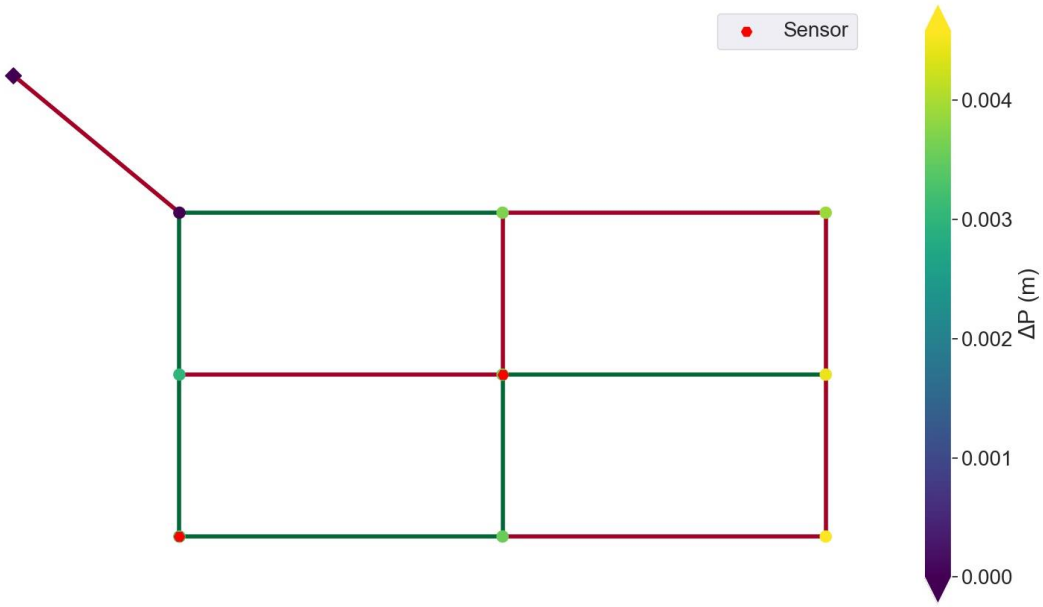

Abbildung 5-5: Visualisierte Netzabdeckung im reduzierten Netz von Poulakis

5.3 Vorbereitung der hydraulischen Modelle

Um ein hydraulisches Modell mittels Leaflet darstellen zu können, müssen die EPANET-Modelle in GeoJSON-Dateien konvertiert werden. Ein hydraulisches Modell kann als Graph, bestehend aus Knoten (z.B. Hochbehälter, Knotenpunkte) und Kanten (z.B. Leitungen, Pumpen), aufgefasst werden. Basierend auf dieser Unterscheidung werden für jedes hydraulische Modell je eine GeoJSON-Datei für Kanten und Knoten erstellt.

Die GeoJSON-Datei für Knoten enthält je Knoten die folgenden Informationen:

- Knoten-ID
- X- und Y-Koordinate
- Knotentyp
- Knotensensitivität

Die Knoten-ID entspricht der ID des Knotens im EPANET-Modell.

X- und Y-Koordinaten werden mittels eines Zoomfaktors an die Darstellung in Leaflet angepasst, aber in kein anderes Koordinatensystem konvertiert.

Der Knotentyp zeigt, ob es sich bei einem Knoten um ein Reservoir, einen Hochbehälter oder einen regulären Knoten handelt.

Die Sensitivität je Knoten wird über die mit einer Leckagegröße von 50 L/s errechneten Sensitivitätsmatrix ermittelt (siehe Gleichung 2-3).

Die Knoten werden in der Weboberfläche basierend auf ihrer Sensitivität mittels einer Farbskala in unterschiedlichen Farben dargestellt. Die dazu verwendete

Farbskala basiert auf der maximalen und der minimalen auftretenden Knotensensitivität. Das bedeutet, dass die Sensitivitäten relativ zueinander dargestellt werden.

Zur einfacheren Handhabung in der Weboberfläche wurden die errechneten mittleren Sensitivitäten $\bar{s}_j$ auf Werte zwischen 0 und 1 normiert, wobei 0 einer geringen Sensitivität und 1 einer hohen Sensitivität entspricht. Die Normierung ermöglicht es, in der Weboberfläche beim Darstellen eines Knotens basierend auf der in der GeoJSON-Datei angegebenen Sensitivität sofort eine Farbe aus der Farbskala zu wählen, ohne dass die Sensitivitäten aller Knoten bekannt sind. Dies spart Rechenzeit bei der Darstellung des Netzwerks, da ansonsten beim Aufbau des Netzwerks in der Weboberfläche erst das Intervall, auf dem die Farbskala basiert, bestimmt werden müsste. So ist dieses Intervall bereits vorgegeben.

Für die Normierung wurden zwei verschiedene Varianten versucht:

1. Normierung der mittleren Sensitivitäten
2. Normierung der logarithmierten mittleren Sensitivitäten

Bei Variante 1 werden die mittleren Sensitivitäten wie folgt normiert:

$$\hat{s}_j = \left| \frac{\bar{s}_j - s_{min}}{s_{max} - s_{min}} \right| \qquad \text{Gleichung 5-17}$$

mit:

$\hat{s}_j$ … *normierte mittlere Sensitivität des Knoten j*

$\bar{s}_j$ … *mittlere Sensitivität des Knoten j*

s_{max} … *Maximum der mittleren Sensitivität aller Knoten*

s_{min} … *Minimum der mittleren Sensitivität aller Knoten*

Bei Variante 2 werden die normierten Sensitivitäten wie folgt bestimmt:

$$\hat{s}_j = \frac{\ln|\bar{s}_j| - \ln|s_{min}|}{\ln|s_{max}| - \ln|s_{min}|} \qquad \text{Gleichung 5-18}$$

mit:

$\hat{s}_j$ … *normierte mittlere Sensitivität des Knoten j*

$\bar{s}_j$ … *mittlere Sensitivität des Knoten j*

s_{max} … *Maximum der mittleren Sensitivität aller Knoten*

s_{min} … *Minimum der mittleren Sensitivität aller Knoten*

Abbildung 5-6 zeigt ein Histogramm der Knotensensitivitäten für C-Town nach der Normierung mit Variante 1, Abbildung 5-7 zeigt ein Histogramm der mit Variante 2 ermittelten Knotensensitivitäten.

Aus diesen Abbildungen wird ersichtlich, dass bei einer regulären Normierung der Großteil der Knoten eine Sensitivität < 0,5 aufweist und nur sehr wenige Knoten eine Sensitivität > 0,8. Werden die Sensitivitäten jedoch vorher logarithmiert, werden deutlich mehr Knoten als sensitiv eingestuft.

Abbildung 5-8 zeigt die Knotensensitivitäten in C-Town basierend auf Variante 1, während in Abbildung 5-9 die Knotensensitivitäten beruhend auf Variante 2 dargestellt sind. Wie bereits in den Histogrammen ersichtlich ist, sind für die Nutzer mehr sensitive Knoten zu erkennen, wenn logarithmierte, normierte Sensitivitäten herangezogen werden. Da den Spielenden nicht der Eindruck vermittelt werden sollte, dass es im Netz nur wenige Knoten gibt, die sensitiv für Druckänderungen sind, und der Großteil des Netzes damit für eine Sensorplatzierung weniger geeignet ist, wurde für die Umsetzung der Weboberfläche für die Darstellung der Sensitivitäten die Variante 2 gewählt.

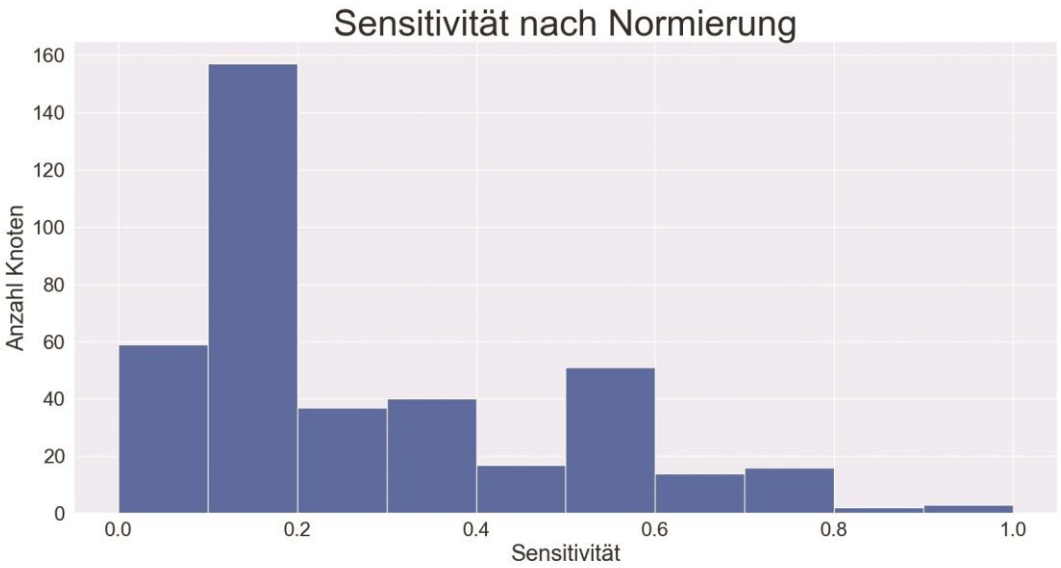

Abbildung 5-6: Histogramm der Knotensensitivitäten für C-Town nach der Normierung

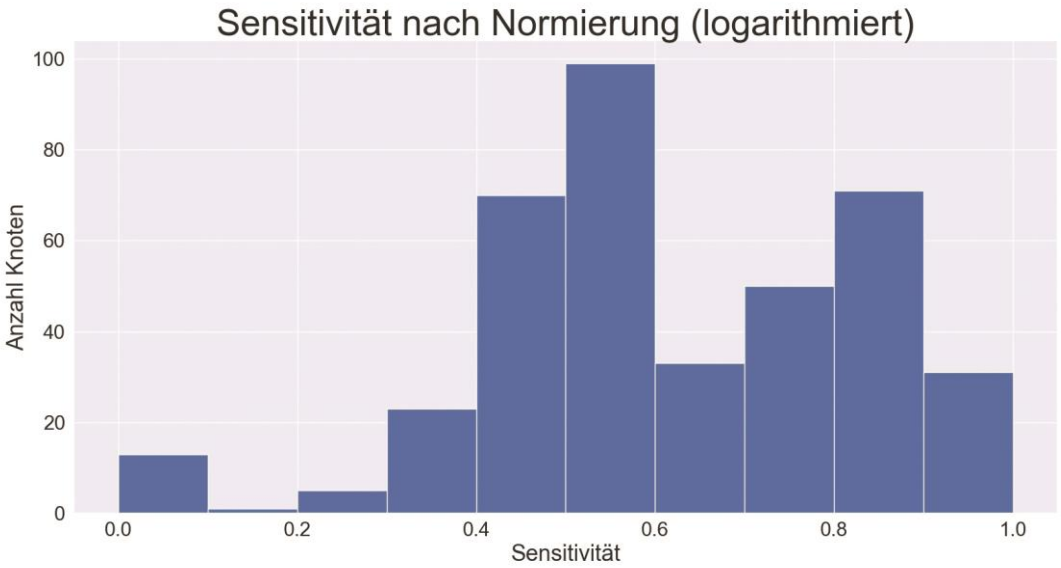

Abbildung 5-7: Histogramm der Knotensensitivitäten für C-Town nach der Normierung (logarithmiert)

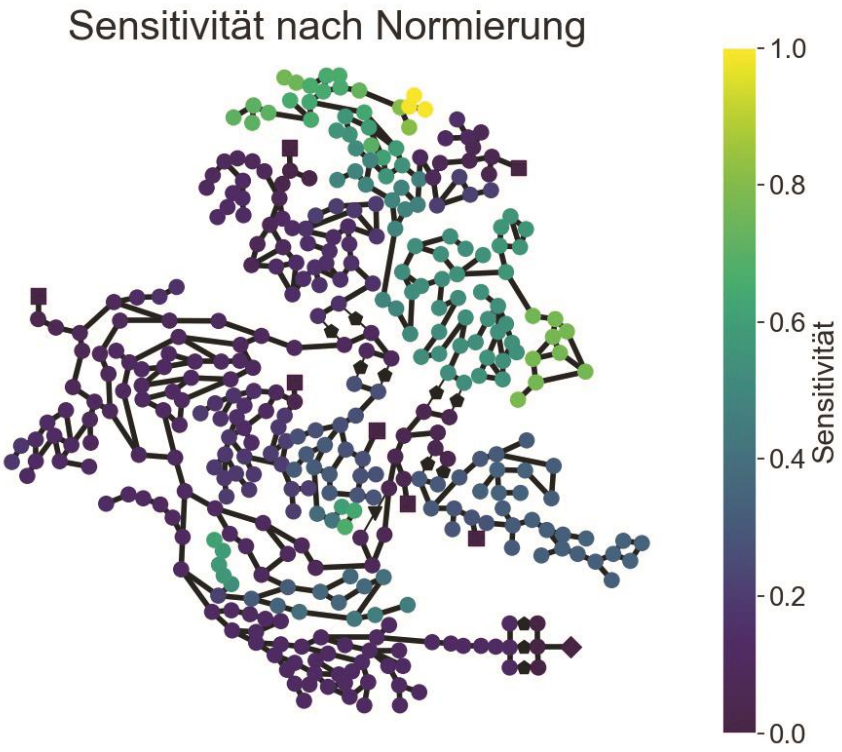

Abbildung 5-8: Knotensensitivitäten für C-Town nach der Normierung

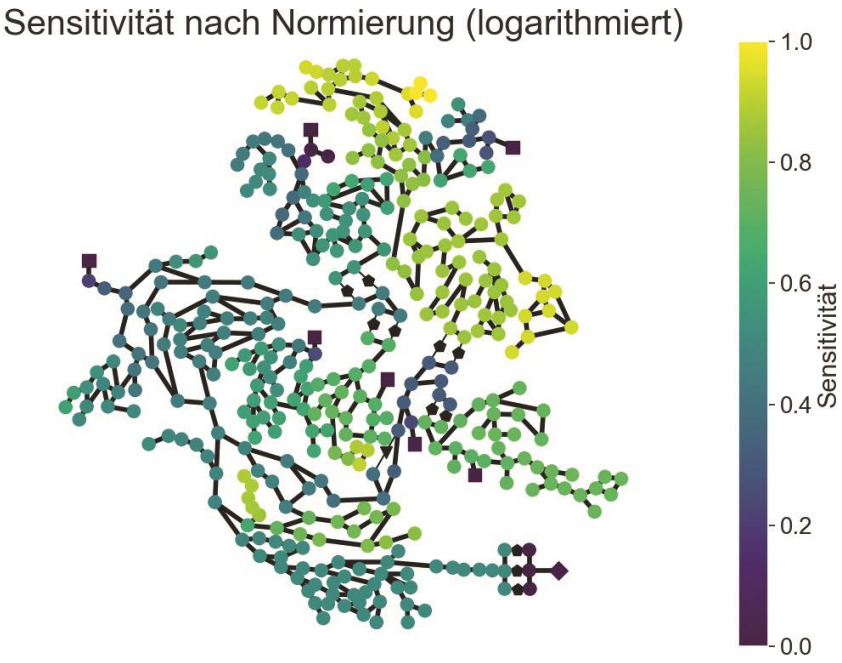

Abbildung 5-9: Knotensensitivitäten für C-Town nach der Normierung (logarithmiert)

Die für die Kanten erstellte GeoJSON-Datei enthält je Kante die folgenden Informationen:

- Kanten-ID
- Kantentyp (EPANET-Bezeichnung)
- Kantentyp (Langform)
- X- und Y-Koordinaten der Kantenenden
- Status
- Durchmesser (bei Rohren)

Wie schon bei den Knoten ist die Kanten-ID dem EPANET-Modell entnommen. In EPANET werden verschiedene Kantentypen unterschieden (Rossman, 2000):

Tabelle 5-1: EPANET-Bezeichnungen für Kanten in hydraulischen Modellen

EPANET-Bezeichnung	**Englische Langform**	**Deutsche Entsprechung**
Pipe	Pipe	Rohr
Pump	Pump	Pumpe
PRV	Pressure Reducing Valve	Druckreduzierventil
PSV	Pressure Sustaining Valve	Druckhalteventil
PBV	Pressure Breaker Valve	Druckeinzelverlust
FCV	Flow Control Valve	Drosselventil
TCV	Throttle Control Valve	Drosselsteuerungsventil
GPV	General Purpose Valve	Mehrzweckventil

Diese unterschiedlichen Kantentypen sollten in der Weboberfläche auch unterschiedlich dargestellt werden. Außerdem sollte in der Weboberfläche die Langform des Kantentyps angezeigt werden. Die Zuordnung der Langform zur EPANET-Bezeichnung hätte auch in der Weboberfläche geschehen können, allerdings wurde dieser Schritt schon vorab erledigt, um Rechenzeit bei der Darstellung der Weboberfläche einzusparen.

Die Koordinaten der Kantenenden (i.e. die Koordinaten der Kantenanfangs- und Endpunkte) werden ebenfalls mittels des bereits bei den Knoten verwendeten Zoomfaktors skaliert.

Kanten können in EPANET unterschiedliche Status aufweisen:

- Open: geöffnet
- Closed: geschlossen
- CV: Rückschlagventil (nur bei Rohren)

Geöffnete und geschlossene Kanten sollten in der Weboberfläche ebenfalls unterschiedlich ausgewiesen werden. Auf eine eigene Visualisierung eines Rohrs

mit einem Rückschlagventil wurde bisher verzichtet, da noch keine funktionierende Möglichkeit zur Darstellung der Fließrichtung mittels Leaflet gefunden werden konnte. Sie werden demnach als reguläre Rohre angezeigt.

Als letztes Attribut wird in der GeoJSON-Datei bei Rohren noch der jeweilige Durchmesser angegeben, um Rohre in der Weboberfläche proportional zum Durchmesser darstellen zu können.

5.4 Bedienelemente und Layout

Gestartet wird das Spiel über eine Startseite, auf der die Spielenden derzeit die Auswahl zwischen C-Town und dem Netz von Poulakis haben. Abbildung 5-10 zeigt besagte Startseite.

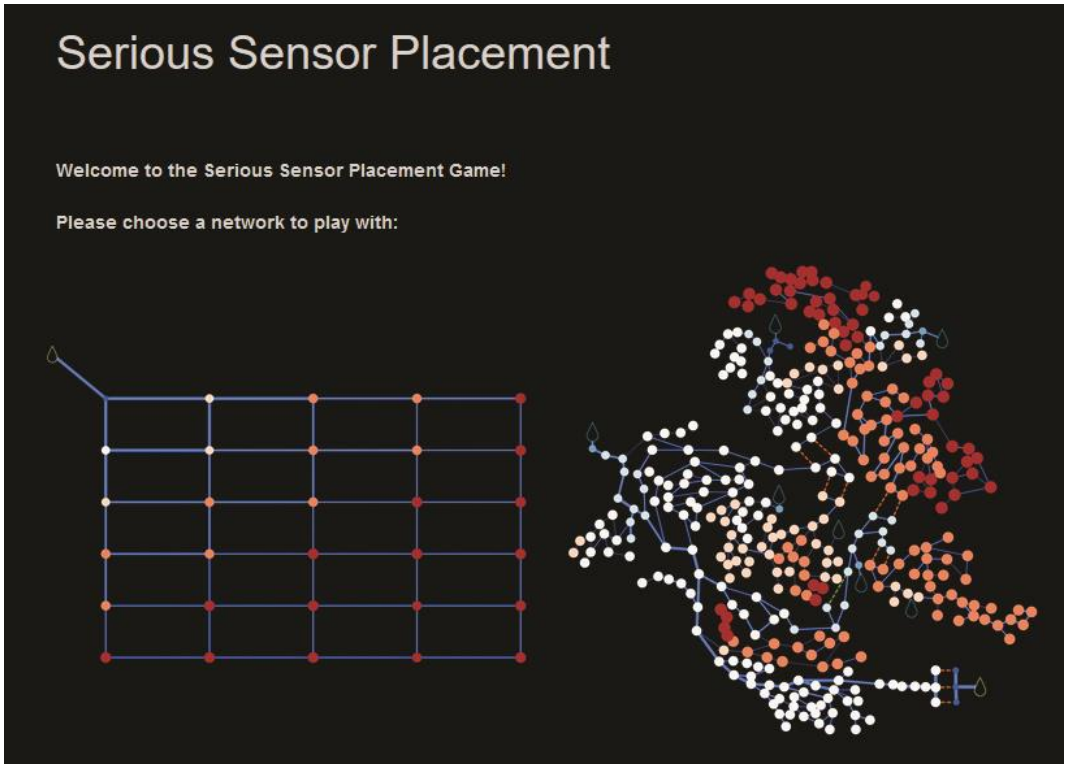

Abbildung 5-10: Startseite von Serious Sensor Placement

In Abbildung 5-11 ist die Weboberfläche mit dem Netz von Poulakis mit zwei platzierten Sensoren abgebildet. Anhand dieses Beispiels sollen die Bedienelemente sowie das Layout des Serious Games erläutert werden.

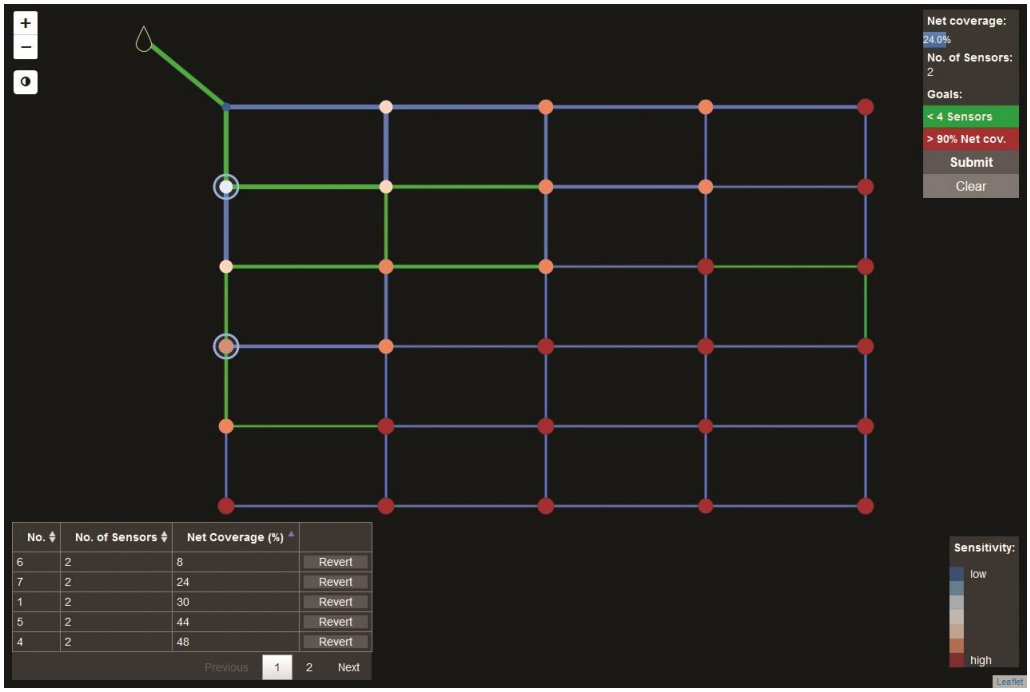

Abbildung 5-11: Darstellung des Poulakis-Netzwerks mit Standardfarbschema

In der linken oberen Ecke befinden sich drei Buttons, von denen zwei für das Zoomen (das alternativ auch mittels Mausrad erfolgen kann) und einer für die Wahl des Farbschemas zuständig ist. In Abbildung 5-11 wird das Standardfarbschema dargestellt. Alternativ dazu steht auch noch ein rein auf Kontrasten basierendes Farbschema zur Verfügung, um Personen mit eingeschränkter Farbwahrnehmung visuell zu unterstützen. Dieses alternative Farbschema ist in Abbildung 5-12 abgebildet. Beide Schemata bestehen aus sieben verschiedenen Farben.

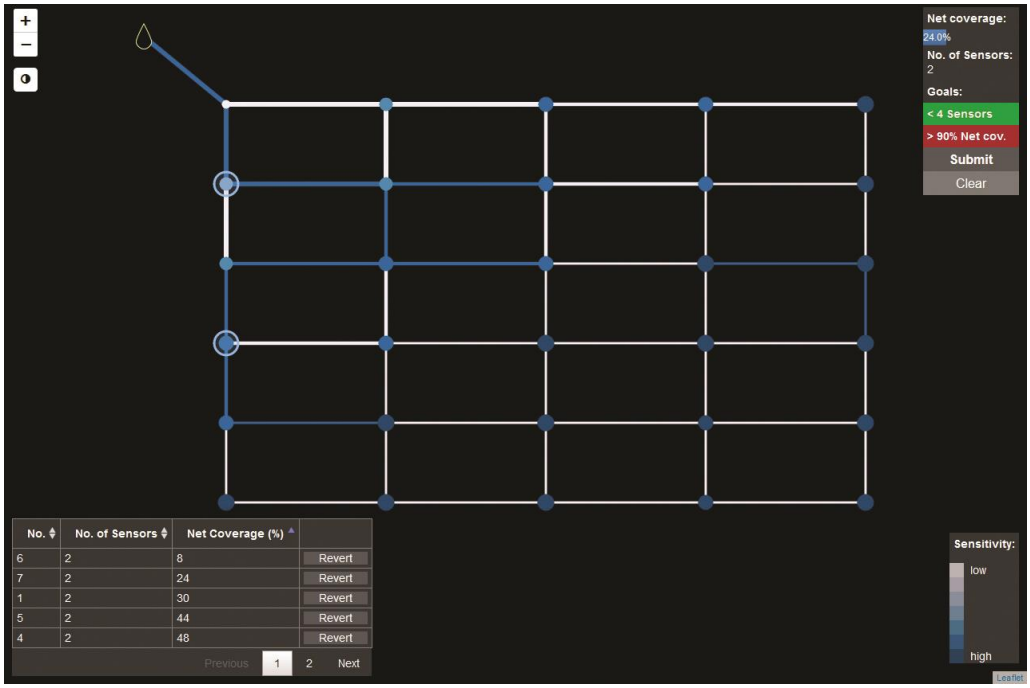

Abbildung 5-12: Darstellung des Poulakis-Netzwerks mit alternativem Farbschema

In der rechten oberen Bildschirmecke befindet sich ein Feld, in dem sowohl die erreichte Netzabdeckung als Fortschrittsbalken und die Anzahl der platzierten Sensoren zu finden sind, als auch die für das jeweilige Netz gesetzten Ziele und zwei Buttons, wobei einer (*Submit*) für die Evaluierung der momentanen Sensorpositionen zuständig ist und mit dem anderen alle platzierten Sensoren gelöscht werden können (*Clear*). Bei einem Klick auf den Button *Submit* wechselt dessen Text von *Submit* auf *Calculating…* bis die Berechnung der Netzabdeckung abgeschlossen ist. In dieser Zeit kann der Button auch nicht mehr betätigt werden. Die gesetzten Ziele werden, solange sie nicht erreicht werden, mit rotem Hintergrund dargestellt, bei Erreichung der Ziele wechselt die Hintergrundfarbe von Rot zu Grün.

Rechts unten befindet sich eine Legende, in der die Knotensensitivitäten mit dem aktuell ausgewählten Farbschema aufgeschlüsselt werden.

In der linken unteren Ecke befindet sich eine Tabelle, in der die bisher vom Spielenden abgegebenen Lösungen eingesehen werden können. Angezeigt wird dabei je Lösung ein fortlaufender Index, die Anzahl der platzierten Sensoren sowie die erreichte Netzabdeckung. Über diese drei Spalten können die Ergebnisse auch jeweils auf- bzw. absteigend sortiert werden. Je Lösung existiert am rechten Ende der Tabelle außerdem ein Button (*Revert*), mit dem auf die jeweilige Lösung zurückgesprungen werden kann. Die Tabelle besteht aus mehreren Seiten, auf denen jeweils fünf Lösungen angezeigt werden. Mittels der Buttons *Previous* und *Next* kann die aktuell angezeigte Lösung ebenso gewechselt werden, wie mit einem Klick auf die jeweilige Seitennummer.

Im Zentrum befindet sich das Netzwerk selbst. Im Netz von Poulakis existiert in der nordwestlichen Ecke ein Reservoir, von dem ausgehend das System mit Wasser versorgt wird. Reservoirs und Hochbehälter werden mit eigens für das Serious Game erstellten Symbolen in Tropfenform dargestellt (siehe Abbildung 5-13).

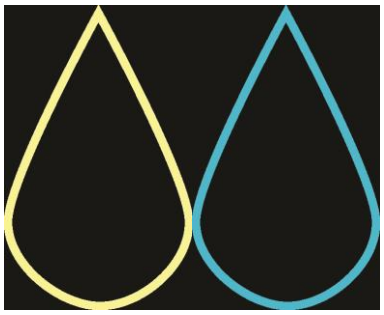

Abbildung 5-13: Symbole für Reservoirs (links) und Hochbehälter (rechts)

Reguläre Knoten werden als Punkte dargestellt, deren Radius und Farbe abhängig von der Knotensensitivität sind. Im Standardfarbschema sind Knoten mit einer geringen Sensitivität dunkelblau und mit hoher Sensitivität dunkelrot, während

beim alternativen Farbschema Knoten mit einer geringen Sensitivität weiß und mit hoher Sensitivität dunkelblau sind. In beiden Fällen werden sensitive Knoten mit einem größeren Durchmesser dargestellt als weniger sensitive Knoten. Knoten, auf denen Sensoren platziert wurden, werden blau eingekreist.

Leitungen werden proportional zum Leitungsdurchmesser mit unterschiedlichen Strichstärken dargestellt, andere Kantentypen werden mit einer konstanten Strichstärke visualisiert. Von den Sensoren abgedeckte Leitungen werden standardmäßig in Grün dargestellt, nicht abgedeckte blau. Beim alternativen Farbschema werden abgedeckte Leitungen dunkelblau eingefärbt, nicht abgedeckte weiß. Geschlossene Leitungen, Pumpen und Ventile werden in Rot dargestellt.

Ventile und Pumpen werden in unterschiedlichen Farben als strichlierte Kanten visualisiert.

5.5 Aufbau und Funktionsweise

Das Serious Game kann erst nach erfolgter Authentifizierung mittels der Shibboleth-Anmeldemaske der Technischen Universität Graz aufgerufen werden. Dies verhindert einerseits Angriffe von außen und spart außerdem die Implementierung einer Registrierung sowie eines eigenen Logins.

Serious Sensor Placement selbst besteht aus drei URL-Routen:

- /: Startseite
- /networks/<network>: Darstellung des Netzwerks network
- /fitness/: Berechnung der Netzabdeckung für ein definiertes Set an Sensorpositionen für ein bestimmtes Netzwerk

Auf der Startseite wird die HTML-Seite *index.html* dargestellt, mittels derer die SpielerInnen jenes Netz auswählen können, mit dem sie spielen wollen. Durch Klicken auf ein Netzwerk werden die Nutzer auf die Route /networks/<network> weitergeleitet, wobei <network> dem gewählten Netz entspricht (z.B. *C-Town*). Beim Aufrufen dieser Route werden etwaige existierende Lösungen des Spielenden für das gewählte Netz aus der PostgreSQL-Datenbank abgerufen und an die HTML-Seite als JSON übergeben. Dargestellt wird das Netzwerk über die HTML-Datei *osp.html*.

Beim Aufrufen der Netzroute werden einige Variablen und Funktionen geladen. So werden zwei Farbschemata geladen, je ein Symbol für Reservoirs und Hochbehälter (siehe Kapitel 5.4), sowie ein maximaler und ein minimaler Knotenradius. Zusätzlich wird eine Variable definiert, die angibt, welches Farbschema momentan in Verwendung ist. Aus den übergebenen existierenden Ergebnissen wird ein JavaScript-Objekt erstellt, um schnell auf die Ergebnisse zugreifen zu können.

Die Darstellung des Netzes geschieht mittels Leaflet. Leaflets ursprünglicher Verwendungszweck ist es, reale Karten mit realen Koordinatensystemen darzustellen, es ist allerdings auch möglich, fiktive Koordinaten zu verwenden. Dies geschieht mittels eines Referenzkoordinatensystems, das auf einem quadratischen Raster basiert. In diesem Raster können beliebige Koordinaten verwendet werden, somit auch fiktive Koordinaten von fiktiven hydraulischen Modellen (Leaflet, 2018). Dazu wird eine Leaflet-Karte (*map*) erstellt und *CRS.Simple* als Referenzkoordinatensystem angegeben.

Außerdem wird in der linken unteren Bildschirmecke eine Tabelle erstellt und mit den von Flask übergebenen bereits existierenden Lösungen des Spielenden befüllt.

In der rechten unteren Ecke wird eine Legende für die Knotensensitivität basierend auf dem momentan ausgewählten Farbschema angezeigt.

In der rechten oberen Ecke wird ein Feld erzeugt, in dem die aktuelle Netzabdeckung und die Anzahl aktuell platzierter Sensoren (beide anfangs 0), sowie die Spielziele und je ein Button zur Berechnung der Netzabdeckung und zum Löschen aller platzierten Sensoren dargestellt werden.

Zur Visualisierung der Knoten und Kanten wird je eine Ebene in Leaflet erstellt (*FeatureGroup*). In diesen Ebenen kann angegeben werden, wie mit den jeweiligen Inhalten (*features*) verfahren werden soll.

In der Ebene für Kanten wird beim Hinzufügen jeder Kante eine Funktion aufgerufen werden, die je nach Kantentyp eine unterschiedliche Darstellungsform für die Kante bestimmt. Außerdem wird bei den Kanten ein Tooltip (ein kleines Pop-Up-Fenster) erzeugt, das angezeigt wird, sobald der Spielende mit dem Cursor über die jeweilige Kante fährt. In diesem Tooltip wird der Kantentyp angezeigt (z.B. *Pipe*), sowie bei geschlossenen Kanten zusätzlich der Status (z.B. *closed Pipe*).

In der Knotenebene wird unterschieden, ob es sich bei dem Knoten um ein Reservoir, einen Hochbehälter oder einen Knotenpunkt handelt. Handelt es sich um ein Reservoir oder einen Hochbehälter, werden die Knoten mittels der dafür vorgesehenen Symbole dargestellt und ein Tooltip erstellt, in dem je nach Knotentyp entweder *Reservoir* oder *Tank* angezeigt wird.

Handelt es sich um einen regulären Knotenpunkt wird ein Punkt erzeugt, dessen Eigenschaften (Farbe und Radius) in einer eigenen Funktion für die jeweilige Knotensensitivität ermittelt wird. Der Knotenradius errechnet sich dabei wie folgt:

$$r_j = \lfloor r_{min} + \hat{s}_j * (r_{max} - r_{min}) + 0{,}5 \rfloor$$ Gleichung 5-19

mit:
r_j … *Radius des Knoten j*
r_{max} … *maximaler Radius*
r_{min} … *minimaler Radius*
$\hat{s}_j$ … *normierte mittlere Sensitivität des Knoten j*

Für Serious Sensor Placement wurde $r_{max} = 8$ und $r_{min} = 4$ gewählt.

Die erzeugten Knoten werden zusätzlich noch in einer Liste gespeichert, um leichter auf sie zugreifen zu können. Außerdem wird angegeben, dass bei einem Klick auf einen Knoten überprüft wird, ob auf diesem Knoten bereits ein Sensor platziert worden ist. Wurde noch kein Sensor an dieser Stelle platziert, wird eine Sensormarkierung zur Oberfläche hinzugefügt, die Koordinaten werden dazu vom ausgewählten Knoten übernommen. Zusätzlich wird die ID des Knotens, auf dem ein Sensor platziert wurde, in einer Liste gespeichert. Wurde hingegen schon ein Sensor an dieser Stelle platziert, wird die Sensormarkierung von der Oberfläche entfernt und die ID des Knoten aus der Liste der Sensorpositionen entfernt. Handelt es sich beim ausgewählten Knoten um ein Reservoir, erscheint eine Fehlermeldung. Da ein Reservoir das System mit konstantem Druck versorgt, können hier auch keine Druckänderungen hervorgerufen werden. Deshalb werden Reservoirs nicht als Sensorpositionen zugelassen.

Nach der Erzeugung der Ebenen werden diese mit den in den GeoJSON-Dateien gespeicherten Knoten und Kanten befüllt, das Netz wird also zur Weboberfläche hinzugefügt und dargestellt.

Die Evaluierung gewählter Sensorpositionen erfolgt über den Button *Submit*. Bei Betätigung dieses Buttons werden der Name des Netzes und die gewählten Sensorpositionen an Flask mittels POST übergeben. Flask übernimmt diese zwei Parameter und übergibt sie wiederum an eine Funktion, die mittels vorab generierter Sensitivitäts- und Residuenmatrizen die Netzabdeckung berechnet und die von den Sensoren abgedeckten Leitungen ausgibt.

Nach erfolgter Berechnung der erzielten Netzabdeckung werden die folgenden Informationen in die PostgreSQL-Datenbank eingespielt:

- Datum
- Name des hydraulischen Modells
- Nutzername
- Netzabdeckung
- Anzahl der platzierten Sensoren
- Knoten-IDs der Sensorpositionen
- Von den Sensoren abgedeckte Leitungen

Der Nutzername wird dabei nicht im Klartext gespeichert, sondern mit MD5 (*Message-Digest Algorithm 5)* verschlüsselt und dadurch anonymisiert.

Die Netzabdeckung und die Liste mit abgedeckten Leitungen werden von Flask wieder an die HTML-Seite übergeben. Mit diesen Daten werden die angezeigte Netzabdeckung und die angezeigte Anzahl platzierter Sensoren aktualisiert und die Einhaltung der gesetzten Ziele überprüft. Zusätzlich werden die Tabelle und das JavaScript-Objekt mit den existierenden Ergebnissen mit den neuen Resultaten ergänzt.

Sollen alle bereits platzierten Sensoren gelöscht werden, ist dies über den Clear-Button möglich. Bei dessen Betätigung werden alle Sensoren in der graphischen Oberfläche gelöscht, die Netzabdeckung und die Anzahl der platzierten Sensoren auf 0 gesetzt und die Farben der Leitungen auf die Standardfarben (blau bzw. weiß) zurückgesetzt. Außerdem werden die Hintergrundfarben der Ziele auf Rot geändert.

Wird ein Revert-Button in der Tabelle der bisherigen Ergebnisse betätigt, um auf ein altes Set an Sensorpositionen zu wechseln, werden zuerst alle momentan gesetzten Sensoren gelöscht, sowie die Farben der Leitungen auf den Ursprungswert gesetzt. Die Revert-Buttons weisen eine ID im Format *revert_i* auf, wobei i dem Index der Zeile entspricht, in welcher der Button platziert ist (beispielsweise hat der Button in der Zeile 3 die ID *revert_3*). Über den Namen des betätigten Buttons können somit der zugehörige Index ermittelt und damit die Ergebnisse aus dem JavaScript-Objekt abgerufen werden. Daraufhin werden die erzielte Netzabdeckung und die angezeigte Sensoranzahl aktualisiert, die Sensoren werden auf den Knoten platziert und die detektierten Leitungen eingezeichnet.

Der Button, der für den Wechsel des Farbschemas zuständig ist, ist ein EasyButton (*Leaflet EasyButton*, 2018) mit einem Glyphicon (Bootstrap - Glyphicons, 2018) als Symbol. Bei Betätigung dieses Buttons wird zuerst überprüft, welches Farbschema momentan in Verwendung ist, indem der Zustand der zugehörigen Statusvariable abgefragt wird (entweder *default* oder *alternative*). Danach werden die Farben der Knoten und der Leitungen auf das neue Farbschema aktualisiert und die Legende neu erstellt. Abschließend wird die Statusvariable auf den jeweils anderen Wert gesetzt.

5.6 Serious Game Klassifizierung

Schlussendlich wird hier die Klassifizierung von Serious Sensor Placement gemäß Kapitel 5.6 gezeigt:

Anwendungsgebiet:

Das Anwendungsgebiet ist die Siedlungswasserwirtschaft. Genauer spezifiziert handelt es sich um ein Spiel aus dem Bereich der Trinkwasserversorgung.

Spielziel:

Das Spielziel ist eine möglichst hohe Netzabdeckung bei möglichst geringer Sensoranzahl.

Initialisierung:

Für diese Arbeit wurde eine Einführung mittels kurzer Einführungspräsentation gewählt.

Anzahl und Art der Spielenden:

Bei Serious Sensor Placement handelt es sich um ein Einpersonenspiel, das auf Personen aus dem Bereich der Trinkwasserversorgung ausgerichtet ist.

Bedienoberfläche:

Die Bedienoberfläche ist eine browserbasierte HTML-Oberfläche, deren Design mit CSS angepasst wurde.

Simulationsmodell:

Für die Simulation wird ein EPANET-Modell in Kombination mit dem Sensorplatzierungsalgorithmus von Casillas verwendet.

Realismus:

Da die genutzten Residuen- und Sensitivitätsmatrizen mit einer Leckagegröße von 50 L/s gerechnet wurden, wird die Netzabdeckung auch für diese Leckagegröße berechnet. Bei kleineren und in der Praxis häufigeren Leckagegrößen kann mit der Methode von Casillas nur eine deutlich geringere Netzabdeckung erreicht werden. Auch Effekte wie das Rauschen der Signale von Drucksensoren werden nicht berücksichtigt.

Leistungsfeedback:

Aufgrund der Rechenzeit bei der Evaluierung der Sensorpositionen (in C-Town ca. 2 Sekunden), kann keine kontinuierliche Evaluierung des Fortschritts des Spielenden vorgenommen werden. Deshalb sind die Spielenden gefordert, aktiv die Bewertung der Sensorpositionen zu starten.

Fortschrittsüberwachung:

Jede bereits evaluierte Lösung kann im Nachhinein wieder aufgerufen werden.

Portabilität:

Es handelt sich um ein Onlinespiel, das im Browser aufgerufen wird.

6 Feldversuch

Am 26. November 2018 wurde Serious Sensor Placement am Institut für Siedlungswasserwirtschaft und Landschaftswasserbau getestet. In diesem Kapitel wird zuerst der Ablauf des Feldversuchs beschrieben, bevor die Ergebnisse des Feldversuchs präsentiert werden.

6.1 Ablauf

Durchgeführt wurde der Feldversuch mit zwölf Probandinnen und Probanden. Alle zwölf waren Studierende oder Angestellte der Technischen Universität Graz und mit einer Ausnahme mit der Thematik der Trinkwasserversorgung vertraut, allerdings in unterschiedlichem Detailgrad. Die Testpersonen hatten vor dem Versuch noch keine Möglichkeit gehabt, das Spiel zu testen und waren nur zum Teil mit der Thematik der Sensorpositionierung vertraut.

Zu Beginn des Feldversuchs wurden den ProbandInnen in einer kurzen Präsentation sowohl Serious Games als auch Sinn und Zweck der Platzierung von Drucksensoren zur Leckagedetektion und -lokalisierung erläutert. Dabei wurde auch erwähnt, dass Algorithmen zur Sensorplatzierung existieren, es wurde allerdings weder erläutert, wie diese funktionieren, noch welcher Algorithmus zur Evaluierung der Sensorpositionen verwendet wird. Anschließend wurde die Bedienung der Weboberfläche anhand des Netzes von Poulakis erklärt, bevor die Testpersonen schließlich rund 20 Minuten lang Zeit hatten, Sensorpositionen in C-Town (Abbildung 6-1) festzulegen und zu optimieren.

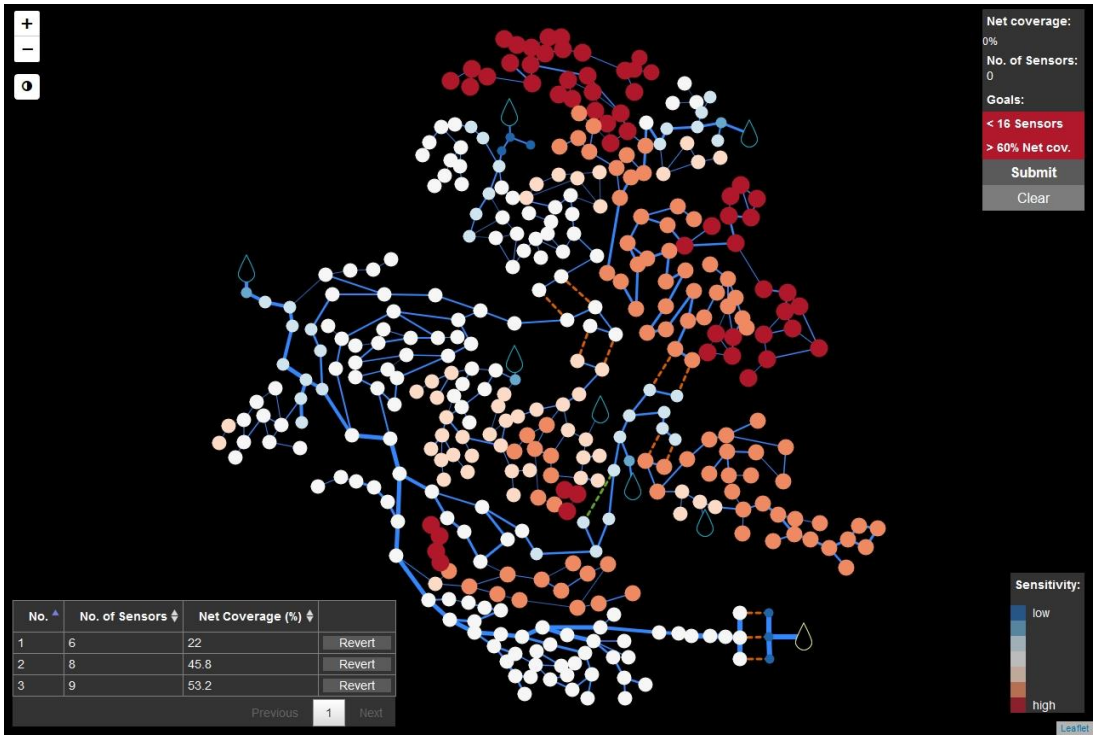

Abbildung 6-1: Serious Sensor Placement Weboberfläche für C-Town

Nach Ablauf der Testzeit wurden Fragebögen verteilt, auf denen das Spiel mittels vierer Bewertungen im Schulnotensystem und dreier Fragen evaluiert werden konnte. Bei den Bewertungen mit Schulnoten bestand außerdem die Möglichkeit, Anmerkungen jeweils in einem Textfeld zu notieren.

Die folgenden Kriterien wurden bewertet:

- Übersichtlichkeit der Weboberfläche
- Intuitivität der Bedienelemente
- Spielspaß bei der Bedienung
- Motivation, immer bessere Ergebnisse zu erzielen

Zusätzlich wurden die folgenden Fragen gestellt:

- Was hat Ihnen am Serious Game gefallen?
- Was hat Ihnen am Serious Game nicht gefallen?
- Haben Sie sonstige Anmerkungen oder Verbesserungsvorschläge?

Die Auswertung dieses Fragebogens folgt in Kapitel 6.3.

6.2 Serious Sensor Placement Ergebnisse

Innerhalb der rund 20 Minuten wurden in Summe 721 Lösungen abgegeben. Wie in Kapitel 5.5 erwähnt wurde, wurden die Nutzernamen anonymisiert, zur einfacheren Lesbarkeit wurden die verschlüsselten Benutzernamen der zwölf Testpersonen für die folgenden Auswertungen durch die Zahlen 1 bis 12 ersetzt.

Abbildung 6-2 zeigt die Anzahl an abgegebenen Lösungen je Testperson. Im Schnitt wurden rund 60 Lösungen von jeder Testperson abgegeben, Testperson 5 hat jedoch mit 122 Lösungen mit Abstand am meisten Lösungen produziert, während Testperson 11 mit 29 Lösungen deutlich weniger Sets von Sensorpositionen evaluiert hat.

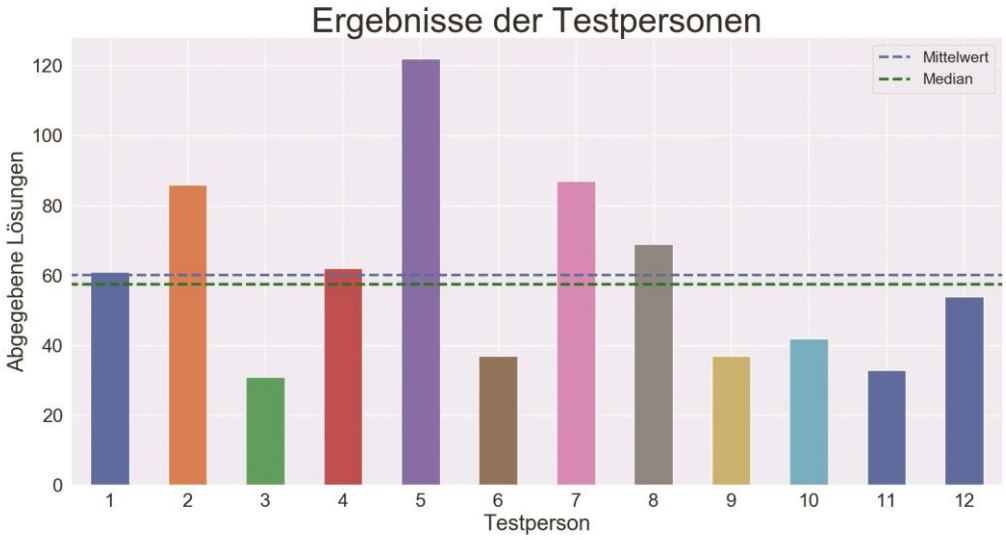

Abbildung 6-2: Anzahl abgegebener Lösungen je Testperson

Tabelle 6-1 zeigt je Testperson die maximal erreichten Netzabdeckungen und die Anzahl an Sensoren, die bei dieser Lösung platziert wurde. Bis auf die Testpersonen 3 und 4 haben alle ProbandInnen 15 Sensoren bei ihrem besten Ergebnis verwendet. Die Testpersonen 3 und 4 erreichten dabei eine Netzabdeckung von 69,9 bzw. 69,2 %, und erreichen daraufhin bei einer Reihung der Testpersonen nach der maximal erzielten Netzabdeckung die Plätze 6 und 8. Bei dieser Analyse wurden nur Lösungen betrachtet, welche die maximale Sensoranzahl nicht überschritten.

Tabelle 6-1: Feldversuchsauswertung je Testperson

Testperson Nr.	**Netzabdeckung b. bester Lösung (%)**	**Sensoranzahl b. bester Lösung**	**Reihung nach Netzabdeckung**
1	70,4	15	5
2	69,7	15	7
3	69,9	14	6
4	69,2	14	8
5	73,1	15	3
6	71,3	15	4
7	57,4	15	12
8	75,0	15	1
9	74,3	15	2
10	65,0	15	9
11	62,0	15	10
12	60,4	15	11

Abbildung 6-3 zeigt in Boxplots je Testperson die Anzahl der gesetzten Sensoren. Es wird dabei ersichtlich, dass fünf der zwölf Testpersonen (Nr. 1, 2, 4, 6 und 9) beinahe ausschließlich Lösungen mit 15 platzierten Sensoren ermittelt haben, also jener Sensoranzahl, die den Testpersonen als Maximalsensoranzahl vorgegeben war. Drei weitere Testpersonen (5, 8 und 10) platzierten ebenfalls rund 15 Sensoren, weisen jedoch eine etwas größere Streuung der Sensoranzahl auf. Die Testpersonen 11 und 12 haben im Schnitt weniger als 15 Sensoren platziert (rd. 13 Sensoren), während Testperson 3 vornehmlich Lösungen mit 12 oder 13 Sensoren abgegeben hat. Die Lösungen von Testperson 7 weisen die mit Abstand größte Streuung in Bezug auf die Sensoranzahl auf, aber auch hier liegt der Median bei rund 13 Sensoren. Testpersonen 3 und 4 versuchten nur Lösungen mit maximal 15 Sensoren, alle anderen Testpersonen übermittelten auch Lösungen mit mehr als 15 Sensoren. Testperson 7 gab die Lösungen mit der höchsten Sensoranzahl ab (19 Sensoren).

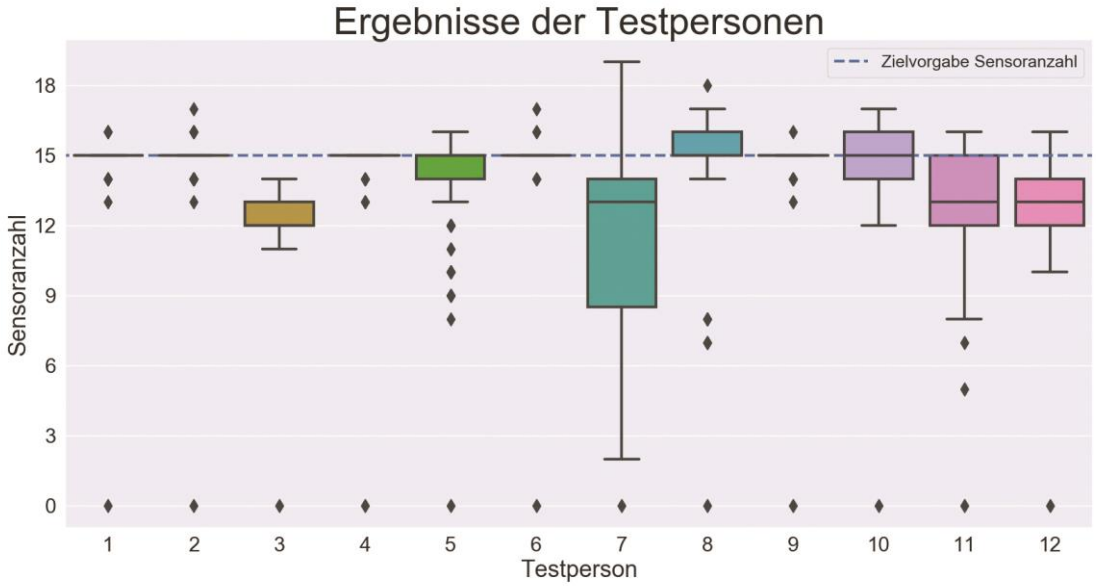

Abbildung 6-3: Anzahl platzierter Sensoren je Testperson

In Abbildung 6-4 sind wiederum mittels Boxplots die erzielten Netzabdeckungen je Testperson abgebildet. Zusätzlich sind die Zielvorgabe in Bezug auf die minimale Netzabdeckung (60 %) sowie die beste mittels Differential Evolution ermittelte Netzabdeckung angegeben (66,4 %).

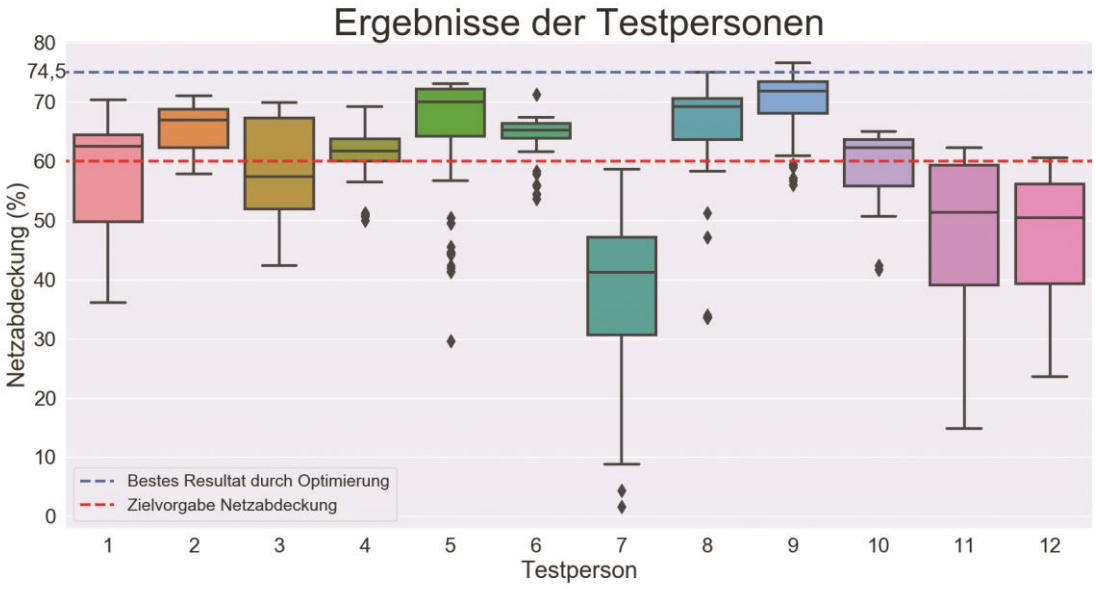

Abbildung 6-4: Erzielte Netzabdeckungen der Testpersonen im Feldversuch

Der Median der erzielten Netzabdeckungen liegt bei den Testpersonen 1, 2, 4, 5, 6, 8, 9 und 10 über der Mindestnetzabdeckung. Fast alle ProbandInnen erzielten bei ihrem besten Ergebnis eine Netzabdeckung von über 60 %, lediglich Testperson 7 erreichte die Mindestnetzabdeckung mit keiner Lösung. Die Testpersonen 11 und 12 erreichten die minimale Netzabdeckung nur knapp, während die besten Lösungen der Testpersonen 1, 2, 3, 4, 5, 6, 8, und 9 sogar über der mittels

Differential Evolution erreichten Netzabdeckung liegen. Testperson 9 weist mit 75 % das in Bezug auf die erreichte Netzabdeckung beste Ergebnis aller Testpersonen auf.

Dies ist auf das stochastische Verhalten von Differential Evolution zurückzuführen. Obwohl die Ergebnisse bei Differential Evolution mit einer Populationsgröße von 100 Individuen über 100 Generationen hinweg optimiert wurde, ist das damit erreichte Ergebnis nicht zwangsläufig ein globales Optimum. Deshalb wurde die Berechnung der optimalen Sensorpositionen bei einer Leckagegröße von 50 L/s für 2 bis 15 Sensoren je Sensoranzahl zehn weitere Male wiederholt. Dies führt dazu, dass je Sensoranzahl bis zu zehn unterschiedliche Sets an Sensorpositionen mit unterschiedlichen Fitness-Werten vorliegen.

Abbildung 6-5 zeigt mittels Boxplots je Sensoranzahl die mit Differential Evolution gefundenen Ergebnisse. An jene Ergebnisse, die je Sensoranzahl die höchste Netzabdeckung erzielten, wurde wiederum eine Kostenfunktion angepasst.

Die Abbildung verdeutlicht das stochastische Verhalten des Optimierers. So wurden beispielsweise bei 6 Sensoren Netzabdeckungen von 48 bis 54 % erreicht. Bei der Platzierung von zwei Sensoren hingegen wurden sehr konstante Netzabdeckungen von rund 25 % erzielt. Die höchste Netzabdeckung bei 15 platzierten Sensoren verbessert sich von 66,5 auf 74,5 %, liegt somit aber immer noch knapp unter dem besten im Feldversuch erreichten Ergebnis von 75,0 %.

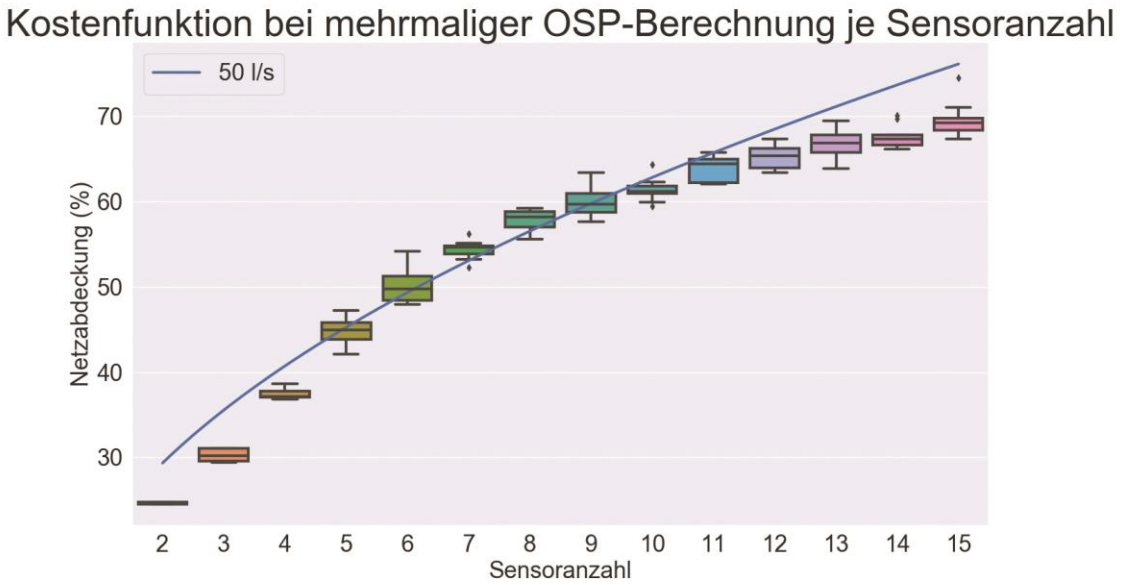

Abbildung 6-5: Berechnete Netzabdeckungen für unterschiedliche Sensoranzahlen

Abbildung 6-6 zeigt eine Gegenüberstellung der besten Feldversuchergebnisse je Sensoranzahl und der Kostenfunktion, die mit einer Leckagegröße von 50 L/s ermittelt wurde und in Abbildung 6-5 dargestellt ist.

Bis zu einer Sensoranzahl von 11 Sensoren liegen die Ergebnisse der Testpersonen unter den errechneten Ergebnissen, ab 12 Sensoren sind die Testpersonen und Differential Evolution ungefähr gleichauf.

Das Ziel von 60 % Netzabdeckung wurde im Feldversuch bei der Platzierung von zwölf Sensoren erreicht, der Optimierer erreichte diese Netzabdeckung bereits mit neun Sensoren. Es lässt sich aus der Darstellung erkennen, dass die ProbandInnen zuerst vornehmlich versucht haben die Netzabdeckung über eine Erhöhung der Sensoranzahl zu erreichen. Gleichzeitig ist allerdings auch zu erkennen, dass sich bei der Platzierung von immer mehr Sensoren ein Lerneffekt eingestellt haben muss, da die Steigung der Kurve der Testpersonenergebnisse wesentlich höher ist, als jene der Kostenfunktion.

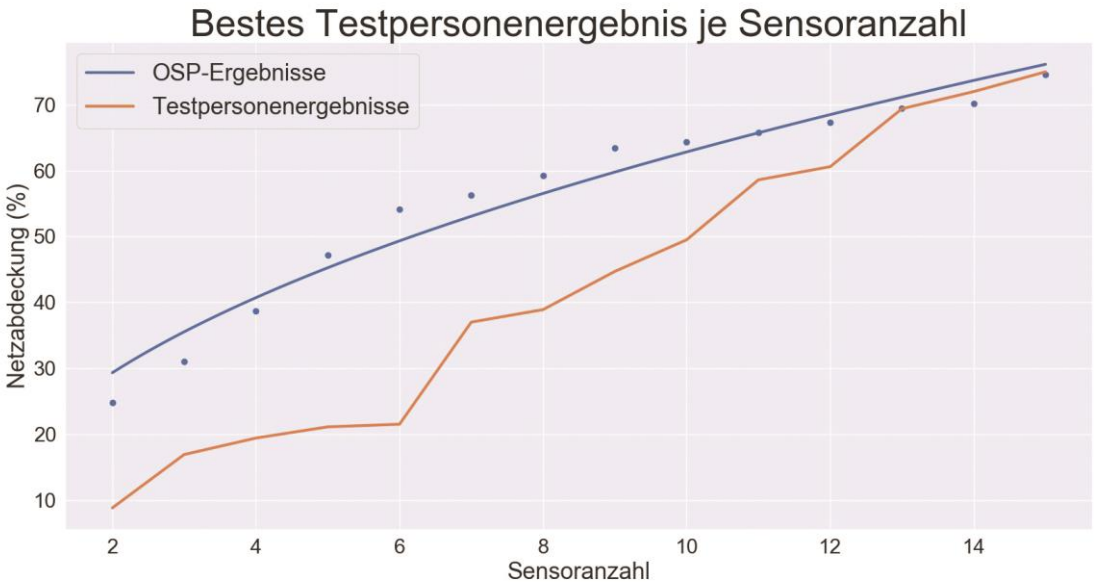

Abbildung 6-6: Bestes Ergebnis je Sensoranzahl und Kostenfunktion

In Bezug auf die gesetzten Spielziele (Netzabdeckung von mind. 60 % und maximal 15 Sensoren), lässt sich feststellen, dass diese Ziele einerseits erreichbar sind, andererseits aber nicht zu niedrig angesetzt wurden. Dies lässt sich daran erkennen, dass viele der im Laufe der 20 Minuten abgegebenen Lösungen die minimale Netzabdeckung nicht erreichen und offensichtlich erst proaktiv nach einer entsprechenden Lösung gesucht werden musste. Eine Testperson konnte die gesetzten Ziele nicht erreichen.

6.2.1 Vergleich der Sensorpositionierungen

Schlussendlich werden noch die Lösungen der drei Spielenden mit der höchsten Netzabdeckung, sowie die mittels Differential Evolution optimierten Sensorpositionen miteinander verglichen.

Zu diesem Zweck wurde das Netzwerk C-Town in sieben Zonen eingeteilt (Zonen A bis F), wobei die Zonengrenzen – bis auf die Grenzen zwischen den Zonen D

und F, sowie zwischen den Zonen A und C – bei den Pumpstationen im Netz gewählt wurden. Die Pumpen wurden als Grenzen definiert, da Druckänderungen sich nicht über diese ausbreiten können und deshalb Sensoren keine Leckagen jenseits einer Pumpe detektieren und lokalisieren können. Die Zoneneinteilung ist in Abbildung 6-7 dargestellt.

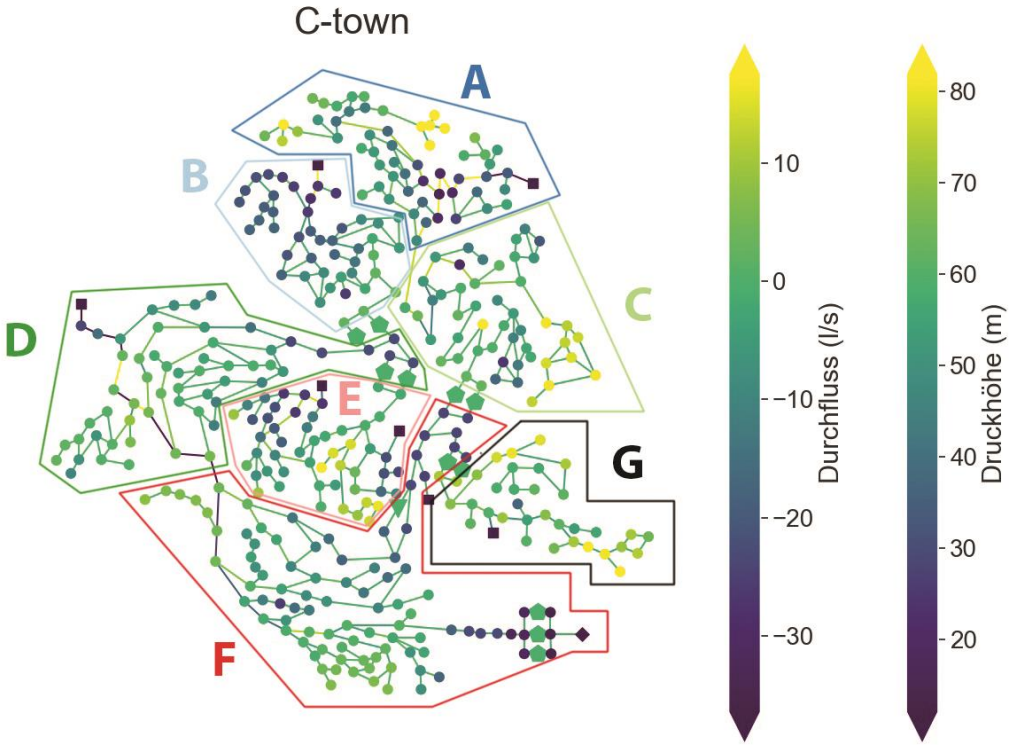

Abbildung 6-7: Zoneneinteilung von C-Town

Zur Gegenüberstellung wurde analysiert, wie viele Sensoren durch die Spielenden je Zone platziert wurden und wo die Sensoren in den Zonen verteilt wurden. Es wurden die drei in Bezug auf die Netzabdeckung besten Spielenden (SpielerIn 8 mit 75,0 %, SpielerIn 9 mit 74,3 % und SpielerIn 5 mit 73,1 % Netzabdeckung), sowie die beste mit Differential Evolution ermittelte Sensorplatzierung (74,5 % Netzabdeckung) herangezogen. Nachdem alle drei betrachteten Spielenden 15 Sensoren bei ihrer besten Lösung platzierten, wurde auch nur die für 15 Sensoren mit Differential Evolution berechnete Lösung betrachtet. In Tabelle 6-2 wird die Anzahl an Sensoren je Lösung und Zone angegeben. Die zugrunde liegenden Sensorplatzierungen sind in Abbildung 6-8, Abbildung 6-9, Abbildung 6-10 und Abbildung 6-11 dargestellt.

Tabelle 6-2: Sensorplatzierung je Zone

Zone	SpielerIn 8	SpielerIn 9	SpielerIn 5	Differential Evolution
A	1	2	2	1
B	2	2	2	2
C	2	3	2	2
D	2	1	3	3
E	2	2	2	1
F	4	3	2	3
G	2	2	2	2

Die Spielenden verteilten ihre Sensoren ähnlich und platzierten zumindest einen Sensor je Zone. Auch Differential Evolution verteilte die Sensoren über das gesamte Netz, in den Zonen D und G wurden Sensoren allerdings sehr nahe beieinander platziert. Die Spielenden verteilten ihre Sensoren auch innerhalb der Zonen gleichmäßiger als der Optimierer. Die betrachteten Testpersonen haben offensichtlich haben ähnliche Wege zur Sensorplatzierung gewählt, indem sie ihre Sensoren gleichmäßig über das Netz verteilten.

In den Zonen A, B und C wurde ein Knoten von allen drei Spielenden gewählt, der Optimierer wählte andere Positionen.

Eine Sensorposition in Zone D wurde von allen betrachteten Testpersonen und dem Optimierer gleichermaßen gewählt.

Die drei SpielerInnen platzierten zwei Sensoren in Zone E, während Differential Evolution lediglich einen Knoten mit einem Sensor versah.

Die von den drei Testpersonen und von Differential Evolution gewählten Sensorpositionen in Zone F ähneln einander ebenfalls. Zwei Testpersonen und der Optimierer platzierten einen Sensor direkt nach der ersten Pumpstation nach dem Reservoir, und auch im darauffolgenden verzweigten Abschnitt positionierten die Spielenden und der Optimierer Sensoren auf ähnlichen Knoten.

In Zone G platzierten die Testpersonen zwei Sensoren auf beinahe denselben Positionen, während der Optimierer zwei Sensoren nahe beieinander zwischen den von den Testpersonen gewählten Positionen situierte.

Die Lösungen der Testpersonen 8 und 9 sind beinahe ident, neun der 15 Sensorpositionen stimmen überein, drei weitere wurden sehr ähnlich platziert.

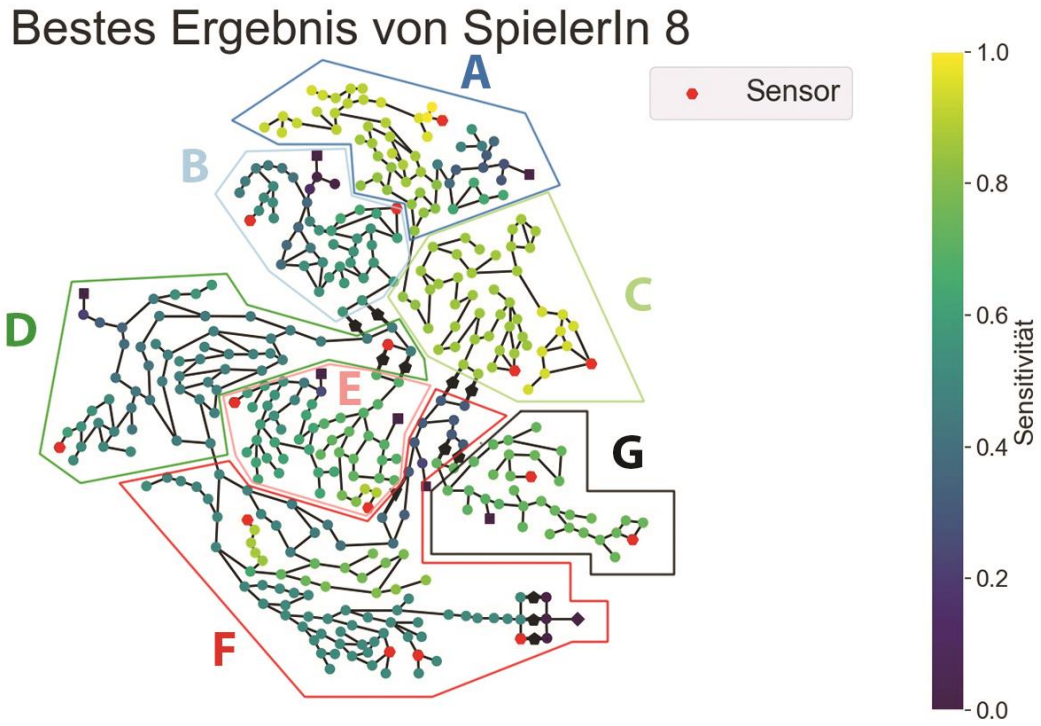

Abbildung 6-8: SpielerIn 8: höchste Netzabdeckung (75,0 %)

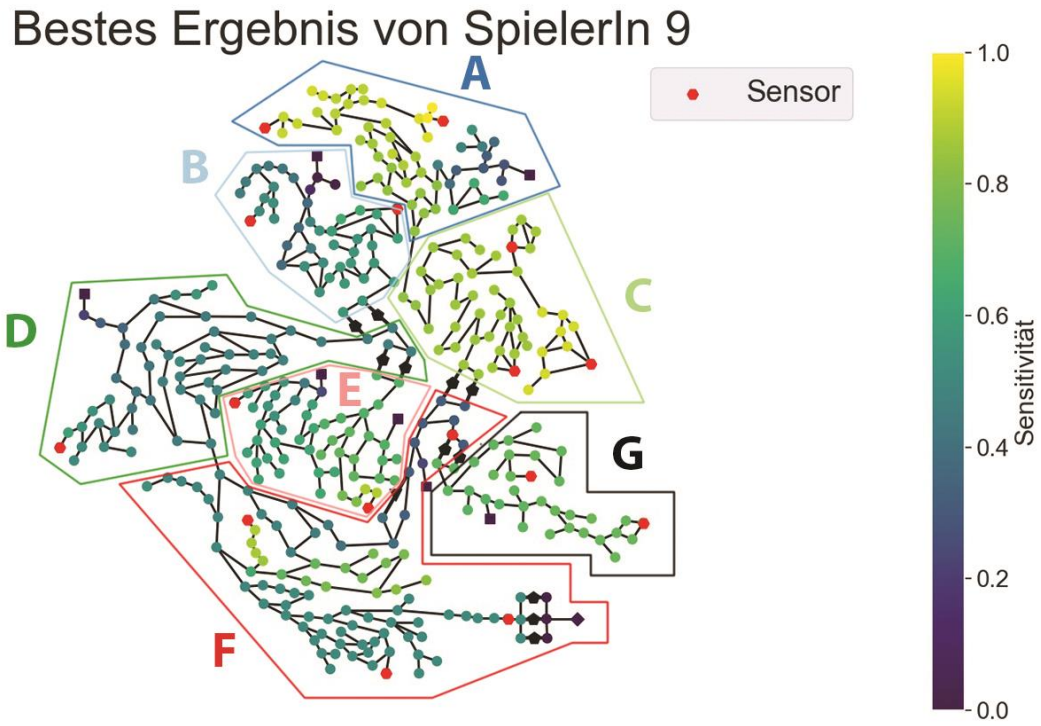

Abbildung 6-9: SpielerIn 9: zweihöchste Netzabdeckung (74,3 %)

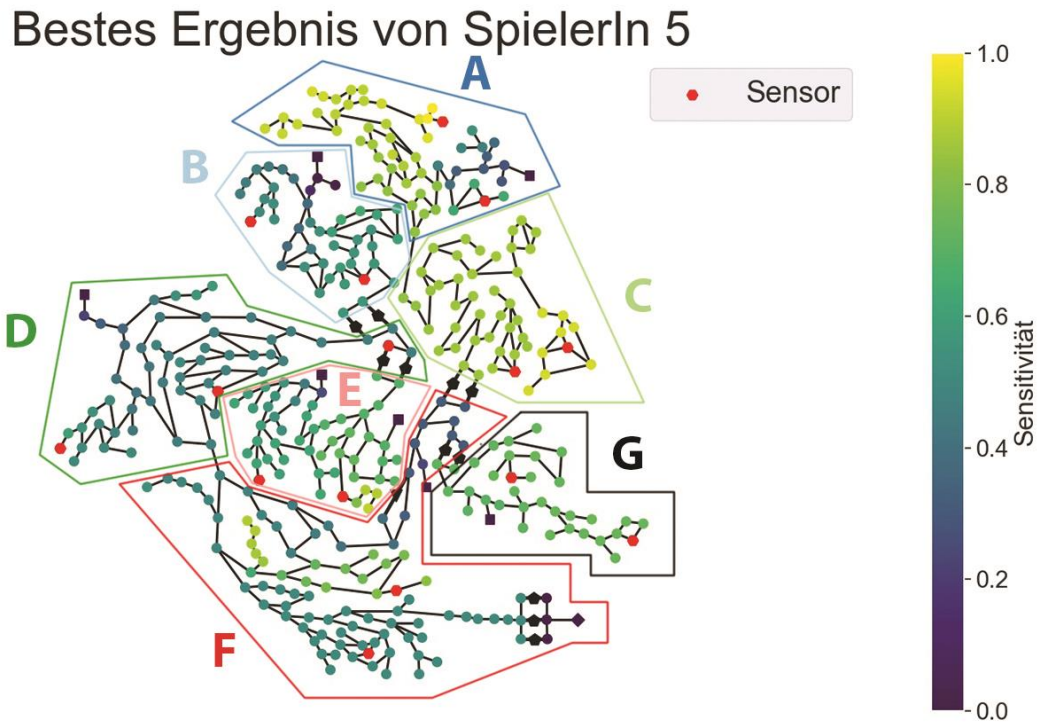

Abbildung 6-10: SpielerIn 5: dritthöchste Netzabdeckung (73,1 %)

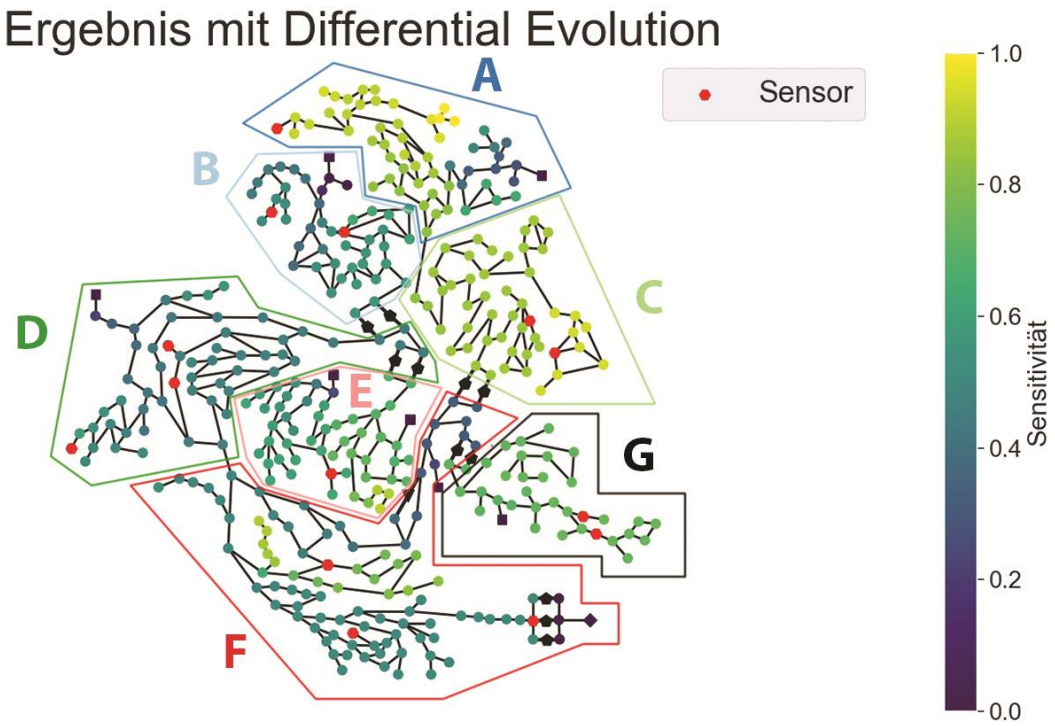

Abbildung 6-11: Differential Evolution (74,5 % Netzabdeckung)

6.3 Fragebogenergebnisse

In diesem Kapitel werden die Ergebnisse der Fragebogenauswertungen dargestellt. Tabelle 6-3 zeigt die mittels Schulnoten bewerteten Kriterien, die im Fragebogen abgefragt wurden.

Tabelle 6-3: Fragebogenauswertung: Bewertungen

Kriterium	Minimum	Maximum	Mittelwert
Übersichtlichkeit der Weboberfläche	1,00	3,00	1,92
Intuitivität der Bedienelemente	1,00	2,00	1,25
Spielspaß bei der Bedienung	1,00	2,00	1,25
Motivation, immer bessere Ergebnisse zu erzielen	1,00	1,00	1,00

Größter Handlungsbedarf besteht demnach bei der Übersichtlichkeit der Weboberfläche.

Die folgenden Punkte wurden von den Probandinnen und Probanden bemängelt:

- Zoomeinstellungen:
  - Die Abstufungen des Zooms sind nicht fein genug.
  - Die Knotengröße ist konstant und unabhängig von der Zoomstufe.
  - Bei Darstellung des gesamten Netzes werden Kanten durch die Knoten verdeckt.
- Legende:
  - Die verschiedenen Kantentypen sollten in der Legende angeführt werden.
  - Der Zusammenhang zwischen Leitungsdurchmesser und Strichstärke der Kanten sollte in der Legende angeführt werden.
  - Die Unterscheidung zwischen detektieren und nicht detektierten Leitungen fehlt in der Legende.
  - Die Legende könnte größer dargestellt werden.
- Die Anzahl platzierter Sensoren wird erst beim Übermitteln der Sensorpositionen zur Evaluierung aktualisiert.
- Der Bereich, indem sich der Cursor über einer Kante befinden muss, um den Kantentyp anzuzeigen, ist zu klein.
- Das Serious Game ist für Geräte mit kleinen Bildschirmen (z.B. Tablets) weniger gut geeignet.
- Die Tabelle, in der die bisher abgegebenen Lösungen abgebildet ist, blättert nicht automatisch auf die letzte Seite.

Positiv hervorgehoben wurden die folgenden Punkte:

- Der Anreiz sich immer weiter zu verbessern

- Der Spielspaß
- Die intuitive und einfache Bedienung
- Die Möglichkeit, zu alten Lösungen zurückzuspringen

Folgende mögliche Verbesserungen wurden von den Testpersonen zusätzlich vorgeschlagen:

- Das Erreichen der Ziele könnte signalisiert werden (beispielsweise mittels eines Pop-ups oder mit einem Soundeffekt).
- Eine *Hall of Fame* mit den bisher besten Ergebnissen könnte angezeigt werden.
- Eine Zeitmessung könnte implementiert werden, um den Spielenden anzuzeigen, wie lange sie schon spielen.
- Tipps, mit denen die Spielenden beim Systemverständnis unterstützt werden, könnten hinzugefügt werden (beispielsweise ein Tipp nach fünf Versuchen).
- Ein Zähler könnte angezeigt werden, der den Spielenden mitteilt, wie viele Sensoren sie noch platzieren können, ohne gegen das vorgegebene Ziel zu verstoßen.
- Es könnten auf Basis realer Werte die Kosten, die durch das Platzieren der Sensoren entstehen, eingeblendet werden.
- Der wissenschaftliche Hintergrund des Serious Games könnte auf der Weboberfläche erklärt werden, sodass keine Präsentation davor nötig ist.

Die von den Testpersonen abgegebenen Kritikpunkte sind allesamt nachvollziehbar und sie könnten in einer zukünftigen Version von Serious Sensor Placement behoben werden.

Insgesamt beschränkt sich die Kritik eher auf Einzelheiten, die Spielmechanik an sich wurde nicht kritisiert und auch sonst lässt sich ein positives Fazit ziehen: Die Spielenden waren ausnahmslos motiviert, ihre Ergebnisse weiter zu verbessern und fanden sich in der Weboberfläche zurecht, auch wenn hier sichtlich noch Verbesserungspotential besteht.

7 Zusammenfassung und Schlussfolgerung

Eingangs wurden in dieser Arbeit bestehende Serious Games und existierende Algorithmen zur optimalen Platzierung von Drucksensoren zur Leckagedetektion und -lokalisierung sowie Optimierungsmethoden für mathematische Probleme betrachtet.

Viele bestehende Serious Games sind vollwertige, ausgereifte Spiele mit 3D-Grafik, die dem Spielenden im Rahmen einer Geschichte Kenntnisse vermitteln. Andere Spiele hingegen beschränken sich auf eine zweidimensionale Oberfläche und verzichten gänzlich auf eine Handlung. Die letztere Variante wurde auch für das erstellte Serious Game *Serious Sensor Placement* gewählt.

Es war für die Erstellung des Serious Games nötig, einen OSP-Algorithmus zur Bewertung der durch die Spielenden gewählten Sensorpositionen zu wählen. Da beim Algorithmus nach Casillas eine eindeutige Unterscheidung in von Sensorpositionen abgedeckte und nicht abgedeckte Knoten möglich ist, fiel die Wahl auf diesen Algorithmus. Durch eine Neuinterpretation der Fitness nach Casillas konnte außerdem eine Netzabdeckung im hydraulischen Modell definiert werden.

Des Weiteren war es notwendig, die Berechnung des gewählten OSP-Algorithmus so zu adaptieren, dass Leckagen auf Leitungen und nicht nur bei bereits bestehenden Knoten simuliert werden können. Außerdem wurde eine Möglichkeit geschaffen, verschiedene OSP-Berechnungen mit einem einfach zu bedienenden Automatisierungsserver durchzuführen. Dies ermöglicht einen zusätzlichen Nutzen des geschriebenen Codes über das Serious Game hinaus.

Im Serious Game wurden den Spielenden zwei Ziele vorgegeben: Sie sollten eine minimale Netzabdeckung bei einer maximalen Anzahl an platzierten Sensoren erreichen. Um diese Ziele festzulegen, wurden mit dem Algorithmus nach Casillas und Differential Evolution als Optimierer Kostenfunktionen für unterschiedliche Leckagegrößen ermittelt. Dabei zeigte sich ein Zusammenhang zwischen Netzabdeckung, Sensoranzahl und Leckgröße. Durch diesen Zusammenhang war es möglich, für eine gewisse Sensoranzahl eine erreichbare Netzabdeckung als Ziel vorzugeben.

Die Umsetzung des Serious Games erfolgte in erster Linie mittels Leaflet, einer ursprünglich zur Visualisierung von Karten gedachten Software. Mittels Python, HTML und CSS wurden verschiedene Bedienelemente zur Oberfläche hinzugefügt, wobei insbesondere auf Nutzerfreundlichkeit Wert gelegt wurde. So steht beispielsweise für Personen mit eingeschränkter Farbwahrnehmung ein alternatives Farbschema zur Verfügung.

Das Serious Game wurde nach Abschluss der Programmiertätigkeiten in einem Feldversuch mit zwölf Personen getestet und von diesen mittels eines Fragebogens bewertet. Evaluiert wurden anschließend die Anzahl abgegebener Lösung und je Testperson die Lösung mit der höchsten Netzabdeckung. Dabei zeigte sich, dass die Testpersonen die vorher mit Differential Evolution berechneten Ergebnisse übertrafen. Dies war auf das stochastische Verhalten des Optimierers zurückzuführen. Durch mehrmaliges Wiederholen der OSP-Berechnungen konnten bessere Netzabdeckungen durch den Optimierer erreicht werden, auch wenn das beste Spielergebnis nach wie vor leicht über dem besten mit Differential Evolution bestimmten Ergebnis liegt.

Von den Testpersonen wurden in den Fragebögen auch Verbesserungsvorschläge unterbreitet, die in einer überarbeiteten Version von Serious Sensor Placement Platz finden können. Beispiele dafür wären eine Anzeige des besten bisher erzielten Spielergebnisses sowie ein Belohnungssystem für jene Personen, welche die gesetzten Ziele erreichen konnten.

Weiteres Verbesserungspotential besteht bei der Dauer der Evaluierung von Sensorpositionen. Dieser Vorgang dauert momentan rund zwei Sekunden, was es einerseits unmöglich macht, die Sensorpositionen in Echtzeit zu bewerten und andererseits den Spielfluss leicht hemmt.

Zusätzlich könnten weitere hydraulische Modelle mit unterschiedlichen Schwierigkeitsgraden in die Weboberfläche eingepflegt werden. Momentan stehen zwei Modelle zur Verfügung: C-Town und das Netz von Poulakis. Der eigentliche Spielinhalt beschränkt sich dabei auf C-Town, da das Netz von Poulakis aufgrund der überschaubaren Anzahl an Knoten und Kanten sowie der rasterförmigen Anordnung der Knoten keine Herausforderung darstellt.

Das Netz von Poulakis könnte jedoch in Zukunft auch anderwärtig genutzt werden: Momentan ist eine Einführung in das Serious Game nötig, um die Bedienelemente sowie den wissenschaftlichen Hintergrund zu vermitteln. In einer zukünftigen Version des Serious Games könnte das Netz von Poulakis zur Einführung genutzt werden, indem die Bedienelemente und die Funktionsweise mittels Pop-ups oder ähnlicher Elemente erklärt werden. In Kombination mit einem einführenden Text auf der Weboberfläche könnte dies die Abhaltung eines Workshops oder einer mündlichen Einführung obsolet machen.

Eine Alternative dazu wäre es, das Serious Game in eine in der Praxis anwendbare Software zu überführen. So könnte die Oberfläche so umgestaltet werden, dass beliebige hydraulische Modelle hochgeladen werden können um eine ideale Platzierung von Sensoren zu erarbeiten. Wird auch noch eine Wahlmöglichkeit der Leckagegröße implementiert, könnten sogar die Netzabdeckung bei unterschiedlicher Leckagegröße ermittelt werden.

Literaturverzeichnis

Abt C. C. (1987) *Serious Games*, University Press of America, Lanham, United States of America.

Bewersdorff J. (2014) *Objektorientierte Programmierung mit JavaScript: Direktstart für Einsteiger*, Springer Vieweg, Wiesbaden.

Blesa J., Nejjari F., & Sarrate R. (2015) Robust sensor placement for leak location: analysis and design. Journal of Hydroinformatics, 136–148.

Bootstrap - Glyphicons (2018) [online] https://getbootstrap.com/docs/3.3/components/ (Zugegriffen 8. Dezember 2018).

Casillas M. V., Puig V., Garza-Castañón L. E., & Rosich A. (2013) Optimal Sensor Placement for Leak Location in Water Distribution Networks Using Genetic Algorithms. Sensors, **13**(11), 14984–15005.

Chew C., Zabel A., Lloyd G. J., Gunawardana I., & Monninkhoff B. (2014) A Serious Gaming Approach For Serious Stakeholder Participation. , (International Conference on Hydroinformatics), 8.

Dörner R., Göbel S., Effelsberg W., & Wiemeyer J. (2016) *Serious Games - Foundations, Concepts and Practice*, Springer International Publishing, Basel.

Flask (2018) [online] http://flask.pocoo.org/ (Zugegriffen 8. Dezember 2018).

Fortin F.-A., Rainville D., Gardner M.-A. G., Parizeau M., Gagné C., & others (2012) DEAP: Evolutionary algorithms made easy. The Journal of Machine Learning Research, **13**(1), 2171–2175.

Goldberg D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Jenkins (2018) Jenkins. [online] https://jenkins.io/index.html (Zugegriffen 8. Februar 2018).

Kickmeier-Rust M. D., Mattheiss E., & Albert D. (2009) An Educational Guide to Planet Earth: Adaptation and Personalization in Immersive Educational Games. Proceedings of the 2nd International Workshop on Story-Telling and Educational Games, 36–45.

Laucelli D. B., Berardi L., & Simone A. (2018) A Teaching Experiment Using a Serious Game for WDNs Sizing. 13th International Conference on Hydroinformatics, **3**, 1104–1112.

Leaflet (2018) [online] https://leafletjs.com/ (Zugegriffen 8. Dezember 2018).

Leaflet EasyButton (2018) CliffCloud. [online] https://github.com/CliffCloud/Leaflet.EasyButton (Zugegriffen 8. Dezember 2018).

Liemberger R. & Marin P. (2006) The Challenge of Reducing Non-Revenue Water in Developing Countries--How the Private Sector Can Help: A Look at Performance-Based Service Contracting.

Meniconi S., Brunone B., van Zyl K., & Mazzetti E. (2017) Aqualibrium Competition: Laboratory Data and EPAnet Simulations. Procedia Engineering, **186**, 522–529.

Ostfeld A., Salomons E., Ormsbee L., Uber J., Bros C., Kalungi P., Burd R., Zazula-Coetzee B., Belrain T., Kang D., Lansey K., Shen H., McBean E., Yi Wu Z., Walski T., Alvisi S., Franchini M., Johnson J., Ghimire S., Barkdoll B., Koppel T., Vassiljev A., Kim J., Chung G., Yoo D., Diao K., Zhou Y., Li J., Liu Z., Chang K., Gao J., Qu S., Yuan Y., Prasad T., Laucelli D., Vamvakeridou Lyroudia L., Kapelan Z., Savic D., Berardi L., Barbaro G., Giustolisi O., Asadzadeh M., Tolson B., & McKillop R. (2011) Battle of the Water Calibration Networks. Journal of Water Resources Planning and Management, **138**(5), 523–532.

PEP 333 - Python Web Server Gateway Interface v1.0 (2018) Python.org. [online] https://www.python.org/dev/peps/pep-0333/ (Zugegriffen 1. Dezember 2018).

PostgreSQL (2018) [online] https://www.postgresql.org/ (Zugegriffen 8. Dezember 2018).

Poulakis Z., Valougeorgis D., & Papadimitriou C. (2003) Leakage detection in water pipe networks using a Bayesian probabilistic framework. Probabilistic Engineering Mechanics, **18**(4), 315–327.

Pudar R. & Liggett J. (1992) Leaks in Pipe Networks. Journal of Hydraulic Engineering, **118**(7), 1031–1046.

Rossman L. A. (2000) *EPANET 2: users manual*, Water Supply and Water Resources Division National Risk Management Research Laboratory - Environmental Protection Agency, Cincinnati, OH, USA. [online] http://ghalambaz.persiangig.com/Amoozeshi/P1007WWU.pdf (Zugegriffen 27. März 2014).

Savic D., Morley M., & Khoury M. (2016) Serious Gaming for Water Systems Planning and Management. Water, **8**(10), 456.

de Schaetzen W. B., Walters G., & Savic D. (2000) Optimal sampling design for model calibration using shortest path, genetic and entropy algorithms. Urban Water, **2**(2), 141–152.

Steffelbauer D. B. (2018) Model-Based Leak Localization in Water Distribution Systems.

Steffelbauer D. B. & Fuchs-Hanusch D. (2016) Efficient Sensor Placement for Leak Localization Considering Uncertainties. Water Resources Management, **30**(14), 5517–5533.

Storn R. (1996) „On the usage of differential evolution for function optimization“ in Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American., 519–523.

Storn R. & Price K. (1997) Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization, **11**(4), 341–359.

Anhang

Beispiel einer Konfigurationsdatei (config.toml) für OSP-Berechnungen mit Jenkins:

```
[SensitivityMatrix]
# the following parameters are only applicable if no sensitivity matrix was given as a file
input_parameter = 'demand'
output_parameter = 'pressure'
perturbation = 2

[ResidualMatrix]
# the following parameters are only applicable if no residual matrix was given as a file
input_parameter = 'demand'
output_parameter = 'pressure'
perturbation = 2

[SensorPlacement]
osp_type = 'Casillas'
a = []

[PositionFiltering]
#id = ['']
#demand = 50.0
#description = ['Feuerhydrant']
#tag = ['']

[ShortestPath1]
# no further input required

[ShortestPath2]
# no further input required

[Optimizer]
# only applicable for SOSP using Casillas or SPUDU
Type = 'DE'
Population = 50
Generations = 50
CR = 0.7
F = 0.5

[Casillas]
# no further input required

[SPUDU]
omega = 0.1
# the following parameters are only applicable if no SigmaP was given as a file
sd = 0.2
mcruns = 100

[Plotting]
robust = 'True'
colormap = 'viridis'

[PostgreSQL]
write = 'True'
```