Projektarbeit: Simulation & Visualisierung des Firefighter-Problems basierend auf OSM-Daten

AIMN AHMED, 3235005, Universität Stuttgart, Germany
SAMUEL HOLDERBACH, 3244630, Universität Stuttgart, Germany
DOMINIK KRENZ, 3236729, Universität Stuttgart, Germany

Das Firefighter-Problem befasst sich mit der strategischen Eindämmung eines Feuers auf einem Graphen. In der allgemeinen Definition des Problems bricht das Feuer auf einem oder mehreren Knoten aus und breitet sich dann in jedem Zeitschritt auf alle benachbarten und unbeschützten Knoten aus. Ziel ist es, über sogenannte Firefighter, Knoten zu beschützen und das Feuer aufzuhalten. Der Fokus dieser Arbeit liegt auf der Simulation und Visualisierung des Firefighter-Problems auf Graphen von realen Straßennetzwerken. Die Kartendaten hierfür entstammen dem freien Geodaten-Service OpenStreetMap [1]. Unter anderem Stellen wir in diesem Projekt verschiedene Strategien zur Platzierung der Firefighter vor und untersuchen diese auf ihre Effizienz und Wirksamkeit. Darüber hinaus stellen wir einen von uns entwickelten Webservice [2] vor, der das Firefighter-Problem mit den vorgestellten Strategien auf beliebigen Straßengraphen simulieren und das Ergebnis anschließend darstellen kann.

Additional Key Words and Phrases: Firefighter Problem, OpenStreetMap, Dijkstra, Contraction Hiearchies, Hub Labels

ACM Reference Format:
Aimn Ahmed, 3235005, Samuel Holderbach, 3244630, and Dominik Krenz, 3236729. 2022. Projektarbeit: Simulation & Visualisierung des Firefighter-Problems basierend auf OSM-Daten. 1, 1 (January 2022), 18 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 EINLEITUNG

Regelmäßig kommt es rund um den Globus zu Bränden, mit denen die lokalen Behörden an ihre Grenzen stoßen. Seien es die Waldbrände in Kalifornien oder Italien oder Buschbrände in Australien, die jeweilige Region gerät stets in einen Notstand, falls das Feuer sich zu weit ausbreitet. Um dies zu verhindern, muss das Feuer strategisch eingedämmt werden. In der Praxis gibt es hier diverse Methoden, um den Verlauf des Feuers vorherzusagen und dagegen vorzugehen.

Ein Versuch, das Szenario eines Flächenbrandes auf Graphen zu abstrahieren, ist das Firefighter-Problem, welches 1995 von Bert Hartnell vorgestellt wurde [5]. Bei der allgemeinen Definition des Problems bricht zum Zeitpunkt $t = 0$ ein Feuer an einem oder mehreren Knoten eines beliebigen ungerichteten Graphen aus. In jedem weiteren Zeitschritt $t = k, k \in \mathbb{N}$ kann dann eine feste Anzahl von unbeschützten und nicht verbrannten Knoten durch sogenannte Firefighter beschützt werden. Anschließend breitet sich das Feuer auf alle unbeschützten Knoten aus, die zu brennenden Knoten adjazent sind. Durch Firefighter beschützte Knoten können hingegen nicht mehr verbrannt werden. Die Prozedur wird solange forgeführt, bis sich das Feuer nicht weiter ausbreiten kann [5].

Authors' addresses: Aimn Ahmed, 3235005, st176364@stud.uni-stuttgart.de, Universität Stuttgart, Stuttgart, Germany; Samuel Holderbach, 3244630, st152101@stud.uni-stuttgart.de, Universität Stuttgart, Stuttgart, Germany; Dominik Krenz, 3236729, st150654@stud.uni-stuttgart.de, Universität Stuttgart, Stuttgart, Germany.

Aus der Definition des Firefighter-Problems lassen sich verschiedene Optimierungsziele für Lösungsstrategien ableiten [5]:

(1) Die zu erwartende Anzahl an brennenden Knoten soll minimiert werden.
(2) Die Anzahl an direkt oder indirekt beschützten Knoten soll maximiert werden.
(3) Die Anzahl an Zeitschritten zwischen Ausbruch und Eindämmung des Feuers soll minimiert werden (alternativ: Die Anzahl an beschützten Knoten soll minimiert werden).
(4) Bestimmen einer Knotenmenge, die als Separator dient, d. h. die, wenn sie beschützt wird, die zu erwartende Menge brennender Knoten von den restlichen Knoten (oder einer gegebenen Knotenmenge) separiert.

Von direkt beschützten Knoten sprechen wir im Nachfolgenden dann, wenn diese durch einen Firefighter dauerhaft beschützt wurden. Als «indirekt beschützt» bezeichnen wir Knoten, welche durch das Feuer nicht mehr erreicht werden können. Dies ist beispielsweise dann der Fall, wenn eine Knotenmenge durch eine Menge von direkt beschützten Knoten vom Rest des Graphen separiert wurde.

Bisherige Arbeiten haben das Firefighter-Problem bereits auf diversen Graphklassen, wie zum Beispiel Bäumen oder Gittergraphen untersucht und stellen verschiedene Lösungsansätze für diese vor, die wiederum unterschiedliche Optimierungsziele verfolgen.

Bei einem Baum, bei dem zum Zeitpunkt $t = 0$ ein Feuer am Wurzelknoten ausbricht, kann man Knoten beispielsweise mit einer Gewichtsfunktion versehen, die jedem Knoten die Anzahl von Knoten zuordnet, die indirekt beschützt würden, wenn ein Firefighter eben diesen Knoten beschützen würde. Ein natürlicher Ansatz, um auf so einem Baum mit einem Firefighter das Feuer einzudämmen und die Anzahl an direkt oder indirekt beschützten Knoten zu maximieren, ist der Greedy-Ansatz. Hierbei würde der Firefighter zu jedem Zeitpunkt stets den Knoten mit dem höchsten Gewicht beschützen. Laut Finbow und MacGillivray [5] liefert der Greedy-Ansatz auf Bäumen, bei denen das Feuer an der Wurzel ausbricht, immer eine Lösung, die mindestens halb so viele Knoten direkt oder indirekt beschützt, wie die optimale Lösung. fig. 1 zeigt zum Beispiel einen Baum, bei dem mit dem Greedy-Ansatz, wie oben beschrieben, 5 Knoten beschützt würden. Die optimale Lösung hingegen beschützt 6 Knoten.

Die Frage, ob ein Feuer mit einer festen Anzahl d von Firefightern eingedämmt werden kann und die Frage nach der minimalen Anzahl von verbrannten Knoten, ist beispielsweise für unendliche Gittergraphen interessant. Sei r die Quelle eines Feuers, das zum Zeitpunkt $t = 0$ auf einem unendlichen quadratischen Gittergraphen G ausbricht. Sei $k \in \mathbb{N}$, V_k die Menge aller Knoten von G mit Abstand k zu r und $B_k \subseteq V_k$ die Menge der Knoten, die zum Zeitpunkt $t = k$ brennen. Sei $N(\cdot)$ die Nachbarfunktion auf V_G. Dann genügen d Firefighter um das Feuer einzudämmen, wenn ein k existiert, sodass $|N(B_k) \cap V_{k+1}| \leq d$. Das bedeutet, dass der «Rand» des Feuers höchstens so viele unbeschützte Nachbarn haben darf, wie in der nächsten Runde Knoten durch Firefighter beschützt werden können [5]. In fig. 2 ist die optimale Strategie auf einem unendlichen quadratischen Gittergraphen mit zwei Firefightern skizziert. Das Feuer ist bei dieser Strategie nach 8 Runden eingedämmt und es brennen insgesamt 18 Knoten. Die Feuerquelle ist durch einen Kreis markiert; verbrannte Knoten sind durchgestrichen und beschützte Knoten sind mit einer Raute gekennzeichnet. Die Zahl neben der Raute bezieht sich auf den Zeitpunkt, zu dem der jeweilige Knoten beschützt wurde.

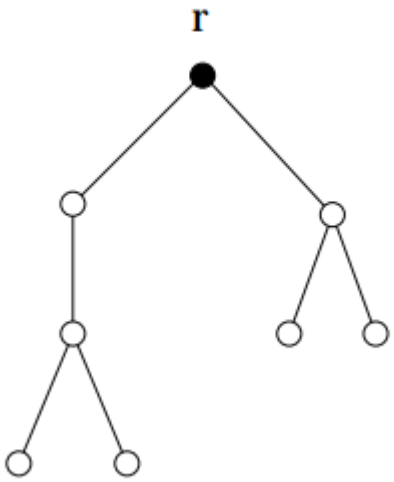


Fig. 1. Beispiel für einen Baum, auf dem der Greedy-Ansatz nicht die optimale Lösung liefert [5]

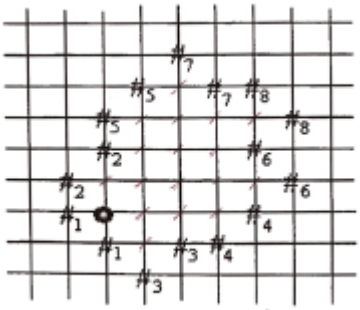


Fig. 2. Optimale Strategie auf einem unendlichen quadratischen Gittergraphen mit $d = 2$ [4]

In diesem Projekt werden wir das Firefighter-Problem im speziellen auf Graphen von realen Straßennetzwerken untersuchen. In section 2 geben wir zunächst einen kurzen Überblick über bisherige Arbeiten, die das allgemeine Firefighter-Problem oder das Firefighter-Problem für diverse Graphklassen untersucht haben. Anschließend erklären wir in section 3 die Motivation für dieses Projekt. section 4 erörtert dann verschiedene Lösungsstrategien für das Firefighter-Problem auf Straßengraphen. Dabei gehen wir zunächst auf die Idee hinter jeder Strategie ein, bevor wir dann deren Umsetzung im Detail erklären. Den Webservice, den wir zur Simulation und Evaluation unserer Strategien entwickelt haben, stellen wir in section 5 vor, bevor wir in section 6 die Ergebnisse der verschiedenen Lösungsansätze sammeln und diskutieren. Die Arbeit wird im Anschluss noch in section 7 durch ein Fazit abgerundet, in dem wir aus dem Projekt gewonnene Kenntnisse zusammenfassen und in einem kurzen Ausblick aufarbeiten.

2 VERWANDTE ARBEITEN

Das Firefighter-Problem wurde bereits auf unterschiedlichen Graphklassen, wie zum Beispiel Bäumen oder Gittergraphen und mit einer Vielzahl von zusätzlichen Restriktionen oder abgewandelten Definitionen untersucht. Einen guten Überblick über den aktuellen Stand der Forschung geben Finbow und MacGillivray [5] in ihrer Arbeit aus dem Jahr 2009.

Unser Projekt setzt sich jedoch ausschließlich mit dem Firefighter-Problem auf Straßengraphen auseinander. Aus diesem Grund beziehen wir uns in den folgenden Kapiteln unserer Arbeit ausschließlich auf unsere eigenen Forschungsergebnisse und verwenden keine Ergebnisse aus anderen Forschungsarbeiten.

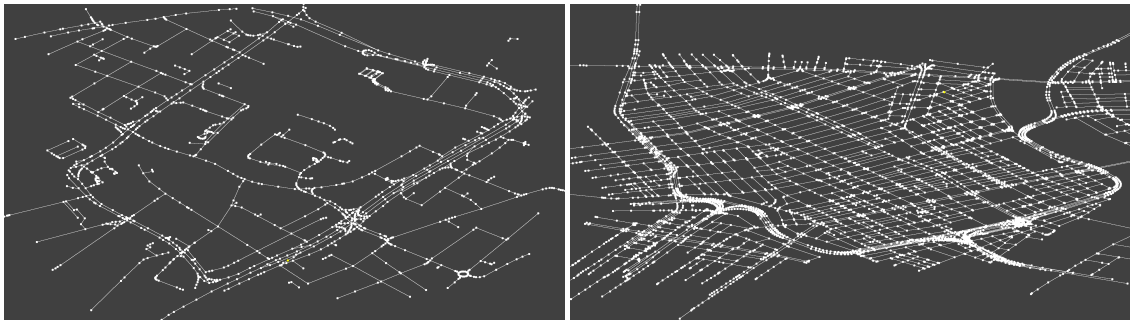
3 MOTIVATION

Das Firefighter-Problem lässt sich auf zahlreiche Szenarien in der Realität übertragen. So gewinnt das Problem unter anderem durch die derzeitige Pandemie-Situation, welche durch das Virus SARS-CoV-2 hervorgerufen wurde, zusätzlich an Relevanz. Ein Graph kann in diesem Fall ein soziologisches Netz darstellen, in welchem die Personen durch Knoten und bestehende Kontakte zwischen den Personen durch Kanten abgebildet werden. Statt von brennenden Knoten sprechen wir in diesem Fall von infizierten Knoten und beschützte Knoten bezeichnen wir als «geimpft». Zudem könnte man die Kanten mit der Inkubationszeit des Virus in Tagen oder Stunden als Gewicht versehen, wodurch man eine leicht veränderte Definition des Firefighter-Problems erhielte, bei der sich das Virus nur alle x Tage oder Stunden auf benachbarte, ungeimpfte Knoten überträgt.

Für dieses Projekt haben wir uns, aufgrund der vorhandenen Software-Basis, jedoch dazu entschieden, Graphen von Straßennetzwerken zu benutzen, um darauf das Firefighter-Problem zu simulieren. Solche Graphen weisen eine ähnliche Komplexität, wie Graphen von soziologischen Netzwerken auf, wodurch sich Lösungsstrategien eventuell übertragen ließen. Straßengraphen sind im Allgemeinen gerichtete Graphen, da Hin- und Rückrichtung einer Straße nicht immer die selbe Distanz haben müssen. Im Falle eines Feuers spielt die Richtung einer Kante jedoch keine Rolle, da das Feuer stets den kürzesten verfügbaren Weg wählen würde, weshalb wir im Folgenden weiterhin von ungerichteten Graphen ausgehen.

Das Firefighter-Problem auf Straßennetzwerken definiert sich demnach wie folgt: Ein Feuer bricht, genau wie in der allgemeinen Definition, zum Zeitpunkt $t = 0$ an einem oder mehreren Knoten des Graphen aus. Alle $m \in \mathbb{N}$ Zeitschritte, d. h. zu den Zeitpunkten $t = m, t = 2m, t = 3m$, etc. kann eine feste Anzahl von Knoten durch Firefighter beschützt werden. Eine Kante e sei definiert als Tupel $e = (\{a, b\}, w)$, bestehend aus der Menge der zu der Kante inzidenten Knoten $\{a, b\}$ und ihrem Gewicht $w \in \mathbb{N}$. Dann breitet sich das Feuer im Allgemeinen mit der Geschwindigkeit $\frac{1}{w}$ aus. Ein bis dahin unbeschützter Knoten, der durch eine Kante mit Gewicht w mit einem zum Zeitpunkt $t = k, k \in \mathbb{N}$ brennenden Knoten verbunden ist, brennt also zum Zeitpunkt $t = k + w$. Auch auf Straßengraphen endet der Prozess, sobald sich das Feuer nicht mehr weiter ausbreiten kann. Die Optimierungsziele für Lösungsstrategien bleiben im Gegensatz zur allgemeinen Definition des Problems unverändert.

Wie in section 1 beschrieben, beziehen sich viele Lösungsansätze für das Firefighter-Problem auf ein bestimmtes Muster von Graphen, beispielsweise Gittergraphen oder Baumstrukturen. Graphen von Straßennetzwerken lassen sich jedoch nur schwer auf ein bestimmtes Muster abstrahieren. Am ehesten ließen sie sich noch den planaren Graphen zuordnen. Jedoch gibt es auch hier schon Ausnahmen, zum Beispiel bei Unterführungen. Hinzu kommt, dass sich Straßengraphen untereinander oft stark unterscheiden. So unterscheiden sich beispielsweise dicht besiedelte



(a) Stuttgarter Zentrum (b) Ausschnitt aus dem Stadtzentrum Tokios

Fig. 3. Visueller Vergleich zwei verschiedener Innenstädte

Innenstädte von ländlicheren Gebieten in ihrer Kantendichte. Selbst Zentren von verschiedenen Städten weisen oft zu große Unterschiede auf, als dass man sie einer bestimmten Klasse von Graphen oder einem Muster zuordnen könnte.

Aus diesem Grund haben wir verschiedene Strategien zur Eindämmung des Feuers auf Graphen von Straßennetzwerken implementiert und getestet. Bei der Entwicklung der Strategien lag unser Fokus darauf, die Ausbreitung des Feuers so wirksam wie möglich zu verhindern, d. h. die Anzahl der verbrannten Knoten zu minimieren. Ein Nebenziel war es, dabei so effizient wie möglich vorzugehen, d. h. die Anzahl an Zeitschritten zwischen Ausbruch und Eindämmung des Feuers so gering wie möglich zu halten.

4 STRATEGIEN ZUR EINDÄMMUNG DES FEUERS

In diesem Abschnitt stellen wir die verschiedenen von uns implementierten Strategien zur Platzierung der Firefighter bei der Simulation des Firefighter-Problems auf Straßengraphen vor. Zu jeder Strategie erklären wir zunächst die zugrundeliegende Idee und gehen anschließend auf ihre praktische Umsetzung ein. Wir definieren dafür k als die Anzahl an Zeitschritten, die zwischen zwei aufeinanderfolgenden Platzierungsrunden liegt und n als die Anzahl an Firefightern, die in jeder Platzierungsrunde zur Verfügung stehen.

4.1 Random

Die Zufallsstrategie dient lediglich als Vergleichsstrategie zur Auswertung der anderen Strategien. Es werden bei der Zufallsstrategie alle k Runden n Knoten zufällig aus der Menge aller Knoten ausgewählt, die weder beschützt noch verbrannt sind. Diese Knoten werden anschließend durch einen Firefighter beschützt. Das Ergebnis einer Ausführung der Strategie ist demnach ebenfalls zufällig.

4.2 Greedy

Die Greedy-Strategie zeichnet sich dadurch aus, dass sie alle k Runden n Knoten mit einem Firefighter beschützt, die zum Zeitpunkt der Auswahl das beste Ergebnis versprechen. Alle k Runden iteriert der Algorithmus über alle brennenden Knoten. Dabei werden alle Kanten in einer Datenstruktur gespeichert, die von einem brennenden Knoten zu einem unbeschützten Knoten führen. Im nächsten Schritt wird diese Datenstruktur in aufsteigender Reihenfolge nach Gewicht der Kanten sortiert. Sollten mehrere Kanten dasselbe Gewicht haben, so wird nach Grad des unbeschützten Ziel-Knotens unterschieden. Knoten mit einem höheren Grad werden vom Greedy-Algorithmus priorisiert, da sie eine

höhere Ausbreitungsgefahr des Feuers darstellen. Sie kommen daher in der Datenstruktur früher in der Reihenfolge vor. Im letzten Schritt werden so viele unbeschützte Ziel-Knoten wie möglich aus der Datenstruktur beschützt. Sollte n größer oder gleich der Anzahl der Elemente in der Datenstruktur sein, so werden alle unbeschützte Ziel-Knoten aus der Datenstruktur mit einem Firefighter beschützt. Der Algorithmus terminiert, sobald dieser Fall eintritt, da keine Kanten von einem brennenden Knoten zu einem unbeschützten Knoten mehr existieren. Das Feuer ist demzufolge eingedämmt und kann sich nicht weiter ausbreiten.

4.3 Minimum-Distance-Sets-Strategien

Die folgenden Strategien platzieren, im Gegensatz zur Greedy-Strategie, ihre Firefighter nicht sobald diese zur Verfügung stehen. Stattdessen wird eine Knotenmenge bestimmt, bei der das Feuer zum Großteil bzw. komplett umschlossen werden kann. Dadurch soll gewährleistet werden, dass die Ausbreitung des Feuers mit einer größeren Wahrscheinlichkeit gestoppt wird, da man nun mehrere Teile des Graphen betrachtet, während man bei der Greedy-Strategie nur die direkten Kanten zum Feuer betrachtet. Um dies umzusetzen, berechnen diese Strategien sogenannte *Minimum-Distance-Sets*. Dabei werden Knotenmengen nach ihrer minimalen Distanz zu einer beliebigen Feuerquelle gruppiert. Diese *Minimum-Distance-Sets* werden nun verwendet, um die Knotenmengen zu bestimmen, die das Feuer am besten eingrenzen und damit die Ausbreitung stoppen.

Berechnet werden die *Minimum-Distance-Sets*, indem man von jeder Feuerquelle aus einen *One-To-All-Dijkstra* ausführt. Anschließend speichert man für jeden Knoten lediglich die kürzeste Distanz aus allen *One-To-All-Dijkstras* ab. Auf diese Weise hat man am Ende eine Datenstruktur, die alle Knoten nach ihrer kürzesten Distanz zu einer beliebigen Feuerquelle gruppiert.

Die ursprüngliche Idee, aus der die folgenden drei Strategien entstanden sind, basiert auf der optimalen Strategie auf einem unendlichen quadratischen Gittergraphen, welche bereits in section 1 aufgegriffen und in fig. 2 skizziert wurde. Sei v ein beliebiger Knoten und d die Distanzfunktion, welche die minimale Distanz von einem Knoten zu einer beliebigen Feuerquelle berechnet. Man betrachtet nun für jede gegebene Distanz $s_0, s_1, ..., s_n \in \mathbb{N}_0$ die Anzahl an Knoten, die sich im *Minimum-Distance-Set* befinden. Da man alle k Zeitschritte n weitere Firefighter zur Verfügung hat, vergleicht man für jede dieser Distanzen die Anzahl der zu Verfügung stehenden Firefighter, mit der Anzahl an Knoten im zugehörigen *Minimum-Distance-Set*. Sollte dabei die Anzahl der Firefighter größer gleich der Anzahl der Knoten sein, platziert man die Firefighter auf genau diese Knoten, da somit die Ausbreitung des Feuer eingedämmt und gestoppt wird.

Um diese Strategie auf Graphen von Straßennetzwerken umzusetzen, reicht das Gruppieren der Knoten nach ihrer minimalen Distanz zum Feuer allerdings nicht aus. Dies liegt daran, dass für eine gegebene Distanz s und Knoten u und w mit $d(u) < s$ und $d(w) > s$ sowie einem Pfad $p := (u, v_0, v_1, ..., w)$ zwischen u und w nicht mehr zwingend gegeben ist, dass ein v_i mit $d(v_i) = s$ auf p liegt. Aufgrund des Kantengewichts können Pfade zwischen Knoten also bestimmte Distanzmengen überbrücken. Durch das Beschützen einer Distanzmenge wird dem Feuer also nicht mehr der Weg zu Knoten mit einer höheren Distanz abgeschnitten. Aufgrund dessen haben wir uns verschiedene Strategien überlegt, die auf der Idee von *Minimum-Distance-Sets* basieren, aber die eben erläuterte Problematik umgehen.

4.3.1 Multiple-Minimum-Distance-Sets. Wie bereits geklärt wurde, ist die ursprüngliche Idee des Algorithmus auf Straßennetzwerken nicht optimal anwendbar, da durch das Beschützen eines *Minimum-Distance-Set* das Feuer nicht garantiert eingedämmt wird. Es werden zunächst die *Minimum-Distance-Sets* berechnet. Im Anschluss werden für alle Distanzen $s_0, s_1, ..., s_n \in \mathbb{N}_0$ die Anzahl an Knoten, die sich im *Minimum-Distance-Set* befinden, mit der Anzahl der

Firefighter die zur Verfügung stehen verglichen. Hierbei werden die Distanzen in aufsteigender Reihenfolge betrachtet. Da eine Distanzeinheit auch gleichzeitig einen Zeitschritt widerspiegelt, lässt sich anhand der zurückgelegten Distanz auch bestimmen wie viele Firefighter aktuell zur Verfügung stehen. Sollte die Anzahl der Firefighter höher oder gleich sein, so werden alle Knoten aus dieser Distanz mit einem Firefighter beschützt. Der Algorithmus terminiert an dieser Stelle noch nicht, da wie bereits erkannt, auch Knoten außerhalb dieser Distanz noch erreicht werden können, da die Kanten gewichtet sind. Es werden solange *Minimum-Distance-Sets* ermittelt, die komplett beschützt werden können, bis entweder das Feuer eingedämmt wurde oder es keine Knoten mehr gibt, die vom Feuer erreicht werden können.

Ein Problem des Algorithmus ist, dass auf diese Weise Feuerquellen weiterhin beschützt werden, obwohl diese bereits eingedämmt sein könnten. Dies liegt daran, dass die *Minimum-Distance-Sets* nur einmalig initial bestimmt werden. Hierbei werden alle Feuerquellen betrachtet. Sobald der Algorithmus nach einer oder mehreren Platzierungsrunden jedoch eine Feuerquelle komplett eingedämmt hat, so werden weiterhin kürzeste Distanzen von dieser Feuerquelle in Betracht gezogen. Eine Optimierung ist es demnach, dass die *Minimum-Distance-Sets* genau dann neu berechnet werden sollten, sobald festgestellt wird, dass eine Feuerquelle sich nicht mehr ausbreiten kann, da sie von Firefightern umkreist wird. Dadurch fallen alle kürzesten Distanzen von dieser Feuerquelle weg, wodurch es nicht mehr passieren kann, dass eine bereits eingedämmte Feuerquelle weiterhin versucht wird zu beschützen. Dass eine Feuerquelle eingedämmt wurde, kann wie folgt festgestellt werden: Alle k Schritte, wenn Firefighter platziert werden dürfen, wird am Ende der Runde überprüft, ob eine Feuerquelle eingedämmt wurde. Dabei wird für jede Feuerquelle eine Datenstruktur angelegt, wobei sich initial die Feuerquelle selbst darin befindet. Diese Datenstruktur stellt die potenzielle Ausbreitungsgefahr der Feuerquelle dar. Wir bezeichnen sie fortan als *risky nodes*. Sie enthält alle nicht brennenden und unbeschützten Knoten, die unmittelbar eine Kante von einem brennenden Knoten entfernt sind und damit ausgehend von dieser Feuerquelle noch erreicht werden können. Da sich der Status der Knoten alle k Schritte ändern kann, werden zunächst alle Knoten aus der Datenstruktur gefiltert, die brennen oder beschützt sind. Beschützte fallen dabei weg, da die Ausbreitung des Feuers an der Stelle endet. Brennende Knoten werden nochmals einzeln betrachtet. Sie werden in einer anderen Datenstruktur gespeichert und so lange abgearbeitet, bis die Datenstruktur leer ist. Wir bezeichnen diese fortan als *burning nodes*. Folgende Schritte werden dabei wiederholt: Man entfernt das erste Element, schaut sich dessen ausgehenden Kanten an und überprüft den Status der Zielknoten. Sollte der Zielknoten weder brennen noch beschützt sein, so wird er in *risky nodes* eingefügt. Sollte der Zielknoten brennen, so wird noch überprüft, ob er bereits besucht wurde. Denn die Methode merkt sich für jede Feuerquelle, ob ein Knoten bereits besucht wurde, da sie sonst in einer Endlosschleife landen kann, falls ein brennender Kreis existiert. Sollte der Zielknoten also noch nicht besucht worden sein und brennen, so wird er in *burning nodes* eingefügt. Zielknoten die beschützt sind werden nicht weiter betrachtet und entfernt. Sobald *burning nodes* leer ist und man für jede Feuerquelle diese Schritte durchgeführt hat, wird überprüft, bei welchen Feuerquellen *risky nodes* leer ist. Sollte *risky nodes* leer sein, existiert von dieser Feuerquelle kein Pfad mehr, der zu einem unbeschützten und nicht brennenden Knoten führt. Die Feuerquelle ist damit eingedämmt. Sollte dies bei mindestens einer Feuerquelle der Fall sein, so können die *Minimum-Distance-Sets* neu berechnet werden, wobei nur Feuerquellen berücksichtigt werden, deren *risky nodes* nicht leer ist.

In fig. 4 ist das Ergebnis einer Simulation auf einem Beispielgraphen dargestellt.

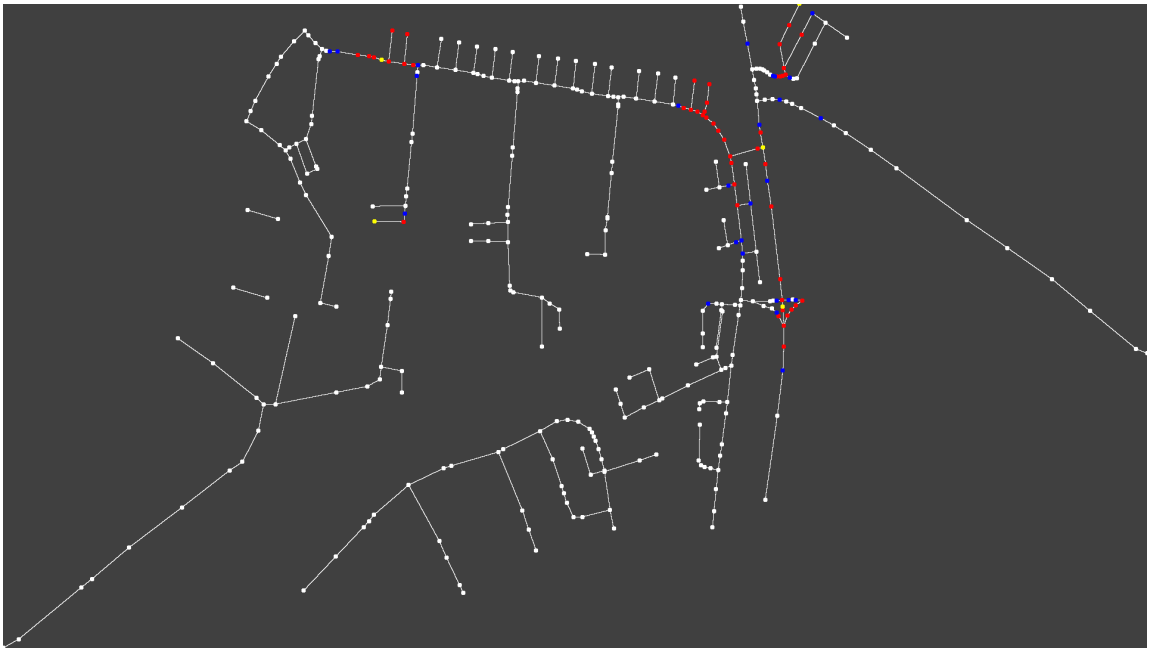


Fig. 4. Ergebnis der Simulation der *Multi-Minimum-Distance-Sets* Strategie auf dem Graphen *bbgrund_undirected.fmi* mit 350 Knoten und 706 Kanten (353 logischen Kanten)

Die initiale Berechnung der *Minimum-Distance-Sets* sowie die Überprüfung auf eingedämmte Feuerquellen und die damit verbundene Neuberechnung wird bei allen Varianten gleich angewendet.

4.3.2 Single-Minimum-Distance-Set. Diese Variante der *Minimum-Distance-Sets* Strategie orientiert sich an der ursprünglichen Idee, für ungewichtete Graphen eine Menge von Knoten zu finden, die weit genug von der bzw. den Feuerquellen entfernt liegen, um vollständig beschützt zu werden. Diese Knotenmenge sollte das Feuer komplett umschließen, sodass Knoten mit einer höheren Distanz zu der bzw. den Feuerquellen nicht mehr brennen können, also indirekt beschützt sind.

Eine Möglichkeit, um diese Strategie dennoch auf einem gewichteten Straßengraphen zu implementieren, wäre es den gewichteten Graphen künstlich aufzublasen und somit in einen ungewichteten Graphen zu transformieren. Dazu müssten Kanten durch Zwichenknoten so aufgeteilt werden, dass ein neuer Graph entsteht, bei dem jede Kante das Gewicht 1 hätte. Dann könnte man die Distanzmengen für diesen neuen Graphen mit einer einfachen Breitensuche berechnen und die Distanzmenge mit der geringsten Distanz beschützen, die vollständig durch Firefighter abgedeckt werden kann, bevor sie durch das Feuer erreicht würde. Ursprüngliche Knoten könnten direkt beschützt werden. Für Zwischenknoten müsste jeweils der zu der ursprünglichen Kante benachbarte Knoten gespeichert werden, der die höhere Distanz zum Feuer hat. Statt dem Zwischenknoten würde dann dieser Knoten beschützt werden.

Das Aufblasen des Graphen bringt jedoch zwei Probleme mit sich. Zum einen ist dieser Ansatz nicht speicher-effizient, da für einen Graphen $G = (V, E)$ mit $m := |E|$ und durchschnittlicher Kantendistanz $\bar{d}$ zusätzlicher Speicherplatz für $m \cdot (\bar{d} - 1)$ Knoten und Kanten benötigt wird. Zum anderen wäre eine naive Breitensuche einer *One-to-all-Dijkstra* Suche mit einem *Min-Heap* als Priority Queue, nicht zuletzt wegen der erhöhten Anzahl von Kanten, deutlich unterlegen.

Bei genauerer Betrachtung des Problems wurde uns jedoch klar, dass das Aufblasen des Graphen lediglich als symbolischer Schritt zu betrachten ist. Die Distanzmengen können also ganz normal mittels einer *One-to-all-Dijkstra* Suche ausgehend von den Feuerquellen berechnet werden. Anschließend muss für jeden Knoten sein Nachbar mit der geringsten ermittelten Distanz zum Feuer bestimmt werden. Gegeben sei ein Knoten v mit Distanz $d(v)$. Dann wird v in die Distanzmengen im Intervall $[d_{min}(N(v)) + 1, d(v)]$ eingefügt.

Bei der Variante des Algorithmus, bei der wir Kanten mit Zwischenknoten erweitern, hätten wir jeweils entweder v selbst oder einen Zwischenknoten, der auf v referenziert, in genau die selben Distanzmengen eingefügt. Dies ist der Fall, da Zwischenknoten immer auf einen Knoten mit höherer Distanz referenzieren und da für jeden Knoten, der vom Feuer erreicht werden kann, mindestens ein Nachbar mit geringerer Distanz zum Feuer existieren muss (ansonsten würde kein Pfad vom Feuer zu diesem Knoten existieren). Dieser Sachverhalt wird in fig. 5 nochmals anschaulich dargestellt. Der gelbe Knoten markiert hier die Feuerquelle. u ist der Nachbar von v mit dem geringsten Abstand zum Feuer, w hingegen liegt weiter entfernt.

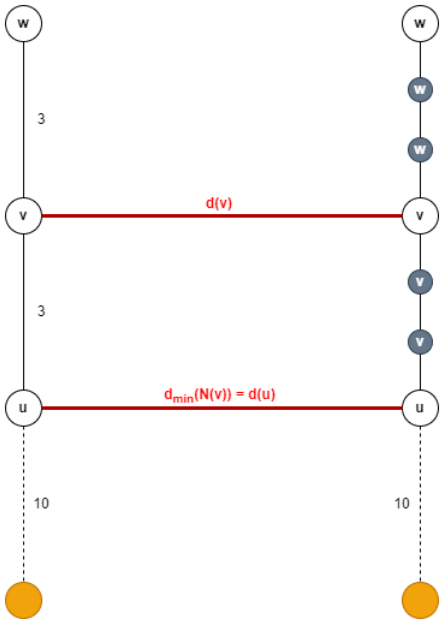


Fig. 5. Distanzmengen-Intervall eines Knotens v für die Variante ohne (links) bzw. mit Aufblasen des Graphen (rechts).

Nach der Berechnung der Distanzmengen wählt der Algorithmus, wie auch bei der ersten Variante, die Menge mit der geringsten Distanz zum Feuer aus, die vollständig beschützt werden kann. Anschließend startet die Simulation des Firefighter-Problems; das Feuer breitet sich aus und im Gegenzug werden die Knoten aus der gewählten Menge durch Firefighter beschützt. Sobald alle Knoten aus der Menge beschützt wurden, ist das Feuer eingekreist und breitet sich nur noch so lange aus, bis es den inneren Rand des Kreises erreicht. Das Feuer ist somit eingedämmt.

fig. 6 zeigt das Ergebnis der Simulation der Strategie auf einem Beispielgraphen.

Fig. 6. Ergebnis der Simulation der *Single-Minimum-Distance-Set* Strategie auf dem Graphen *bbgrund_undirected.fmi*

4.3.3 Priority-Minimum-Distance-Sets. Die Idee für diese Variante kam auf, als uns beim Testen aufgefallen ist, dass oft Knoten beschützt wurden, die in einer Sackgasse enden, während Knoten mit einer höheren Ausbreitungsgefahr verbrannt sind. Aufgrund dessen kam die Idee, dass jedem Knoten ein sogenannter Prioritätswert zugewiesen wird. Anhand dieses Werts lässt sich feststellen, wie gefährlich es wäre, wenn dieser Knoten brennen würde. Da Sackgassen oder auch Straßen ohne Kreuzungen weniger gefährlich sind, als Kreuzungen oder ähnliches, sollten die Prioritätswerte genau das widerspiegeln. Der Prioritätswert eines Knotens entspricht dem Grad des Knotens. Dadurch haben Sackgassen einen niedrigen Wert, während er bei Kreuzungen höher ist. Anfangs haben wir ebenfalls das Gewicht g der Kanten in die Berechnung miteinbezogen. Hierbei berechnete sich der Prioritätswert eines Knotens mit Grad n und Kantengewicht g_i der i-ten Kante mit der Formel $\sum_{i=0}^{n-1} \frac{1}{g_i}$, wobei $g_i \in \mathbb{N}$. Grund dafür war, dass Knoten einem niedrigeren Grad genau dann einen höheren Prioritätswert haben sollten, wenn die Gewichte der Kanten sichtlich niedriger sind, da der Zielknoten schneller vom Feuer erreicht wird. Diese Formel stellte sich jedoch später auf Straßengraphen als nicht effektiv heraus. Stattdessen beschränkten wir uns auf den Grad des Knotens.

Wie auch bei der ersten Variante berechnet man bei dieser ebenfalls die *Minimum-Distance-Sets* und überprüft ebenfalls alle k Schritte, ob es eingedämmte Feuerquellen gibt. Im Anschluss berechnet man für alle Knoten die Prioritätswerte. Nachdem diese berechnet wurden, sortiert man sie in aufsteigender Reihenfolge, um das 25-%-Quantil, welches wir fortan als $q_{0.25}$ bezeichnen, bestimmen zu können. Nun berechnet man zwei verschiedene *Minimum-Distance-Sets*: Zum einen ein *high prio nodes* und ein *low high prio nodes*. Diese sind wie die *Minimum-Distance-Sets* aufgebaut. Sie unterscheiden sich darin, dass die Knoten im *high prio nodes* einen Prioritätswert haben der $\geq q_{0.25}$ ist, während er im *low prio nodes* $\leq q_{0.25}$ ist. Zuerst iteriert man über die *high prio nodes*. Hierbei werden der Reihe die Knoten von kleiner nach großer Distanz betrachtet. Alle Knoten die man dabei beschützen kann, werden beschützt, da

man zu jedem Zeitpunkt weiß wie viele Firefighter zur Verfügung stehen. Nachdem alle *high prio nodes* abgearbeitet wurden, schaut sich der Algorithmus die *low prio nodes* an und beschützt nach dem selben Vorgehen so viele Knoten wie möglich, angefangen bei der kleinsten bis zur größten Distanz. Nachdem er alle berechneten Knoten beschützt hat, terminiert er wie jeder Algorithmus, sobald kein weiterer Knoten mehr vom Feuer erreicht werden kann.

Anfangs verwendeten wir den Median, um die *Minimum-Distance-Sets* zu separieren. Dieser stellte sich jedoch als nicht sinnvoll für Straßengraphen heraus, da viele Knoten einen ähnlichen Grad haben und der höchste Grad nicht all zu hoch ist. Dadurch wurden viele Knoten beschützt, die weit entfernt von den Feuerquellen und daher nicht von Bedeutung waren.

fig. 7 stellt erneut das Ergebnis einer Beispiel-Simulation dar.

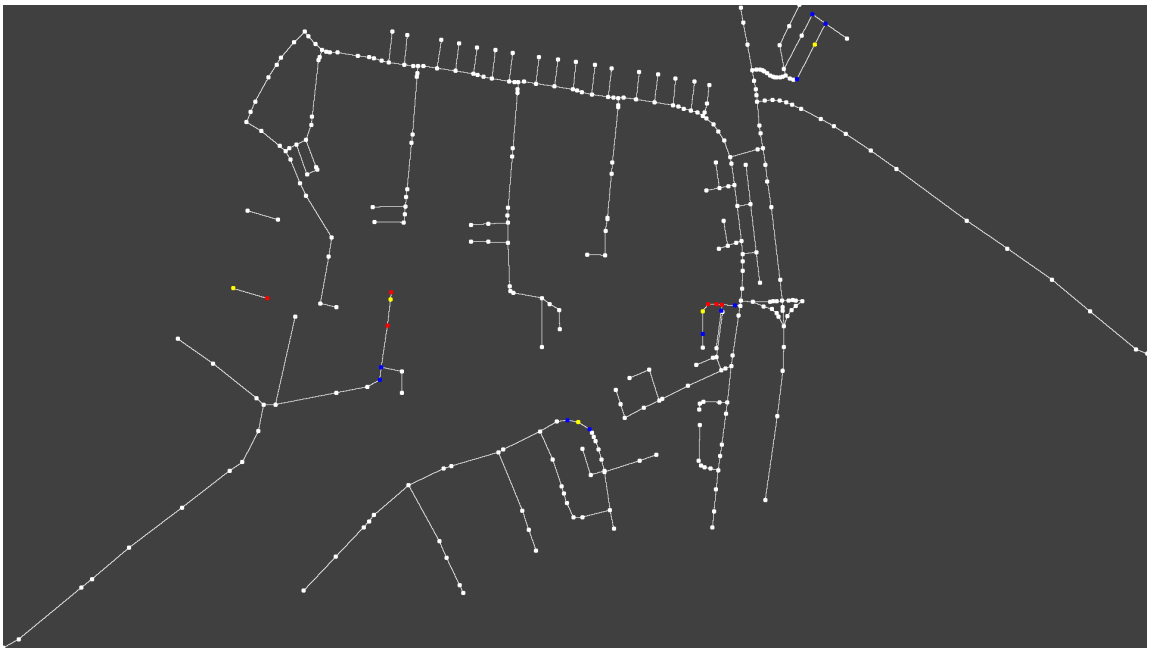


Fig. 7. Ergebnis der Simulation der *Priority-Minimum-Distance-Sets* Strategie auf dem Graphen *bbgrund_undirected.fmi*

5 WEBSERVICE ZUR SIMULATION DER FIREFIGHTER-STRATEGIEN

Ziel des Projekts war es eine Anwendung zu entwickeln, welche das Firefighter-Problem simulieren und visualisieren soll. Das Projekt wird als Web-Anwendung gestartet, die, wenn der Server hochgefahren ist, von einem beliebigen Internet-Browser benutzt werden kann. Hierzu besteht die Anwendung aus einer Frontend und einer Backend Komponente. Bei dem Frontend handelt es sich um ein Angular Projekt. Das Backend wurde in der Programmiersprache Rust entwickelt.

5.1 Konfigurieren einer Simulation

Sind sowohl die Backend-Komponente, als auch die Frontend-Komponente hochgefahren, kann die Anwendung über den Browser geöffnet werden. Zu Beginn ist noch keine Karte zu sehen, da noch keine Simulation gestartet wurde. Wird jedoch auf den Button «Start new Simulation» geklickt erscheint ein Dialog wie in Abbildung 8.

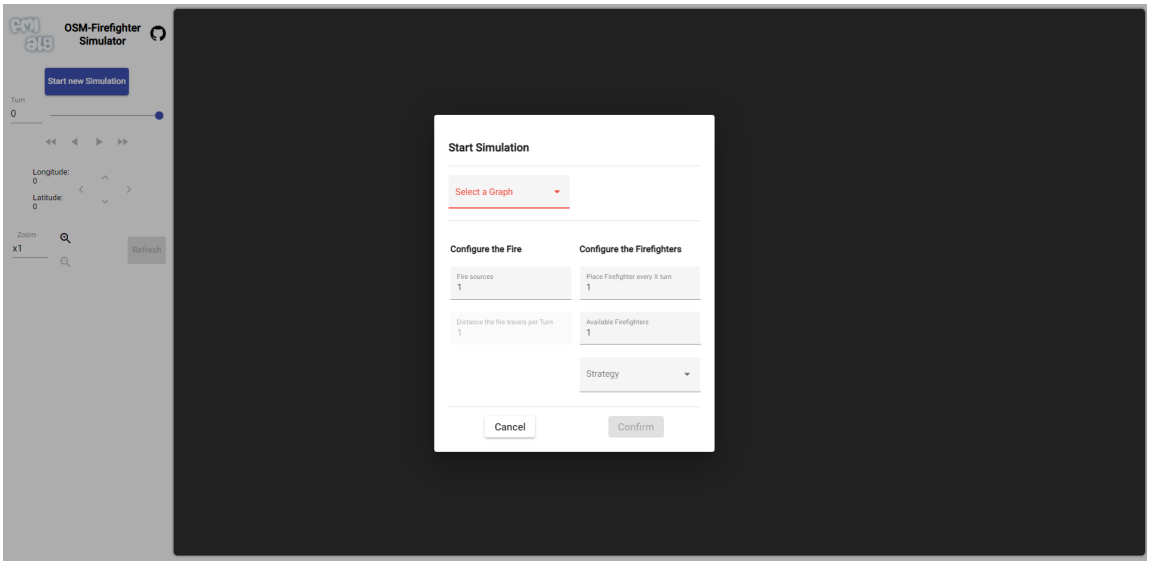


Fig. 8. Der Dialog zum Konfigurieren der Simulationsparameter.

In diesem Dialog werden diverse Parameter festgelegt, um das Verhalten des Firefighter-Problems zu manipulieren. Zuerst muss ein Graph ausgewählt werden. Die Liste an Graphen wird von dem Backend zur Verfügung gestellt und beinhaltet sämtliche zum Start der Anwendung eingelesenen Graphen. Sollte ein neuer Graph hinzugefügt werden sollen, muss die Backend-Komponente neu gestartet werden. Als nächstes können ein paar Parameter bezüglich dem Feuer eingestellt werden. Im Moment kann lediglich die Anzahl der Feuerquellen festgelegt werden. Zu Beginn der Simulation werden dann entsprechend viele zufällig ausgewählte Knoten als Feuerquelle definiert. In Zukunft könnte aber noch die Distanz, die das Feuer zurücklegt, parametrisiert werden. Im Moment wurde die Geschwindigkeit jedoch auf 1 festgelegt. Zuletzt können diverse Parameter bezüglich der Firefighter festgelegt werden. «Place Firefighter every X Turn» steht im Grunde für die Frequenz, mit der Firefighter auf dem Graphen platziert werden. Dieser Parameter steht in starker Verbindung mit der Distanz, die das Feuer pro Runde zurücklegt. Da diese gemeinsam die effektive Distanz darstellen, die das Feuer zurücklegt bevor die Firefighter erneut platziert werden können. Im nächsten Eingabefeld kann die Menge an Firefightern definiert werden, die pro Platzierungsrunde verteilt werden. Zuletzt kann die anzuwendende Strategie ausgewählt werden. Die Liste an Strategien entspricht der im Backend implementierten Strategien.

5.2 Ansicht der Simulation

Wurde eine Simulation gestartet, wird diese für einen kurzen Moment komplett durchlaufen. Ist die Simulation fertig, so ist das Frontend in der Lage die Ansicht anzufordern. Hierbei handelt es sich um eine im Backend generierte Bilddatei. Dies ist eine Komprimierungsmethode, damit nicht sämtliche Graph- und Simulationsinformationen an den Client gesendet werden müssen. Denn im Normalfall werden die von Open-Street-Map generierten Kartendaten sehr schnell sehr groß. Das Endresultat ist in Abbildung 9 dargestellt.

Sobald eine Simulation geladen wurde, werden die Inputkomponenten im Sidebar aktiviert. Oben ist der Nutzer in der Lage, die derzeit angezeigte Runde zu verändern. Dies funktioniert entweder über ein Inputfeld, einen Slider oder die Buttons. Darunter ist der Nutzer in der Lage, die Ansicht zu bewegen. Die Ansicht liegt zu Beginn so, dass der

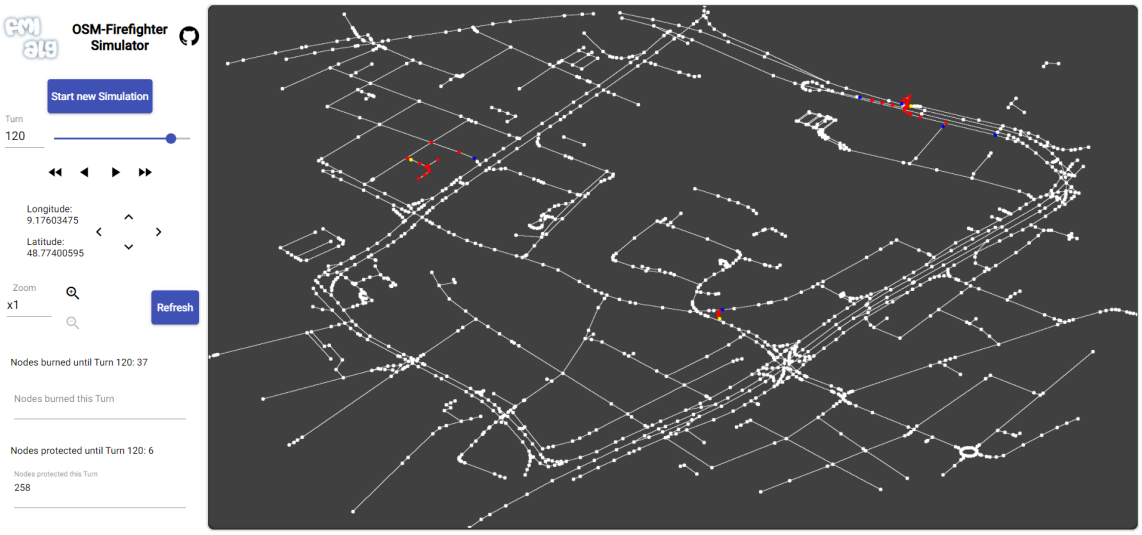


Fig. 9. Die Ansicht des Frontends wenn eine Simulation geladen ist.

Graph mittig liegt und komplett zu sehen ist. Mit einem Klick auf die Buttons wird die Ansicht in die entsprechende Richtung bewegt. Alternativ sind auch die Pfeiltasten benutzbar. Zuletzt ist noch ein Inputfeld für der Zoomlevel zu sehen, welcher auch über Buttons verändert werden kann. Außerdem gibt es einen weiteren Button, um die Ansicht zu aktualisieren. Dieser Button wird aber auch automatisch betätigt, sobald eine Änderung in den zuvor beschriebenen Komponenten erkannt wird.

Zusätzlich erscheinen unten zwei Textboxen. Hier werden Meta-Informationen der derzeitigen Runde angezeigt. Zu diesen gehören die Anzahl an verbrannten und beschützen Knoten bis zu der aktuellen Runde. Außerdem werden die Knoten angezeigt, die in der aktuellen Runde verbannt oder beschützt wurden.

In der Graphansicht sind alle Knoten und Kanten des Graphen dargestellt. Die Graphdaten basieren auf den von Open-Street-Map generierten Kartendaten. In dem Bild sind die Knoten durch Punkte dargestellt. Ihre Position entspricht der geographischen Lage aus den Kartendaten. Jeder Knoten kann vier verschiedene Farben annehmen:

- Gelb: Der Knoten ist eine Feuerquelle.
- Rot: Der Knoten ist verbrannt.
- Blau: Der Knoten wurde durch einen Firefighter beschützt.
- Weiß: Der Knoten ist weder verbrannt noch beschützt.

Wie bereits erwähnt, ist das dargestellte Bild eine vom Backend generierte Bilddatei. Diese Komporimierungsstrategie hat den Nebeneffekt, dass die Ansicht ganz leicht durch einen Rechtsklick exportiert werden kann.

6 EVALUIERUNG/DISKUSSION

Zur Evaluation unser Strategien haben wir eine kleine Konsolenanwendung geschrieben, der die Parameter für die Simulation des Firefighter-Problems als Kommandozeilenargumente übergeben werden können. Zudem kann spezifiziert werden, wie oft eine Konfiguration des Problems simuliert werden soll. Das Programm berechnet dann die

Durchschnittswerte für die Anzahl von verbrannten und beschützten Knoten, sowie die benötigten Zeitschritte und gibt diese auf der Konsole aus.

Für unseren Testgraphen haben wir OSM-Daten aus dem Stuttgarter Zentrum exportiert und mithilfe des *OsmGraphCreator* [3] in das FMI-Textformat umgewandelt. Anschließend haben wir mithilfe einer selbst programmierten Anwendung den Graphen in einen ungerichteten Graphen transformiert. Der dabei entstandene Graph *stgcenter_undirected.fmi* hat 1311 Knoten und 2796 Kanten (1398 logische Kanten).

Wir haben das Firefighter-Problem mit den von uns implementierten Strategien auf insgesamt vier verschiedenen Konfigurationen zu jeweils 1000 Iterationen in unserer Benchmark-Anwendung simuliert und die Ergebnisse anschließend ausgewertet. Die vier Konfigurationen sind in table 1 aufgelistet.

#	**R**	**F**	**S**
1	1	1	10
2	2	1	10
3	2	2	10
4	5	1	10

R – Anzahl Feuerquellen

F – Anzahl Firefighter

S – Anzahl Zeitschritte zwischen zwei Runden, in denen Firefighter Knoten beschützen dürfen

Table 1. Konfigurationen für die Firefighter Simulation auf dem Stuttgart Graphen

Als Vergleichsstrategie dient die *Random*-Strategie. Die Ergebnisse für die jeweiligen Strategien befinden sich in table 2. Das jeweils beste Ergebnis in jeder Spalte – entsprechend der in section 1 definierten Optimierungsziele – wurde hervorgehoben.

	ØB				**ØD**				**ØE**			
Strategy	**# 1**	**# 2**	**# 3**	**# 4**	**# 1**	**# 2**	**# 3**	**# 4**	**# 1**	**# 2**	**# 3**	**# 4**
Random	592,3	785,8	527	966,6	128,6	124,8	200	99,3	1290,7	1252,5	1002,1	997,7
Greedy	**5,3**	**22,8**	**7,8**	**274,6**	**3**	**7,9**	**5,5**	37	**29,8**	**78,9**	**30**	370,1
Multiple Minimum Distance Sets	33,1	154,3	19,8	712,4	14,9	44,2	22,8	100,7	157,2	477,9	115,4	1111,6
Single Minimum Distance Sets	12	87,6	22,9	576,2	3,3	10,2	6,8	**31,7**	33,7	104,1	36,6	**322,2**
Priority	24,9	111,7	15,7	592,4	12,2	36,3	17,7	94,7	127,1	386,9	89,4	1025,1

B – Anzahl verbrannter Knoten

D – Anzahl beschützter Knoten

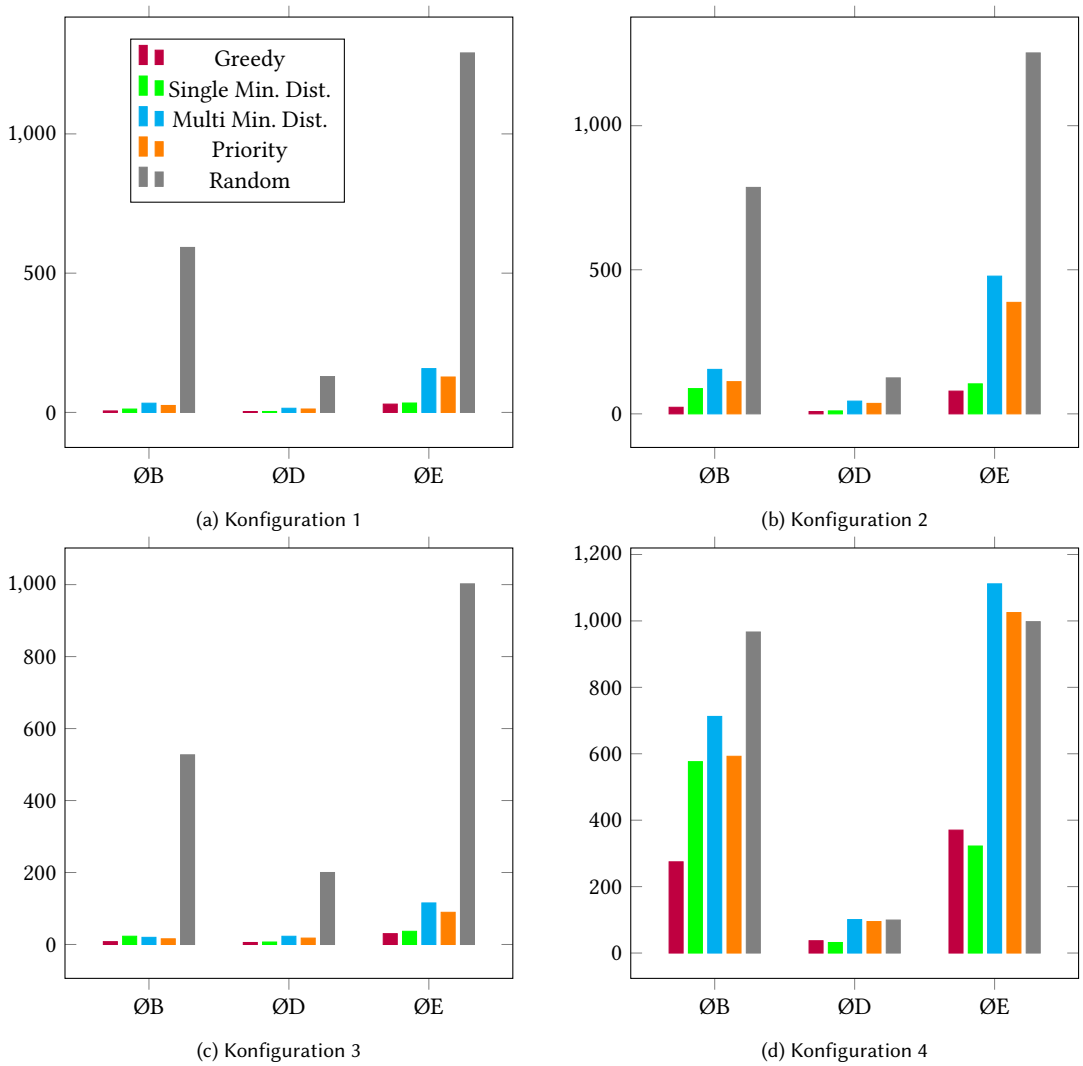
E – Simulationsdauer in Zeitschritten

Table 2. Ergebnisse für die Simulation der Strategien auf dem Stuttgart Graphen

Das Hauptziel der *Multiple Minimum Distance Sets* Strategie sowie der *Priority*-Strategie war die Minimierung der verbrannten und die Maximierung der direkt oder indirekt beschützten Knoten (Optimierungsziele 1 und 2). Die *Single*

Minimum Distance Set Strategie hingegen versucht mit der minimalen Anzahl von durch Firefighter beschützten Knoten, dem Feuer den Weg abzuschneiden und es so einzudämmen (Optimierungsziele 3 und 4). Die Ergebnisse zeigen jedoch einen klaren Vorteil der *Greedy*-Strategie gegenüber diesen Strategien für die Konfigurationen 1 bis 3. Lediglich die *Single Minimum Distance Set* Strategie benötigt mit steigender Anzahl von Feuerquellen tendenziell weniger beschützte Knoten und Zeit, um das Feuer einzudämmen.

fig. 10 stellt die Ergebnisse der Algorithmen unter den verschiedenen Konfigurationen grafisch dar.



(a) Konfiguration 1

(b) Konfiguration 2

(c) Konfiguration 3

(d) Konfiguration 4

Fig. 10. Benchmarkergebnisse für die verschiedenen Konfigurationen

Alle *intelligenten* Firefighter-Algorithmen schnitten für die Konfigurationen 1 bis 3 wesentlich besser ab, als der Zufallsalgorithmus. Mit Hinblick auf die Anzahl von verbrannten Knoten (Optimierungsziele 1 und 2) waren unsere Strategien der *Random*-Strategie sogar für alle vier Konfigurationen überlegen.

Neben der *Greedy*-Strategie erzielt auch die *Single Minimum Distance Set* Strategie oft eine sehr geringe Anzahl von verbrannten Knoten (Optimierungsziele 1 und 2). Dazu kommt ihre wesentlich höhere Effizienz bezüglich der Anzahl beschützter Knoten und der zur Eindämmung des Feuers benötigten Zeit (Optimierungsziele 3 und 4), im Gegensatz zu der *Multi Minimum Distance Sets* oder der *Priority*-Strategie. Letztere versuchen lediglich die Anzahl an nicht verbrannten Knoten zu maximieren (Optimierungsziele 1 und 2), konnten die *Single Minimum Distance Set* Strategie diesbezüglich allerdings nur mit Konfiguration 3 knapp schlagen. Die *Priority*-Strategie stellt zudem über alle Konfigurationen hinweg in allen Bereichen eine leichte Verbesserung gegenüber der *Multi Minimum Distance Sets* Strategie dar.

Die guten Ergebnisse des *Greedy*-Algorithmus auf Straßengraphen sind unserer Meinung nach vor allem auf die geringe Kantendichte solcher Graphen zurückzufühen. Knoten haben in der Regel einen Grad zwischen zwei (z.B. ein Zwischenknoten auf einer Straße) und vier (z.B. ein Knoten an einer Kreuzung). Um einen einzelnen brennenden Knoten an seiner Ausbreitung zu hindern, müssen also lediglich wenige benachbarte Knoten beschützt werden. Hinzu kommt, dass der *Greedy*-Algorithmus immer die Knoten, mit dem höchsten Grad, die am nächsten zum Feuer stehen und damit unmittelbar einer Brandgefahr ausgesetzt sind. Wenn der Algorithmus also genügend Firefighter zur Verfügung hat, so kann er die Ausbreitung des Feuers auf Knoten mit geringerem Grad beschränken. Dadurch wird die exponentielle Ausbreitung des Feuers verhindert bzw. verlangsamt und das Feuer kann mit begrenzten Ressourcen und in relativ kurzer Zeit eingedämmt werden. Ein weiterer Vorteil der *Greedy*-Strategie ist zudem, dass kein globaler Zustand gespeichert werden muss, da der Algorithmus stets in der jeweiligen Runde entscheidet, welche Knoten als nächstes beschützt werden. Die anderen Algorithmen hingegen, berechnen die zu beschützenden Knoten und deren Reihenfolge schon im Vorhinein.

Der *Single Minimum Distance Set* Algorithmus geht im Gegensatz zum *Greedy*-Algorithmus defensiver bei der Eindämmung des Feuers vor, was sich auch in der Anzahl der verbrannten Knoten widerspiegelt. Dennoch sind die Ergebnisse der *Single Minimum Distance Set* Strategie ebenfalls vielversprechend. Zum einen spielt sie für eine höhere Anzahl von Feuerquellen ihren Effizienzvorteil gegenüber der *Greedy*-Strategie aus. Zum anderen schlägt sie die *Multi Minimum Distance Sets* Strategie sowie die *Priority*-Strategie – die ebenfalls auf der Idee der Berechnung von Distanzmengen basieren – in fast allen Belangen und ist dabei auch noch wesentlich effizienter.

Letztere Strategien sind in Bezug auf die Optimierungsziele 1 und 2 immer noch deutlich stärker als die *Random*-Strategie; jedoch zeigt fig. 10 keinen Fall in dem man eine der beiden Strategien der *Single Minimum Distance Set* oder *Greedy*-Strategie vorziehen würde.

7 ZUSAMMENFASSUNG

Das Firefighter-Problem hat in den letzten Jahren immer mehr an Relevanz gewonnen. Bisher hat man sich weitestgehend auf bestimmte Graphklassen bzw. -muster beschränkt; darunter beispielsweise Bäume oder Gittergraphen. Praktische Anwendungsfälle sind zum Beispiel die Bekämpfung von Waldbränden oder einer Virus-Pandemie. In unserer Arbeit haben wir versucht mit Graphen von Straßennetzwerken einen weiteren möglichst realistischen Answendungsfall zu kreieren. Zu diesem Zweck haben wir uns verschiedene Strategien zur Eindämmung des Feuers auf Straßengraphen überlegt und implementiert. Für die Simulation und Visualisierung des Problems sowie unserer Strategien stellen wir mit dieser Arbeit außerdem eine simple Web-Anwendung bereit.

Einen natürlichen Ansatz, um das Firefighter-Problem zu lösen, haben wir mit der *Greedy*-Strategie modelliert. Diese versucht zu jedem Zeitpunkt den bestmöglichen Folgezustand zu ermitteln. Intelligentere Strategien verfolgen einen Ansatz, bei dem zunächst die zu beschützenden Knoten nach ihrer kürzesten Distanz zu den Feuerquellen sortiert

werden. Die *Single-Minimum-Distance-Set* Strategie bestimmt daraus eine Knotenmenge, die das Feuer vollständig umschließt, bei der jeder Knoten mindestens eine bestimmte Distanz zum Feuer hat. Knoten innerhalb des «Kreises» werden dabei garantiert verbrannt. Die *Multi-Minimum-Distance-Sets* Strategie hingegen beschützt Knoten auf mehreren Distanzebenen mit größer werdender Distanz, so lange, bis sich das Feuer nicht weiter ausbreiten kann. Die *Priority-Minimum-Distance-Sets* Strategie verfolgt grundlegend den selben Ansatz, wie die *Multi-Minimum-Distance-Sets* Strategie, weißt den Knoten innerhalb der Distanzmengen jedoch noch eine Priorität zu. Beim Beschützen von Knoten werden dann jene Knoten mit niedriger Priorität erst später betrachtet. Um die Ergebnisse der Strategien in ein Verhältnis setzen zu können, haben wir außerdem noch die *Random*-Strategie implementiert. Diese wählt zufällig aus nicht verbrannten und unbeschützten Knoten eine Teilmenge aus, die anschließend beschützt wird.

Die Auswertung der Strategien brachte ein Ergebnis, das auf den ersten Blick etwas überrascheind sein mag. Die *Greedy*-Strategie konnte in beinahe allen Fällen und für alle Optimierungsziele aus section 1 ein besseres Ergebnis erzielen, als die von uns implementierten intelligenten Strategien. Bei näherer Betrachtung konnte man jedoch schnell sehen, warum die *Greedy*-Strategie auf Straßengraphen so gut funktioniert. Zudem zeigte sich die *Single-Minimum-Distance-Set* Strategie als brauchbare Alternative und konnte in Sachen Effizienz – zumindest für eine Konfiguration mit hoher Anzahl von Feuerquellen – sogar den *Greedy*-Algorithmus knapp ausstechen. Alle von uns implementierten Strategien schnitten außerdem deutlich besser ab, als die Zufallsstrategie. Gerade in Bezug auf die Minimierung verbrannter Knoten können mit intelligenten Algorithmen also deutlich bessere Ergebnisse erzielt werden als durch zufälliges beschützen von Knoten durch Firefighter.

Ausblick

Beim Testen unserer Anwendung ist uns eine Optimierungsmöglichkeit bezüglich der Performance aufgefallen. Beispielsweise werden bei den *Minimum-Distance-Sets* die Distanzmengen lediglich neu berechnet, sobald eine Feuerquelle eingedämmt wird. Durch das Platzieren von Firefightern werden jedoch Pfade von einer Feuerquelle ebenfalls blockiert. Dadurch ändern sich die kürzesten Distanzen, die zuvor über diesen Knoten verlaufen sind, der nun beschützt wird. Ein effektiverer Lösungsansatz wäre, dass man nach jedem Platzieren eines Firefighters den jeweiligen Knoten und seine Nachbarknoten genauer betrachtet. Man startet von jedem Nachbarknoten eine Dijkstra-Suche zu jedem seiner Nachbarknoten. Verläuft der einzige kürzeste Weg über diesen beschützten Knoten, so weiß man, dass sich der kürzeste Pfad für das Feuer verändert hat. Die *Minimum-Distance-Sets* werden entsprechend aktualisiert.

Strategien wie die *Multiple-Minimum-Distance-Sets* oder auch *Priority-Minimum-Distance-Sets* beschützen oft Knoten nicht sehr sinnvoll. Beispielsweise könnte man, sobald alle *Minimum-Distance-Sets* abgearbeitet wurden, einen Greedy-Algorithmus ausführen, der das Feuer noch weiter versucht so gut wie möglich einzudämmen.

An der Priority Strategie könnte man ebenfalls weiter forschen, indem man beispielsweise die Prioritätsberechnung ändert oder auch statt dem 25-%-Quantil einen anderen Separator berechnet. Auch beim Bestimmen der Knotenmengen könnte man anders vorgehen. Beispielsweise könnte man, wie bei der *Multi-Minimum-Distance-Sets* Strategie, Distanzmengen vollständig beschützen, statt nur Teile von ihnen zu beschützen.

Bei der Greedy-Strategie könnte man die Bestimmung der zu beschützenden Knoten ebenfalls ändern. Statt erst nach dem Gewicht der Kanten zu sortieren, sortiert man zuerst nach dem Grad der Zielknoten und erst in zweiter Instanz nach dem Kantengewicht. Ob der Algorithmus dadurch besser oder schlechter abschneidet, bleibt jedoch offen.

Eine weitere Strategie die man implementieren könnte, haben wir aus der *Single-Minimum-Distance-Set* Strategie abgeleitet. Diese versucht der Reihe nach Feuerquellen einzudämmen. Im Grunde ist das Vorgehen sehr ähnlich zu

der *Single-Minimum-Distance-Set* Strategie, mit dem Unterschied, dass man nicht mit einer Gruppe von Knoten alle Feuerquellen umschließen möchte, sondern einzelne Feuerquellen der Reihe nach abarbeitet.

Es ist also deutlich, dass zu dem Firefighter-Problem noch Raum für weitere Forschungen und Optimierungen offen ist.

REFERENCES

[1] 2004–2021. OpenStreetMap. https://www.openstreetmap.org
[2] Aimn Ahmed, Dominik Krenz, and Samuel Holderbach. 2021. OSM-Firefighter Webservice. https://github.com/MScAimnAhmed/OSM-Firefighter
[3] Daniel Bahrdt. 2013–2021. OsmGraphCreator. https://github.com/fmi-alg/OsmGraphCreator
[4] Sarah Days-Merrill. 2019. Firefighter Problem Played on Infinite Graphs. (may 2019).
[5] Stephen Finbow and Gary MacGillivray. 2009. The Firefighter Problem: A survey of results, directions and questions. *Australian Journal of Combinatorics* 43 (2009), 57–77.