Institut für Formale Methoden der Informatik

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Optimierte Platzierung von Ladevorrichtungen für E-Fahrzeuge

Helmut Waldschmidt

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Stefan Funke
Betreuer/in:	Claudius Proissl, M.Sc.
Beginn am:	1. März 2023
Beendet am:	1. September 2023

Kurzfassung

Batterie-Elektrofahrzeuge leiden an einer kurzen Reichweite und benötigen daher ein dichtes Netzwerk von Ladestationen. Um die Kosten zu senken und ein sorgloses Laden gewährleisten zu können, ist eine robuste Platzierung erforderlich.

In dieser Ausarbeitung wird das Problem der Platzierung von Ladestationen untersucht. Dabei wird eine minimale Anzahl an Ladestationen gesucht sodass diese alle kürzesten Pfade einer bestimmten Mindestlänge abdecken. Dazu wurde in dieser Ausarbeitung ein Verfahren entwickelt, das auf hierarchischen Hub-Labels basiert. Dabei wird aus dem Hub-Labeling das entsprechende Hitting-Set extrahiert, welches der Menge der Ladestationen entspricht. Des Weiteren wurde die Korrektheit dieses Verfahrens nachgewiesen. Ergänzend wurden einige Heuristiken und erste Verbesserungen untersucht. Schließlich wurde die Laufzeit und Qualität des implementierten Verfahrens ausführlich auf verschiedenen Graphen mit unterschiedlichen Permutationen getestet. Der Algorithmus liefert dabei eine vielfache Laufzeitverbesserung bei etwas schlechteren Ergebnissen im Vergleich zu anderen Verfahren.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

Abkürzungsverzeichnis

Bwd Backward. 18

CH Contraction Hierachies. 19

Dijkstra Dijkstra-Algorithmus. 30

E-Fahrzeuge Batterie-Elektrofahrzeuge. 5

ESC EV Shortest Path Cover. 7

Fwd Forward. 18

HS Hitting-Set. 17

HSP Hitting-Set-Problem. 17

IS Independent-Set. 19

ISs Independent-Sets. 27

SPHS Shortest-Path Hitting-Set. 17

TT Travel Time. 7

1 Einleitung

1.1 Motivation

In den letzten Jahren haben sich E-Fahrzeuge als nachhaltige Alternative zu konventionellen Fahrzeugen etabliert. Allerdings haben viele E-Fahrzeuge eine geringe Reichweite, weshalb ein dichtes Netzwerk von Ladestationen erforderlich ist. Ein solches dichtes Netzwerk existiert jedoch noch nicht, weshalb sich E-Fahrzeuge nicht für den Einsatz auf langen Strecken eignen.

Um diesem Problem entgegenzuwirken und in Zukunft ein sorgenfreises Aufladen von E-Fahrzeugen gewährleisten zu können, ist eine robuste Platzierung der Ladestationen notwendig. Dabei sollen möglichst wenige Ladestationen platziert werden, um die Kosten gering zu halten und das Laden sollte keine Umwege verursachen. Das bedeutet, dass auf jeder kürzesten Route immer genügend Ladestationen vorhanden sind, falls die Länge der Route die Reichweite der E-Fahrzeuge überschreitet.

Das Ziel dieser Arbeit ist es ein Verfahren zu entwickeln, das für ein gegebenes Straßennetz und eine gegebene Reichweite unter Berücksichtigung der genannten Bedingungen möglichst schnell ein Netzwerk von Ladestationen berechnen kann. Dazu soll die Qualität und Laufzeit dieses Verfahrens als implementierte Version auf realen Straßennetzen getestet werden.

1.2 Aufbau der Arbeit

Zur Lösung der gestellten Aufgabe gliedert sich diese Ausarbeitung wie folgt:

Kapitel 2 - Verwandte Arbeiten: Dieses Kapitel gibt einen Überblick über Arbeiten, die sich mit ähnlichen oder verwandten Problemen befassen.

Kapitel 3 - Grundlagen: Die grundlegende Idee die Menge der Ladestationen aufzubauen besteht darin, diese aus einem hierarchischen Hub-Labeling zu extrahieren. Damit dies möglich ist, müssen zunächst einige Grundlagen definiert werden. Dazu gehören die Definition der Problemstellung, des Hitting-Set-Problems, des Hub-Labelings und der Contraction Hierarchies.

Kapitel 4 - Verfahren: Aufbauend auf den Grundlagen wird hier das Verfahren zur Lösung der Fragestellung beschrieben. Weiterhin werden verschiedene Heuristiken zur Verbesserung des Ergebnisses definiert.

Kapitel 5 - Evaluation: Ziel dieses Kapitels ist das ausführliche Testen des beschriebenen Verfahrens als implementiertes Programm im Hinblick auf Qualität und Laufzeit.

Kapitel 6 - Zusammenfassung und Ausblick: Abschließend werden im letzten Kapitel die Ergebnisse zusammengefasst und ein Ausblick gegeben.

2 Verwandte Arbeiten

Die hier vorliegende Fragestellung wurde bereits in mehreren Ausarbeitungen untersucht. Zu den ersten gehören A. Y. S. Lam et. al. [LLC14]. In dieser Ausarbeitung wird die gegebene Problemstellung definiert und gezeigt, dass diese NP-schwer ist. Das Problem wird analog zu dieser und anderen Ausarbeitungen als Hitting-Set-Problem (siehe Abschnitt 3.3) definiert. Dies ist ein wichtiger Schritt, da es die Verwendung von etablierten Methoden ermöglicht, die ein Hitting-Set für andere Probleme konstruieren. Darüber hinaus werden in der Ausarbeitung einige erste Verfahren zum Lösen des Problems vorgestellt. Diese Verfahren haben zwar eine kurze Laufzeit, liefern jedoch ziemlich große Hitting-Sets. Allerdings können die Verfahren nicht garantieren, dass die Ladestationen sich auf kürzesten Pfaden befinden. Dies kann dazu führen, dass große Umwege zum Laden in Kauf genommen werden müssen.

Diese Einschränkung der kürzesten Pfade ist in der Ausarbeitung von S. Funke et. al. [FNS15] berücksichtigt. Außerdem wird eine effiziente Methode gegeben, welche die untere Schranke des berechneten Hitting-Sets annähert. Diese untere Schranke erlaubt es die Qualität des Verfahrens zu analysieren. Es wird auch ein Lösungsverfahren vorgestellt, das der berechneten unteren Schranke sehr nahe kommt. Jedoch hat dieses Verfahren eine ziemlich lange Laufzeit. Da die Ausarbeitung von S. Funke et. al. das gleiche Problem wie hier behandelt, werden die Problemdefinition (Abschnitt 3.2) und die Berechnung der unteren Schranke (Abschnitt 5.1.5) wiederverwendet.

In der Realität ist es nicht immer notwendig, dass sich Ladestationen direkt auf dem kürzesten Pfad befinden. Es ist auch akzeptabel einen kleinen Umweg zu machen. Aus diesem Grund haben S. Funke et. al. ein weiteres Verfahren entwickelt [FNS16], welches kurze Umwege für Ladestationen erlaubt. Dieser Ansatz kann für bestimmte Graphen die Anzahl der platzierten Ladestationen im Vergleich zum Ansatz ohne Umwege drastisch reduzieren. Das funktioniert jedoch nicht für beliebige Graphen.

Während die diskutierten Arbeiten die Fragestellung als Hitting-Set-Problem definieren, wird das Problem von Y. Xiong et. al. [XGA+17] mit Hilfe von Spieltheorie definiert und gelöst. Der Grund dafür ist, dass bei der Definition durch das Hitting-Set-Problem das dynamische Verhalten der E-Fahrzeug-Fahrer nicht berücksichtigt wird. Beispielsweise können Ladestationen in Großstädten überfüllt oder teurer sein, weshalb Fahrer diese meiden und Ladestationen außerhalb der Stadt aufsuchen. Hinzu kommen die Ladezeiten an den einzelnen Ladestationen und weitere Faktoren, die das Verhalten der Fahrer beeinflussen. Aus diesem Grund haben die Autoren ein Verfahren entwickelt, welches mit der Hilfe von Spieltheorie diese dynamischen Faktoren bei der Platzierung von Ladestationen beachtet.

Dazu betrachten einige Ausarbeitungen das Problem aus der Perspektive eines Ingenieurs. Zum Beispiel wurde von M. Z. Zeb et. al. [ZIK+20] ein Verfahren entwickelt, welches Ladestationen so auswählt, dass die Kosten für die Platzierung minimiert werden. Dabei werden Parameter wie Kosten, Arten von Ladestationen etc. in die Berechnung einbezogen.

3 Grundlagen

In diesem Abschnitt werden einige grundlegende Begriffe, Algorithmen und Definitionen vorgestellt, die für die vorliegende Problemstellung relevant sind. Außerdem wird eine genaue Definition der Problemstellung gegeben.

3.1 Graph

In dieser Ausarbeitung entspricht ein Graph der folgenden Definition:

Definition 3.1.1 (gerichteter, gewichteter Graph)
Ein gerichteter, gewichteter Graph G ist ein Tupel $G = (V, E)$.
$V = \{v_1, v_2, ..., v_n\}$ entspricht der Menge aller Knoten und n der Anzahl der Knoten. In der Implementierung stimmt jeder Knoten mit seinem Index i überein, was den Zugriff auf die Datenstruktur des Graphen erleichtert. $E \subseteq \{(v_i, v_j) | v_i, v_j \in V\}$ entspricht der Menge der gerichteten Kanten im Graph.
Dazu existiert eine Kostenfunktion $c : E \rightarrow \mathbb{N} \setminus \{0\}$, welche jeder Kante ein Kantengewicht zuweist. Dabei sind Kanten mit einem Gewicht von 0 ausgeschlossen.

Ein solcher Graph besteht aus Pfaden, die von einem Start- zu einem Endknoten führen. Nachfolgend die Definitionen eines Pfades und wann dieser minimal ist:

Definition 3.1.2 (Pfad und Pfadlänge)
Sei $P = (v_1, v_2, ..., v_k)$ ein Pfad im Graph G. Es müssen entsprechend die Kanten zwischen den Knoten im Graph existieren. Die Länge des Pfades $l(P)$ lässt sich aus der Summe der Kantengewichte bestimmen. In manchen Fällen wird ein Pfad auch als Menge der Knoten auf diesem Pfad betrachtet. Dies entspricht dann $P = \{v_1, v_2, ..., v_k\}$.

Definition 3.1.3 (kürzester Pfad)
Es ist $dist(v_1, v_k) = min(l(P'))$ die minimale Distanz von v_1 nach v_k aus allen Pfaden P' welche von v_1 nach v_k führen. Ein Pfad $P = (v_1, v_2, ..., v_k)$ ist ein kürzester Pfad wenn $l(P) = dist(v_1, v_k)$.

3.2 Problemstellung

Wie bereits erwähnt, ist das Ziel dieser Arbeit, ein Verfahren zu entwickeln, das aus einem gegebenen Graphen eine möglichst kleine Menge von Knoten auswählt, welche alle kürzesten Pfade abdeckt, die länger als eine gegebene Reichweite sind. Bevor ein Verfahren entwickelt werden kann, muss das Problem formal definiert werden. Im Rahmen dieser Arbeit wird die Definition von S. Funke et al. [FNS15] verwendet. Diese findet sich nochmals in Definition 3.2.1.

Definition 3.2.1 (EV Shortest Path Cover (ESC))
Seien ein Graph $G = (V, E)$ und eine Funktion $c : E \to \mathbb{N} \setminus \{0\}$ nach Definition 3.1.1 gegeben. Dazu sei $\mu > 0$ die maximale Reichweite des E-Fahrzeuges.
Ein ESC ist eine Menge $C \subseteq V$ in welcher jeder Knoten $v \in C$ eine Ladestation darstellt. Dabei soll die Menge C für jeden kürzesten Pfad P in G mit $l(P) > \mu$ garantieren, dass auf dem Pfad genug Ladestationen vorhanden sind, um das Ziel zu erreichen. Es wird angenommen, dass das E-Fahrzeug beim Startknoten voll geladen ist, um weitere Probleme zu vermeiden.

Das Ziel des ESC-Problems ist es nun, eine möglichst kleine Menge an Ladestationen zu finden.

Um störende Sonderfälle zu vermeiden, wird die folgende Annahme getroffen:

Annahme 1 (Mindestreichweite)
Sei k_{max} die maximale Kantenlänge im Graphen G. Es gilt $\mu \geq k_{max}$ für die Reichweite μ. Diese Bedingung wird, um Probleme beim Algorithmus in Abschnitt 4 zu vermeiden, auf $\frac{\mu}{2} \geq k_{max}$ erweitert. Ist diese Bedingung nicht erfüllt, so existiert keine gültige ESC-Menge.

Darüber hinaus kann die Bedingung für das ESC etwas verschärft werden, wie in Abbildung 3.1 zu sehen ist. Kürzeste Pfade P, deren Länge ein Vielfaches von der Reichweite μ beträgt, werden bei der Lösung des Problems implizit mehrfach betrachtet, da Teilpfade P', die ebenfalls $l(P') \geq \mu$ erfüllen ggf. bereits betrachtet wurden und somit der Pfad P schon abgedeckt ist. Dadurch kann die Anzahl der betrachteten Pfade reduziert werden. Das kann sogar weiter verschärft werden. Sei k_{max} die maximale Kantenlänge. Pfade, die länger als $\mu + k_{max}$ sind, werden implizit bereits durch kürzere Pfade berücksichtigt und müssen daher nicht noch einmal betrachtet werden. Dadurch wird trotzdem ein allgemeingültiges ESC erzeugt. Die Pfade mit $\mu + k_{max} \geq l(P) \geq \mu$ entsprechen dabei den kürzesten Pfaden, welche nur eine Ladestation benötigen, um das Ziel zu erreichen. In [FNS15] wurde bewiesen, dass es ausreicht, wenn eine ESC-Menge nur diese Pfade abdeckt. Dadurch werden dann implizit alle anderen Pfade mit $> \mu + k_{max}$ abgedeckt. Daraus kann nun das erweiterte ESC definiert werden.

Definition 3.2.2 (Erweitertes ESC)
Sei k_{max} die maximale Kantenlänge. Eine Menge $C \subseteq V$ ist ein valides ESC, wenn für jeden kürzesten Pfad P mit $\mu + k_{max} \geq l(P) > \mu$ mindestens eine Ladestation vorhanden ist. Diese Menge deckt auch alle Pfade mit $l(P) > \mu + k_{max}$ ab.

Diese Definition führt nun dazu, dass jeder Pfad P der Länge $\mu + k_{max} \geq l(P) \geq \mu$ nur einmal abgedeckt werden muss. Was den Vorteil mit sich bringt, dass das ESC als Hitting-Set-Problem formuliert werden kann (Abschnitt 3.3).

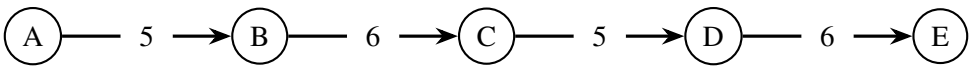

Abbildung 3.1: Beispiel ESC: Sei die Reichweite $\mu = 10$. Durch die Pfade (A, B, C) und (C, D, E) wurden die Knoten B, E als Ladestationen ausgewählt. Damit ist der Pfad (A, B, C, D, E) bereits abgedeckt und muss nicht mehr weiter betrachtet werden.

Dabei ist zu beachten, dass das ESC-Problem, wie bereits in [FNS15] gezeigt, NP-schwer ist, weshalb in dieser Ausarbeitung der Fokus auf Heuristiken (Abschnitt 4.3) mit kurzer Laufzeit und einer kleinen ESC-Menge gesetzt wird.

3.3 ESC als Hitting-Set-Problem

Das Hitting-Set-Problem (HSP) [KAR72] ist ein NP-vollständiges Problem mit dem Ziel, aus einer Menge von Teilmengen und einem Universum, eine Menge zu extrahieren, welche aus jeder Teilmenge mindestens ein Element enthält. Haufig ist auch die maximale Größe der extrahierten Menge vorgegeben.

Wie bereits in [FNS15] gezeigt, kann das ESC als HSP formuliert werden. Dies hat den Vorteil, dass sich schon bekannte Verfahren zur Konstruktion von Hitting-Sets verwenden lassen.

Im folgenden wird nun die klassische und eine auf die Fragestellung angepasste Version des Hitting-Sets definiert:

3.3.1 Das klassische Hitting-Set-Problem

Definition 3.3.1 (l-Hitting-Set (HS))
Sei U ein Universum und $S = \{S_1, S_2, ..., S_k\}$ eine Sammlung von Mengen mit $\forall S_i \in S : S_i \subseteq U$. Ein l-Hitting-Set (HS) mit $l > 0$ ist eine Menge $H \subseteq U$ mit den Eigenschaften $|H| \leq l$ und $\forall S_i \in S : |S_i \cap H| \neq \emptyset$.

Das Ziel des HSP ist es nun so ein l-HS für die Sammlung von Mengen S zu finden. Dabei ist $l > 0$ ein vorgegebener Parameter.

3.3.2 Shortest-Path Hitting-Set-Problem

Die Idee bei der Umformulierung des ESC-Problem zum HSP ist, als Teilmengen kürzeste Pfade zu verwenden, welche länger als die Reichweite sind, was auch in [FNS15] gemacht wurde. Hier wird ein ähnlincher Ansatz verwendet. Dies führt zur Definition 3.3.2

Definition 3.3.2 (Shortest-Path Hitting-Set (SPHS))
Sei k_{max} die maximale Kantenlänge und $K = \{P|\mu + k_{max} \geq l(P) > \mu\}$ eine Menge von kürzesten Pfaden. Es ist $C \subseteq V$ ein SPHS für die Menge K mit der Eigenschaft $\forall P \in K : |P \cap V| \neq \emptyset$. Dieses deckt nach Abschnitt 3.2 alle kürzesten Pfade mit Länge $> \mu$ ab.

Aus den Definitionen 3.2.2 und 3.3.2 geht hervor, dass das (erweiterte) ESC äquivalent zum SPHS ist. Bei beiden Problemen wird eine Menge gesucht, die alle kürzesten Pfade einer gegebenen Länge abdeckt. Die Äquivalenz wurde schon in [FNS15] gezeigt.

3.4 Distanzmetriken

In dieser Ausarbeitung werden zwei verschiedene Metriken/Kostenfunktionen für die Kantengewichte der Graphen verwendet. Die erste ist die Reisezeit (Travel Time(TT)), die standardmäßig beim Einlesen des Graphen gegeben ist. Dabei entspricht 1 $TT = \frac{1}{100}s$. Die Zweite Metrik ist die Distanz in Metern. Diese muss zunächst aus den gegebenen Koordinaten, bestehend aus Längen- und Breitengrad, berechnet werden. Dazu wird die Haversine-Formel [Sin84] (siehe Definition 3.4.1) verwendet, die ein ziemlich exaktes Ergebnis liefert.

Definition 3.4.1 (Haversine-Formel [Sin84])
Seien die Koordinaten zweier Punkte $x = (lat_x, long_x), y = (lat_y, long_y)$ und der Radius der betrachteten Kugel r (für Erde $r \approx 6371000m$ [NAS23]) gegeben. Die Distanz von x nach y $dist(x, y)$ lässt sich dann mit der folgenden Formel berechnen:

$$dist(x, y) = 2 \cdot r \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{lat_y - lat_x}{2}\right) + \cos(lat_x) \cdot \cos(lat_y) \cdot \sin^2\left(\frac{long_y - long_x}{2}\right)}\right)$$

3.5 Hub-Labeling

Hub-Labeling hat zum Ziel, kürzeste Distanzen zwischen zwei beliebigen Punkten zu berechnen. Dabei ist dieses um ein Vielfaches schneller im Vergleich zu herkömmlichen Kürzeste-Distanz-Algorithmen, wie dem Dijkstra-Algorithmus [DIJ59]. Um eine so kurze Laufzeit zu erreichen, müssen einige Informationen vorberechnet werden.

Dabei werden beim Hub-Labeling für jeden Knoten zwei Mengen gebildet. Dies sind die Forward (Fwd)- und Backward (Bwd)-Label. Diese Mengen bestehen aus Tupeln von Hub-Knoten und Distanzen vom Ausgangsknoten zum Hubknoten. In der Praxis sind diese Mengen relativ klein, so dass sich sehr effizient die Distanzen zwischen zwei beliebigen Knoten finden lassen. Man findet die Distanzen, indem man die Fwd-Menge des Startknotens und die Bwd-Menge des Endknotens nimmt und einen gemeinsamen Hubknoten sucht. Ist ein Knoten gefunden, lassen sich die jeweiligen Distanzen addieren, welche den Abstand zwischen Start- und Endknoten ergeben. Im Folgenden ist das Hub-Labeling nochmal formal definiert:

Definition 3.5.1 (Hub-Labeling)
Ein Hub-Labeling besteht aus den zwei Mengen $fwd = \{fwd(v)|v \in V\}$ und $bwd = \{bwd(v)|v \in V\}$. Wobei $fwd(v) = \{(u, k)|u \in V \wedge k = dist(v, u)\}$ und $bwd(v) = \{(u, k)|u \in V \wedge k = dist(u, v)\}$ Dabei muss für zwei beliebige Knoten $u, v \in V$, für die ein Pfad von u nach v existiert, $\exists h \in V : (h, k_1) \in fwd(u) \wedge (h, k_2) \in bwd(v) \wedge dist(u, v) = k_1 + k_2$ gelten.

Es gibt mehrere Ansätze zur Berechnung von Hub-Labels für jeden Knoten. Einer der ersten Ansätze wurde von E. Cohen et. al. [CHKZ03] entwickelt. Dieser Ansatz hat jedoch eine relativ lange Laufzeit, weshalb sich der hierarchische Ansatz von I. Abraham et. al. [ADGW12] durchgesetzt hat. Der hierarchische Ansatz ist nochmal kurz in Abschnitt 3.6.1 erklärt. Das hierarchische Hub-Labeling wird im Folgenden zur Berechnung der Label verwendet.

3.6 Contraction Hierarchies

Bei den Contraction Hierachies (CH) handelt es sich um einen Algorithmus, welcher in einen Graphen G neue Abkürzungskanten einfügt [GSSD08]. Um dies zu erreichen, werden nacheinander alle Knoten kontrahiert. Beim Kontrahieren wird der ausgewählte Knoten entfernt. Liegt der entfernte Knoten auf dem kürzesten Weg von einem Nachbarknoten zu einem anderen Nachbarknoten, so wird eine Abkürzungskante mit der entsprechenden Länge eingefügt. Die neuen Kanten werden anschließend wieder in den Graphen G eingefügt. Diese nehmen keinen Einfluss auf die maximale Kantenlänge k_{max}. Dabei haben die Contraction Hierarchies analog zum Hub-Labeling das Ziel Distanzen zwischen zwei beliebigen Punkten schneller als beim Dijkstra-Algorithmus zu berechnen. Es sind zwar die Distanzberechnungen mit den Contraction Hierarchies nicht so schnell, wie mit dem Hub-Labeling, jedoch werden diese für das hierarchische Hub-Labeling benötigt.

Bei der Kontraktion hat die Knotenreihenfolge einen sehr großen Einfluss auf die Qualität der eingefügten Kanten und die Geschwindigkeit bei der Berechnung von Punkt-zu-Punkt-Distanzen. Die Knotenreihenfolge ist dabei eine Funktion, die jedem Knoten eine Zahl zuordnet (siehe Definition 3.6.1). Diese Funktion legt die Hierarchie der Knoten fest. Dabei entspricht eine große Zahl einer hohen Wichtigkeit des Knotens.

Definition 3.6.1 (Hierarchiefunktion)
Die Hierarchiefunktion $level : V \to \mathbb{N}$ ordnet jedem Knoten $v \in V$ eine natürliche Zahl zu, die der Wichtigkeit (Position in der Hierarchie) des Knotens entspricht.

Die Reihenfolge der Knoten wird während des CH-Algorithmus mit Hilfe von Heuristiken dynamisch berechnet. Durch lazy update [GSSD08] wird ein Knoten ausgewählt, der als nächster kontrahiert wird. Dem ausgewählten Knoten wird dann die aktuelle Wichtigkeit (Position in der Hierarchie) zugewiesen. Die berechnete Knotenreihenfolge kann auch nach dem CH-Algorithmus verwendet werden (Abschnitt 3.6.1). Es ist zu beachten, dass die Knotenreihenfolge durch die Heuristik bestimmt wird und somit einen großen Einfluss auf die Qualität der Hierarchie und der eingefügten Kanten hat. Somit hat die Heuristik auch einen Einfluss auf das Hub-Labeling und das finale ESC. Aus diesem Grund werden verschiedene Heuristiken zur Verbesserung der Hierarchie in Abschnitt 4.3 nochmals aufgegriffen. Der genaue CH-Algorithmus ist in [GSSD08] zu finden.

In der beschriebenen Version des CH-Algorithmus gilt für alle Knoten:

$$\forall u, v \in V : u \neq v \implies level(u) \neq level(v)$$

Da insbesondere die ersten Knoten, die kontrahiert werden, sehr ähnlich sind, ist es sinnvoll, diese gleichzeitig zu kontrahieren. Dadurch erhalten diese Knoten den gleichen Wert von der Hierarchiefunktion. Genau dies wurde in der Version von T. Kieritz et al. [KLSV10] des CH-Algorithmus gemacht. Anstatt einen Knoten zur Kontraktion auszuwählen, wird ein Independent-Set (IS) aufgebaut. Diese Menge enthält alle Knoten, die unabhängig voneinander kontrahiert werden können. Der Rest des CH-Algorithmus bleibt gleich. Da nun viele Knoten den gleichen Wert in der Hierarchiefunktion haben, können Funktionen, die mit der Hierarchiefunktion arbeiten, parallelisiert werden. Beide Versionen des CH-Algorithmus werden verwendet und insbesondere in Abschnitt 5 auf ihre Qualität bezüglich des ESC-Problems untersucht.

Im Folgenden wird das hierarchische Hub-Labeling basierend auf den Contraction Hierarchies beschrieben.

3.6.1 Hierarchisches Hub-Labeling

Der hierarchische Ansatz verwendet die Hierarchiefunktion, die durch den Contraction Hierarchies Algorithmus aufgebaut wird. Man beginnt mit dem Knoten, der den höchsten Wert der Hierarchiefunktion erhält. Dann wird die Hierarchie schrittweise nach unten abgelaufen. Für jeden Knoten wird der Knoten selbst als Label in der Fwd- und Bwd-Menge eingefügt. Dann werden für die Fwd-Menge alle ausgehenden Kanten betrachtet. Wenn der Zielknoten einen höheren Wert in der Hierarchiefunktion hat, werden die Fwd-Label des Zielknotens in die Fwd-Label des betrachteten Knotens eingefügt. Anschließend werden Duplikate aus der Fwd-Menge entfernt. Weitere Knoten werden aus der Fwd-Menge entfernt, wenn die Distanz vom betrachteten Knoten zum Hub-Knoten nicht stimmt (das kann auch schon mit den Hub-Label-Mengen getestet werden, um Zeit zu sparen). Das gleiche wird nun mit den eingehenden Kanten für die Bwd-Label gemacht.

Durch den hierarchischen Ansatz kann die Berechnung der Label parallelisiert werden, wenn die Knoten den gleichen Wert in der Hierarchiefunktion haben.

4 Verfahren

Das Ziel dieses Kapitels ist es, das Verfahren zu beschreiben, das eine ESC-Menge für einen gegebenen Graphen G und eine Reichweite μ liefert. Dazu wird zunächst die Idee des Verfahrens und die formale Definition gegeben, die als Grundlage für den Algorithmus zur Berechnung des ESC genommen wird. Weiterhin werden verschiedene Heuristiken beschrieben, die zur Minimierung des ESC beitragen sollen. Abschließend wird eine Verbesserung des Verfahrens entwickelt und auf die Parameter von E-Fahrzeugen eingegangen.

4.1 Idee und formale Definition

Ein erster Ansatz, das ESC aufzubauen war, alle Pfade zu konstruieren und aus diesen das Hitting-Set zu extrahieren. Dies ist jedoch, wie bereits in [FNS15] gezeigt, sehr zeitaufwendig. Selbst bei Punkt-zu-Punkt-Anfragen mit Hub-Labeling, das um ein Vielfaches schneller ist als der Dijkstra-Algorithmus, führt das zu einer sehr hohen Laufzeit und einem hohen Speicherverbrauch.

Die Idee des hier verwendeten neuen Ansatzes ist es, nicht mehr alle Pfade zu konstruieren, sondern das ESC aus dem Hub-Labeling zu extrahieren. Denn nach [ADF+16] lässt sich das Hub-Labeling als ein Hitting-Set betrachten. Dazu ist gemäß Abschnitt 3.3.2 das ESC äquivalent zum SPHS und somit auch ein Hitting-Set-Problem. Um das ESC zu erhalten, lässt sich also die entsprechende Teilmenge aus dem Hub-Labeling extrahieren. Dies führt uns zum Theorem 1, das zeigt, wie das ESC mit dem Hub-Labeling zusammenhängt.

Theorem 1 (ESC ist Teilmenge von Hub-Labeling)
Sei $H = \bigcup_{v \in V}(fwd(v) \cup bwd(v))$ die Vereinigung aller Hub-Label gemäß Definition 3.5.1.
Es ist $C = \{v \in V | (v,k) \in H \wedge \mu \geq k \geq \frac{\mu}{2}\}$ ein SPHS. Das ist nach Definition äquivalent zu ESC für die Reichweite $\mu > 0$. (Intuitiv könnte man sich denken die Hub-Label im Bereich $2\mu \geq k \geq \mu$ zu verwenden. Das Beispiel in Abbildung 4.1 verdeutlicht nochmal warum dies nicht funktioniert.)

Es ist nun zu beweisen, dass die Menge C aus Theorem 1 eine gültige ESC-Menge ist.

Beweis 1 (Theorem 1: C ist ein gültige ESC-Menge für $\mu > 0$)
Widerspruchsbeweis:
Sei die Menge C definiert nach Theorem 1 und k_{max} die maximale Kantenlänge. Sei C keine gültige ESC-Menge. Damit existiert mindestens ein kürzester Pfad P mit $\mu + k_{max} \geq l(P) > \mu$, Startknoten s und Endknoten t, der nicht von C abgedeckt wird. Nach der Definition des Hub-Labeling gibt es zwei Tupel $(u, k_1) \in fwd(s), (u, k_2) \in bwd(t)$ mit $l(P) = k_1 + k_2$ und dem Hub-Knoten u. Es muss $(k_1 > \mu \wedge k_2 < \frac{\mu}{2}) \vee (k_1 < \frac{\mu}{2} \wedge k_2 > \mu)$ gelten, da sonst nach der Definition von C der Knoten u in der Menge sein müsste. O. B. d. A. sei P' der Pfad von s nach u mit $l(P') > \mu$ (Fall mit Pfad von u nach t geht analog). Da $\frac{\mu}{2} \geq k_{max}$ gilt, existiert ein Teilpfad P'' von P' mit $\mu \geq l(P'') \geq \frac{\mu}{2}$. Es

ist P'' ein Pfad von einem Knoten s' zum Hubknoten u. Da hier ein hierarchisches Hub-Labeling verwendet wird, muss der Knoten u entsprechend ein Hub für s' sein. Nach der Definition von C gilt somit $u \in C$. Damit ist aber der Pfad P durch den Knoten u abgedeckt, was ein Widerspruch zur Annahme ist.

□

In Theorem 1 wurde die obere Schranke auf μ gesetzt. Diese lässt sich jedoch aufgrund des hierarchischen Hub-Labelings beliebig wählen, solange diese mindestens $\frac{\mu}{2} + k_{max}$ ist. Dabei hat die obere Schranke keinen Einfluss auf die Größe und die Korrektheit der ESC-Menge. Die untere Schranke lässt sich dagegen nicht beliebig wählen (siehe Abschnitt 5.1.6).

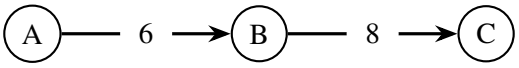

Abbildung 4.1: Beispiel ESC Teilmenge von Hub-Labeling: Sei die Reichweite $\mu = 10$. Gemäß Abschnitt 3.5 ergeben sich die folgenden Fwd-Label: $fwd(A) = \{(A, 0), (B, 6), (C, 14)\}$, $fwd(B) = \{(B, 0), (C, 8)\}$, $fwd(C) = \{(C, 0)\}$ (Bwd-Label in diesem Beispiel nicht relevant). Nun können die ESC-Mengen $C_1 = \{B, C\}$ (Verwendung von $\mu \geq k \geq \frac{\mu}{2}$) und $C_2 = \{C\}$ (Verwendung von $2\mu \geq k \geq \mu$) gebildet werden. Die Menge C_2 ist kein valides ESC, da C von A nicht erreichbar ist (keine Ladestation dazwischen). Dagegen ist C_1 ein valides ESC, da C nun von A erreichbar ist. Somit müssen die Label mit der Bedingung $\mu \geq k \geq \frac{\mu}{2}$ ausgewählt werden.

4.2 Algorithmus

Der Algorithmus 4.1 beschreibt die genaue Konstruktion eines ESC aus einem hierarchischen Hub-Labeling. In diesem Fall werden CH und Hub-Label jedes Mal neu berechnet. Dies ist nicht notwendig. Diese können einmal berechnet werden und dann kann ein ESC mit beliebiger Reichweite μ ermittelt werden.

Das Hub-Labeling kann viel Speicherplatz in Anspruch nehmen. Daher wird in Abschnitt 4.4 ein Ansatz diskutiert, der diesem Problem entgegenwirkt. Dadurch wird die Laufzeit und der Speicherverbrauch stark reduziert.

Algorithmus 4.1 ESC aus Hub-Labeling: Zunächst werden die Contraction Hierarchies und das Hub-Labeling berechnet. Anschließend kann aus dem Hub-Labeling gemäß Theorem 1 die ESC-Menge erstellt werden.

```
1:  procedure calculateESC(G = (V, E), μ, heuristic)
2:      // augment G with shortcuts and get node hierachy (levels[i] returns level of node i)
3:      (G, levels) ← calculateCH(G, heuristic)
4:      // fwd[i] returns Fwd-Labels of node i (analogous for Bwd-Labels)
5:      (fwd, bwd) ← calculateHubLabeling(G, levels)
6:      C ← ∅
7:      for all v ∈ V do
8:          for all (u, k) ∈ fwd[v] do
9:              if k ≥ μ/2 ∧ k ≤ μ then
10:                 C ← C ∪ {u}
11:             end if
12:         end for
13:         for all (u, k) ∈ bwd[v] do
14:             if k ≥ μ/2 ∧ k ≤ μ then
15:                 C ← C ∪ {u}
16:             end if
17:         end for
18:     end for
19:     return C
20: end procedure
```

4.3 Heuristiken

Im Folgenden werden einige Heuristiken vorgestellt, die für beide Varianten des CH-Algorithmus verwendet werden können. Bei einer Heuristik wird einem ausgewählten Knoten $v \in V$ ein Wichtigkeitswert zugeordnet. Ist dieser Wert minimal, so wird der entsprechende Knoten als nächster kontrahiert.

Eine gute Heuristik minimiert die Anzahl der eingefügten Kanten. Dazu muss die Heuristik in diesem Fall die Größe des Hub-Labelings und des ESC minimieren. Die meisten Heuristiken sind Kompositionen aus den Heuristiken 1, 2 und einigen anderen, die von R. Geisberger et. al. [GSSD08] entwickelt wurden. Daraus wurde dann die Heuristik 3 in Kombination mit einer eigenen Heuristik abgeleitet. Die letzte Heuristik 4 kombiniert die Heuristik aus [ADGW11], die ein sehr kleines Hub-Labeling liefert, mit der eigenen Heuristik. Die Qualität der hier behandelten Heuristiken wird in Kapitel 5 ausführlich untersucht.

Heuristik 1 (In and Out Product)
Bei der In and Out Product Heuristik wird die Anzahl der ausgehenden und eingehenden Kanten, die zu nicht kontrahierten Knoten führen, des betrachteten Knotens multipliziert. Dies wird dann als Wichtigkeit des Knotens zurückgegeben. Siehe Beispiel in Abbildung 4.2.
Die Heuristik repräsentiert die maximale Anzahl an eingefügten Abkürzungskanten, weshalb diese als gute Grundlinie für die anderen Heuristiken dient.

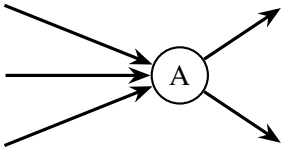

Abbildung 4.2: Beispiel für In and Out Product Heuristik: Der Knoten *A* hat 3 eingehende und 2 ausgehende Kanten, daher wird ihm der Heuristikwert $3 \cdot 2 = 6$ zugewiesen.

Heuristik 2 (Edge Difference)
Diese Heuristik berechnet zunächst die num_shortcuts und num_deleted_edges. Der Parameter num_shortcuts zählt die Anzahl der Abkürzungskanten, die eingefügt werden, wenn der Knoten als nächstes kontrahiert wird. Die Variable num_deleted_edges zählt analog zur Heuristik 1 die Summe der ausgehenden und eingehenden Kanten, die zu noch nicht kontrahierten Knoten führen. Als Ergebnis wird die Differenz num_shortcuts − num_deleted_edges zurückgegeben. Siehe Beispiel in Abbildung 4.3. Diese Heuristik repräsentiert die genaue Anzahl an eingefügten und entfernten Kanten, weshalb diese in der Regel ein besseres Ergebnis im Vergleich zu Heuristik 1 liefert.

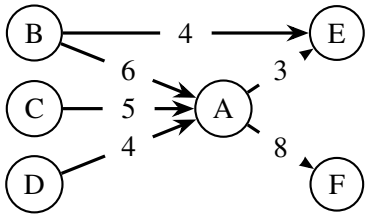

Abbildung 4.3: Beispiel für Edge Difference Heuristik: Der Knoten *A* hat 3 eingehende und 2 ausgehende Kanten, also ist $num\_deleted\_edges = 5$. Es ist $num\_shortcuts = 5$. Daher ist der Heuristikwert von *A* $num\_shortucts - num\_deleted\_edges = 5 - 5 = 0$

Heuristik 3 (Maximale Kosten)
Auch hier werden zwei Parameter berechnet. Zum einen max_cost und zum anderen in_out_product. Für max_cost wird das maximale Kantengewicht der Abkürzungskanten gesucht. Die Variable in_out_product wird analog zur Heuristik 1 berechnet. Schließlich wird $0.8 \cdot max\_cost + 0.2 \cdot in\_out\_product$ *zurückgegeben. Intuitiv reduziert diese Heuristik die Größe der ESC-Menge, da durch die Variable max_cost kurze Abkürzungskanten präferiert werden. Dies bestätigt sich auch in den Ergebnissen in Abschnitt 5.2.*

Nimmt man den Graphen aus Abbildung 4.3 und kontrahiert den Knoten A. Dann ist $in\_out\_product = 6$ *und* $max\_cost = 14$*, da die Abkürzungskante von B nach F die längste ist. Dies ergibt den Heuristikwert* $0.8 \cdot max\_cost + 0.2 \cdot in\_out\_product = round(12.4) = 12$ *(Werte werden als Integer gespeichert).*

Heuristik 4 (Mixed)
Diese Heuristik kombiniert die fünf Parameter max_cost, edge_difference, num_contracted_neighbours, cur_level und num_underlying_shortcuts. Dabei wird max_cost analog zur Heuristik 3 und edge_difference analog zur Heuristik 2 berechnet.

Der Parameter num_contracted_neighbours zählt die Nachbarn, die bereits kontrahiert wurden, num_underlying_shortcuts zählt die Anzahl der Abkürzungskanten in den erzeugten Abkürzungskanten und cur_level ist der nächste Wert der Hierarchiefunktion. Alle Parameter werden dann folgendermaßen kombiniert:

$$0.001 \cdot max\_cost + 2 \cdot edge\_difference + num\_contracted\_neighbours + 5 \cdot cur\_level + num\_underlying\_shortcuts$$

In Abbildung 4.4 werden die Komponenten der Heuristik anhand eines Beispiels nochmals erläutert. Diese Heuristik kombiniert die Heuristik aus [ADGW11], welche ein vergleichsweise kleines Hub-Labeling liefert und den Teil aus Heuristik 3, welcher zu einer kleinen ESC-Menge führt, um möglicherweise ein besseres Ergebnis für die ESC-Menge zu erreichen.

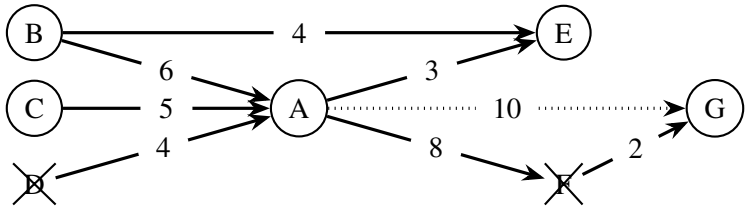

Abbildung 4.4: Beispiel für Mixed Heuristik: D und F wurden bereits kontrahiert. Durch die Kontraktion von F wurde eine Abkürzungskante von A nach G eingefügt. Für die Heuristik wird nun A betrachtet. Es ist $num\_contracted\_neighbours = 2$, $edge\_difference = 0$ und $max\_cost = 16$. Die Variable $cur\_level = 3$, da bereits zwei Knoten kontrahiert wurden. $num\_underlying\_shortcuts = 2$, da für die Abkürzungen von C nach G und B nach G die Abkürzung von A nach G verwendet wird (beim Kontrahieren merken sich die neuen Kanten die Anzahl der Abkürzungskanten aus denen diese bestehen (In diesem Fall 1)). Der Knoten A erhält schließlich den Heuristikwert nach der Formel aus Heuristik 4 $0.001 \cdot 16 + 2 \cdot 0 + 2 + 5 \cdot 3 + 2 = round(19.016) = 19$.

Die $0.8/0.2$ Gewichtung der Heuristik 3 und die 0.001 Gewichtung des Parameters max_cost in der Heuristik 4 wurden durch Testen auf einigen Graphen abgeleitet. Näheres dazu in Abschnitt 5.2.

4.4 Pruned Hub-Labeling

Das Hub-Labeling verbraucht den meisten Speicher, daher ist es von Vorteil, den Speicherverbrauch zu reduzieren. Der erste Ansatz dazu ist die Verwendung einer guten Heuristik für die CH-Berechnung. Es ist jedoch möglich den Speicher in diesem Zusammenhang weiter zu reduzieren. Da der vorgestellte Algorithmus 4.1 nur die Hub-Labels mit den Distanzen im Bereich $[\frac{\mu}{2}, \mu]$ verwendet, sind alle anderen Labels nicht notwendig und können weggelassen werden. Die Labels mit den Distanzen $< \frac{\mu}{2}$ können nicht weggelassen werden, da sie zur Berechnung der Labels mit größeren Distanzen benötigt werden. Dies führt dann zum Pruned Hub-Labeling, das bei der Hub-Label-Berechnung die Labels mit den Distanzen $> \mu$ nicht berechnet. Dies spart Speicher durch die weggelassenen Labels. Außerdem wird die Laufzeit reduziert, da weniger Label berechnet werden müssen.

4.5 Parameter für E-Fahrzeuge

Die Reichweite von E-Fahrzeugen hängt von vielen Parametern ab. Einer dieser Parameter ist die Steigung zwischen zwei Knoten, welcher in der Ausarbeitung von S. Funke et. al. [FNS15] direkt in die Berechnung einfließt. Weitere Parameter sind die Jahreszeit oder die Geschwindigkeit, mit der sich die E-Fahrzeuge fortbewegen. In dieser Ausarbeitung werden diese Parameter nicht beachtet, da diese nur die Kantengewichte beeinflussen und somit keine Aussage über die Qualität des Verfahrens geben. Falls die Parameter notwendig sind, lassen sich diese ohne Weiteres beim Aufbauen der Graphdatenstruktur ergänzen.

5 Evaluation

In diesem Abschnitt wird das entwickelte Verfahren hinsichtlich seiner Laufzeit und der Qualität der Ergebnisse untersucht. Dazu wird zunächst auf die Methodik eingegangen, in der die genauen Ziele in Form von Forschungsfragen und das Testverfahren definiert werden. Abschließend werden die Ergebnisse dargestellt und die Forschungsfragen beantwortet.

5.1 Methodik

5.1.1 Forschungsfragen

1. Welche Laufzeit hat der Algorithmus?
2. Wie gut sind die Ergebnisse des Algorithmus?
3. Welchen Einfluss hat die Heuristik auf die Laufzeit und die Qualität des Ergebnisses?
4. Wie unterscheiden sich die Laufzeit und die Qualität der Ergebnisse bei Verwendung des klassischen CH-Algorithmus und der Version mit Independent-Sets (ISs)?
5. Inwieweit verbessert das Pruned Hub-Labeling die durchschnittliche Labelgröße und die Laufzeit des Hub-Labelings?
6. Wie vergleichen sich die Laufzeit und Ergebnisse mit dem Ansatz von S. Funke et. al. [FNS15]?

5.1.2 Experimentalaufbau

Alle hier verwendeten Algorithmen wurden in C++17 implementiert und mit gcc Version 9.4.0 kompiliert. CH und Hub-Labeling, das mit ISs arbeitet, wurden mit OpenMP parallelisiert. Alle Experimente wurden auf einem AMD Ryzen Threadripper 1950X mit 16 Kernen (32 Threads) und 256 GB RAM durchgeführt. Bei der Verwendung von OpenMP wurden 14 von den 32 verfügbaren Threads genutzt.

Die getesteten Graphen sind Straßennetzwerke und sind auf der Webseite des Instituts für Formale Methoden der Informatik der Universität Stuttgart [Ins22] zu finden. Diese Graphen basieren auf den Daten von OpenStreetMap [Ope23]. Im Folgenden sind die verwendeten Graphen und die Dimensionen dieser aufgelistet:

Name	Region	$\|V\|$	$\|E\|$	$k_{max,tt}$	$k_{max,m}$
STGT	Stuttgart Regierungsbezirk	1, 132, 113	2, 292, 887	16977	1199
BW	Baden-Württemberg	3, 600, 520	7, 300, 290	27090	1822
GER	Deutschland	25, 115, 477	50, 790, 030	74778	4013

Tabelle 5.1: Graphen und deren Eigenschaften: Es entspricht $|V|$ der Anzahl der Knoten, $|E|$ der Anzahl der Kanten, $k_{max,tt}$ der längsten Kante mit TT als Metrik und $k_{max,m}$ der längsten Kante mit der Distanz in Metern als Metrik.

Auf allen Graphen wurden alle möglichen Permutationen der folgenden Parameter durchgeführt:

- **Heuristik:**
  Dieser Parameter legt fest, welche Heuristik verwendet wird. Es werden die in Abschnitt 4.3 aufgeführten Heuristiken verwendet.
- **Distanzmetrik:**
  Bei diesem Parameter wird die verwendete Metrik/Kostenfunktion festgelegt. Die verwendeten Metriken sind die Reisezeit (TT) und die Distanz in Metern. Für die Umrechnung von TT in Meter wird die Formel aus Definition 3.4.1 verwendet.
- **IS:**
  Legt den definierten CH-Algorithmus fest. Es wird dabei jeweils der CH-Algorithmus ohne oder mit ISs verwendet. Wird die Version mit ISs verwendet, so wird der CH-Algorithmus und die Erstellung der Hub-Label mit OpenMP parallelisiert.

Der einzige Fall, in dem nicht alle Permutationen durchgeführt wurden, ist der GER Graph. Für diesen Graphen wurden die Heuristiken In and Out Product und Edge Difference (Heuristik 1 und 2) nicht getestet. Außerdem wurden nicht alle Fälle mit der Meter-Metrik getestet. Alle anderen Fälle wurden getestet.

Zusätzlich wurden zwei mögliche Verbesserungen am STGT Graphen mit der Heuristik 3 durchgeführt. Da diese Verbesserungen kein valides Ergebnis liefern, wurden sie in Kapitel 4 nicht behandelt. Diese Verbesserungen sind in Abschnitt 5.1.6 zu finden. Die Korrektheit der Verbesserungen wurde mit dem in Abschnitt 5.1.4 definierten Algorithmus getestet.

Die Reichweite der E-Fahrzeuge wurde für den STGT Graph auf 40 Kilometer und für die übrigen Graphen auf 125 Kilometer festgelegt. Diese Werte entsprechen den in [FNS15] verwendeten Reichweiten. Es muss jedoch eine Umrechnung zwischen TT und der Entfernung in Metern gefunden werden. Dazu wird für jeden Graphen ein Mittelwert berechnet, der mit dem Algorithmus aus Abschnitt 5.1.3 ermittelt wird.

Um die gestellten Forschungsfragen beantworten zu können, werden während jeder Permutation einige Statistiken berechnet und ausgegeben. Folgende Werte werden ausgegeben

- **Laufzeiten:**
  Für die folgenden Schritte werden die Laufzeiten ausgegeben:
  - Zeit zum Berechnen der CH
  - Zeit zur Erstellung der Hub-Labels

  - Zeit zur Berechnung der ESC-Menge

- **Durchschnittliche Labelgröße**
  Sei $label\_sum$ die Summe aller Fwd- und Bwd-Label über alle Knoten. Dann ist $avg\_labe\_size = \frac{label\_sum}{2 \cdot |V|}$ die durchschnittliche Labelgröße.
- **Größe der ESC-Menge**
  Sei C die Menge welche von Algorithmus 4.1 zurückgegeben wird. Es ist $|C|$ die Größe der ESC-Menge
- **Größe der unteren Schranke**
  Ausgehend von der ESC-Menge wird mit dem in Abschnitt 5.1.5 beschriebenen Algorithmus eine untere Schranke berechnet. Deren Größe ist ein Maß für die Qualität der ESC-Menge. Um die verschiedenen Permutationen vergleichen zu können, wird für jeden Graphen und jede Metrik die beste untere Schranke gewählt. Dabei haben die TT- und die Meter-Metrik jeweils unterschiedliche untere Schranken, da sich durch die Verwendung unterschiedlicher Metriken unterschiedliche kürzeste Pfade ergeben.
- **Konvertierungsrate TT zu Meter**
  Diese wird nur ausgegeben, wenn die Reisezeit als Distanzmetrik verwendet wird. Um sicherzustellen, dass alle Permutationen mit der TT-Metrik die gleiche Reichweite haben, wird für jeden Graphen eine durchschnittliche Konvertierungsrate bestimmt.

Dabei sind insbesondere die Laufzeiten und die Größen der ESC-Menge und der unteren Schranke relevant. Es kann der Quotient $q = \frac{|C|}{|lowerbound|}$ (wobei $lowerbound$ die untere Schranke darstellt) gebildet werden, der die Qualität der ESC-Menge repräsentiert. Der Quotient kann schließlich auch mit den Ergebnissen aus [FNS15] verglichen werden.

Abschließend werden die Laufzeit und die durchschnittliche Labelgröße des Pruned Hub-Labeling für einige Permutationen gemessen. Hierbei wird keine ESC-Menge und keine untere Schranke berechnet, da das Pruned Hub-Labeling keinen Einfluss auf diese hat. Das Ziel dabei ist es die Reduktion der Labelgröße und die Verbesserung der Laufzeit zu messen.

5.1.3 Konvertierung Reichweite von Meter zu Reisezeit

Algoritmus 5.1 liefert die Konvertierungsrate. Diese gibt an, wie viele Meter einem TT entsprechen. Gegeben sei nun die Reichweite μ in Meter und die Konvertierungsrate c. Um die entsprechende Reichweite μ_{TT} in TT zu erhalten, kann die Formel $\mu_{TT} = \frac{\mu}{c}$ verwendet werden.

Algorithmus 5.1 Berechung der Konvertierungsrate von Meter zu TT

```
1: procedure getConversionRate(graph_file)
2:     G_TT ← getTTGraph(graph_file)
3:     G_Meter ← getMeterGraph(graph_file)
4:     sum_TT ← 0.0
5:     sum_Meter ← 0.0
6:     iter ← 0
7:     while iter ≠ 100000 do
8:         (start, end) ← getRandomPair // Make sure to avoid duplicates
9:         // Dijkstra-Algorithmus (Dijkstra) returns distance. Other shortest path algorithms can
   also be used here
10:        // Also make sure distance is not infinity
11:        sum_TT ← sum_TT + Dijkstra(G_TT, start, end)
12:        sum_Meter ← sum_Meter + Dijkstra(G_Meter, start, end)
13:        iter ← iter + 1
14:    end while
15:    return sum_Meter / sum_TT
16: end procedure
```

5.1.4 Verifikation einer ESC-Menge

Die Idee der Verifikation besteht darin, von jedem Knoten bis zu einer bestimmten Länge eine Dijkstra-Suche durchzuführen und dann zu testen, ob sich auf jedem Pfad mindestens ein Knoten aus der berechneten ESC-Menge befindet. Sobald ein Pfad ohne einen Knoten aus der ESC-Menge gefunden wird, ist die ESC-Menge ungültig. Der Ansatz entspricht der ersten Idee, alle möglichen Pfade mit Hilfe des Dijkstra zu konstruieren. Da dieser, wie bereits diskutiert, eine extrem hohe Laufzeit hat, gilt das gleiche für die Verifikation. Für mittelgroße Graphen beträgt die Laufzeit mehrere Wochen (siehe [FNS15]). Aus diesem Grund wurde die Verifikation nur für sehr kleine Reichweiten durchgeführt, um das Verfahren und die fehlgeschlagenen Verbesserungen (Abschnitt 5.1.6) zu testen. Die genauen Schritte der Verifikation sind in Algorithmus 5.2 beschrieben.

Algorithmus 5.2 Verifikation einer ESC-Menge

```
1: procedure verifyESC(G = (V, E), C ⊆ V, μ > 0) // C has been calculated by algorithm 4.1
2:     for all v ∈ V do
3:         // modifiedDijkstra runs until it selects a node with distance > 1.1 · μ.
4:         // Additionaly keeps track of all paths and if they contain a node c ∈ C.
5:         // If length of a path is > μ and no node c ∈ C on path, then return false
6:         covered ← modifiedDijkstra(G, v, μ)
7:         if covered == false then
8:             return false // C is invalid ESC-set
9:         end if
10:    end for
11:    return true // C is valid ESC-set
12: end procedure
```

5.1.5 Berechnung der unteren Schranke

Um die echte untere Schranke zu erhalten, müsste man das minimale ESC finden. Dies ist jedoch aufgrund der NP-härte von ESC nicht möglich, weshalb eine Approximation notwendig ist. In dieser Ausarbeitung wird der Ansatz von S. Funke et. al. [FNS15] verwendet. Die genaue Berechnung ist nochmals für Reproduzierbarkeit in Algorithmus 5.3 gegeben.

Algorithmus 5.3 Berechnung der unteren Schranke

```
1:  procedure getLowerBound(G = (V, E), C ⊆ V, μ > 0) // C has been calculated by algorithm 4.1
2:      lower_bound ← ∅
3:      nodes_covered ← 0
4:      marked ← [false] · |V| // array of size |V| with values false
5:      while nodes_covered ≠ |C| do
6:          // selects node from C, which hasn't been selected yet
7:          cur_node ← randomSelect(C)
8:          nodes_covered ← nodes_covered + 1
9:          if marked[cur_node] == true then
10:             continue // goes directly to next iteration
11:         end if
12:         // modifiedDijkstra runs until it selects a node with distance > 2 · μ.
13:         // Additionaly keeps track of all paths and returns the shortest path with the minimal number of nodes in C and length > μ.
14:         path ← modifiedDijkstra(G, cur_node, μ)
15:         path_unmarked ← true
16:         for all v ∈ path do
17:             if marked[v] == true then
18:                 path_unmarked ← false
19:                 break // stop looping through path
20:             end if
21:         end for
22:         if path_unmarked == true then
23:             for all v ∈ path do
24:                 marked[v] ← true
25:             end for
26:             lower_bound ← lower_bound ∪ {cur_node}
27:         end if
28:     end while
29:     return |lower_bound|
30: end procedure
```

5.1.6 Fehlgeschlagene Verbesserungen

Insgesamt wurden zwei Verbesserungen getestet. Die erste Verbesserung bestand darin, den Bereich der Hub-Labels, die für die ESC-Menge verwendet werden, zu verkleinern. Dies betrifft die Zeilen neun und vierzehn des Algorithmus 4.1. Statt von $\frac{\mu}{2}$ bis μ geht der Bereich beispielsweise von $\frac{9\mu}{10}$ bis μ. Dass dieser Ansatz nicht funktioniert, lässt sich aus Theorem 1 ableiten.

Die zweite Verbesserung hat einen etwas komplexeren Ansatz. Die Idee besteht darin, für jeden Knoten der ESC-Menge C alle ein- und ausgehenden Pfade bis zur Länge $\frac{\mu}{2}$ zu betrachten. Wenn alle Pfade mindestens einen anderen Knoten aus der Menge C haben, dann ist der betrachtete Knoten nicht relevant und kann aus C entfernt werden. Alle Schritte sind nochmals unter Algorithmus 5.4 aufgeführt.

Da beide Ansätze nach dem Algorithmus 5.2 ein ungültiges Ergebnis liefern, werden sie in den Ergebnissen (Abschnitt 5.2) nicht weiter betrachtet.

Algorithmus 5.4 Fehlgeschlagende Verbesserung zwei

```
1:  procedure ImprovementTwo(G = (V, E), C ⊆ V, μ > 0) // C has been calculated by algorithm 4.1
2:      for all v ∈ C do
3:          // modifiedForwardDijkstra runs until it selects a node with distance > μ/2.
4:          // It works similarly to the bidirectional Dijkstra with CH but only doing the Fwd-search
5:          // Additionaly keeps track of all paths and if they contain a node c ∈ C, which is not v.
6:          // If there is at least one path with no such node, then return true, else return false
7:          fwd ← modifiedForwardDijkstra(G, v, μ)
8:          // modifiedBackwardDijkstra works the same but with Bwd-search
9:          bwd ← modifiedBackwardDijkstra(G, v, μ)
10:         if fwd == false ∧ bwd == false then
11:             C ← C − {v}
12:         end if
13:     end for
14:     return C
15: end procedure
```

5.2 Ergebnisse

Die Forschungsfragen aus Abschnitt 5.1.1 können nun beantwortet werden. Aus den Tabellen 5.2, 5.3 und 5.4 geht für die erste Forschungsfrage hervor, dass die Heuristik zwar die Laufzeit der CH-Berechnung erhöht, dafür aber die Laufzeit des Hub-Labelings stark reduziert. Das Hub-Labeling macht den größten Teil der Gesamtlaufzeit aus. Ein Grund dafür ist, dass durch eine bessere Heuristik das Hub-Labeling insgesamt kleiner wird und somit weniger Labels berechnet und überprüft werden müssen. Außerdem verkürzt die Verwendung von ISs durch die Parallelisierung die Laufzeit des Hub-Labeling-Algorithmus um ein Vielfaches. Allerdings steigt die Laufzeit des CH-Algorithmus wieder an. Der Grund dafür ist, dass in die Optimierung der CH-Implementierung mit ISs weniger Zeit investiert wurde als in die Version ohne ISs. Bei der Heuristik 4 ist dies nicht

der Fall. Die Laufzeit der CH-Berechnung wird bei dieser Heuristik durch ISs stark reduziert, da bei dieser Heuristik mehr Komponenten berechnet werden müssen als bei den anderen Heuristiken und somit die Heuristik 4 stärker von der Parallelisierung profitiert. Eine weitere Erkenntnis ist, dass die Bestimmung der ESC-Menge im Vergleich zur Berechnung der CH und des Hub-Labelings kaum Zeit benötigt. Es ist hier wichtig zu erwähnen, dass die Laufzeiten für den Deutschland Graphen im Vergleich zu anderen Verfahren sehr kurz sind. Somit ist es nun möglich, eine ESC-Menge für relativ große Graphen in kurzer Zeit zu bestimmen.

Heuristik	IS	Distanzmetrik	CH	Hub-Label	ESC	Summe
In and Out (1)	nein	TT	4	83	2	89
In and Out (1)	nein	Meter	5	122	2	129
Edge Difference (2)	nein	TT	9	76	1.5	86.5
Edge Difference (2)	nein	Meter	10	115	2	127
Max. Kosten (3)	nein	TT	13	64	1	78
Max. Kosten (3)	nein	Meter	12	93	2	107
Mixed (4)	nein	TT	32	57	1	90
Mixed (4)	nein	Meter	43	91	2	136
In and Out (1)	ja	TT	5	16	2	23
In and Out (1)	ja	Meter	6	18	2	26
Edge Difference (2)	ja	TT	10	11	1.5	22.5
Edge Difference (2)	ja	Meter	11	16	2	29
Max. Kosten (3)	ja	TT	20	10	1	31
Max. Kosten (3)	ja	Meter	24	17	2	43
Mixed (4)	ja	TT	16	7	1	24
Mixed (4)	ja	Meter	17	10	1.5	28.5

Tabelle 5.2: Laufzeiten Stuttgart Graph: Alle Laufzeiten in Sekunden.

Heuristik	IS	Distanzmetrik	CH	Hub-Label	ESC	Summe
Max. Kosten (3)	nein	TT	7.5	86	1	94.5
Max. Kosten (3)	nein	Meter	8.3	221	2.35	231.65
Mixed (4)	nein	TT	39	103	1.2	143.2
Mixed (4)	nein	Meter	83.3	255	2.2	340.5
Max. Kosten (3)	ja	TT	12	11.3	1	24.3
Mixed (4)	ja	TT	9.85	7.7	0.85	18.4

Tabelle 5.3: Laufzeiten Deutschland Graph: Alle Laufzeiten in Minuten.

Heuristik	IS	Distanzmetrik	CH	Hub-Label	ESC	Summe
In and Out (1)	nein	TT	17	564	9	590
In and Out (1)	nein	Meter	20	912	10	942
Edge Difference (2)	nein	TT	35	371	5	411
Edge Difference (2)	nein	Meter	41	709	8.5	758.5
Max. Kosten (3)	nein	TT	49	315	3	367
Max. Kosten (3)	nein	Meter	48	560	7.5	615.5
Mixed (4)	nein	TT	128	304	3	435
Mixed (4)	nein	Meter	195	582	6.5	783.5
In and Out (1)	ja	TT	20	83	7	110
In and Out (1)	ja	Meter	28	144	12	184
Edge Difference (2)	ja	TT	38	47	4.5	89.5
Edge Difference (2)	ja	Meter	45	85	9	139
Max. Kosten (3)	ja	TT	77	43	3	123
Max. Kosten (3)	ja	Meter	97	87	8	192
Mixed (4)	ja	TT	60	33	3	96
Mixed (4)	ja	Meter	66	59	6	131

Tabelle 5.4: Laufzeiten Baden-Württemberg Graph: Alle Laufzeiten in Sekunden.

Mit den Ergebnissen der Tabellen 5.5, 5.6 und 5.7 kann nun die zweite Forschungsfrage beantwortet werden. Wie zu erwarten, hat die Heuristik einen großen Einfluss auf die durchschnittliche Größe der Hub-Label und ESC-Menge. Dabei liefert die Heuristik 3 die besten Ergebnisse für die ESC-Menge, wenn keine ISs verwendet werden. Entsprechend liefert die Heuristik 4 die besten Ergebnisse für die ESC-Menge, wenn ISs verwendet werden. Im Allgemeinen sind die Ergebnisse ohne ISs jedoch besser, obwohl in vielen Fällen die durchschnittliche Labelgröße mit ISs kleiner ist. Bei der Betrachtung des Quotienten q und der Heuristiken 3 und 4 fällt auf, dass die Ergebnisse für den BW Graphen im Allgemeinen etwas schlechter sind als die Ergebnisse für den STGT oder GER Graphen. Ein Grund dafür ist, dass die Gewichtungen in den Heuristiken (z.B. 0.8 Gewichtung in Heuristik 3) durch die besten Ergebnisse auf dem STGT Graphen bestimmt wurden. Daher ist es wahrscheinlich, dass mit den gleichen Heuristiken und einer anderen Gewichtung ein besseres Ergebnis auf dem BW Graphen erzielt werden kann. Dazu lässt sich erkennen, dass der Quotient q beim Verwenden der Meter-Metrik immer besser ausfällt als bei der TT-Metrik. Ein Beispiel dafür lässt sich in Tabelle 5.5 in den Zeilen fünf und sechs finden. Der Quotient ist hier sehr unterschiedlich, obwohl die ESC-Mengen fast gleich groß sind. Der Grund dafür ist, dass die untere Schranke der TT-Metrik sich nicht als untere Schranke für die Meter-Metrik verwenden werden kann, da sich bei der Verwendung unterschiedlicher Metriken neue kürzeste Pfade entstehen und somit eine andere untere Schranke entsteht, welche, wie in den Tabellen, in der Regel größer ist als die untere Schranke mit der TT-Metrik. Zusammenfassend lässt sich sagen, dass das vorgestellte Verfahren mit der Heuristik 3 ohne ISs bei der TT-Metrik um einen Faktor sechs bis acht und mit der Heuristik 4 und ISs um einen Faktor vier bis sechs von der unteren Schranke entfernt ist. Des weiteren lassen sich mit der Heuristik 4 und ISs etwas schlechtere Ergebnisse mit einer kürzeren Laufzeit erreichen.

Die dritte Forschungsfrage wurde bereits indirekt beantwortet. Eine gute Heuristik reduziert die Laufzeit des Hub-Labelings stark und führt zu einem besseren Ergebnis für das Hub-Labeling und die ESC-Menge. Dabei ist die Heuristik 3 am besten, wenn keine ISs verwendet werden und die Heuristik 4 am besten, wenn diese verwendet werden.

Durch die Verwendung von ISs kann das Hub-Labeling parallelisiert und somit die Laufzeit um ein Vielfaches reduziert werden. Dabei führt die Heuristik 4 für die mittlere Labelgröße zu den besten Ergebnissen. Allerdings sind die Ergebnisse für die ESC-Menge und damit für den Quotienten q bei Verwendung von ISs etwas schlechter.

Analog wurde auch schon die vierte Forschungsfrage indirekt beantwortet. Die Verwendung von ISs erlaubt es das Hub-Labeling zu parallelisieren und somit die Laufzeit um ein vielfaches zu reduzieren. Dazu führt die Heuristik 4 zu den besten Ergebnissen für die durchschnittliche Labelgröße. Jedoch sind die Ergebnisse für die ESC-Menge und somit für den Quotienten q beim Verwenden von ISs etwas schlechter.

Heuristik	**IS**	**Metrik**	**TT zu Meter**	**R in TT/m**	**L-Größe**	**|C|**	**|LB|**	**q = $\frac{\|C\|}{\|LB\|}$**
(1)	nein	TT	0.223457	179005	92	951	104	9.14
(1)	nein	Meter	-	40000	111	1007	133	7.57
(2)	nein	TT	0.223457	179005	84	1130	104	10.86
(2)	nein	Meter	-	40000	102	1229	133	9.24
(3)	nein	TT	0.223457	179005	68	628	104	6.04
(3)	nein	Meter	-	40000	85	634	133	4.76
(4)	nein	TT	0.223457	179005	67	1111	104	10.68
(4)	nein	Meter	-	40000	88	1274	133	9.58
(1)	ja	TT	0.223457	179005	90	1043	104	10.03
(1)	ja	Meter	-	40000	103	1137	133	8.55
(2)	ja	TT	0.223457	179005	73	1024	104	9.84
(2)	ja	Meter	-	40000	94	1062	133	7.98
(3)	ja	TT	0.223457	179005	68	889	104	8.54
(3)	ja	Meter	-	40000	96	973	133	7.32
(4)	ja	TT	0.223457	179005	56	713	104	6.86
(4)	ja	Meter	-	40000	73	824	133	6.19

Tabelle 5.5: Ergebnisse Stuttgart Graph: TT zu Meter $\widehat{=}$ Konvertierungsrate, R $\widehat{=}$ Reichweite, L-Größe $\widehat{=}$ durschnittlichen Labelgröße und LB $\widehat{=}$ untere Schranke

Heuristik	IS	Metrik	TT zu Meter	R in TT/m	L-Größe	|C|	|LB|	q = $\frac{\|C\|}{\|LB\|}$
(3)	nein	TT	0.229944	543609	129	2517	406	6.38
(3)	nein	Meter	-	125000	226	4163	889	4.68
(4)	nein	TT	0.229944	543609	154	6528	406	16.08
(4)	nein	Meter	-	125000	248	9366	889	10.53
(3)	ja	TT	0.229944	543609	139	3836	406	9.45
(4)	ja	TT	0.229944	543609	117	3280	406	8.07

Tabelle 5.6: Ergebnisse Deutschland Graph: TT zu Meter ≙ Konvertierungsrate, R ≙ Reichweite, L-Größe ≙ durschnittlichen Labelgröße und LB ≙ untere Schranke

Heuristik	IS	Metrik	TT zu Meter	R in TT/m	L-Größe	|C|	|LB|	q = $\frac{\|C\|}{\|LB\|}$
(1)	nein	TT	0.221872	563387	134	537	32	16.78
(1)	nein	Meter	-	125000	168	756	69	10.95
(2)	nein	TT	0.221872	563387	103	558	32	17.44
(2)	nein	Meter	-	125000	142	888	69	12.87
(3)	nein	TT	0.221872	563387	84	242	32	7.56
(3)	nein	Meter	-	125000	121	417	69	6.04
(4)	nein	TT	0.221872	563387	88	475	32	14.84
(4)	nein	Meter	-	125000	128	795	69	11.52
(1)	ja	TT	0.221872	563387	133	651	32	20.34
(1)	ja	Meter	-	125000	169	930	69	9.13
(2)	ja	TT	0.221872	563387	97	491	32	15.34
(2)	ja	Meter	-	125000	134	784	69	11.36
(3)	ja	TT	0.221872	563387	87	422	32	13.18
(3)	ja	Meter	-	125000	136	746	69	10.81
(4)	ja	TT	0.221872	563387	74	374	32	11.68
(4)	ja	Meter	-	125000	104	538	69	7.79

Tabelle 5.7: Ergebnisse Baden-Württemberg Graph: TT zu Meter ≙ Konvertierungsrate, R ≙ Reichweite, L-Größe ≙ durschnittlichen Labelgröße und LB ≙ untere Schranke

Tabelle 5.8 zeigt die Ergebnisse für die fünfte Forschungsfrage. Es sind nicht alle Permutationen aufgelistet, da sich die Verbesserungen bei den anderen Permutationen ähnlich verhalten. Wenn die Reichweite μ im Vergleich zum Durchmesser des Graphen relativ klein ist, dann lässt sich einiges an Speicher sparen. Für die Graphen STGT und GER sind es in diesem Fall ca. 20%. Beim BW Graph ist die Einsparung nicht sehr groß, da die verwendete Reichweite nah an den Durchmesser von Baden-Württemberg heran kommt. Ebenso wird beim BW Graph nicht viel Laufzeit eingespart, da die Labelgröße in etwa gleich bleibt. Ähnliches gilt für die Laufzeit auf dem STGT Graphen, da die Laufzeit für kleine Graphen bereits ohne Pruned Hub-Labeling gering ist. Für große Graphen wird dagegen einiges an Laufzeit eingespart. Dies zeigt sich insbesondere bei der Version ohne ISs. Hier können für den GER Graphen ca. 30% an Laufzeit eingespart werden.

Graph	Heuristik	IS	Metrik	μ in TT/m	L-Größe	Hub-Label Berechnungszeit
STGT	(3)	nein	TT	179005	59	61 **s**
STGT	(3)	nein	Meter	40000	69	72 **s**
STGT	(4)	ja	TT	179005	48	7 **s**
STGT	(4)	ja	Meter	40000	59	9 **s**
BW	(3)	nein	TT	563387	83	5.85 **min**
BW	(3)	nein	Meter	125000	114	8.58 **min**
BW	(4)	ja	TT	563387	72	0.57 **min**
BW	(4)	ja	Meter	125000	96	0.89 **min**
GER	(3)	nein	TT	543609	102	59.44 **min**
GER	(4)	ja	TT	125000	92	6.30 **min**

Tabelle 5.8: Laufzeiten und Ergebnisse Pruned Hub-Labeling: Es entspricht L-Größe der durschnittlichen Labelgröße.

Abschließend kann die letzte Forschungsfrage beantwortet werden. Die Laufzeit des hier vorgestellten Verfahrens ist in allen Fällen schneller als das Verfahren von S. Funke et. al. [FNS15]. Für große Graphen ist das Verfahren um ein Vielfaches schneller. Nimmt man z.B. den GER Graphen, der in [FNS15] nur aus 17.7 Millionen Knoten besteht, so beträgt die Laufzeit ohne ISs mit der Heuristik 3 und der TT-Metrik 94.5 Minuten. Das ist mehr als doppelt so schnell wie die schnellste parallelisierte Zeit aus [FNS15]. Vergleicht man dies mit der nicht parallelisierten Zeit, so ist das hier angegebene Verfahren fast siebenmal so schnell. Für diesen Test aus [FNS15] ist die Qualität der ESC-Mengen sogar identisch. Der Algorithmus aus [FNS15] erreicht für den GER Graph einen besten Quotienten von $q = 3.83$. Mit diesem kann das hier vorgestellte Verfahren mit den gegebenen Heuristiken nicht mithalten. Allerdings erhöht sich auch hier die Laufzeit für die nicht-parallele Zeit um das Doppelte und für die parallele Zeit um ca. 50%. Für kleinere Graphen sind die Laufzeitunterschiede ähnlich. Für diese liefert das Verfahren von S. Funke et. al. jedoch zuverlässigere Ergebnisse mit einem Quotienten im Bereich von drei bis vier. Bezüglich des Speicherverbrauchs verhält sich das hier vorgestellte Verfahren vermutlich etwas effizienter, wenn man es mit dem Verfahren von [FNS15] vergleicht. Nimmt man z.B. den BW-Graphen welcher in [FNS15] ca. 14 GB braucht, ist dieses deutlich größer als die ca. 10 GB mit der Heuristik 3 ohne ISs. Dazu kann durch das Pruned Hub-Labeling der Speicherverbrauch um weitere 20% und die Laufzeit um bis zu 30% reduziert werden, wodurch die Laufzeit- und Speicherunterschiede noch größer werden. Da der Speicherverbrauch in [FNS15] für den GER Graphen nicht gegeben ist, lässt sich jedoch diese Aussage bezüglich des Speicherverbrauchs nicht verallgemeinern. Zusammenfassend kann also gesagt werden, dass das hier vorgestellte Verfahren eine deutlich bessere Laufzeit mit etwas schlechteren Ergebnissen aufweist.

Die schlechteren Ergebnisse können auch positiv interpretiert werden. Denn rein realistisch betrachtet sind die 728 (bestes Ergebnis aus [FNS15] für GER Graph) Ladestationen für ein Land von der Größe Deutschlands zu wenig. E-Fahrzeuge brauchen im Vergleich zu Fahrzeugen mit anderen Kraftstoffen viel länger, um ihre Reichweite aufzuladen. Dies kann zu Staus an den Ladestationen führen, da dort mehr Zeit verbracht werden muss. Niemand möchte jedoch stundenlang sein Fahrzeug aufladen. Dieses Problem lässt sich durch mehr Ladestationen lösen, die man implizit

durch das hier vorgestellte Verfahren bekommt. In der Realität dürften die 2.517 (Tabelle 5.6, Zeile 1) mit diesem Verfahren platzierten Ladestationen immer noch nicht ausreichen, wenn man sie mit der Anzahl der Tankstellen (14.107 Stand 2021 [ADA21]) vergleicht.

6 Zusammenfassung und Ausblick

In dieser Ausarbeitung wurde das Problem der Platzierung von Ladestationen für E-Fahrzeuge untersucht. Dabei wurde das Problem so eingeschränkt, dass sich auf jedem kürzesten Pfad, der länger als die vorgegebene Reichweite ist, genug Ladestationen vorhanden sind. Dadurch werden Umwege zum Laden auf längeren Strecken vermieden. Es wurde ein Verfahren, basierend auf hierarchischen Hub-Labels, entwickelt und mathematisch gezeigt, dass die Menge der Ladestationen aus den Hub-Labels extrahiert werden kann. Anschließend wurde die Implementierung des Verfahrens ausführlich mit verschiedenen Heuristiken, Distanzmetriken und Ansätzen für die Berechnung der Contraction Hierarchies getestet. Dazu wurden erste Verbesserungen des Verfahrens getestet. Im Vergleich zu anderen Ansätzen, die das gleiche Problem lösen, liefert das vorgestellte Verfahren eine sehr große Laufzeitverbesserung bei einer leichten Verschlechterung der Ergebnisse. Die Laufzeitunterschiede werden noch größer, wenn das vorgestellte Pruned Hub-Labeling verwendet wird. Dazu wird bei diesem der Speicherverbrauch reduziert.

In zukünftigen Ausarbeitungen können einige Verbesserungen und Erweiterungen des vorgestellten Verfahrens untersucht werden. Eine mögliche Verbesserung besteht darin, bessere Heuristiken zu finden. Insbesondere eine Heuristik, die für die Version mit Independent-Sets ein besseres Ergebnis liefert, das mit der Version ohne Independent-Sets mithalten kann. Außerdem kann eine gute Heuristik den Speicherverbrauch und die Laufzeit des Hub-Labelings drastisch reduzieren. Des Weiteren kann die Auswirkung von Verfahren, die im Machine Learning automatisch die besten Parameter für gegebene Datensätze finden, auf den vorgestellten Heuristiken und mit weiteren Graphen getestet werden. Dadurch können ggf. die besten Gewichtungen der Heuristik für jeden Graphen automatisch gefunden werden. Schließlich lässt sich das Verfahren durch das Einbeziehen verschiedener Parameter, die für E-Fahrzeuge relevant sind, erweitern. Zu diesen Parametern zählen die Steigung, die Jahreszeit, die Art der Ladestation, die Anbindung an das Stromnetz oder auch dynamische Parameter, wie das Verhalten der Fahrer von E-Fahrzeugen.

Literaturverzeichnis

[ADA21] ADAC. *So viele Tankstellen gibt es in Deutschland*. 2021. URL: https://www.adac.de/verkehr/tanken-kraftstoff-antrieb/deutschland/tankstellen-in-deutschland/ (zitiert auf S. 38).

[ADF+16] I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, R. F. Werneck. „Highway dimension and provably efficient shortest path algorithms“. In: *Journal of the ACM (JACM)* 63.5 (2016), S. 1–26 (zitiert auf S. 21).

[ADGW11] I. Abraham, D. Delling, A. V. Goldberg, R. F. Werneck. „A hub-based labeling algorithm for shortest paths in road networks“. In: *Experimental Algorithms: 10th International Symposium, SEA 2011, Kolimpari, Chania, Crete, Greece, May 5-7, 2011. Proceedings 10*. Springer. 2011, S. 230–241 (zitiert auf S. 23, 25).

[ADGW12] I. Abraham, D. Delling, A. V. Goldberg, R. F. Werneck. „Hierarchical hub labelings for shortest paths“. In: *European Symposium on Algorithms*. Springer. 2012, S. 24–35 (zitiert auf S. 18).

[CHKZ03] E. Cohen, E. Halperin, H. Kaplan, U. Zwick. „Reachability and distance queries via 2-hop labels“. In: *SIAM Journal on Computing* 32.5 (2003), S. 1338–1355 (zitiert auf S. 18).

[DIJ59] E. DIJKSTRA. „A Note on Two Problems in Connexion with Graphs“. In: *Numerische Mathematik* 1 (1959), S. 269–271 (zitiert auf S. 18).

[FNS15] S. Funke, A. Nusser, S. Storandt. „Placement of loading stations for electric vehicles: No detours necessary!“ In: *Journal of Artificial Intelligence Research* 53 (2015), S. 633–658 (zitiert auf S. 13, 15–17, 21, 26–31, 37).

[FNS16] S. Funke, A. Nusser, S. Storandt. „Placement of loading stations for electric vehicles: Allowing small detours“. In: *Proceedings of the international conference on automated planning and scheduling*. Bd. 26. 2016, S. 131–139 (zitiert auf S. 13).

[GSSD08] R. Geisberger, P. Sanders, D. Schultes, D. Delling. „Contraction hierarchies: Faster and simpler hierarchical routing in road networks“. In: *Experimental Algorithms: 7th International Workshop, WEA 2008 Provincetown, MA, USA, May 30-June 1, 2008 Proceedings 7*. Springer. 2008, S. 319–333 (zitiert auf S. 19, 23).

[Ins22] U. S. Institut für formale Methoden der Informatik. *Useful Stuff*. 2022. URL: https://fmi.uni-stuttgart.de/alg/research/stuff/ (zitiert auf S. 27).

[KAR72] R. KARP. „Reducibility among combinatorial problems“. In: *Complexity of Computer Computation* (1972), S. 85–104 (zitiert auf S. 17).

[KLSV10] T. Kieritz, D. Luxen, P. Sanders, C. Vetter. „Distributed time-dependent contraction hierarchies“. In: *Experimental Algorithms: 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings 9*. Springer. 2010, S. 83–93 (zitiert auf S. 19).

[LLC14] A. Y. Lam, Y.-W. Leung, X. Chu. „Electric vehicle charging station placement: Formulation, complexity, and solutions“. In: *IEEE Transactions on Smart Grid* 5.6 (2014), S. 2846–2856 (zitiert auf S. 13).

[NAS23] NASA. *Solar System Exploration: Planet Compare*. 2023. URL: https://solarsystem.nasa.gov/planet-compare/ (zitiert auf S. 18).

[Ope23] OpenStreetMap contributors. *Planet dump retrieved from https://planet.osm.org*. https://www.openstreetmap.org. 2023 (zitiert auf S. 27).

[Sin84] R. W. Sinnott. „Virtues of the Haversine“. In: *Sky and telescope* 68.2 (1984), S. 158 (zitiert auf S. 18).

[XGA+17] Y. Xiong, J. Gan, B. An, C. Miao, A. L. Bazzan. „Optimal electric vehicle fast charging station placement based on game theoretical framework“. In: *IEEE Transactions on Intelligent Transportation Systems* 19.8 (2017), S. 2493–2504 (zitiert auf S. 13).

[ZIK+20] M. Z. Zeb, K. Imran, A. Khattak, A. K. Janjua, A. Pal, M. Nadeem, J. Zhang, S. Khan. „Optimal placement of electric vehicle charging stations in the active distribution network“. In: *IEEE Access* 8 (2020), S. 68124–68134 (zitiert auf S. 13).

Alle URLs wurden zuletzt am 07. 08. 2023 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift