Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik
Professur für Algorithmen und Datenstrukturen

Bachelorarbeit

Wohin mit den Ladesäulen?

Lösungsansätze zur optimalen Platzierung von Ladeinfrastruktur für Elektrofahrzeuge

vorgelegt von

Katharina Matulla

betreut von

Patrick Brosi

geprüft von

Prof. Hannah Bast

Diese Bachelorarbeit ist entstanden in Kooperation mit dem
Fraunhofer-Institut für Solare Energiesysteme (ISE)

Einreichung in Freiburg, Juni 2019

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum: _____

Unterschrift: _____

Abstract

Recent growth of the electric vehicles – especially electric cars – market calls for setting up an efficient infrastructure for electric charging stations that doesn't only accommodate this growth but also accounts for optimizing implementation costs and reducing drivers' range anxiety. While there already exist multiple methods for calculating optimal locations for charging stations, most of these methods require costly or unavailable data. This thesis showcases three different algorithms that rely only on open-source Points of Interest (POIs) from OpenStreetMap (OSM) to calculate optimal placements of electric charging stations in a given area. The first algorithm is k-means clustering based on Euclidean distances. The second is a clustering method based on the betweenness centrality of POIs in a graph using shortest paths. The third is a genetic algorithm which solves the p-median problem while also using shortest paths. All three algorithms lead to consistent output; clusters of POIs where one POI is chosen as an optimal charging station for its cluster. For the purpose of this thesis, the aforementioned algorithms were applied to three areas: North-Amsterdam, North-Seattle and Davutpaşa-Campus (Istanbul). The computed optimal outputs in all three areas are displayed on an interactive map in a webpage and are evaluated against optimal solutions gathered from past research.

This undergraduate thesis is developed in connection to a project managed by the Fraunhofer Institute for Solar Energy Systems (ISE) called „EnStadt:Pfaff". The project aims to transform a former industrial area in Kaiserslautern, Germany into a modern, climate neutral, urban quarter. The work presented here is supposed to not only be applicable for this area but several other areas in the future.

Contact

Katharina Matulla (author)– katharina.matulla@googlemail.com

Patrick Brosi (supervisor) – brosi@informatik.uni-freiburg.de

Matti Sprengeler (Fraunhofer ISE) – matti.sprengeler@ise.fraunhofer.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der marktdurchbrechende Hype um Elektroautos als umweltschonende Alternative zu konventionellen Fahrzeugen lässt länger auf sich warten als zunächst angenommen. Ehrgeizige politische Ziele, wie beispielsweise die Erhöhung von derzeit etwa 83.000 Elektroautos auf Deutschlands Straßen auf ca. 1.000.000 Elektroautos bis im Jahr 2020, drohen mit Sicherheit zu scheitern (Statista GmbH (Hrsg.), 2018; Spiegel ONLINE (Hrsg.), 2018; Zeit ONLINE (Hrsg.), 2018). Um die Ziele dennoch zu erreichen, sind einige Hindernisse aus dem Weg zu räumen. Vor allem die Dauer des Ladevorgangs und die Reichweitenangst werden häufig als Gründe genannt, keine Umstellung auf Elektrofahrzeuge vorzunehmen (Nationale Plattform Elektromobilität NPE, 2018; Spiegel ONLINE (Hrsg.), 2018; Zeit ONLINE (Hrsg.), 2018). Um diesen beiden Argumenten entgegenzutreten und den zukünftigen Bedarf an Ladeenergie zu decken, ist ein durchdachter Ausbau der Ladeinfrastruktur vonnöten wozu auch die Bereitstellung von Ladesäulen für Elektroautos gehört (Nationale Plattform Elektromobilität NPE, 2018). Doch wohin mit den Ladesäulen? Wie können diese so platziert werden, dass der Bedarf gedeckt ist und die Kosten für die Platzierung minimal sind?

Diese Fragen werden unter anderem vom Fraunhofer-Institut für Solare Energiesysteme (ISE) im Zusammenhang mit dem Leuchtturmprojekt „EnStadt:Pfaff“ bearbeitet, in dessen Rahmen diese Bachelorarbeit eingebunden ist. Beim Projekt „EnStadt:Pfaff“ geht es darum, ein klimaneutrales Stadtquartier auf einem ehemaligen Fabrikgelände in Kaiserslautern zu entwickeln, das derzeit brach liegt. Hierzu gehört auch die Berücksichtigung der zukünftigen Elektromobilität (Stadt Kaiserslautern (Hrsg.), 2019). Das übergeordnete Ziel des Fraunhofer ISE ist es, ein Tool zu entwickeln, dass geeignete Ladestandorte in unterschiedlichen Gebieten berechnen kann, wobei das Pfaff-Gelände nun das erste Gebiet ist, in dem das Tool angewendet werden soll. Um dieses Tool zu entwickeln, sind jedoch einige Vorarbeiten nötig, die in dieser Bachelorarbeit erfolgen.

Das Ziel dieser Arbeit ist demnach die Beantwortung folgender Leitfragen:

- Welche frei verfügbaren Daten können zur Berechnung von optimalen Ladestandorten in verschiedenen Gebieten verwendet werden?
- Welche Algorithmen eignen sich zur Berechnung von optimalen Ladestandorten?

- Wie können die berechneten optimalen Ladestandorte in Hinblick auf ihre Qualität evaluiert werden?

Die Bacheloarbeit ist dabei in folgende Kapitel unterteilt:
Nach der Einleitung in Kapitel 1 werden in Kapitel 2 Grundlagen zur Ladeinfrastrukturforschung erläutert. In Kapitel 2.1 werden dabei die zentralen Aspekte der bisherigen Forschung zur Platzierung von Ladeinfrastruktur dargelegt. Kapitel 2.2 widmet sich dann einigen ausgewählten Forschungsarbeiten, aus denen entweder Konzepte für die im Rahmen dieser Arbeit erfolgte Implementierung oder optimale Lösungen zur Evaluation entnommen werden. Damit die Ergebnisse dieser Arbeit nachvollziehbar, reproduzierbar und hinsichtlich ihrer Qualität beurteilbar sind, werden in Kapitel 3 die Implementierung und die dahinter liegenden algorithmischen Theorien beschrieben. Hierzu werden in Kapitel 3.1 die verwendeten Daten und Datenstrukturen vorgestellt sowie in Kapitel 3.2 die Funktionsweise und Eigenschaften der drei implementierten Algorithmen erläutert. Anschließend wird in Kapitel 3.3 auf die Auswahl der Untersuchungsgebiete eingegangen und in Kapitel 3.4 die Funktionalität der Webseiten zur Darstellung der Ergebnisse beschrieben. In Kapitel 3.5 werden dann drei verschiedene Ansätze zur empirischen Analyse der Ergebnisse erläutert. Kapitel 4 widmet sich schließlich den Ergebnissen und der Ergebnisevaluation der drei Untersuchungsgebiete Nord-Amsterdam, Nord-Seattle und des Davutpaşa-Campus, wobei in Kapitel 4.4 ein zusammenfassender Vergleich der Ergebnisse vorgenommen wird. Abschließend werden im Fazit in Kapitel 5.1 die in dieser Einleitung aufgeworfenen Leitfragen zusammenfassend beantwortet und in Kapitel 5.2 ein Ausblick gegeben.

2 Grundlagen zur Ladeinfrastrukturforschung

Um die Arbeit in den aktuellen Forschungskontext einzuordnen und eine Wissensgrundlage über die Planung von Ladeinfrastruktur zu schaffen, werden in Kapitel 2.1 die zentralen Forschungsschwerpunkte dieses Forschungsgebietes beschrieben. Anschließend werden in Kapitel 2.2 Lösungskonzepte derjenigen Forschungsarbeiten vorgestellt, die entweder als Inspiration für die Implementierung dienen oder die optimale Lösungen bereitstellen, welche zur qualitativen Evaluation der im Rahmen dieser Arbeit berechneten Ergebnisse dienen.

2.1 Zusammenstellung zentraler Forschungsschwerpunkte

Der Forschungsaufwand und damit einhergehend die Anzahl der veröffentlichten Publikationen zur Planung von Ladeinfrastruktur für Elektrofahrzeuge wächst seit 2008 nahezu exponentiell an (Pagany et al., 2018, S. 3). Die meisten Forschungsergebnisse und Fallstudien werden in den Ländern China, USA und Deutschland bzw. Westeuropa produziert, in denen auch der größte Markt an Elektromobilität herrscht (Pagany et al., 2018, S. 10). Einen guten Überblick über die bisherigen Forschungsarbeiten und -ergebnisse liefern Pagany et al., Shareef et al. und Rahman et al. in Review-Publikationen. Die für diese Arbeit bedeutenden Inhalte dieser Publikationen werden in diesem Kapitel näher erläutert. Diese sind:

- Definition von Ladeinfrastruktur
- Anzahl und Art von Ladestandorten
- Die Bedeutung von Points of Interest (POIs)
- Datenquellen
- Typische Optimierungsziele und Optimierungsalgorithmen
- Untersuchungsgebiete

Definition von Ladeinfrastruktur

Es gibt keine allgemeingültige Definition von Ladeinfrastruktur. Eine umfassende Ladeinfrastruktur für Elektroautos besteht jedoch aus mehr als nur der optimalen

Platzierung von Ladestandorten in einem Gebiet. Neben der Platzierung gehören beispielsweise auch die Bestimmung des Bedarfs an elektrischer Energie sowie die Anzahl und Art an benötigten Ladesäulen und Ladepunkten zu einer Ladeinfrastruktur (Gnann et al., 2017, S. 5; Shareef et al., 2016, S. 404). Ein Ladestandort – wie zum Beispiel ein Parkplatz – kann dabei aus mehreren Ladesäulen bestehen, die wiederum mehrere Ladepunkte umfassen können. An jedem Ladepunkt kann genau ein Fahrzeug laden (Gnann et al., 2017, S. 5). Ein weiterer Aspekt, der ebenfalls in einigen Forschungsarbeiten zur Ladeinfrastruktur berücksichtigt wird, ist die Eignungsprüfung des Stromnetzes, den Mehrenergiebedarf an entsprechenden Standorten zur Verfügung zu stellen (Shareef et al., 2016, S. 413). Während energetische Fragestellungen im Rahmen dieser Arbeit nicht weiter beachtet werden, spielt die Anzahl und Art der zu installierenden Ladestandorte eine Rolle bei der Berechnung der optimalen Platzierung.

Anzahl und Art von Ladestandorten

Die Anzahl der benötigten Ladesäulen bzw. Ladestandorte wird häufig mittels Annahmen zur Marktdurchdringung von Elektrofahrzeugen approximiert, wobei die Annahmen von ca. sechs Prozent bis zu 100 Prozent, also von einer marginalen bis hin zu einer kompletten Umstellung auf Elektroautos reichen. Von der Anzahl der Elektroautos wird dann - häufig unter Zuhilfenahme von Fahrprofilen - auf die benötigte Energiemenge und damit auf die Anzahl der benötigten Ladestandorte geschlossen (Pagany et al., 2018, S. 6f.). In dieser Arbeit wird die Anzahl der benötigten Ladestandorte aus den Ergebnissen anderer Forschungsarbeiten entnommen (vgl. Kap. 2.2, Kap. 3.5.1). Hierbei werden jedoch keine Annahmen über die Anzahl der benötigten Ladestationen bzw. Ladepunkte getroffen, sondern lediglich über optimale Ladestandorte. Einige Forschungsarbeiten berechnen zudem noch zeitliche Bedarfsschwankungen im Tagesablauf, die zum Beispiel durch den Berufsverkehr verursacht werden (Chen et al., 2013, S. 36). Dies wird im Rahmen dieser Arbeit jedoch aufgrund fehlender Daten nicht berücksichtigt.

Die Art der zu installierenden Ladesäulen richtet sich nach der Bedarfsmenge und der Verweildauer an einem Ladestandort. Im Gegensatz zu herkömmlichen Fahrzeugen dauert der Ladevorgang für Elektrofahrzeuge in Abhängigkeit der Ladetechnik zwischen 20 Minuten (Level 3-Ladesäulen) bis zu acht Stunden (Level 2-Ladesäulen) oder 16 Stunden (Level 1-Ladesäulen) (Shareef et al., 2016, S. 413f.). Während Ladestationen zu Hause

oder am Arbeitsplatz mit einer mehr als achtstündigen Ladezeit akzeptabel sind, sind für den öffentlichen Raum erst Ladestationen mit einer maximalen Ladezeit von 8 Stunden (Level 2) sinnvoll (Brooker and Qin, 2015, S. 84; Xi et al., 2013, S. 65). Da in dieser Arbeit die Ladeinfrastruktur im öffentlichen Raum betrachtet und eine maximale Ladezeit von acht Stunden angenommen wird, sind die berechneten Ladestandorte für Level 2 oder 3 Ladesäulen geeignet.

Die Bedeutung von Points of Interest (POIs)

Das Ladeverhalten im öffentlichen Raum wird sich im Gegensatz zum Ladeverhalten an Tankstellen dahingehend verändern, dass ein Fahrzeug nicht in Anwesenheit einer Person geladen wird. Vielmehr ist das Ziel, das Aufladen so in den Alltag der Menschen zu integrieren, dass zum Beispiel Besorgungen erledigt werden können, während das Fahrzeug lädt. Das bedeutet, dass Ladestandorte so platziert werden sollen, dass sie möglichst nah am Zielstandort der Besorgungen und auf den ursprünglichen Routen der Personen liegen, damit kein Umweg zum Aufladen des Elektroautos in Kauf genommen werden muss (Philipsen et al., 2016, S. 127). Viele Forschungsarbeiten berücksichtigen dies, indem sie die Zielkoordinaten von Fahrtstrecken, die Parkdauer oder sogenannte Points of Interest (POIs) wie zum Beispiel Supermärkte, Bildungseinrichtungen und Freizeiteinrichtungen etc. als mögliche Ladestandorte im öffentlichen Raum definieren (Wagner et al., 2014, S. 5).

Neben der Bedeutung der POIs als mögliche Ladestandorte dienen die POIs in Forschungsarbeiten auch häufig als Proxy für den Ladebedarf, beispielsweise unter der Annahme, dass das heutige Parkverhalten an den POIs das Ladeverhalten für elektronische Fahrzeuge imitiert (Chen et al., 2013, S. 33). Je mehr POIs in einem Gebiet vorkommen, desto höher wird der Ladebedarf in diesem Gebiet eingeschätzt. Dabei beschäftigen sich einige Studien auch damit, die Relevanz der verschiedenen POI-Kategorien zu analysieren. So finden Chen et al. beispielsweise heraus, dass es lange Parkplatznutzungen für Universitäten, religiöse Einrichtungen und Erholungseinrichtungen gibt und diese POI-Kategorien damit besonders geeignet für Ladestandorte sind (Chen et al., 2013, S. 31). Ein weiterer Vorteil der POIs ist außerdem, dass sie im Gegensatz zum kontinuierlichen Wandel der verschiedenen Elektrofahrzeuge und ihrer fortschreitenden (Lade-)Technologie relativ stabil bleiben (Sathaye and Kelley, 2013, S. 16). In dieser Arbeit werden relevante

POIs sowohl als mögliche Ladestandorte bewertet als auch deren Verteilung im Raum bei der Berechnung der optimalen Ladestandorte berücksichtigt.

Datenquellen

Forschungen zur Platzierung von Ladeinfrastruktur nutzen zahlreiche unterschiedliche Datenquellen, Algorithmen und Annahmen, wobei die Herangehensweise der jeweiligen Forschung von den verfügbaren Daten abhängt und Datenmangel ein Haupthindernis vieler Forschungsarbeiten ist (Pagany et al., 2018, S. 11). In dieser Arbeit werden zur Berechnung der optimalen Ladestandorte lediglich frei zugängliche Daten aus OpenStreetMap sowie zur Evaluation bereits veröffentlichte Ergebnisse anderer Forschungsarbeiten als Datengrundlage verwendet.

Weitere häufig verwendete Datenquellen, die eventuell eine genauere Berechnung zur Platzierung von Ladestandorten oder die Betrachtung weiterer Aspekte wie zum Beispiel eine Berücksichtigung des Ladebedarfs im Tagesablaufs ermöglichen, sind (Pagany et al., 2018, S. 6):

- sozioökonomische Daten, um Nutzergruppen von Elektromobilität und deren Verteilung im Raum, die Durchdringung, den elektrischen Bedarf und den Bedarf an Ladestationen zu bestimmen.
- Verkehrsdaten von Elektrofahrzeugen, wie z. B. GPS-Daten, um Fahrzeiten, Parkzeiten, den Batterieverbrauch und die räumliche sowie zeitliche Bewegung im Raum zu erhalten.
- Verkehrsdaten herkömmlicher Fahrzeuge als Approximation von Elektrofahrzeugdaten.
- Simulations- oder Testdaten als Approximation von Elektrofahrzeugdaten.
- Daten von Verkehrszählungen oder Verkehrstagebüchern.
- Daten und Koordinaten bereits installierter Ladesäulen, um Aussagen über deren Auslastung und Wirtschaftlichkeit zu treffen.
- Points of Interest von Google Places mit Informationen zur Frequentierung als Approximation des Ladebedarfs und Parkverhaltens im Tagesverlauf.

Typische Optimierungsziele und Optimierungsalgorithmen

Aufgrund der unterschiedlichen Datenlage und der verschiedenen Aspekte, die bei der Platzierung von Ladeinfrastruktur berücksichtigt werden können, werden unterschiedliche Optimierungsprobleme bearbeitet. Die Optimierungen haben gemeinsam, dass sie den Ladebedarf mit einer minimalen Anzahl an Ladestationen sicherstellen wollen. Zum Teil werden auch die Einhaltung von Budgetauflagen und eine zu erreichende Abdeckung mitberücksichtigt (Pagany et al., 2018, S. 8).

Die Optimierungsziele, die üblicherweise zur Standortbestimmung von Ladestationen genutzt werden, sind (Pagany et al., 2018, S. 9):

- Minimierung der Installations- und Instandhaltungskosten von Ladestationen,
- Minimierung der Distanzen der Bedarfspunkte (POIs) zu Ladestandorten,
- Minimierung der Laufdistanzen vom eigentlichen Fahrtziel zu Ladestandorten,
- Minimierung der Fahrtwegestrecken vom Fahrtursprung zu Ladestationen,
- Maximierung der räumlichen Abdeckung durch Ladestationen.

Um diese Optimierungsprobleme zu lösen, werden typischerweise Clusteralgorithmen, lineare Programme, genetische Algorithmen oder Partikelschwarmoptimierungen genutzt (Rahman et al., 2016, S. 1041).

Untersuchungsgebiete

In den Fallstudien werden meist einzelne Städte oder urbane Metropolregionen untersucht, selten ganze Länder oder ländliche Gebiete. In urbanen Regionen wird in naher Zukunft auch mehr Elektromobilität bzw. ein höherer Ladebedarf erwartet als in ländlichen Gebieten (Pagany et al., 2018, S. 10). Die Darstellung der Ergebnisse erfolgt wie in dieser Arbeit in der Regel mittels Karten. Häufig werden in Untersuchungen anders als in dieser Arbeit keine exakt georefferenzierten Ladestationen berechnet, sondern lediglich geeignete Zonen ausgewiesen (Pagany et al., 2018, S. 8).

2.2 Lösungskonzepte ausgewählter Forschungsarbeiten

In diesem Kapitel werden Forschungsarbeiten vorgestellt, aus denen Konzepte für die Implementierung, die im Rahmen dieser Bachelorarbeit entwickelt wird, übernommen

werden. Weiterhin wird dargelegt, wie die optimalen Ergebnisse berechnet wurden, die dieser Arbeit als Evaluationsgrundlage dienen.

Übernahme von Konzepten aus ausgewählten Forschungsarbeiten

Eine Möglichkeit, optimale Ladestandorte zu bestimmen, ist die Verwendung sogenannter Clusteralgorithmen wie zum Beispiel des k-Means-Algorithmus (Andrenacci et al., 2016; Shukla et al., 2016). Bei Andrenacci et al. werden hierfür GPS-Koordinaten von ca. 60.000 Fahrten in der Metropolregion Rom ausgewertet und anhand der Zielpunkte dieser Fahrten wird eine Clusteranalyse mit einem fuzzy k-Means-Algorithmus durchgeführt. Die Mittelpunkte (Zentroide) der erhaltenen Cluster werden dann als optimale Ladestandorte für Elektrofahrzeuge ausgegeben. Anschließend wird jede der Fahrten einem Cluster zugeteilt und der elektrische Energiebedarf eines Clusters berechnet, indem die benötigte Energie aller Fahrten eines Clusters summiert wird. Andrennacci et al. gehen davon aus, dass die Zielpunkte der Fahrten alle Bereiche abdecken und ihre Dichte von der Verteilung der POIs abhängt (Andrenacci et al., 2016, S. 42). Die Anzahl der Cluster muss von vornherein bestimmt werden und wird in dieser Forschungsarbeit mittels der erwarteten Anzahl an Ladevorgängen pro Ladestation hergeleitet. Da die exakten Koordinaten der Zentroide mit hoher Wahrscheinlichkeit nicht als tatsächliche Ladestandorte geeignet sind, wird empfohlen, die Platzierung der endgültigen Standorte zwar möglichst nah am berechneten Zentroid, aber unter Beachtung der lokalen Gegebenheiten vorzunehmen (Andrenacci et al., 2016, S. 43). Der Ansatz, Bedarfspunkte mit Hilfe des k-Means-Algorithmus zu clustern und die Zentroide als Standorte zu verwenden, wird übernommen und die Implementierung dieses Ansatzes in Kapitel 3.2.1 beschrieben.

Da viele der Forschungsarbeiten Verkehrsprofile mit realen Distanzen nutzen, die beim k-Means-Algorithmus nur mit euklidischen Distanzen approximiert werden, erscheint die Implementierung eines weiteren Algorithmus, der auf realen Wegdistanzen basiert, sinnvoll. Da sich der k-Means-Algorithmus mit diesem Distanzmaß nicht in einer zufriedenstellenden Laufzeit implementieren lässt, ist ein Übergang zu realen Distanzen, die mittels eines gewichteten Graphen abgebildet werden können, sinnvoll (Jain, 2009, S. 654; Storandt and Funke, 2013, S. 1). Hier können dann mit gängigen Algorithmen bedeutende Knoten und kürzeste Wege berechnet werden. Die Implementation eines solchen Ansatzes wird in Kapitel 3.2.2 beschrieben.

Ein weiterer Ansatz, der für diese Arbeit übernommen wird, ist die Implementation eines genetischen Algorithmus. Genetische Algorithmen haben zwar in der Regel eine längere Laufzeit, sind jedoch in vielen Fällen anderen Algorithmen qualitativ überlegen. Sie werden daher häufig genutzt, um optimale Standorte für Ladestationen zu berechnen (Efthymiou et al., 2017, S. 3). Ein Beispiel hierfür liefern Mehar und Senouci, die eine Methodik entwickeln, welche die Installationskosten für neue Ladestationen sowie die Transportkosten (Abstand von Bedarfspunkten zu Ladestationen) minimiert. Das erhaltene Ergebnis zeigt am Beispiel Köln, welche der möglichen Standorte für Ladestationen ausgesucht werden (Mehar and Senouci, 2013). Auch Efthymiou et al. nutzen einen genetischen Algorithmus, um optimale Standorte in Thessaloniki zu finden. Optimiert wird in dieser Studie die Abdeckung des Gesamtbedarfs durch die berechneten Ladestandorte. Hierzu werden zuerst Fahrstrecken generiert und diese anschließend in eine Distanzmatrix überführt. Als Input werden eine Liste der möglichen Ladestandorte mit dem jeweiligen erwarteten Ladebedarf sowie die Distanzen zwischen den möglichen Ladestandorten übergeben. Im Gegensatz zu Mehar und Senouci wird hier jedoch die nötige Anzahl der Ladestandorte nicht berechnet, sondern von den Nutzenden als Eingabeparameter mit übergeben. Auch die Parametereinstellungen, die es bei der Verwendung eines genetischen Algorithmus vorzunehmen gilt, werden anders als in dieser Arbeit komplett den Nutzerinnen und Nutzern überlassen (Efthymiou et al., 2017, S. 7).

Lösungen ausgewählter Forschungsarbeiten

Da die Ergebnisevaluation der Berechnungen in dieser Arbeit anhand von Ergebnissen aus der Forschungsliteratur durchgeführt wird, werden in diesem Abschnitt die drei Studien vorgestellt, deren optimale Ergebnisse verwendet werden.

In der Studie von Wagner et al. wird, wie auch in dieser Arbeit, ein POI-basierter Ansatz verfolgt. Hierzu entwickeln Wagner et al. eine Regressionsanalyse. Bei der Analyse werden durch Berücksichtigung der POI-Kategorien und der Auslastung der bereits installierten Ladesäulen in einem Gebiet, diejenigen POI-Kategorien berechnet, die besonders ausschlaggebend für die Eignung eines Gebietes als Ladestandort sind. Die berechnete Eignung der unterschiedlichen POI-Kategorien wird dann wiederum genutzt, um Aussagen darüber zu treffen, welche Standorte in anderen Gebieten für Ladestandorte besonders geeignet sind. Hierfür wird ein iterativer Optimierungsalgorithmus implementiert,

der die Abdeckung maximiert, indem er sehr nah beieinanderliegende Ladestationen bestraft, um eine Clusterbildung dieser zu vermeiden (Wagner et al., 2013). Abbildung 2.1 zeigt das Ergebnis von Wagner et al. für Nord-Amsterdam, das für diese Arbeit als optimal angenommen wird.

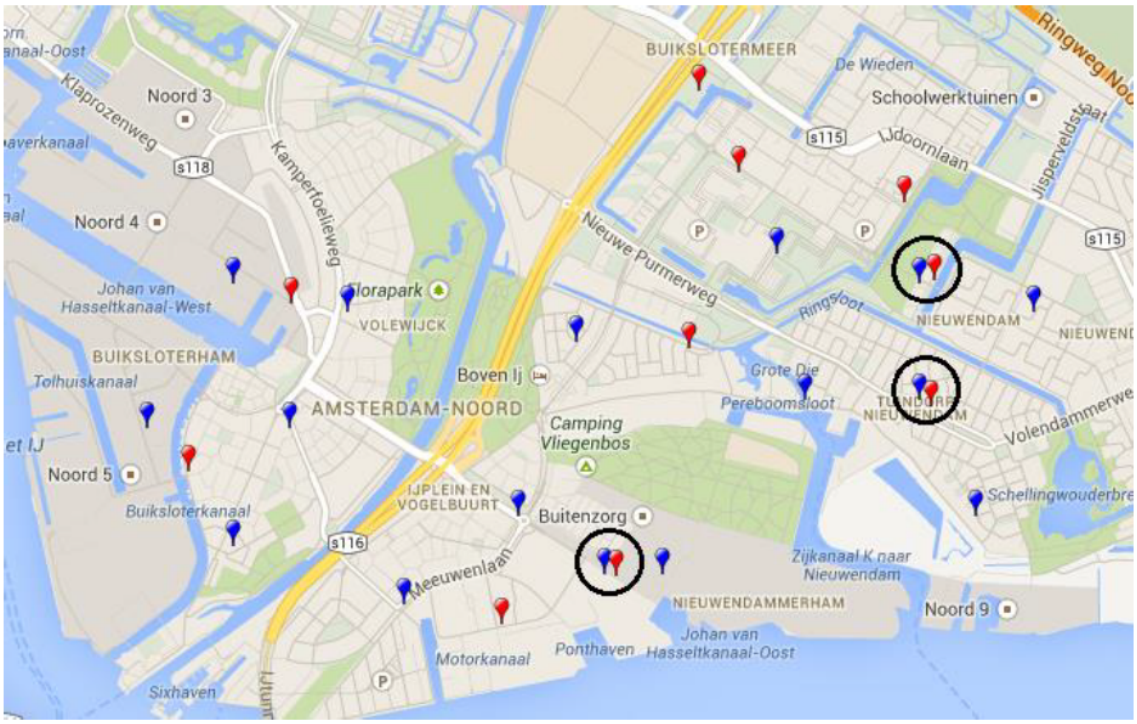


Abbildung 2.1: Optimale Ladestandorte (blau) und bereits installierte Ladesäulen (rot) für Nord-Amsterdam. Quelle: Wagner et al., 2013, S. 13.

Eine Studie von Chen et al. beschreibt eine Methodik, in der Parkinformationen im Großraum Seattle von etwa 30.000 Fahrten herkömmlicher Fahrzeuge mittels elektronischer Fahrtenbücher gesammelt und innerhalb von Verkehrsanalysezonen (sogenannter TAZ – Traffic Analysis Zones) aggregiert werden. Aus diesen Parkinformationen werden dann die Parkdauer sowie optimale Parkstandorte bestimmt. Anschließend wird eine Optimierung durchgeführt, welche die Laufdistanzen vom Parkplatz bis zur eigentlichen Zielzone (TAZ) minimiert. Auch in dieser Studie wird eine Mindestdistanz zwischen den Ladestationen festgesetzt, um eine Clusterbildung von Ladestationen zu verhindern. Eine der durch diese Methodik erhaltene Lösung platziert die Ladestationen in den Zonen mit dem höchsten Parkbedarf (Chen et al., 2013, S. 35). Diese Lösung wird in dieser Arbeit als optimal angenommen und ist in Abbildung 2.2 zu sehen. Da jedoch anders als in dieser Lösung Punktkoordinaten statt Zonen benötigt werden, wird jeweils der Mittelpunkt einer Zone als optimaler Ladestandort angenommen. Dieser Mittelpunkt wird mittels der Open

Source-GIS-Software QGIS ermittelt (vgl. Abbildung A0.1 im Anhang).

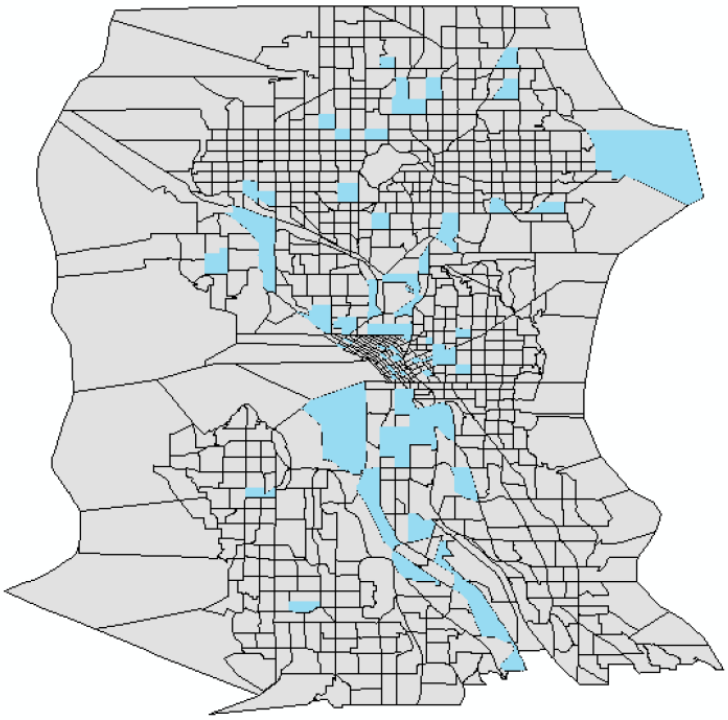


Abbildung 2.2: Optimale Ladestandorte für Seattle. Die blau gefärbten Verkehrsanalysezonen (TAZ) sind die Zonen mit dem höchsten Parkbedarf und daher optimale Standorte für Ladestationen.
(Quelle: Chen et al., 2013, S. 35).

Yagcitekin et al. berechnen in einer Studie die nötige Anzahl und Platzierung der Ladestandorte auf dem 131 Hektar großen Davutpaşa-Campus der Yildiz Technical Universität in Istanbul mittels einer hierarchischen Warteschlangentheorie. Dabei wird die Wartezeit, die in der Schlange verbracht werden muss, um ein Elektroauto zu laden, minimiert. Zuerst wird dabei die optimale Anzahl an Ladestationen in einem Gebiet berechnet, dann die optimale Platzierung dieser anhand von verschiedenen Kriterien wie zum Beispiel Nutzungsdichte und Zugänglichkeit (Yagcitekin et al., 2016, S. 1298). Abbildung 2.3 zeigt das Ergebnis dieser Studie, das in dieser Arbeit als optimal angenommen wird.

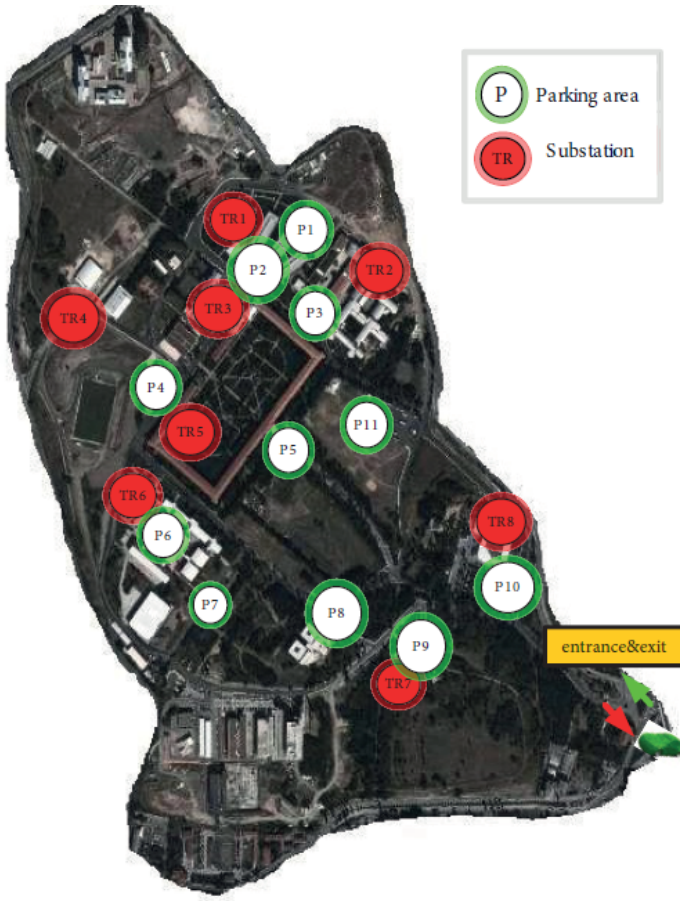


Abbildung 2.3: Die optimalen Ladestandorte (TR1 bis TR8) und Parkplätze (P1 bis P11) des Davutpaşa-Campus in Istanbul. Quelle: Yagcitekin et al., 2016, S. 1300.

3 Beschreibung und Theorie der Implementierung

In diesem Kapitel werden die Vorgehensweisen bei der Implementierung und die dahinterliegenden Theorien dargelegt. Hierzu werden die Daten und Datenstrukturen, die zur Berechnung der optimalen Ladestandorte verwendet werden, sowie die drei hierfür implementierten Algorithmen beschrieben. Weiterhin wird die Auswahl der Untersuchungsgebiete begründet und die Methodik der Ergebnisevaluation erläutert. Zusätzlich werden die Funktionalitäten der beiden Webseiten, welche die Ergebnisse kartografisch darstellen, vorgestellt. Abbildung 3.1 gibt eine Übersicht über den Aufbau und die Vorgehensweise der Implementierung. Sofern nicht anders erwähnt, erfolgt die Implementierung aller nachfolgenden Algorithmen mit Python 3.6.

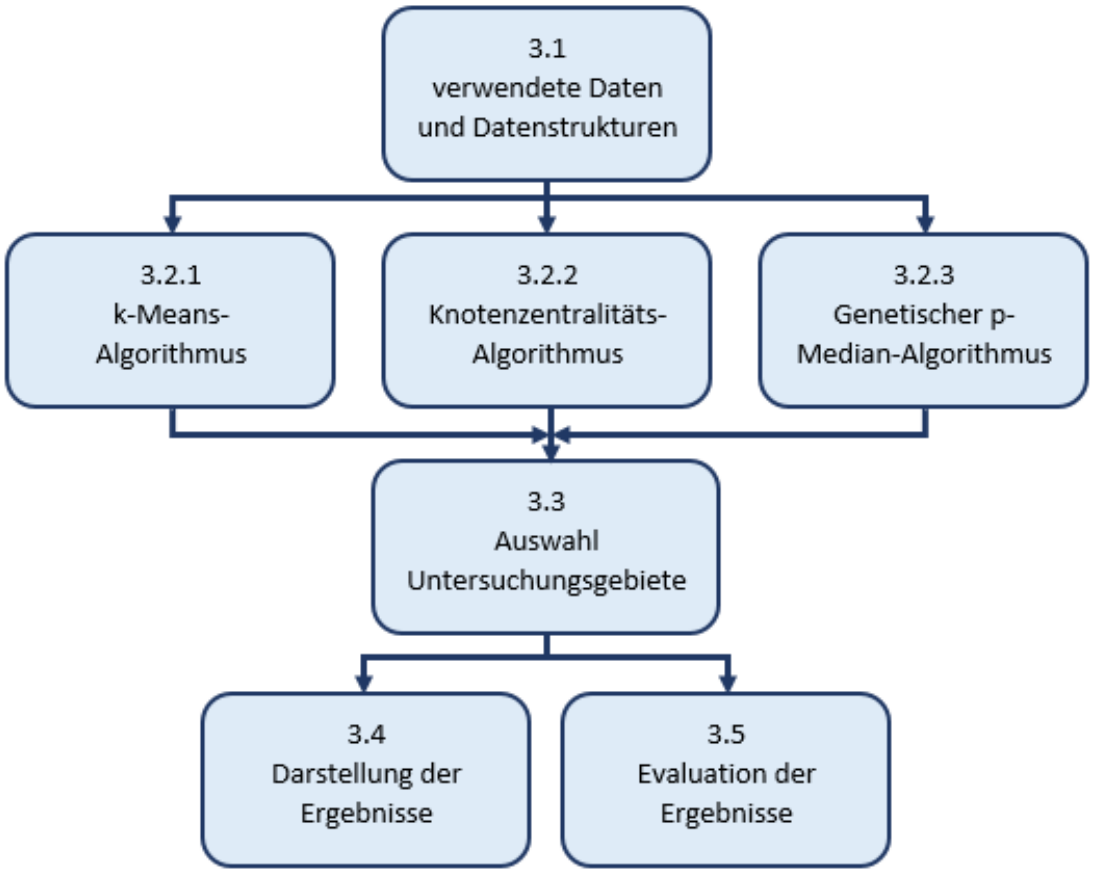


Abbildung 3.1: Überblick über die Implementierung zur Berechnung von optimalen Ladestandorten. Die Nummern verweisen auf die entsprechenden Kapitel.
Quelle: Eigene Darstellung.

3.1 Verwendete Daten und Datenstrukturen

Die in dieser Arbeit verwendete Datengrundlage zur Berechnung der optimalen Ladestandorte in einem Gebiet, bildet für alle Algorithmen die jeweils immer gleiche Menge von Points of Interest (POIs) aus OpenStreetMap (OSM). Diese POIs dienen wie bereits erwähnt einerseits als Proxy für den lokalen Bedarf von Ladestandorten und andererseits als Platzierungsstandorte für Ladesäulen (vgl. Kap. 2.1). Zur Analyse der Ergebnisse werden außerdem manuell digitalisierte Ladestandorte anderer Publikationen verwendet (vgl. Kapitel 2.2 und Kapitel 3.5.1). Damit sind alle in dieser Arbeit verwendete Daten frei verfügbar und eine Ergebnistransparenz ist ermöglicht.

Die Datenqualität der von über einer Million Nutzerinnen und Nutzer zusammengetragenen OSM-Daten wird vor allem in Industrieländern der Regel als hoch eingestuft und ist damit in den ausgewählten Untersuchungsgebieten ausreichend (Boeing, 2017, S. 128). Die Datenqualität der POIs, die bei Google hinterlegt sind, ist zwar höher, die Daten sind jedoch nicht in beliebiger Menge frei zugänglich (von Rueden, 2017, S. 5).

Die bei OSM hinterlegten Daten zu einer Region bzw. einer Bounding Box werden über die Exportfunktion von OSM mittels der Overpass API heruntergeladen und als XML-Datei abgespeichert. Abbildung 3.2 zeigt ein Beispiel einer hierarchischen XML- bzw. OSM-Datenstruktur. Diese kann als Baumstruktur mit dem Wurzelelement <osm> verstanden werden und wird beim Parsen durch die Datei in der Regel auch als solche gebildet.

```
<osm>
  <bounds minlat="52.000" minlon="4.000" maxlat="52.2000" maxlon="4.7000"/>
  <node id="1" lat="52.0258" lon="4.5076">
    <tag k="amenity" v="parking"/>
  </node>
  <node id="2" lat="52.0826" lon="4.6858">
    <tag k="leisure" v="fitness_centre"/>
  </node>
  <way id="3">
    <nd ref="1"/>
    <nd ref="2"/>
    <tag k="amenity" v="parking"/>
  </way>
  <relation id="4">
  </relation>
</osm>
```

Abbildung 3.2: Beispiel einer vereinfachten XML-Datenstruktur aus OpenStreetMap. Tags sind blau, Attribute sind rot und Werte sind grün eingefärbt. Das Beispiel besteht aus einer Bounding Box, zwei Knoten, einem Weg und einer Relation.
Quelle: Eigene Darstellung.

Die von OpenStreetMap heruntergeladene XML-Datei enthält zahlreiche Informationen, von denen hier nur die für diese Arbeit relevanten Details näher erläutert werden. So enthält sie beispielsweise neben dem Tag <bounds>, in dem die Bounding Box angegeben wird, Tags für Knoten, Wege und Relationen. Für diese Arbeit sind lediglich die Knoten- und Wege-Tags interessant, da in diesen gegebenenfalls die relevanten Informationen zu den POIs gespeichert sind. Knoten können neben einer eindeutigen ID und den Koordinaten, die in Längen- und Breitengrad angegeben werden, weitere Tags enthalten, in denen Attribute und die dazugehörigen Werte abgespeichert werden (Schild and Mochol, 2007). Die k-Attribute „amenities“, „leisure“ und „building“ bestimmen hierbei, ob es sich ggf. um einen für Ladestandorte relevanten POI handelt. Wenn das v-Attribut zu einem der genannten k-Attribute ebenfalls relevant ist, also z. B. den Wert „parking“, „doctors“ und „restaurant“ hat, wird der Knoten als relevanter POI abgespeichert. Die Erstellung und Auswahl der relevanten Attribut-Wert-Kombinationen erfolgt manuell mithilfe der OSM-Wikipedia, welche die wichtigsten und häufigsten Attribut-Wert-Kombinationen listet (OpenStreetMap-Wikipedia, 2019). Werte, die keine Bedeutung für die Ladestandorte haben, wie beispielsweise „baby_hatch“, „fountain“ und „post_box“, werden nicht von der Wikipediaseite übernommen. In Tabelle A0.1 im Anhang dieser Arbeit sind alle als relevant eingestuften Werte der Wikipediaseite zu finden. Weiterhin werden auch bereits installierte Ladesäulen bzw. der Wert „charging_station“ nicht berücksichtigt, um die Ergebnisse nicht zu verfälschen. Des Weiteren wird auch bei Wegen, die aus mehreren Knoten bestehen, jeweils nur der erste relevante Knoten als POI berücksichtigt. Dies hat den Vorteil, dass die Menge an POIs, die eine große Auswirkung auf die reale Laufzeit hat (siehe Kapitel 3.2.2), geringer gehalten wird und dass die Bedeutung von Wegen aufgrund zu vieler ausgewählter POIs nicht übermäßig überhöht wird. Die relevanten POIs eines Untersuchungsgebietes werden in einem Vorbereitungsschritt mithilfe der Python-Bibliothek lxml aus der XML-Datei gefiltert und in einer CSV-Datei zwischengespeichert, die von allen Algorithmen zur Berechnung der Ladestandorte eingelesen wird. Die Nutzung der lxml-Bibliothek hat sich bewährt, da im Gegensatz zu anderen XML-Bibliotheken in Python nicht der komplette Baum aller bisher eingelesenen Attribute im Arbeitsspeicher beibehalten werden muss, sondern zwischenzeitlich gelöscht werden kann. Hierdurch wird eine Überlastung des Arbeitsspeichers vermieden und somit auch das Parsen der zum Teil sehr großen XML-Dateien von OSM auf gewöhnlichen Computern ermöglicht (Behnel,

2019). Das Speichern der Zwischenlösung in einer CSV-Datei ist nicht zwingend notwendig, verringert jedoch den Zeitbedarf des Einlesens der relevanten POIs eines Gebietes bei mehrfacher Ausführung der nachfolgenden Algorithmen.

Neben den POIs aus OpenStreetMap benötigen alle drei Algorithmen auch die Anzahl der benötigten Ladestandorte in einem Gebiet als Eingabe. Da die Bestimmung des benötigten energetischen Ladebedarfs, aus dem die benötigte Anzahl an Ladestandorten unter anderem resultiert, nicht Teil dieser Arbeit ist, wird die benötigte Anzahl aus wissenschaftlichen Publikationen entnommen (vgl. Kapitel 2.2).

Zur qualitativen Evaluation der berechneten Ergebnisse wird außerdem eine optimale Lösung pro Untersuchungsgebiet benötigt. Hier werden die Ergebnisse anderer Forschungsarbeiten als optimal angenommen und deren Lösungen manuell digitalisiert (vgl. Kapitel 2.2).

Die Speicherung dieser digitalisierten sowie der in dieser Arbeit berechneten Lösungen erfolgt in einer GeoJSON-Datenstruktur, die in einer .geojson-Datei abgelegt wird. GeoJSON ist ein Dateiformat, das sich seit der ersten Spezifikation im Jahr 2008 aufgrund seiner Schlankheit an zunehmender Beliebtheit erfreut und strukturell auf der JavaScript Object Notation (JSON) basiert. Jedes GeoJSON-Objekt hat ein Elternobjekt üblicherweise eine Sammelobjekt wie eine FeatureCollection in dem die Features Punkte, Linien und Polygone definiert werden. Da die Ladestandorte und POIs alle als Punkte dargestellt werden können, werden in dieser Arbeit nur Punkt-Features definiert. Zu jedem Feature bzw. jedem Punkt können weitere nichträumliche Attribut-Wert-Kombinationen abgespeichert werden, wobei die Spezifikation keine Attributnamen vorgibt (Butler et al., 2016). In dieser Arbeit werden folgende Attribute und Werte zu jedem POI abgespeichert:

- osm_keyword: zugehöriger Wert ist der Attributwert, den OSM zu diesem POI gespeichert hat, wie z. B. parking, doctors und restaurant.
- osm_id: zugehöriger Wert ist die eindeutige ID, unter welcher der POI in OSM hinterlegt ist.
- charging_station: zugehöriger Wert ist eine boolesche Variable, die den Wert 1 hat, wenn es sich um einen Ladestandort handelt und sonst den Wert 0.
- clusternumber: zugehöriger Wert ist die berechnete Clusternummer des POIs.

Die optimalen Ladestandorte, die nicht in den OSM-Daten enthalten sind, erhalten jeweils nur die Information „clusternumber: 0“ und „charging_station: 1“. Abbildung 3.3 zeigt ein Beispiel einer GeoJSON-Datenstruktur, wie sie in dieser Arbeit verwendet wird. Die Koordinaten der GeoJSON-Datenstruktur sind als Breiten- und Längengrad hinterlegt, wobei anders als sonst üblich zuerst die Länge und dann die Breite angegeben werden muss. Das Standardkoordinatensystem ist wie bei OSM der WGS84 Geoid, was dazu führt, dass die GeoJSON-Dateien ohne Koordinatentransformation aus den OSM-Daten erstellt werden und auf OSM-Karten angezeigt werden können (Marsden, 2009). In Python steht für die Verwaltung von GeoJSON-Datenstrukturen das Modul geojson zur Verfügung.


```
{
    "features": [
        {
            "geometry": {
                "coordinates": [
                    4.9083182,
                    52.3817779
                ],
                "type": "Point"
            },
            "properties": {
                "amenity": "ferry_terminal",
                "centroid": 0,
                "id": "46434756",
                "label": 0
            },
            "type": "Feature"
        },
    ],
    "type": "FeatureCollection"
}
```


Abbildung 3.3: Beispiel einer GeoJSON-Datenstruktur, die einen POI als Punkt enthält. In den „properties“ sind die jeweiligen Attribut-Wert-Kombinationen gespeichert. Quelle: Eigene Darstellung.

3.2 Algorithmen zur optimierten Platzierung von Ladestandorten

In diesem Kapitel werden die drei implementierten Algorithmen zur Berechnung der Ladestandorte beschrieben. Hierbei wird die grundlegende Funktionsweise erklärt, auf die spezifischen Hintergründe zur Berechnung der Ladestandorte in dieser Arbeit eingegangen und Informationen zu den theoretischen Laufzeiten gegeben.

Alle drei Algorithmen haben gemeinsam, dass sie den gleichen Input (die CSV-Datei mit den relevanten POIs eines Gebietes) bekommen und einen gleichartigen Output liefern. Der Output ist eine GeoJSON-Datei mit geclusterten POIs, wobei für jedes Cluster ein POI als Ladestandort ausgewählt ist.

3.2.1 K-Means-Algorithmus

Der k-Means-Algorithmus wird seit den 50er Jahren verwendet und zählt bis heute aufgrund seiner Einfachheit und Schnelligkeit zu den beliebtesten Clusteralgorithmen. In dieser Arbeit wird er daher als Baseline-Algorithmus ausgewählt (Jain, 2009, S. 651). Er zählt zu den unüberwachten maschinellen Lernmethoden, weil im Gegensatz zu den überwachten Lernmethoden an keinem optimalen Ergebnis trainiert wird (Swamynathan, 2017, S. 195).

Der k-Means Algorithmus partitioniert eine Eingabemenge bestehend aus n Objekten in k disjunkte Teilmengen C, sogenannte Cluster. Dabei wird jedes Cluster von einem Element μ, im Nachfolgenden Zentroid genannt, repräsentiert. Die Clustereinteilung der Eingabemenge erfolgt basierend auf einer Ähnlichkeits- bzw. Distanzmessung dahingehend, dass die Distanzen zwischen den Objekten innerhalb eines Clusters minimiert und die Distanzen zwischen den Objekten in unterschiedlichen Clustern maximiert werden (Jain, 2009, S. 652).

Als Distanzmaß im zweidimensionalen Raum wird hierzu üblicherweise die euklidische Distanz genommen. Gleichung 3.1 zeigt, wie der euklidische Abstand zwischen zwei Punkten bzw. einem Punkt x und einem Zentroid μ berechnet wird (Swamynathan, 2017, S. 196).

$$d(x, \mu) = \sum_{i=1}^{2} (x_i - \mu_i)^2 = ||x - \mu||^2 \tag{3.1}$$

Das Ziel des k-Means-Algorithmus ist es, die Bildung der Cluster so vorzunehmen, dass die Summe der quadrierten Abweichungen von den Cluster-Zentroiden (Residuenquadratsumme) minimal ist. Mathematisch entspricht dies der Optimierung der Zielfunktion von Gleichung 3.2 (Pedregosa et al., 2011):

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2) \tag{3.2}$$

Die Minimierung dieser Zielfunktion mittels des k-Means-Algorithmus für eine Menge an POIs wird durch den Pseudocode (siehe Algorithmus 1) skizziert.

Data: k, Liste an POI-Koordinaten
Result: Clustereinteilung, Zentroide
Initialisierung: Wähle zufällig k Zentroide aus
repeat
  for *jeden POI* **do**
    finde den nächstgelegenen Zentroid und weise den POI dem Cluster dieses Zentroids zu.
  end
  Berechne für die k-Cluster neue Zentroide, indem jeweils die Durchschnittskoordinaten der clusterzugehörigen Elemente berechnet werden.
until *Terminierungskriterium ist erfüllt*;

Algorithmus 1: Pseudocode des k-Means-Algorithmus.
Quelle: Eigene Darstellung nach: Jin and Han, 2010, S. 695.

Die zwei sich wiederholenden Iterationsschritte des k-Means-Algorithmus führen dazu, dass sich die Residuenquadratsumme in jedem Schritt verringern, jedoch niemals erhöhen kann (Manning et al., 2009, S. 363).

Die theoretische Analyse des k-Means-Algorithmus zeigt, dass es insgesamt k^n Möglichkeiten gibt, n Punkte in k Cluster einzuteilen. Die optimale Lösung des Problems ist somit NP-schwer. Die Laufzeit des k-Means-Algorithmus liegt in $O(nki)$, wobei i die Anzahl der Iterationen ist (Jin and Han, 2010, S. 696).

Der im Rahmen dieser Arbeit implementierte Algorithmus ist so konfiguriert, dass er entweder nach 400 Iterationsschritten oder wenn sich die Residuenquadratsumme um weniger als 0,0001 zum vorherigen Iterationsschritt unterscheidet, terminiert. Durch die zufällige Auswahl der Zentroide im Initialisierungschritt können sich die Ergebnisse zudem unterscheiden. Aus diesem Grund ist es empfehlenswert, den Algorithmus mehrmals hintereinander auszuführen und das beste Ergebnis beizubehalten. In dieser Implementierung ist eine Konfiguration von maximal 50 Wiederholungen eingestellt. Die Berechnung wird durchgeführt mit Hilfe des KMeans-Moduls der Python-Bibliothek scikit-learn (Pedregosa et al., 2011).

Ein Nachteil des k-Means-Algorithmus ist, dass er in der Regel kein globales Optimum liefert. Nur wenn die zugrundeliegenden Daten bereits eine ball- bzw. kugelförmige Struktur aufweisen, findet der k-Means-Algorithmus mit einer hohen Wahrscheinlichkeit auch das globale Optimum (Jain, 2009, S. 654). Liegt diese Datenstruktur nicht natürlich vor, findet der k-Means-Algorithmus keine guten Cluster. Die gegebene Punktemenge wird vom Algorithmus aber in jedem Fall in Cluster eingeteilt, unabhängig davon, ob natürliche Cluster vorliegen oder nicht. Das kann unter anderem dazu führen, dass das Vorhandensein von Ausreißern einen starken Einfluss auf das Ergebnis hat. Bevor der k-Means-Algorithmus angewendet wird, sollte folglich eine grundlegende Clustertendenz der Daten sichergestellt werden (Jain, 2009, S. 656f.). Bei den POIs der ausgewählten Untersuchungsgebiete ist diese Eigenschaft jeweils gegeben (vgl. Kapitel 3.3).

Das Ergebnis des k-Means-Algorithmus in dieser Arbeit ist ähnlich wie bei Andrenacci et al. so zu interpretieren, dass die Koordinaten der berechneten Zentroide die optimalen Ladestandorte darstellen (Andrenacci et al., 2016, S. 43). Der Eingabeparameter k entspricht also der Menge an Ladestandorten, die in einem Gebiet benötigt werden. Da diese Standorte jedoch unter Umständen nicht geeignet sind (beispielsweise könnte ein Zentroid mitten auf einer Wasserfläche liegen), wird stattdessen jeweils der zum Zentroid nächstgelegene POI innerhalb des Clusters als Ladestandort ausgewählt (Andrenacci et al., 2016, S. 43). Für diesen Nachbereitungsschritt wird jedoch nicht die euklidische Distanz, sondern die Länge der kürzesten Wege zwischen den Zentroiden und den POIs als Distanzmaß verwendet. In Kapitel 3.2.2 wird gezeigt, wie diese Distanz ermittelt werden kann. Die Laufzeit für diesen Nachbereitungsschritt beträgt $O(n + k)$, da jeder Zentroid und jeder POI genau einmal angeschaut wird.

3.2.2 Clustering basierend auf Knotenzentralitätswerten

Einer der großen Nachteile des k-Means-Algorithmus ist, dass standardmäßig nur das euklidische Distanzmaß, nicht jedoch reale Distanzen verwendet werden können (Swamynathan, 2017, S. 196). Die realen Distanzen zwischen zwei Punkten in den Untersuchungsgebieten weichen jedoch in der Regel stark von den euklidischen Luftliniendistanzen ab. Aus diesem Grund wird in dieser Arbeit ein weiterer Baseline-Algorithmus getestet, der auf realen Distanzen beruht.

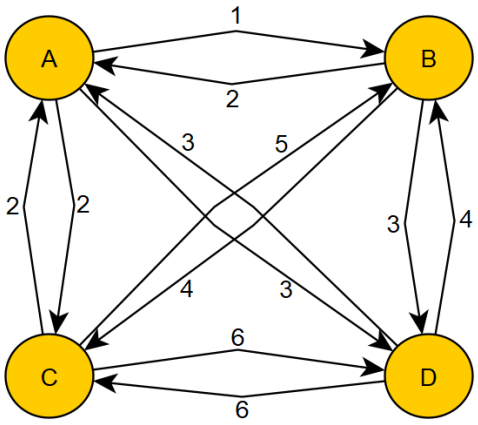
Um eine Punktemenge unter Berücksichtigung der realen Distanzen zwischen den Punkten zu clustern, bietet sich die Nutzung einer Graphenstruktur und eines Graphen-Clusteralgorithmus an. Dabei werden die POIs eines Gebietes so in eine Graphenstruktur überführt, dass die POIs die Knoten und die Distanzen zwischen den POIs die Kanten des Graphen bilden. Dabei können sich die Distanzen von einem Punkt A zu einem Punkt B beispielsweise aufgrund von Einbahnstraßen und anderen unregelmäßigen Straßenführungen unterscheiden. Die Kanten sind folglich gerichtet und verlaufen in beide Richtungen zwischen den POIs. Da natürlicherweise jeder POI von jedem POI erreicht werden kann, ist der Graph außerdem vollständig. Während ein ungerichteter vollständiger Graph aus $\binom{n}{2} = \frac{n \cdot (n-1)}{2}$ Kanten besteht, besteht der so erzeugte gerichtete vollständige Graph folglich aus $\frac{n \cdot (n-1)}{2} \cdot 2 = n \cdot (n-1)$ Kanten (Wikimedia Foundation Inc. (Hrsg.), 2019). Um den Graphen zu bilden, wird die Python-Bibliothek NetworkX genutzt. Um die benötigten Distanzen zwischen den POIs für die Kantengewichte zu erhalten, wird die Python-Bibliothek OSMnx genutzt, die auf das Straßennetzwerk von OpenStreetMap zurückgreift und dieses als gewichteten gerichteten Graphen runterlädt, bei dem Einbahnstraßen abgebildet werden können (Boeing, 2017, S. 131). Im Gegensatz zu anderen Routingmaschinen ist der Vorteil des Routings mittels der OSMnx-Bibliothek, dass diese zum einen kostenlos ist und zum anderen das Herunterladen des Straßennetzwerks in der angegebenen Bounding Box in den Arbeitsspeicher ermöglicht. Zugriffe wie eine Distanzabfrage auf das Straßennetz liefern dann nicht nur schneller Ergebnisse, sondern insbesondere werden auch HTTP-Fehler wie „429: Too Many Requests“ vermieden, die bei der Nutzung anderer Routingmaschinen aufgetreten sind (OpenStreetMap Foundation (Hrsg.), 2019). Nachteil dieser Implementierung ist, dass die Größe der Bounding Box, innerhalb derer das Straßennetzwerk heruntergeladen werden kann, beschränkt ist. Dieses Problem tritt beispielsweise auf, wenn das Straßennetz für die komplette Metropolregion Seattle angefragt wird. Aus diesem Grund wird in dieser Arbeit das Untersuchungsgebiet für Seattle auf einen interessant scheinenden nördlichen Teilbereich der Metropolregion beschränkt.

Die theoretische Laufzeit zur Erstellung der Distanzmatrix mit deren Hilfe der Graph gebildet wird, ist jedoch in jedem Fall quadratisch in Abhängigkeit der Anzahl der POIs, da von jedem POI zu jedem anderen POI die Distanz abgefragt werden muss. Da sich die Distanzen von POI A zu B und B zu A unterscheiden können, ist die resultierende

NxN-Matrix auch nicht symmetrisch. Abbildung 3.4 zeigt beispielhaft eine Distanzmatrix und die Überführung dieser in einen gerichteten vollständigen Graphen.

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & \ddots & 4 & 3 \\ 2 & 5 & \ddots & 6 \\ 3 & 4 & 6 & 0 \end{pmatrix} \end{array}$$

(a) 4x4-Distanzmatrix der Punkte A, B, C und D.



(b) Vollständiger, gewichteter, gerichteter Graph.

Abbildung 3.4: Beispiel einer Distanzmatrix mit dazugehörigem Graphen. Die POIs A, B, C und D bilden die Punkte und die Distanzen die gewichteten Kanten des Graphen. Quelle: Eigene Darstellung.

Weiterhin sei noch erwähnt, dass es unter Umständen vorkommen kann, dass die OSMnx Bibliothek nicht in der Lage ist, Wege zu bestimmten POIs zu finden. Dies ist jedoch in der Realität gar nicht möglich, da es keine nicht erreichbaren POIs geben kann. Ein Grund für diese Problematik könnte die Begrenzung des Straßennetz auf die Bounding Box sein. Liegt ein POI beispielsweise nah an der Grenze der Bounding Box und innerhalb einer Einbahnstraße, ist es in diesem Straßennetz tatsächlich nicht möglich den POI zu erreichen. In dieser Arbeit werden unerreichbare POIs manuell über ihre ID ausgeschlossen, um das Ergebnis nicht zu verfälschen. Eine Liste der ausgeschlossenen POI-IDs ist im Anhang in Tabelle A0.2 zu finden. Für fremde Gebiete wird bei einem solchen Fehler standardmäßig die Haversine-Distanz genutzt, die im Gegensatz zur euklidischen Distanz die Erdkrümmung mitberücksichtigt und als Distanzmaß auch in kleinen Gebieten empfohlen wird (Wagner et al., 2013, S. 6). Die Haversine-Distanz ist dabei eine untere Annäherung an die kürzesten Wege. Würde stattdessen eine Distanz von 0, eine negative Distanz oder eine sehr hohe Distanz genommen, würde dies die Ergebnisse der nachfolgenden Clusterbildung

verfälschen.

Nachdem die POIs in eine Graphenstruktur überführt sind, gilt es nun, diesen Graphen zu clustern und k POIs des Graphen als Ladestandorte auszuwählen. Eine Möglichkeit, einen Graphen in k Cluster aufzuteilen, ist die Bildung eines k-Spanning Trees. Dabei wird zuerst ein Minimaler Spannbaum gebildet und aus diesem dann die $k-1$ Kanten mit den größten Kantengewichten (also den größten Distanzen) entfernt (Samatova, 2013). Eine weitere Möglichkeit, einen Graphen in k Cluster einzuteilen, ist die Verwendung des Girvan-Neuman-Algorithmus, der iterativ genau die Kanten, auf denen die meisten kürzesten Wege verlaufen, aus dem Graphen entfernt, bis k Cluster übrig sind (Despalatovic et al., 2014, S. 998). Beide Clustermethoden führen zu Ergebnissen, bei denen sehr viele Knoten in einem Cluster sind und einige Cluster nur aus einem einzigen Knoten bestehen. Da diese Ergebnisse unerwünscht sind und der Girvan-Neuman-Algorithmus zudem noch eine sehr lange Laufzeit hat, werden die beiden Clusteransätze nicht weiter verfolgt und ein anderer Algorithmus implementiert.

Der letztendlich in dieser Arbeit implementierte Algorithmus wird in der Regel nicht verwendet, um Graphen zu clustern, sondern um bedeutende Knoten in Netzwerken zu ermitteln. Er beruht auf dem Prinzip der Intermediationszentralität (betweenness centrality) von Knoten, die ein Maß für die Bedeutung eines Knotens in einem Graphen ist. Je höher die Intermediationszentralität bzw. der Zentralitätswert eines Knotens ist, desto bedeutender ist der Knoten. Hierbei liegt die Annahme zugrunde, dass ein Knoten umso wichtiger ist, je mehr kürzeste Wege durch ihn verlaufen. Die Intermediationszentralität $c_B(v)$ eines Knotens $v \in V$ ist der Anteil aller kürzesten Wege, die durch ihn verlaufen und wird in Gleichung 3.3 beschrieben, wobei V die Anzahl der Knoten, $\sigma(s,t)$ die Anzahl der kürzesten (s,t)-Wege und $\sigma(s,t|v)$ die Anzahl derjenigen Wege ist, die durch v verlaufen aber nicht durch s und t (NetworkX Developers (Hrsg.), 2018).

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)} \tag{3.3}$$

Die Berechnung der Intermediationszentralität erfolgt mit der Python-Bibliothek Networkx, die eine Liste der Zentralitätswerte der einzelnen Knoten zurückgibt. Networkx nutzt zur Berechnung der Knotenzentralität den Algorithmus von Ulrik Brandes und gibt eine

Laufzeit von $O(nm+n^2logn)$ bei einem Platzverbrauch von $O(n+m)$ an (Brandes, 2001, S. 9). Algorithmus 2 zeigt den Pseudocode zur Berechnung der Zentralitätswerte aller Knoten. Dabei wird auch Dijkstras kürzester Wege-Algorithmus verwendet, dessen Pseudocode in Algorithmus 3 dargestellt ist.

```
Data: gewichteter gerichteter Graph G
Result: Liste der Zentralitätswerte aller Knoten v von G
Initialisiere: Für alle Knoten v ∈ V setze Zentralität(v) := 0
for jeden Knoten v ∈ V do
    Wege, Distanzen = Dijkstra_Kürzeste_Wege(G, v, Gewicht)
    Zentralität = Akkumuliere(Zentralität, Wege, Distanzen)
end
```

Algorithmus 2: Pseudocode des Knotenzentralitäts-Algorithmus (Betweenness Centrality). Quelle: Eigene Darstellung nach: (NetworkX Developers (Hrsg.), 2018)

Die so erhaltenen Zentralitätswerte können nun genutzt werden, um den Graphen in k Cluster einzuteilen. Dazu werden die Knoten bzw. POIs absteigend nach ihren Zentralitätswerten sortiert und die k Knoten mit dem höchsten Wert als Ladestandorte ausgewählt. Anschließend werden die übrigen POIs dem Ladestandort zugeteilt, dem sie am nächsten liegen, wobei wieder die kürzesten Wege als Distanzmaß verwendet werden. Da die Knoten mit den höchsten Zentralitätswerten sehr häufig nahe beieinander liegen, kann außerdem ein Mindestabstand zwischen den Knoten bzw. Ladestandorten angegeben werden (vgl. Kapitel 4). Knoten, die diesen Mindestabstand zu bereits ausgewählten Ladestandorten nicht erfüllen, werden dann in der sortierten Liste übersprungen und der erste nachfolgende Knoten, der den Mindestabstand erfüllt, als Ladestandort gewählt. In dieser Arbeit wird für die berechneten Ergebnisse standardmäßig jeweils ein Mindestabstand von 100 Metern zwischen den Ladestandorten eingefordert.

Die Berechnung der kürzesten Wegedistanz der übrigen Knoten zu den mittels der Zentralitätswerte ausgewählten Ladestandorten erfolgt ebenfalls mit der Python-Bibliothek Networkx. Um alle kürzesten Wege von allen Knoten zu allen anderen Knoten eines Graphens zu erhalten, wird der auf Dijkstra basierende „all-pair-shortest-paths-and-length-Algorithmus“ auf den gewichteten Graphen angewandt. Algorithmus 3 zeigt den auf Dijktra basierenden Pseudocode, der alle kürzesten Wege von einem Knoten zu einem anderen Knoten zurückgibt. Durch n Aufrufe dieses Algorithmus, wobei n der Anzahl der Knoten im Graphen entspricht, können alle kürzesten Wege von allen Knoten zu allen

anderen Knoten gefunden werden.

```
Data: gewichteter gerichteter Graph, s
Result: kürzeste Wege und Wegelängen von s zu allen anderen Knoten
Initialisiere: Für alle Knoten u ∈ V setze dist(u) := ∞
Initialisiere: Warteschlange Q mit Startknoten s und setzte dist(s) := 0.
repeat
    nehme Knoten u mit der vorläufig kleinsten Distanz dist(u) in Q
    for alle Nachbarknoten v von u do
        setze neue-dist := dist(u) + w(u, v)
        if new-dist < dist(v) then
            if dist(v) = ∞ then
                füge Nachbarknoten v in Q ein mit Priorität neue-dist
            end
            else
                setze Priorität des Nachbarknoten v in Q auf neue-dist
            end
        end
        setzte dist(v) := new − dist
    end
until Warteschlange Q ist leer;
```

Algorithmus 3: Pseudocode von Dijkstra's Algorithmus der kürzesten Wege. Quelle: Wagner and Willhalm, 2007, S. 25.

Networkx gibt keine näheren Informationen zu Laufzeit- oder Implementierungsdetails des „all-pair-shortest-paths-and-length-Algorithmus“ an. Eine naive Implementierung dieses auf Dijkstra basierenden Algorithmus benötigt kubische Laufzeit. Es sind jedoch verschiedene Verbesserungsmöglichkeiten, wie beispielsweise eine Parallelisierung, das Ausnutzen von Highway-Hierarchien, das Labeln von Knoten und Kanten oder verschiedene Kombinationen hiervon, bekannt, welche die Laufzeit verringern (Wagner and Willhalm, 2007).

3.2.3 Genetischer p-Median-Algorithmus

Da die Nutzung genetischer Algorithmen bei der Platzierung von Ladestandorten häufig zu besseren Ergebnissen als die Nutzung anderer Algorithmen führt, wird in dieser Arbeit neben den beiden Baseline Algorithmen auch ein genetischer Algorithmus implementiert, der das p-Median-Problem löst (Efthymiou et al., 2017, S. 27).

Das p-Median-Problem besteht darin, dass p Standorte aus einer Menge von n möglichen Bedarfspunkten so gewählt werden sollen, dass die Gesamtdistanz von allen

n Bedarfspunkten zum nächstgelegenen ausgewählten Standort minimal ist (Chaudhry et al., 2003, S. 87). In dieser Arbeit entsprechen sowohl die möglichen Standorte als auch die Bedarfspunkte den POIs eines Untersuchungsgebietes. Es gilt also, p POIs als Ladestandorte in einem Gebiet auszuwählen, so dass die Gesamtdistanz von allen anderen $n - p$ POIs zu diesen Ladestandorten minimiert wird. Als Distanzmaß werden bei dieser Implementierung wieder die realen kürzesten Wege zwischen den POIs verwendet, die, wie in Kapitel 3.2.2 beschrieben, mit der Python-Bibliothek OSMnx ermittelt und in einer Distanzmatrix abgespeichert werden. Das p-Median-Problem ist durch die Zielfunktion in Gleichung 3.4 als lineares Optimierungsproblem formuliert, wobei p die Anzahl der benötigten Ladestandorte, n die Anzahl der Bedarfspunkte und d_{ij} die Distanz zwischen Bedarfspunkt i und Ladestandort j angibt. Falls ein Bedarfspunkt i einem Ladestandort j zugewiesen wird, beträgt $x_{ij} = 1$, sonst 0.

$$min \sum_{j=1}^{p} \sum_{i=1}^{n} d_{ij} x_{ij} \tag{3.4}$$

Folgende Nebenbedingungen müssen ebenfalls eingehalten werden:

$$X_{ij} \leq X_{ij} \qquad \forall i, j \tag{3.5}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad \forall i \tag{3.6}$$

$$\sum_{j=1}^{n} X_{ij} = p \qquad \forall i \tag{3.7}$$

Dabei bedeutet Nebenbedingung 3.5, dass ein Bedarfspunkt nur einem Ladestandort, nicht jedoch einem anderen Bedarfspunkt zugeteilt werden kann. Nebenbedingung 3.6 bedeutet, dass jeder Bedarfspunkt nur genau einem Ladestandort zugeteilt werden kann und Nebenbedingung 3.7 bedeutet, dass es insgesamt genau p Ladestandorte gibt (Chaudhry et al., 2003, S. 87).

Insgesamt ergeben sich hiermit $n^2 + n + 1$ Nebenbedingungen und $\binom{n}{p} = \frac{n!}{(n-p)! \cdot p!}$ Möglichkeiten, p Punkte aus n möglichen Punkten auszuwählen. Grundsätzlich ist die Lösung des Optimierungsproblems damit NP-schwer. Aus diesem Grund wird die

Berechnung mit Hilfe eines genetischen Algorithmus implementiert, der die optimale Lösung approximiert (Mangla et al., 2015, S. 781; Studimup.de (Hrsg.), 2019).

Genetische Algorithmen sind Optimierungsalgorithmen, die in ihrer Funktionsweise vom Prozess der natürlichen Selektion inspiriert sind, was sich auch an den Begrifflichkeiten zeigt. Statt einer Zielfunktion wird eine Fitnessfunktion optimiert, die in dieser Implementierung durch die Zielfunktion des p-Median-Problems (Gleichung 3.4) beschrieben wird. Fitnessfunktionen genetischer Algorithmen werden üblicherweise folgendermaßen optimiert: Zu Beginn wird eine Population gebildet, die aus einer Menge von zufälligen zulässigen Lösungen des Problems besteht. Jedes Individuum dieser Population, also jede zufällige Lösung, wird Chromosom genannt. Im Kontext dieser Arbeit repräsentiert jedes Chromosom eine zulässige Lösung des p-Median-Problems, d. h., dass aus der Menge der n POIs p POIs als Ladestandort ausgewählt werden. Dargestellt wird dieses Chromosom als eine Liste der Länge n, wobei jedes Listenelement einen POI repräsentiert. In der Liste steht am Index i der Wert 1, wenn der POI i als Ladestandort ausgewählt ist, sonst 0. Anschließend wird für jedes Chromosom der Fitnesswert ausgerechnet und je geringer dieser ausfällt, desto besser ist die Lösung, die das Chromosom repräsentiert. Abbildung 3.5 zeigt beispielsweise auf, wie der Fitnesswert zu einer Lösung berechnet werden kann (Jaramillo et al., 2002).

Nachdem auf diese Weise eine Anfangspopulation erstellt ist, werden in einem iterativen Verfahren immer neue Generationen gebildet, indem ein Teil der Vorgängergeneration durch neu gebildete Chromosomen ersetzt wird, die im Schnitt einen besseren Fitnesswert haben sollen. Hierzu werden einige Chromosomen aus der aktuellen Population selektiert und mit diesen selektierten Chromosomen durch Kreuzungen (Crossover) und Mutationen neue Chromosomen gebildet. Für die Selektion gibt es verschiedene Verfahrensweisen. In dieser Arbeit wird das Verfahren Binary-Tournament-Selection ausgewählt, da dieses zu guten Ergebnissen führt (Beasley and Chu, 1994, S. 394). Bei diesem Selektionsverfahren werden zufällig zwei verschiedene Chromosomen aus der Vorgängergeneration ausgewählt und das Chromosom beibehalten, das den besseren Fitnesswert hat. Nachdem auf diese Weise zwei verschiedene Chromosomen (Eltern) ausgewählt wurden, wird ein Crossover durchgeführt, um ein neues Chromosom (Kind) zu bilden. In dieser Arbeit wird das Crossover mit einem Fusions-Operator implementiert. Dabei erhält das Kind

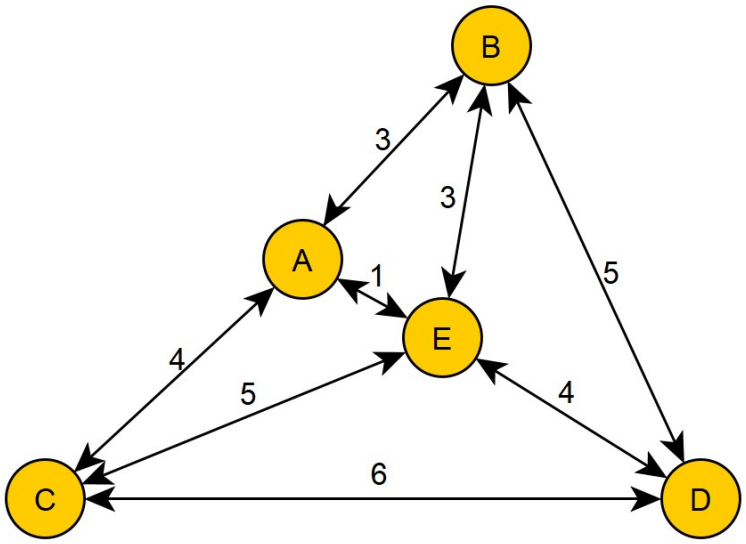


Abbildung 3.5: Beispielgraph zur Berechnung der Fitnessfunktion des p-Median-Problems. Sei $p = 2$, die Punkte „ABCDE“ die POIs eines Gebietes und „10010“ eine zufällige Lösung, die besagt, dass die Punkte A und D als Ladestandorte ausgewählt sind. Dann berechnet sich der Fitnesswert mit $3 + 4 + 1 = 8$, da dies die Gesamtsumme der Bedarfspunkte B, C und E ist, um einen nächstliegenden Ladestandort A oder D zu erreichen. Quelle: Eigene Darstellung.

an Index i den gleichen Wert der Elternchromosomen, wenn an dieser Stelle die Werte der Elternchromosomen übereinstimmen. Haben die Elternchromosomen an Index i unterschiedliche Werte (0 und 1), erhält das Kind mit einer höheren Wahrscheinlichkeit den Wert desjenigen Elternchromosoms, das einen besseren Fitnesswert aufweist (Fusions-Operator). Mathematisch wird dies durch die Gleichungen 3.8 und 3.9 beschrieben, wobei C für die Chromosomenliste des Kindes, P_1 und P_2 für die Chromosomenlisten der Eltern und f_{P_1} bzw. f_{P_2} für die Fitnesswerte der Eltern steht (Beasley and Chu, 1994, S. 395).

$$C[i] := P_1[i] \text{ mit Wahrscheinlichkeit } p = \frac{f_{P_2}}{f_{P_1} + f_{P_2}} \tag{3.8}$$

$$C[i] := P_2[i] \text{ mit Wahrscheinlichkeit } 1 - p \tag{3.9}$$

Da ein Crossover dazu führen kann, dass im Chromosom des Kindes nicht genau p POIs als Ladestandorte ausgewählt sind und die erhaltene Lösung somit unzulässig ist, muss das Chromosom eventuell noch in eine gültige Lösung überführt werden. Anschließend wird mit einer festgelegten Mutationswahrscheinlichkeit gegebenenfalls eine Mutation am neuen Chromosom durchgeführt. Mutationen erfüllen bei genetischen Algorithmen den Zweck, lokale Optima zu überwinden (Jaramillo et al., 2002, S. 765). Hierbei werden an

zwei beliebigen Stellen des Chromosoms die Werte vertauscht, indem eine 0 in eine 1 und eine 1 in eine 0 überführt wird.

Des Weiteren wird empfohlen, Duplikate, d. h. gleiche Chromosomen bzw. Lösungen in einer Population, nicht oder nur in einer geringen Anzahl zuzulassen (Beasley and Chu, 1994, S. 397). Der implementierte Algorithmus stellt sicher, dass beim Erstellen der Population sowie durch Mutation keine Duplikate zur bereits existierenden Population zugefügt werden.

Abbildung 3.6 gibt eine beispielhafte Übersicht über den erläuterten Ablauf des Algorithmus. Die Distanzen zur Berechnung der Fitnesswerte des Beispiels ergeben sich aus dem Graphen in Abbildung 3.5.

Die Parameter Populationsgröße, Mutationswahrscheinlichkeit und die Anzahl der Iterationen, die dem Algorithmus als Input übergeben werden, können die Qualität der Ergebnisse entscheidend beeinflussen. Der genetische Algorithmus dieser Arbeit ist daher so implementiert, dass er eine Lösung mit folgenden von der Literatur empfohlenen Standardeinstellungen berechnet, sofern keine anderen Parameterwerte an ihn übergeben werden (Efthymiou et al., 2017, S. 27; Beasley and Chu, 1994; Jaramillo et al., 2002):

- Die Populationsgröße soll zwischen n und $2n$ betragen, wobei n der Anzahl der POIs eines Gebietes entspricht. Der implementierte Algorithmus erstellt standardmäßig eine Population der Größe n.
- Die Mutationswahrscheinlichkeit soll gering und etwa 5 bis 10 Prozent betragen. Der implementierte Algorithmus verwendet standardmäßig eine Mutationswahrscheinlichkeit von 5 Prozent.
- Die Anzahl der Iterationen kann beispielsweise durch eine Maximalzahl oder durch eine Mindeständerung des besten Fitnesswertes zum besten Fitnesswert der Vorgängergeneration festgelegt werden. Der implementierte Algorithmus führt standardmäßig 50 Iterationen aus, auch wenn schon vorher keine nennenswerte Verbesserung des Fitnesswertes mehr erreicht wird.

Nachdem der genetische Algorithmus die p-Median-Lösung berechnet und folglich p POIs als Ladestandorte ausgewählt hat, werden in einem Nachbereitungsschritt die POIs, die nicht als Ladestandorte ausgewählt sind, dem jeweils nächstgelegenen Ladestandort

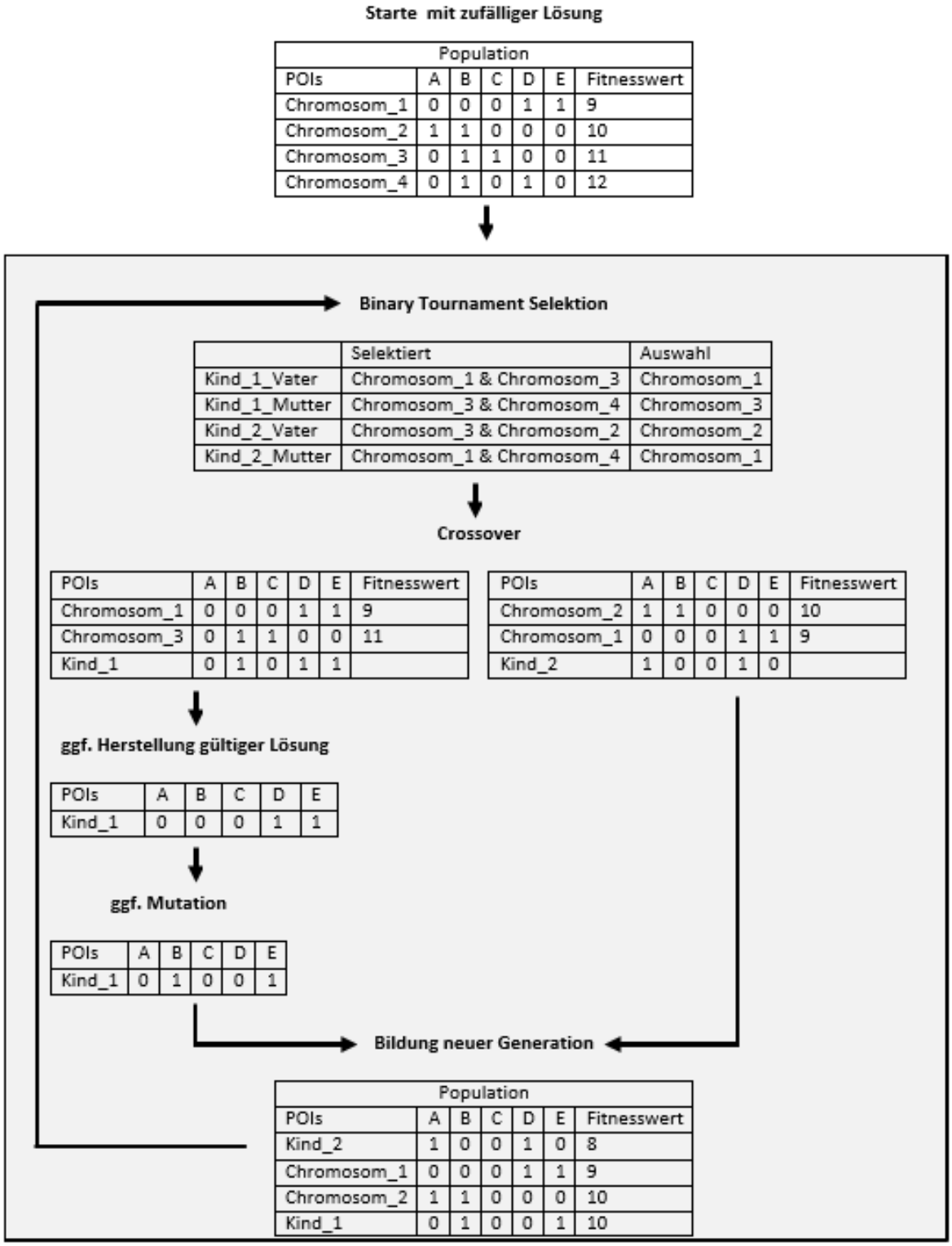


Abbildung 3.6: Überblick über die Funktionsweise des genetischen Algorithmus. Es wird eine Beispieliteration zur Lösung des 2-Median-Problems mit den POIs A, B, C, D und E dargestellt. Die Distanzen zur Berechnung der Fitnesswerte ergeben sich aus dem Graphen in Abbildung 3.5. Quelle: Eigene Abbildung nach: Jaramillo et al., 2002, S. 762.

zugeteilt, wobei wieder der kürzeste Weg als Distanzmaß verwendet wird. Das Ergebnis dieses Nachbereitungsschrittes sind p Cluster, wobei in jedem Cluster ein POI als Ladestandort ausgewählt ist. Die Implementierung erfolgt mittels einer existierenden Lösung für das p-Median-Problem die von GitHub heruntergeladen und auf den hier beschriebenen Anwendungsfall angepasst und erweitert wird (Devran, 2018).

3.3 Auswahl der Untersuchungsgebiete

Die genannten Algorithmen werden auf die drei Untersuchungsgebiete Nord-Amsterdam, Nord-Seattle und den Davutpaşa-Campus der Yildiz Universität in Istanbul angewandt.

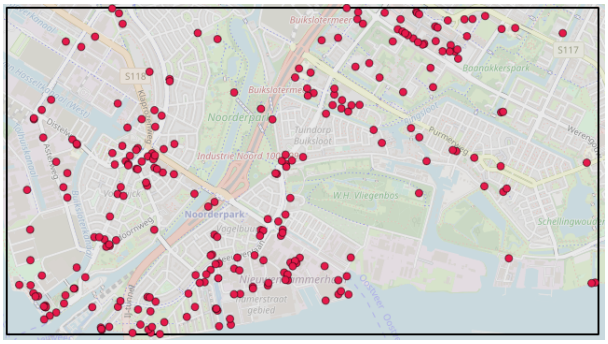
Der Hauptgrund für die Auswahl dieser Gebiete ist, dass diese in anderen Forschungsarbeiten bereits auf die Platzierung optimaler Ladestandorte hin untersucht wurden und diese Ergebnisse somit zur Evaluation der in dieser Arbeit berechneten Ergebnisse verwendet werden können (vgl. Kapitel 2.2).

Alle drei Untersuchungsregionen haben gemeinsam, dass es sich um städtische Gebiete handelt, bei denen sich Clusterverfahren eher als in ländlichen Gebieten oder an Autobahnen anbieten (Sathaye and Kelley, 2013, S. 29). Neben dieser Gemeinsamkeit sind aber auch einige Unterschiede zwischen den Gebieten zu verzeichnen, die für die Anwendung der Algorithmen interessant sein können. Tabelle 3.1 listet einige Gebietsmerkmale auf.

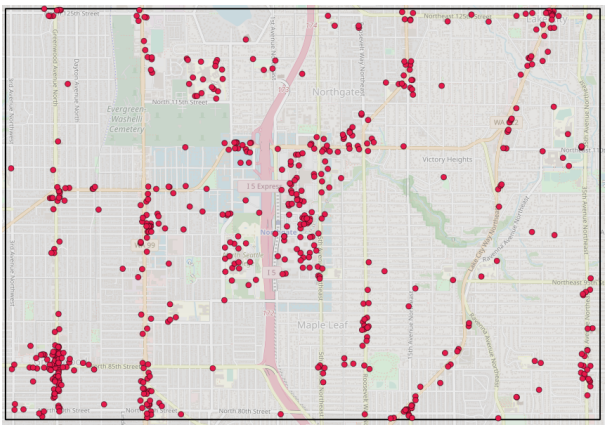
Tabelle 3.1: Eigenschaften der Untersuchungsgebiete. Quelle: Eigene Darstellung.

Parameter	**Nord-Amsterdam**	**Nord-Seattle**	**Davutpaşa-Campus**
Größe der OSM-Bounding Box in km^2	7,5	20,4	3,0
Größe des berücksichtigten Gebietes in km^2	7,5	20,4	1,1
Straßenführung	organisch gemischt	schachbrett-artig	organisch
Vorkommen natürlicher Cluster	ja	ja	ja
Anzahl der berücksichtigen POIs im Gebiet	257	552	68
Anzahl optimaler Ladestandorte	16	10	8

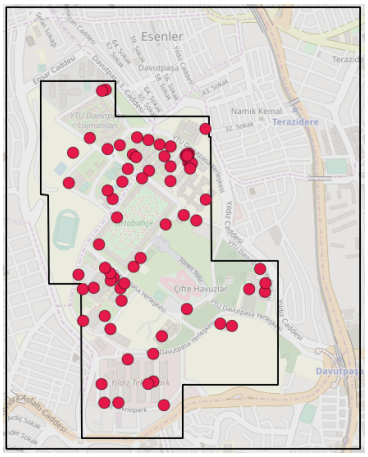
Die Größe der Gebiete wird dabei mithilfe der Software GoogleEarth bestimmt, indem manuell ein Polygon in der Größe der Bounding-Box erstellt wird. Abbildung 3.7 zeigt die drei Gebiete und die in ihnen liegenden POIs aus OpenStreetMap, die als relevant für die Berechnung der Ladestandorte erachtet werden (vgl. Kapitel 3.1).



(a) Die OSM-POIs in der Bounding Box des Untersuchungsgebietes Nord-Amsterdams.



(b) Die OSM-POIs in der Bounding Box des Untersuchungsgebietes Nord-Seattles.



(c) Die OSM-POIs des Untersuchungsgebietes in der Bounding Box (innere) des Davutpaşa-Campus. Die äußere Bounding Box dient als Begrenzung des Straßennetzwerks beim Routing, da hier nur eine viereckige Bounding Box angegeben werden kann.

Abbildung 3.7: Überblick über die ungeclusterten OSM-POIs der Untersuchungsgebiete. Quelle: Eigene Darstellung.

3.4 Webseitenbasierte Darstellung der Ergebnisse

Da sich die Aussagekraft der GeoJSON-Dateien nicht auf den ersten Blick erschließt, werden die Ergebnisse mittels der JavaScript-Bibliothek Leaflet auf zwei Webseiten in interaktiven Karten präsentiert.

Eine der beiden Webseiten dient dazu, die in Kapitel 4 vorgestellten Ergebnisse darzustellen

und somit die Transparenz der erhaltenen Ergebnisse sicherzustellen. Abbildung 3.8 gibt einen Überblick über die Funktionalitäten der Webseite. Um ein bestimmtes Ergebnis auf der Karte anzuzeigen, muss mit einem Klick auf die entsprechenden Buttons zuerst ein Untersuchungsgebiet und dann ein Algorithmus ausgewählt werden. Für jedes Gebiet gibt es außerdem die Möglichkeit, die optimalen Ladestandorte anzeigen zu lassen. Weiterhin können mit einem Klick auf einen POI auch dessen Attribute, die in der GeoJSON-Datei zum POI abgespeichert werden, eingesehen werden (vgl. Kapitel 3.1). Zudem enthält sie auch die Funktionalität, über die Legende auszuwählen, ob alle POIs oder nur die berechneten Ladestandorte eines Gebietes angezeigt werden sollen. Bei letztgenannter Funktionalität empfiehlt es sich, die Webseite neu zu laden, bevor ein anderes Gebiet oder ein anderer Algorithmus ausgewählt werden.

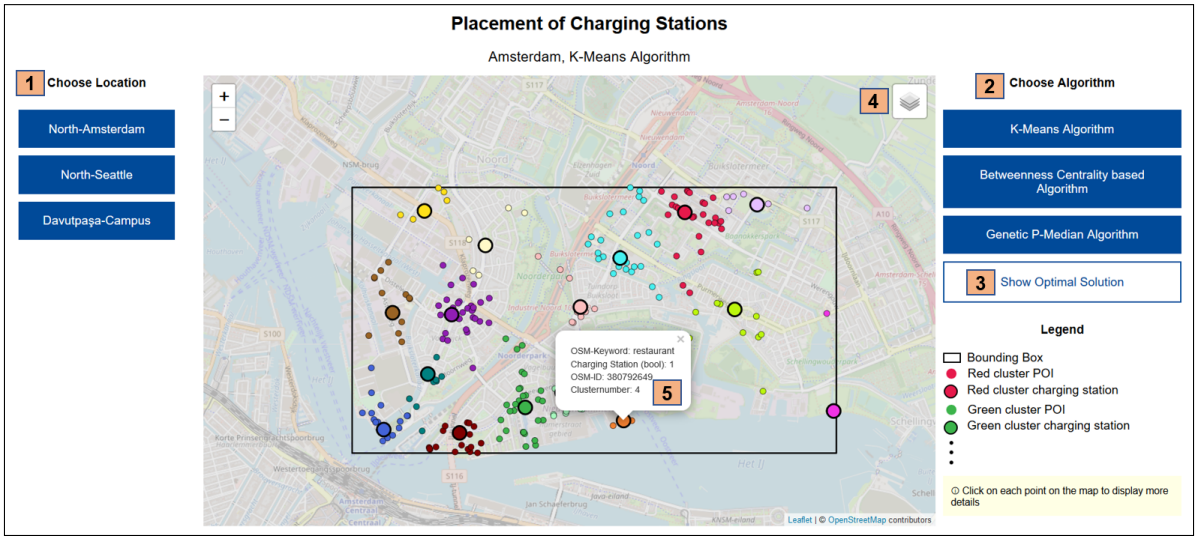


Abbildung 3.8: Screenshot der Webseite zum Anzeigen der berechneten Ergebnisse. Die roten Nummern weisen auf diese Funktionalitäten hin: 1: Auswahl Untersuchungsgebiet, 2: Auswahl Algorithmus, 3: Anzeigen der optimalen Ladestandorte eines ausgewählten Untersuchungsgebiets, 4: Auswahl, ob alle POIs oder nur die Ladestandorte angezeigt werden, 5: Attributwerte eines POIs anzeigen lassen. Quelle: Eigene Darstellung.

Die zweite Webseite zeigt jeweils immer nur das zuletzt berechnete Ergebnis an und dient damit zum einen zum Debuggen und zum anderen dazu, Transparenz über die berechneten Ergebnisse und die Funktionalitäten der Algorithmen herzustellen. Sie verfügt ebenfalls über die Funktionalität mit einem Klick auf einen POI dessen Attribute einzusehen. Wird eine neue Berechnung durchgeführt, ist ein Neuladen der Seite notwendig, um das neu berechnete Ergebnis anzusehen.

Da die Webseiten nur zum Betrachten der Ergebnisse dienen, aber ansonsten keinen

wissenschaftlichen Anspruch haben, wird an dieser Stelle nicht weiter auf die hierfür verwendeten Datenstrukturen oder benötigten Laufzeiten eingegangen.

3.5 Methodik zur empirischen Analyse der Ergebnisse

In derzeitigen wissenschaftlichen Publikationen sind qualitative Analysen zu berechneten optimalen Ladestandorten eine Ausnahme. Aus diesem Grund kann in dieser Arbeit auf keine standardisierte Evaluationsmethode zurückgegriffen werden und es werden eigene Ansätze zur Evaluierung der berechneten Ergebnisse entwickelt und in Kapitel 3.5.1 bis 5.5.3 beschrieben.

3.5.1 Bestimmung der Differenz zur optimalen Lösung mittels der Ungarischen Methode

Bei dieser Evaluationsmethode wird die Differenz zwischen dem berechneten Ergebnis und einem Ergebnis aus veröffentlichen Forschungsarbeiten, das als optimal definiert wird, bestimmt (vgl. Kapitel 2.2). Um die Differenz zu erhalten, wird der mindestens nötige Abstand in Metern berechnet, um die berechnete Platzierung in die optimale Platzierung zu überführen. Als Abstandsmaß dient hierbei wiederum die kürzeste Wegdistanz zwischen zwei Ladestandorten. Damit ist die Differenz ein Maß dafür, wie weit das berechnete Ergebnis vom optimalen Ergebnis abweicht. Die Überführung der berechneten in die optimalen Standorte mittels der Ungarischen Methode setzt voraus, dass die Anzahl der optimalen und berechneten Ladestandorte gleich ist, was bei den gewählten Szenarien der Fall ist, da die optimale Menge an Ladestandorten den optimalen Lösungen entnommen wird (vgl. Kapitel 3.1).

Die Überführung eines berechneten Ergebnisses in ein optimales Ergebnis erfolgt, indem jeder Ladestandort des berechneten Ergebnisses einem Ladestandort des optimalen Ergebnisses zugeordnet wird. Dabei kann jeder optimale Standort nur genau eine Zuweisung erhalten. Die Summe der Distanzen von den berechneten Ladestandorten zu den zugewiesenen optimalen Standorten ergibt dann die Differenz der beiden Ladestandortplatzierungen und sollte möglichst minimal sein. Mathematisch kann dies mit der Zielfunktion von Gleichung 3.10 beschrieben werden, wobei C die Distanz- bzw. Kostenmatrix ist und X eine boolesche Matrix. $X[i, j] = 1$ genau dann, wenn Reihe i zur

Spalte j zugeordnet wird, was gleichbedeutend damit ist, dass der berechnete Ladestandort i dem optimalen Ladestandort j zugewiesen wird.

$$min \sum_{i} \sum_{j} C_{ij} X_{ij} \quad \text{so dass,} \tag{3.10}$$

$$\sum_{i} X_{ij} = 1, \qquad \forall i \tag{3.11}$$

$$\sum_{j} X_{ij} = 1, \qquad \forall j \tag{3.12}$$

Nebenbedingung 3.11 sorgt dafür, dass jeder berechnete Ladestandort genau einem optimalen Ladestandort zugewiesen wird. Nebenbedingung 3.12 stellt sicher, dass jeder optimale Ladestandort genau einem berechnetem Ladestandort zugewiesen wird (Srinivasan, 2009). Die Berechnung der Distanzmatrix, die auch in dieser Implementation benötigt wird, liegt bereits in $O(n^2)$, wobei n die Gesamtanzahl der Ladestandorte ist, da n^2 Distanzen zwischen den berechneten und optimalen Ladestandorten berechnet werden müssen.

Das naive Ausprobieren aller möglichen Zuordnungen ist in diesem Fall nicht empfehlenswert, da es für eine Menge von $2n$ Ladestandorten $n!$ mögliche Zuordnungen gibt (n Möglichkeiten für die erste Zuordnung, $n-1$ Möglichkeiten für die zweite Zuordnung, usw.) (Varyani, 2013). Auch eine naive Implementation eines gierigen (greedy) Algorithmus, der nacheinander jeden berechneten Ladestandort zum jeweils nächstgelegenen optimalen Ladestandort zuweist, bietet sich nicht an, da dies zu beliebig schlechten Ergebnissen führen kann, wie in Abbildung 3.9 exemplarisch gezeigt wird.

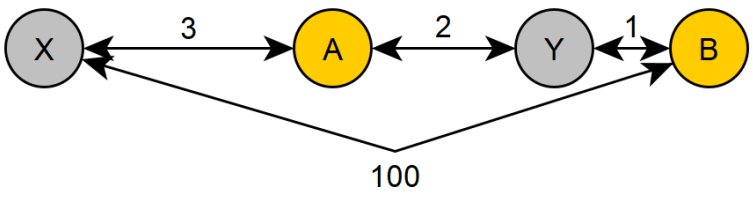


Abbildung 3.9: Beispiel einer suboptimalen Zuordnung der berechneten Ladestandorte (gelb) zu den optimalen Ladestandorten (grau) eines gierigen Algorithmus. Statt A zu X zuzuordnen und B zu Y (Distanz = 4), wird A zu Y zugeordnet und B zu X (Distanz = 102). Quelle: Eigene Darstellung.

Stattdessen bietet sich die Überführung des Zuordnungsproblems der Ladestandorte in eine

bipartite Graphenstruktur an. Dabei bilden die berechneten und optimalen Ladestandorte die Knoten und die kürzesten Wegedistanzen zwischen jedem berechneten und jedem optimalen Ladestandort bilden gewichte Kanten. Dabei verläuft jedoch keine Kante innerhalb der Knotenmenge der berechneten Ladestandorte bzw. der Knotenmenge der optimalen Ladestandorte. Die optimale Lösung des beschriebenen Zuordnungsproblems entspricht dann einem perfekten Matching mit minimalen Kosten wie Abbildung 3.10 exemplarisch aufzeigt (Kuhn, 2013).

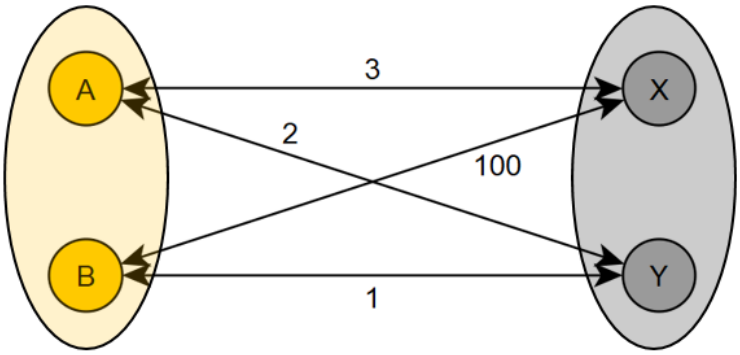


Abbildung 3.10: Überführung des Zuordnungsproblems in eine bipartite Graphenstruktur. Die berechneten Ladestandorte (gelb) und die optimalen Ladestandorte (grau) bilden jeweils eine Menge. Die Kantengewichte entsprechen den Distanzen zwischen den Ladestandorten. Eine optimale Zuordnung der Ladestationen entspricht dann einem perfekten Matching mit minimalen Kosten. Quelle: Eigene Darstellung.

Diese Lösung kann mittels der von James Munkres entwickelten Ungarischen Methode berechnet werden, die eine polynomielle Laufzeit benötigt (Varyani, 2013). Auf eine ausführliche Beschreibung der Funktionsweise wird in dieser Arbeit verzichtet, da diese nicht relevant für die Ergebnisse dieser Arbeit ist. Srinivasan liefert jedoch beispielsweise eine gute und nachvollziehbare Erklärung des Algorithmus (Srinivasan, 2009).

Für die Implementation der Ungarischen Methode in dieser Arbeit wird das Linear-Sum-Assignment-Tool der Python-Bibliothek Scipy genutzt, welches das genannte Zuordnungsproblem mittels der Ungarischen Methode löst (SciPy community (Hrsg.), 2019, S. 1355). Da zur Evaluation nicht die eigentliche Zuordnung der Knoten interessant ist, sondern nur die Differenz zur optimalen Lösung, werden auch nur folgende Distanzen in Metern ausgegeben:

- Total Cost: die Gesamtdistanz zwischen den berechneten und den zugeordneten optimalen Ladestandorten

- Mean Cost per Station: die durchschnittliche Distanz zwischen den berechneten und den zugeordneten optimalen Ladestandorten
- Minimum Cost: die minimale Distanz zwischen einem berechneten Ladestandort und einem zugeordneten optimalen Ladestandort
- Maximum Cost: die maximale Distanz zwischen einem berechneten Ladestandort und einem zugeordneten optimalen Ladestandort

Die durchschnittliche Distanz ist für den Vergleich der Qualität der verschiedenen Algorithmen bzw. Ladestandortplatzierungen dabei am aussagekräftigsten.

3.5.2 Nutzung der p-Median-Fitnessfunktion

Ein weiteres Qualitätsmerkmal, das zur Evaluation der Ergebnisse genutzt werden kann, ist die Anwendung der Fitnessfunktion des p-Median-Problems auf die berechneten Ladestandortlösungen. Die Fitnessfunktion des p-Median-Problems erlaubt es, mit einem implementierten Hilfsprogramm auch den Fitnesswert von bereits berechneten Ergebnissen einer GeoJSON-Datei einzulesen, auszurechnen und auszugeben. Somit können die Fitnesswerte verschiedener Ergebnisse bzw. Algorithmen miteinander verglichen werden. Erwartet wird, dass der genetische Algorithmus, der genau diesen Wert optimiert, hier die besten Ergebnisse liefert.

3.5.3 Vergleich mit einem zufälligen Ergebnis

Um einen weiteren Referenzwert für die Evaluation der Algorithmen zur Verfügung zu haben, wird außerdem pro Untersuchungsgebiet ein zufälliges Ergebnis berechnet und mit den genannten Evaluationsmethoden untersucht. Ein Vergleich dieser zufälligen Evaluationswerte mit den Evaluationswerten der (Optimierungs-)Algorithmen ermöglicht es, eine Aussage darüber zu treffen, ob letztgenannte Algorithmen überhaupt ein Ergebnis berechnen können, das besser ist als der Zufall. Zur Berechnung der zufälligen Lösung wird das Python Modul random verwendet, das aus der gegebenen Liste von POIs uniform verteilt k verschiedene Listenelemente bzw. POIs auswählt, diese als Ladestandorte zurückgibt und ebenfalls in einer GeoJSON-Datei abspeichert. Im Anhang in Abbildung A0.2 sind die für diese Arbeit mit dem Zufalls-Algorithmus berechneten Ergebnisse in Karten dargestellt.

4 Ergebnisse und Ergebnisevaluation

In diesem Kapitel werden die Ergebnisse jedes Untersuchungsgebietes beschrieben und eine qualitative Bewertung dieser vorgenommen (Kapitel 4.1 bis 4.3). In Kapitel 4.4 werden anschließend die verschiedenen Ergebnisse zusammengefasst und miteinander in Beziehung gesetzt, um allgemeingültige Aussagen über die Eignung der implementierten Algorithmen für die drei Untersuchungsgebiete zu treffen. Weiterhin werden die realen Laufzeiten, welche für die Berechnung der Ergebnisse benötigt werden, aufgelistet und miteinander verglichen. Alle hier vorgestellten Berechnungen werden mit einem gewöhnlichen Laptop, der über 8 GB RAM, 2.3 GHz und einen Intel® Core™ i5-6200U verfügt, berechnet.

4.1 Nord-Amsterdam

Abbildungen 4.1 bis 4.3 zeigen die Clusterergebnisse der drei implementierten Algorithmen sowie die Ladestandorte, die für das Untersuchungsgebiet Nord-Amsterdam jeweils aus den Clustern ausgewählt werden. In den Abbildungen ist erkennbar, dass die mit euklidischer Distanz berechneten k-Means-Cluster in sich geschlossener und punktförmiger ausfallen, während die Cluster der beiden anderen Algorithmen, die auf den realen kürzesten Wegen beruhen, eher durch den Verlauf breiterer Straßen in die Länge gezogen werden. Die Verwendung der kürzesten Wege als Distanzmaß führt außerdem zu Ergebnissen, bei denen POIs, die auf der Karte sehr nahe beieinander liegen, zum Teil in unterschiedliche Cluster eingeordnet werden. Bei allen drei Algorithmen ist die Anzahl der POIs in den Clustern uneinheitlich.

Abbildung 4.4 (d) zeigt die optimale Lösung aus der Literatur für die Platzierung der Ladestandorte in der ausgewählten Bounding Box. Die Ladestandorte dieser optimalen Lösung sind sehr gleichmäßig und flächendeckend räumlich verteilt sowie häufig in der Nähe oder an größeren Straßen zu finden.

Die berechneten Platzierungen der Ladestandorte durch die drei in dieser Arbeit implementierten Algorithmen, die in Abbildungen 4.4 (a) bis (c) dargestellt werden, zeigen, dass sowohl der k-Means-Algorithmus als auch der p-Median-Algorithmus die Ladestandorte weitestgehend räumlich verteilt. Der auf Knotenzentralitätswerten beruhende Algorithmus platziert Ladestandorte dagegen eher an bedeutenden Straßen,

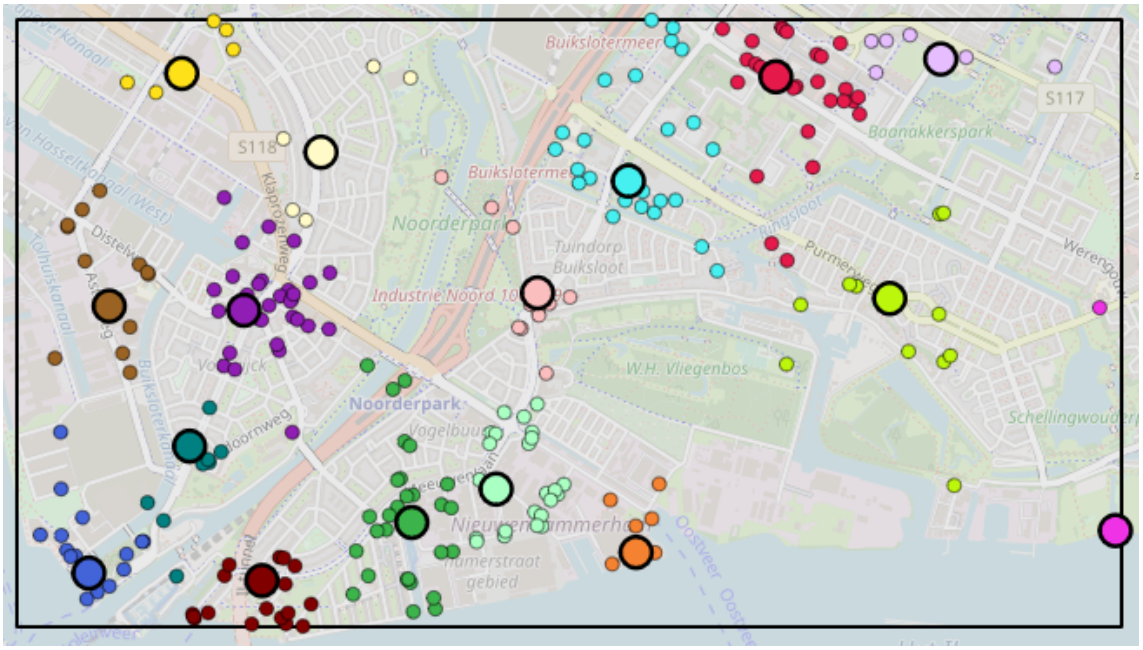


Abbildung 4.1: Die Ergebniscluster des k-Means-Algorithmus in Nord-Amsterdam. Jedes Cluster ist in einer Farbe dargestellt. Die POIs mit der schwarzen Umrandung sind die ausgewählten Ladestandorte des Clusters. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

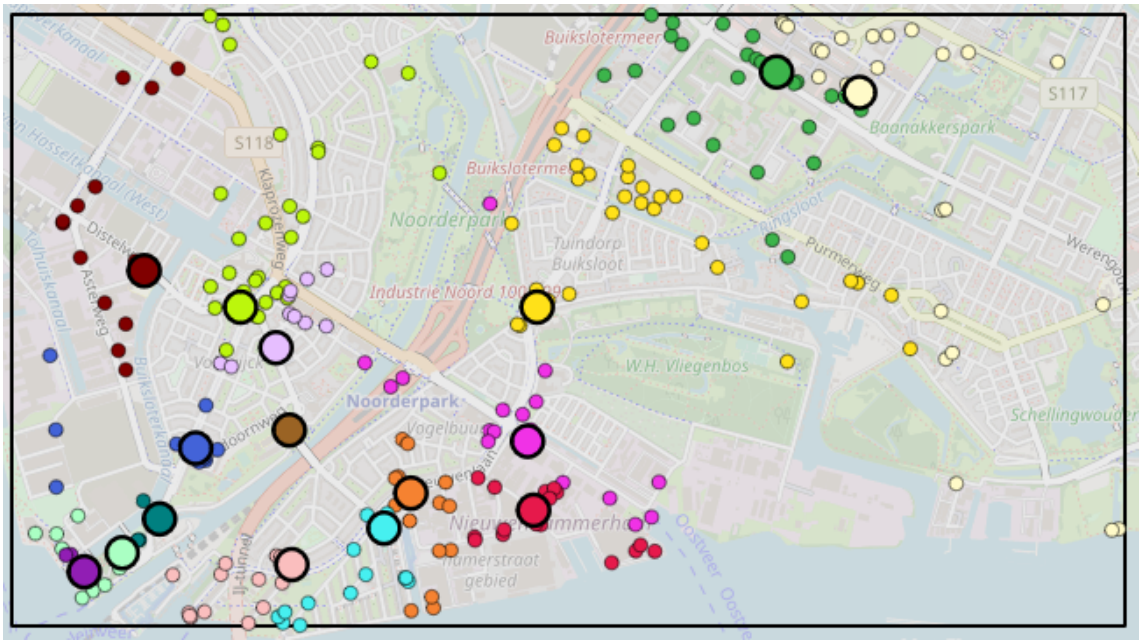


Abbildung 4.2: Die Ergebniscluster des Knotenzentralitätswerte-Algorithmus in Nord-Amsterdam. Jedes Cluster ist in einer Farbe dargestellt. Die POIs mit der schwarzen Umrandung sind die ausgewählten Ladestandorte des Clusters und haben einen Mindestabstand von 100 Metern. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

da hierüber die meisten kürzesten Wege verlaufen. Da als Mindestabstand zwischen den Ladestandorten nur 100 m festgesetzt werden, ist hier folglich keine räumlich

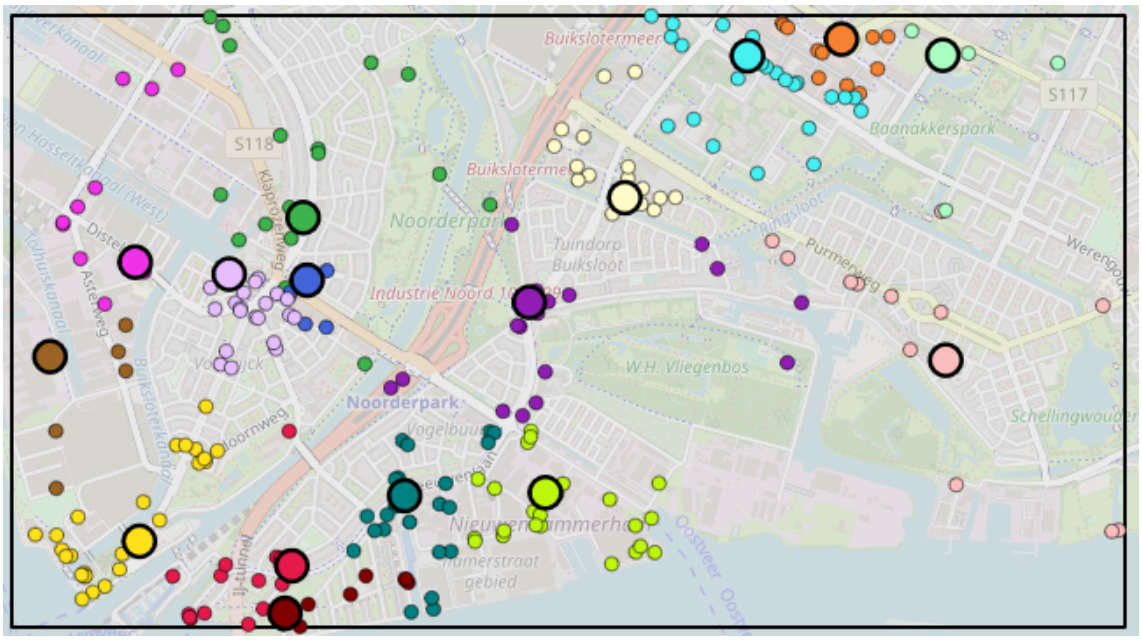
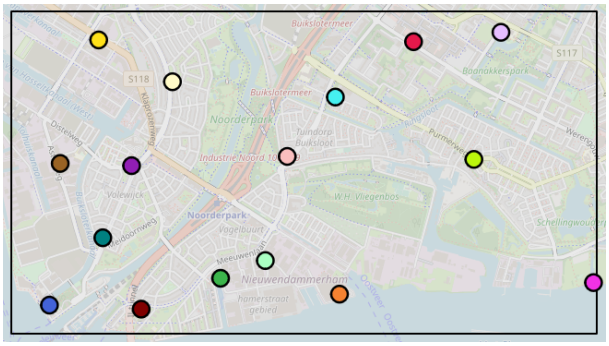


Abbildung 4.3: Die Ergebniscluster des genetischen p-Median-Algorithmus in Nord-Amsterdam. Jedes Cluster ist in einer Farbe dargestellt. Die POIs mit der schwarzen Umrandung sind die ausgewählten Ladestandorte des Clusters. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

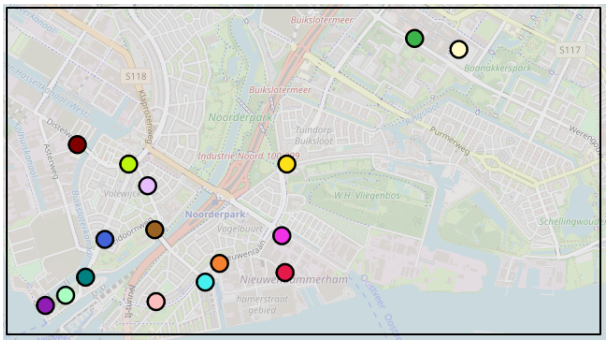
flächendeckende Verteilung der berechneten Ladestandorte gegeben. Da das letztgenannte Ergebnis unter Umständen nicht gewünscht ist, zeigt Abbildung 4.5 ein Ergebnis des auf Knotenzentralitätswerten beruhenden Algorithmus, bei dem ein Mindestabstand von 500 Metern zwischen den Ladestandorten eingehalten wird. Bei diesem Ergebnis sind die Ladestandorte bereits besser räumlich verteilt.

Ein qualitativer Vergleich der erhaltenen Werte mit der optimalen Lösung, der in Tabelle 4.1 festgehalten ist, zeigt, dass die Gesamtdifferenz zur optimalen Lösung beim k-Means-Algorithmus am geringsten ist. Die Berechnung der Knotenzentralitätswerte ist nur etwa halb so gut. Der Fitnesswert ist jedoch wie zu erwarten beim p-Median-Algorithmus, der genau diesen Wert optimiert, am besten. Er konnte bei dieser Ausführung von anfänglich 91 km auf 79 km gesteigert werden. Der k-Means-Algorithmus kommt jedoch mit nur etwa 2 km Unterschied und einer deutlich schnelleren Laufzeit bereits sehr nahe an diesen Wert heran.

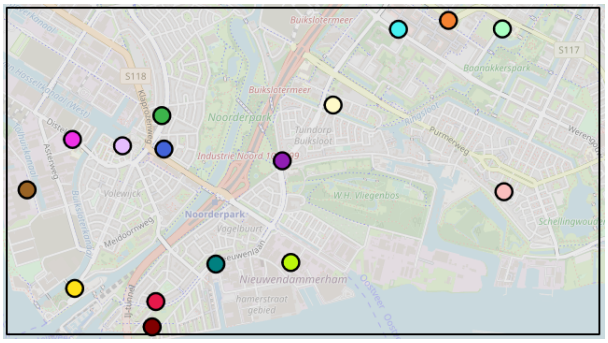
Die realen Laufzeiten der Algorithmen unterscheiden sich genau wie die theoretischen Laufzeiten, bei der Ausführung der drei Algorithmen auf das Untersuchungsgebiet Nord-Amsterdam, deutlich (vgl. Kapitel 3.2.1 bis 3.2.3). Allein die Berechnung der 257x257-Distanzmatrix, die beim Knotenzentralitätswerte-Algorithmus und dem genetischen p-



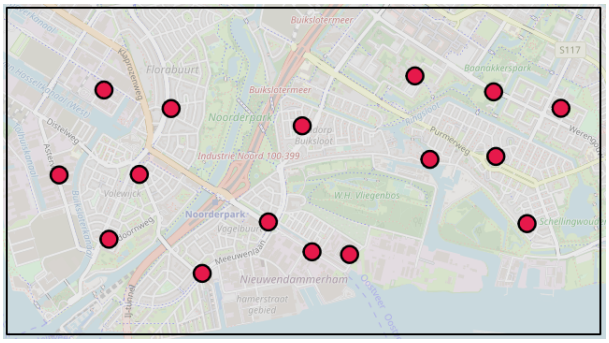
(a) Ladestandorte des k-Means-Algorithmus.



(b) Ladestandorte des Knotenzentralitätswerte-Algorithmus mit einem Mindestabstand von 100 Metern.



(c) Ladestandorte des genetischen p-Median-Algorithmus.



(d) Ladestandorte der optimalen Lösung.

Abbildung 4.4: Die mit den verschiedenen Algorithmen berechneten Ladestandorte für Nord-Amsterdam. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

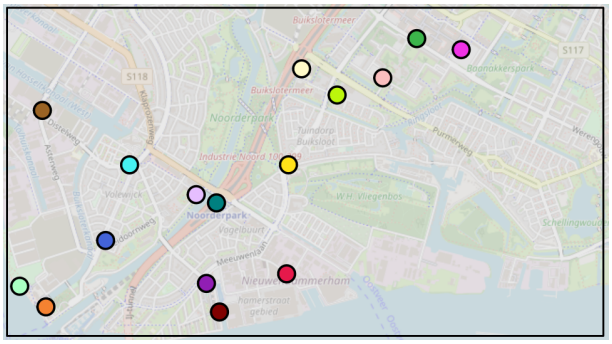


Abbildung 4.5: Die berechneten Ladestandorte des Knotenzentralitätswerte-Algorithmus für Nord-Amsterdam. Der Mindestabstand zwischen den Ladestandorten beträgt bei dieser Lösung 500 Meter. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

Median-Algorithmus verwendet wird, benötigt etwa 22 Minuten. Die Ausführung des k-Means-Algorithmus benötigt dagegen etwa 14 Sekunden.

Tabelle 4.1: Qualitative Evaluation der berechneten Ergebnisse für Nord-Amsterdam. Alle Angaben in Metern. Angegeben sind die Fitnesswerte sowie die jeweiligen Distanzen der berechneten Ergebnisse zu den optimalen Lösungen. Die Mindestdistanz zwischen den Ladestandorten beträgt bei dem auf Knotenzentralitätswerten basierten Algorithmus 100 Meter (a) bzw. 500 Meter (b). Quelle: Eigene Darstellung.

Parameter	***k*-Means-Algorithmus**	**Knoten-zentralitäts-werte (a)**	**Knoten-zentralitäts-werte (b)**	***p*-Median-Algorithmus**	**Zufällige Platzierung**
Gesamtdistanz	11702	20444	14174	13216	23547
Mittlere Distanz pro Ladestandort	731	1278	886	826	1472
Minimale Distanz	101	19	19	43	43
Maximale Distanz	5066	4928	4928	4924	5000
Fitnesswert	81391	99194	90259	79452	126216

4.2 Nord-Seattle

Die Clusterergebnisse des k-Means-Algorithmus für Nord-Seattle sind trotz der schachbrettartigen Straßenverläufe aufgrund der Anordnung der POIs ähnlich ballförmig wie die von Nord-Amsterdam. Während die Clusterergebnisse des k-Means-Algorithmus und des genetischen p-Median-Algorithmus außerdem große Ähnlichkeiten aufweisen, unterscheidet sich das auf Knotenzentralitätswerten basierende Clusterergebnis von Nord-Seattle deutlich, was in den Abbildungen 4.6 bis 4.8 dargestellt ist.

Insbesondere die drei Cluster in der südwestlichen Ecke der Bounding Box von Abbildung 4.7 zeigen, dass hier drei Knoten bzw. Ladestandorte, durch die sehr viele kürzeste Wege verlaufen, sehr nah beieinander liegen. Da der Mindestabstand zwischen den Ladestandorten nur 100 Meter beträgt, werden auch alle drei Knoten als Ladestandorte ausgewählt. Das führt dazu, dass einige Cluster (z.B. das rote Cluster unten links) sehr klein sind und nur über wenige zugehörige POIs verfügen, während andere Cluster (z. B. das pinke und violette Cluster rechts) jeweils fast ein Sechstel des Gebietes einnehmen.

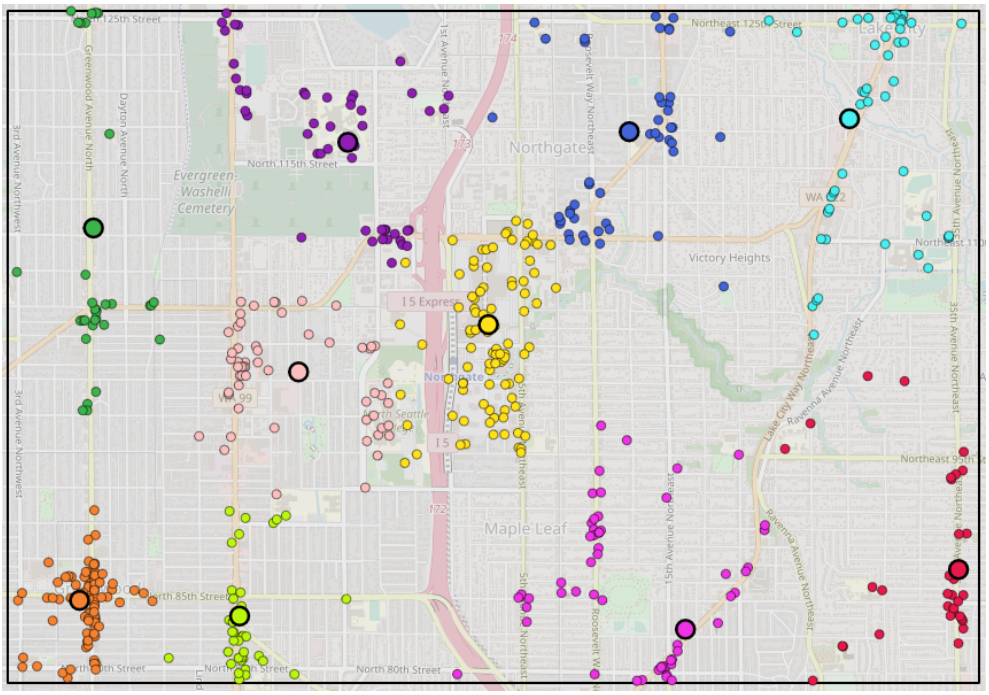


Abbildung 4.6: Die Ergebniscluster des k-Means-Algorithmus in Nord-Seattle. Jedes Cluster ist in einer Farbe dargestellt. Die POIs mit der schwarzen Umrandung sind die ausgewählten Ladestandorte des Clusters. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

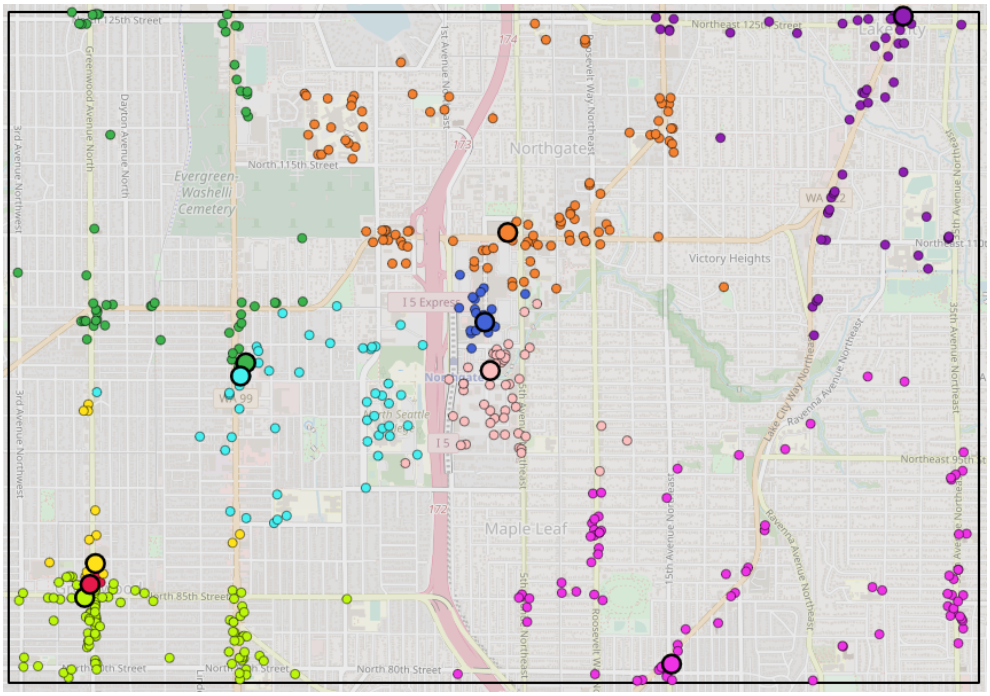


Abbildung 4.7: Die Ergebniscluster des Knotenzentralitätswerte-Algorithmus in Nord-Seattle. Jedes Cluster ist in einer Farbe dargestellt. Die POIs mit der schwarzen Umrandung sind die ausgewählten Ladestandorte des Clusters und haben einen Mindestabstand von 100 Metern. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

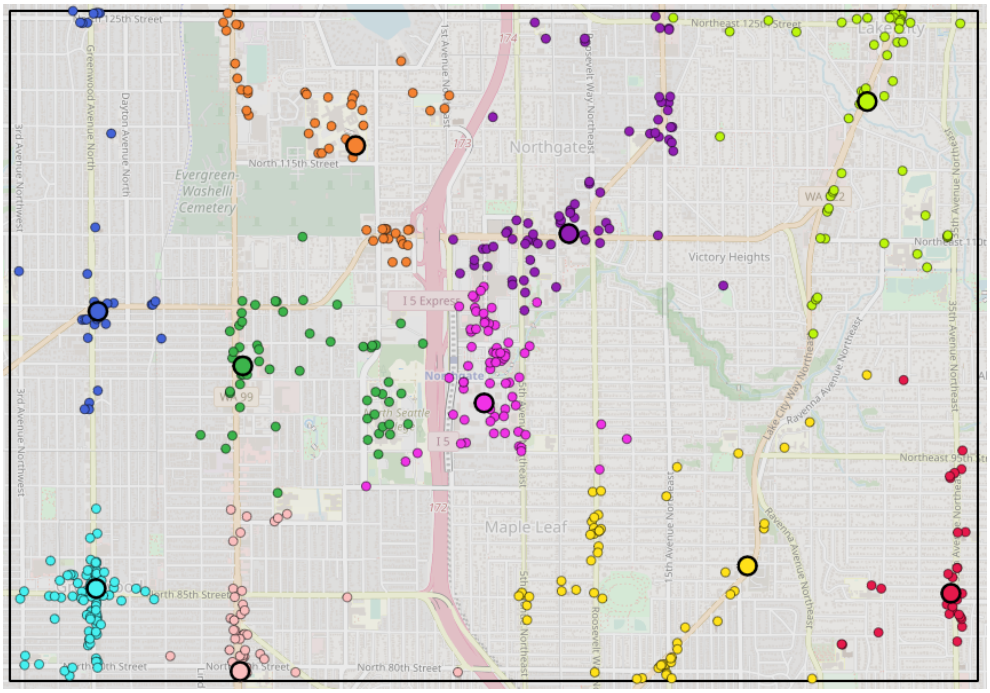
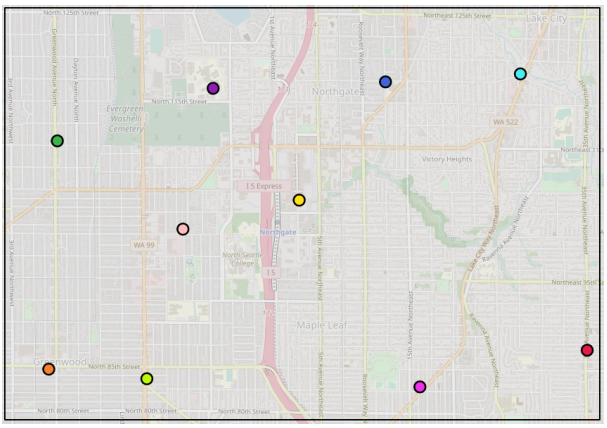


Abbildung 4.8: Die Ergebniscluster des genetischen p-Median-Algorithmus in Nord-Seattle. Jedes Cluster ist in einer Farbe dargestellt. Die POIs mit der schwarzen Umrandung sind die ausgewählten Ladestandorte des Clusters. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

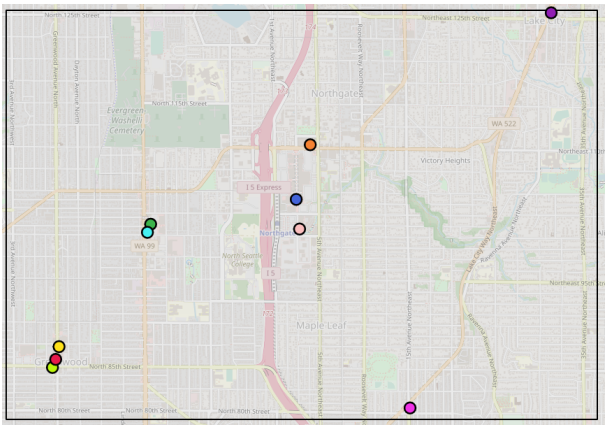
Abbildung 4.9 (a) bis (c) zeigt die Platzierung der Ladestandorte der optimalen Lösungen und der drei implementierten Algorithmen für Nord-Seattle im Vergleich. Auch hier ist wieder erkennbar, dass der k-Means-Algorithmus und der genetische p-Median-Algorithmus eine gute räumliche Abdeckung erreichen, während der auf Knotenzentralitätswerten basierende Algorithmus mehrere Ladestandorte sehr nah beieinander platziert. Da die optimale Lösung (siehe Abbildung 4.9 (d)) in Nord-Seattle anders als die von Nord-Amsterdam nicht über deren Gleichverteilung im Raum verfügt, schneidet dieser Algorithmus im qualitativen Vergleich bei der Differenzberechnung zur optimalen Lösung jedoch deutlich besser ab.

Tabelle 4.2 listet die Werte der qualitativen Ergebnisevaluation auf, die mit den verschiedenen Algorithmen für das Untersuchungsgebiet Nord-Seattle erreicht werden können.

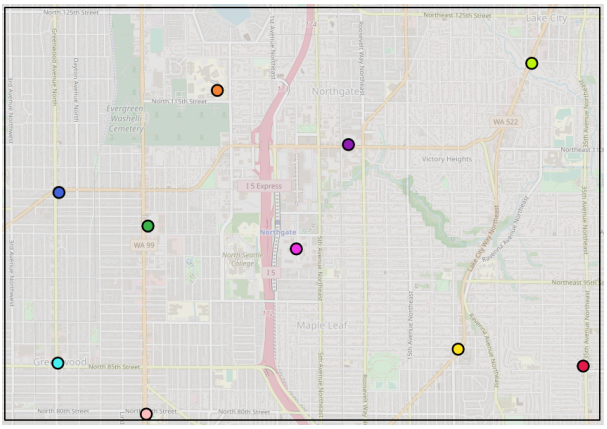
Ähnlich wie in Nord-Amsterdam unterscheiden sich auch hier die realen Laufzeiten erheblich. Während der k-Means-Algorithmus insgesamt knapp 40 Sekunden benötigt, benötigt allein die Berechnung der 552x552-Distanzmatrix, die in den anderen beiden



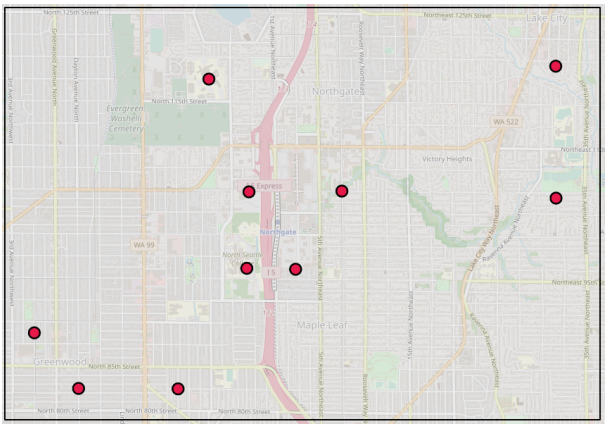
(a) Ladestandorte des k-Means-Algorithmus.



(b) Ladestandorte des Knotenzentralitätswerte-Algorithmus mit einem Mindestabstand von 100 Metern.



(c) Ladestandorte des genetischen p-Median-Algorithmus.



(d) Ladestandorte der optimalen Lösung.

Abbildung 4.9: Die mit den verschiedenen Algorithmen berechneten Ladestandorte für Nord-Seattle. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

Tabelle 4.2: Qualitative Evaluation der berechneten Ergebnisse für Nord-Seattle. Alle Angaben in Metern. Angegeben sind die Fitnesswerte sowie die jeweiligen Distanzen der berechneten Ergebnisse zu den optimalen Lösungen. Quelle: Eigene Darstellung.

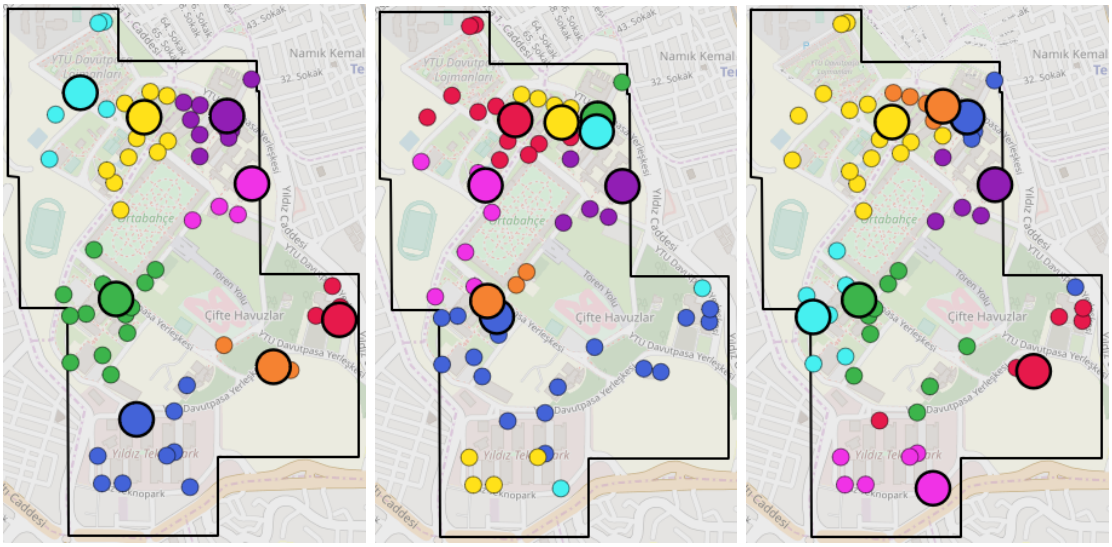
Parameter	**k-Means-Algorithmus**	**Knoten-zentralitäts-werte**	**p-Median-Algorithmus**	**Zufällige Platzierung**
Gesamtdistanz	11344	11236	10522	14641
Mittlere Distanz pro Ladestandort	1134	1124	1052	1464
Minimale Distanz	52	362	121	547
Maximale Distanz	6874	7205	6792	7205
Fitnesswert	310366	440464	321937	554012

Algorithmen verwendet wird, etwa 3 Stunden und zehn Minuten. Damit ergibt sich für den auf Knotenzentralitätswerten basierenden Algorithmus eine Gesamtlaufzeit von 3 Stunden und 17 Minuten und für den p-Median-Algorithmus sogar eine Gesamtlaufzeit von 5 Stunden und 35 Minuten (bzw. 2 Stunden und 25 Minuten ohne die Berechnung der Distanzmatrix). Aufgrund dieser hohen Laufzeit und der zum k-Means-Algorithmus nahezu gleichen Ergebnisqualität ist die Verwendung des genetischen p-Median-Algorithmus für dieses Untersuchungsgebiet kaum zu rechtfertigen. Eine Anpassung der Parameter, mit denen der genetische p-Median-Algorithmus ausgeführt wird oder eine Laufzeitoptimierung des implementierten Codes ist daher dringend angeraten (vgl. Kapitel 3.2.3). Eine erneute Berechnung der Ladestandorte mittels des p-Median-Algorithmus, bei der nur eine Populationsgröße von 50 statt einer der Anzahl POIs entsprechenden Populationsgröße von 552 verwendet wird, benötigt 13,5 Minuten ohne die Berechnung der Distanzmatrix. Zudem ergibt die Berechnung bereits ein qualitativ vergleichbares Ergebnis mit einem errechneten Fitnesswert von 341359 Metern und einem durchschnittlichen Abstand von 1377 Metern zur optimalen Lösung.

4.3 Davutpaşa-Campus

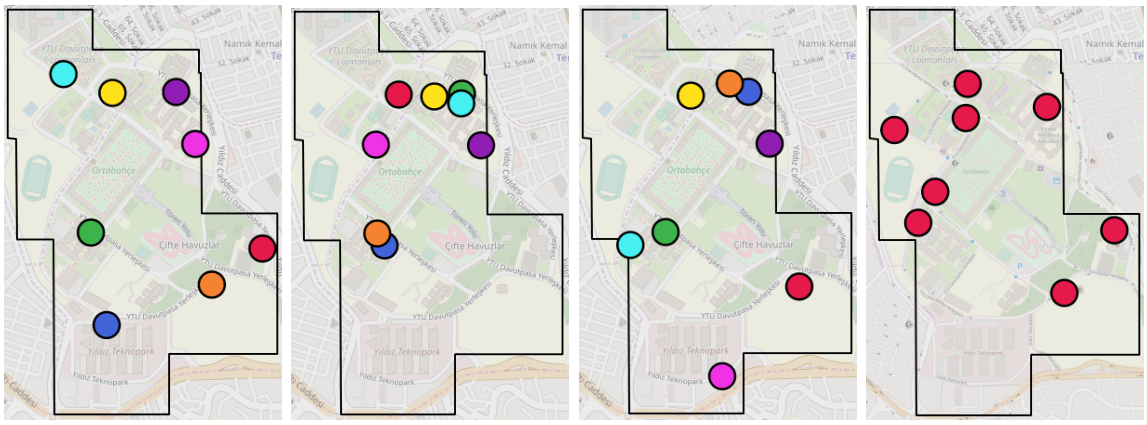
Die berechneten Cluster des Davutpaşa-Campus sind in Abbildung 4.10 (a) bis (c) zu sehen. Während die Ergebniscluster des k-Means-Algorithmus vergleichsweise deutlich voneinander abgegrenzt sind, sind die Cluster der beiden anderen, auf realen Wegstrecken basierenden Algorithmen, deutlich diverser. Nicht nur die Anzahl der POIs in den Clustern unterscheidet sich stärker, auch die räumliche Abdeckung der einzelnen Cluster weist größere Unterschiede auf. Insbesondere erscheinen bei diesen beiden Algorithmen einige POIs als Außenseiter, da sie scheinbar weit gestreuten Clustern zugeteilt werden. Dies könnte daran liegen, dass der direkte Zugang von dieser Stelle durch den Campus hindurch länger ist als der Weg mittels der Ringstraße außen herum. Der k-Means-Algorithmus ignoriert wie zu erwarten solche Gegebenheiten.

Die berechneten Platzierungen der Ladestandorte, die in Abbildung 4.11 (a) bis (c) dargestellt sind, zeigen auch in diesem kleinen Gebiet, dass der k-Means-Algorithmus und der genetische p-Median-Algorithmus eine bessere räumliche Abdeckung erreichen als der auf Knotenzentralitätswerten basierende Algorithmus.



(a) Ergebniscluster des k-Means-Algorithmus. **(b)** Ergebniscluster des Knotenzentralitätswerte-Algorithmus. **(c)** Ergebniscluster des genetischen p-Median-Algorithmus.

Abbildung 4.10: Die Ergebniscluster der verschiedenen Algorithmen für den Davutpaşa-Campus. Jedes Cluster ist in einer Farbe dargestellt. Die POIs mit der schwarzen Umrandung sind die ausgewählten Ladestandorte des Clusters. Die Ladestandorte von (b) haben einen Mindestabstand von 100 Metern. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.



(a) Ladestandorte des k-Means-Algorithmus. **(b)** Ladestandorte des Knotenzentralitäts-werte-Algorithmus. **(c)** Ladestandorte des genetischen p-Median-Algorithmus. **(d)** Ladestandorte der optimalen Lösung.

Abbildung 4.11: Die mit den verschiedenen Algorithmen berechneten Ladestandorte für den Davutpaşa-Campus. Die Ladestandorte von (b) haben einen Mindestabstand von 100 Metern. Der schwarze Rahmen bildet die Bounding Box des Untersuchungsgebietes. Quelle: Eigene Darstellung.

Die optimale Lösung sieht für den Davutpaşa-Campus in der südwestlichen Ecke der Bounding Box überhaupt keine Ladestandorte vor. Ebenso wie der auf

Knotenzentralitätswerte basierende als auch der genetische p-Median-Algorithmus erachtet die optimale Lösung den nördlicheren Teil als wichtiger.

Tabelle 4.3 zeigt die qualitativen Evaluationsergebnisse der Berechnungen zum Davutpaşa-Campus. Hier zeigt sich, dass der k-Median-Algorithmus die geringste Differenz zum optimalen Ergebnis, der p-Median Algorithmus jedoch mit Abstand den besten Fitnesswert aufweist. Da alle implementierten Algorithmen für dieses kleine Gebiet in einem erträglichen realen Zeitaufwand liegen, indem die 68x68-Distanzmatrix in etwa einer Minute berechnet werden kann, werden diese beiden Algorithmen für dieses kleine Gebiet empfohlen.

Tabelle 4.3: Qualitative Evaluation der berechneten Ergebnisse für den Davutpaşa-Campus. Alle Angaben in Metern. Angegeben sind die Fitnesswerte sowie die jeweiligen Distanzen der berechneten Ergebnisse zu den optimalen Lösungen. Quelle: Eigene Darstellung.

Parameter	**k-Means-Algorithmus**	**Knoten-zentralitäts-werte**	**p-Median-Algorithmus**	**Zufällige Platzierung**
Gesamtdistanz	2761	3116	5637	8694
Mittlere Distanz pro Ladestandort	345	390	705	1087
Minimale Distanz	0	0	0	0
Maximale Distanz	2838	2147	3892	4005
Fitnesswert	24604	27416	11741	36291

4.4 Zusammenfassender Vergleich

Ein Vergleich der Ergebnisse aus den drei verschiedenen Untersuchungsgebieten zeigt, dass die Eigenschaften der drei implementierten Algorithmen, trotz der unterschiedlichen Charakteristika der Untersuchungsgebiete im Hinblick auf die Größe und die Anzahl der zu berücksichtigten POIs, erhalten bleiben.

Generell sind Clusteralgorithmen gut geeignet, um eine räumliche Abdeckung in Gebieten herzustellen, in denen noch keine Ladestandorte zu finden sind (Sathaye and Kelley, 2013, S. 28). Vor allem die Ergebnisse, die mit dem k-Means-Algorithmus, aber auch diejenigen, die mit dem p-Median-Algorithmus berechnet werden, weisen immer eine zufriedenstellende räumliche Verteilung auf. Die Ergebnisse, die mit dem auf Knotenzentralitätswerten

basierenden Algorithmus berechnet werden, enthalten dagegen immer einige Ladestandorte, die sehr nahe beieinander platziert sind. Die Vorgabe eines Mindestabstandes kann hier Abhilfe schaffen. Die optimalen Lösungen für Nord-Seattle und den Davutpaşa-Campus, die ebenfalls keine räumliche Gleichverteilung aufweisen, zeigen jedoch, dass die mittels des auf Knotenzentralitätswerten basierenden Algorithmus berechneten Platzierungen ähnliche räumliche Tendenzen wie die optimalen Lösungen aufweisen. Wird jedoch wie im Untersuchungsgebiet Nord-Amsterdam eine nahezu gleichverteilte Lösung angestrebt, ist dieser Algorithmus nicht geeignet.

Aus der qualitativen Ergebnisevaluation mittels der Ungarischen Methode und den Fitnesswerten geht hervor, dass in der Regel entweder der k-Means-Algorithmus oder der p-Median-Algorithmus präferiert werden sollte, wobei letzterer insbesondere in Gebieten mit vielen POIs eine deutlich längere Laufzeit hat. Hier ist die Anpassung der Parameter oder eine verbesserte Implementierung des Codes anzuraten.

Tabelle 4.4 zeigt die realen Laufzeiten der Algorithmen im Vergleich, wobei immer der arithmetische Mittelwert von drei verschiedenen Ausführungen genommen wird. Die quadratische Laufzeit zur Berechnung der Distanzmatrizen, die beim auf Knotenzentralitätswerten basierenden Algorithmus und beim genetischen p-Median-Algorithmus benötigt wird, wirkt sich dabei am stärksten auf die reale Laufzeit aus. Der größte Anteil der Laufzeit des k-Means-Algorithmus wird für den Nachbereitungsschritt benötigt, der die Zentroide auf den jeweils nächstgelegenen POI eines Clusters überführt (siehe Kapitel 3.2.1). Ausführliche Tabellen aller Algorithmenlaufzeiten inklusive der Vor- und Nachbereitungsschritte sind im Anhang in den Tabellen A0.3 und A0.4 zu finden.

Tabelle 4.4: Die realen Laufzeiten der Algorithmen im Vergleich für die verschiedenen Untersuchungsgebiete. Alle Angaben sind in Stunden. Quelle: Eigene Darstellung.

Parameter	**Nord-Amsterdam**	**Nord-Seattle**	**Davutpaşa-Campus**
k-Means-Algorithmus	0:00:13.769	0:00:37.233	0:00:08.564
Knotenzentralitätswerte (ohne Distanzmatrix)	0:00:42.407	0:06:14.246	0:00:01.810
p-Median-Algorithmus (ohne Distanzmatrix)	0:21:05.515	2:25:26.367	0:00:32.284
Distanzmatrix	0:21:98.366	3:10:72.916	0:01:01.668

5 Fazit

In diesem Kapitel werden die in der Einleitung dieser Bachelorarbeit gestellten Leitfragen noch einmal in Kürze beantwortet und ein Ausblick auf weitere Aspekte des Forschungsbereichs zur Platzierung von Ladeinfrastruktur für Elektrofahrzeuge gegeben.

5.1 Beantwortung der Leitfragen

Die zusammenfassenden Antworten auf die Leitfragen resultieren aus dem Inhalt von Kapitel 2 bis 4 dieser Bachelorarbeit.

Welche frei verfügbaren Daten können zur Berechnung von optimalen Ladestandorten in verschiedenen Gebieten verwendet werden?
Viele Forschungsarbeiten zur optimalen Platzierung von Ladestandorten werden durch Daten, bzw. durch die eingeschränkte Verfügbarkeit von Daten, begrenzt. Auch für diese Untersuchung stehen nicht viele Daten als Grundlage zur Verfügung. Die Verwendung der Points of Interest (POIs) von OpenStreetMap ist jedoch in fast allen Untersuchungsgebieten, in denen derzeit Ladestandorte untersucht werden, aufgrund der ausreichenden Datenqualität der POIs möglich. Dabei dienen POIs sowohl als Proxy für den erwarteten Bedarf in einem Gebiet als auch als Platzierungsstandorte für Ladesäulen. Zwar können mittels der POIs keine endgültigen Ladestandorte, die alle lokalen Gegebenheiten berücksichtigen, bestimmt werden, aber es können räumliche Tendenzen zur optimalen Platzierung aufgezeigt werden.

Welche Algorithmen eignen sich zur Berechnung von optimalen Ladestandorten?
In städtischen Gebieten eignen sich Clusteralgorithmen zur Berechnung von optimalen Ladestandorten. Insbesondere wenn noch keine Ladeinfrastruktur vorhanden ist, hilft die Clusterbildung, eine erste räumliche Abdeckung mit Ladestandorten zu erreichen. Sowohl der in dieser Arbeit implementierte k-Means-Algorithmus als auch der genetische p-Median-Algorithmus sind gut dafür geeignet, Cluster bzw. optimale Ladestandorte zu berechnen, die eine räumlich gleichmäßige Abdeckung sicherstellen. In Gebieten mit bereits installierten Ladesäulen sind diese Algorithmen weniger gut geeignet, da es hier eher darauf ankommen wird, den noch nicht gedeckten Bedarf zu lokalisieren und die

vorhandene Ladestruktur an bestimmten Stellen auszubessern (Sathaye and Kelley, 2013, S. 29).

Wie können die berechneten optimalen Ladestandorte im Hinblick auf ihre Qualität evaluiert werden?
Zur Evaluation der Ergebnisse wird die Differenz zu einer optimalen Lösung berechnet. Außerdem werden die Algorithmen mittels des p-Median-Fitnesswertes der erhaltenen Lösungen miteinander verglichen. Weiterhin wird gezeigt, dass jede durch die Algorithmen berechnete Platzierung besser sein kann, als eine zufällige Platzierung der Ladestandorte. Generell ist auch ein Abgleich hinsichtlich der Wirtschaftlichkeit und Auslastung der bereits installierten Ladesäulen zur Ergebnisevaluation nützlich. Die Verfügbarkeit dieser Daten ist jedoch nicht immer gegeben (Wagner et al., 2013).

5.2 Ausblick

Die in dieser Arbeit hergestellten Ergebnisse liefern einen Beitrag zum wachsenden Forschungsbereich der Platzierung von Ladeinfrastruktur, indem drei verschiedene Algorithmen implementiert und insbesondere auch evaluiert werden.

Es existieren zwar bereits einige Algorithmen, die in der Regel deutlich komplizierter aufgebaut sind und häufig größere Datenmengen und weitere Aspekte der Ladeinfrastruktur berücksichtigen, jedoch werden die so erhaltenen Lösungen in der Regel nicht evaluiert oder deren Evaluation nicht publiziert. Eine Vergleichbarkeit der Qualität der Algorithmen der verschiedenen Forschungsarbeiten ist somit nicht gegeben.

Wird ein Implementierungsansatz dieser Arbeit weiter verfolgt, ist es sinnvoll, zusätzliche Aspekte bei der Berechnung der optimalen Ladestandorte mit einzubeziehen. Hierzu zählen beispielsweise die Berücksichtigung der bereits installierten Ladesäulen in einem Gebiet sowie die energetischen Auswirkungen der Ladestandorte auf das Stromnetz.

Die Tatsache, dass das Ladeverhalten schon heute zwischen verschiedenen Ländern und Gebieten variiert und die Vermutung, dass sich das heutige Ladeverhalten auch noch einmal ändern wird, sobald Elektrofahrzeuge eine größere Marktmacht erreicht haben, wird dazu führen, dass sich die Anforderungen an optimal platzierte Ladestandorte mit der Zeit wandeln werden (Brey et al., 2016, S. 145; Philipsen et al., 2016, S. 127). „Wohin

mit den Ladesäulen?“ ist damit keine Frage, die einmalig beantwortet werden kann, sondern eine Frage, die immer wieder neu und unter Berücksichtigung der aktuellen Gegebenheiten bearbeitet werden sollte, um eine optimale Platzierung von Ladestandorten für Elektrofahrzeuge zu ermöglichen.

Literaturverzeichnis

Andrenacci, N., Ragona, R., and Valenti, G. (2016). A demand-side approach to the optimal deployment of electric vehicle charging stations in metropolitan areas. *Applied Energy*, 182(128):39–46.

Beasley, J. and Chu, C. P. (1994). A Genetic Algorithm for the Set Covering Problem. *European Journal of Operational Research*, 94:392–404.

Behnel, S. (2019). The lxml.etree Tutorial. URL: https://lxml.de/tutorial.html. Letzter Abruf am 22.06.2019.

Bill, R. (2016). *Grundlagen der Geo-Informationssysteme*, volume 6. Wichmann, Berlin.

Boeing, G. (2017). OSMnx: New methods for acquiring, construction, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems.*, 65(65):126–139.

Brandes, U. (2001). A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177.

Brey, J. J., Brey, R., Carazo, A. F., Ruiz-Montero, M., and Tejada, M. (2016). Incorporating refuelling behaviour and drivers' preferences in the design of alternative fuels infrastructure in a city. *Transportation Research Part C: Emerging Technologies*, 65:144–155.

Brooker, R. P. and Qin, N. (2015). Identification of potential locations of electric vehicle supply equipment. *Journal of Power Sources*, 299:76–84.

Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., and Schaub, T. (2016). The GeoJSON Format. Internet Engineering Task Force (IETF) (Hrsg.). URL: https://tools.ietf.org/html/rfc7946. Letzter Abruf am 22.06.2019.

Chaudhry, S. S., He, S., and Chaudhry, P. E. (2003). Solving a class of facility location problems using genetic algorithms. *Expert Systems*, 20(2):86–91.

Chen, T. D., Kockelman, K. M., and Khan, M. (2013). Locating Electric Vehicle Charging Stations. Parking-Based Assignment Method for Seattle, Washington. *Transportation Research Record: Journal of Transportation Research Board.*, 1(2385):28–36.

Despalatovic, L., Vojkovic, T., and Vukicevic, D. (2014). Community structure in networks: Girvan-Newman algorithm improvement. pages 997–1002.

Devran, I. T. (2018). The-P-median-Problem-Python. GitHub Inc. (Hrsg.). URL: https://github.com/ilkayDevran/The_Pmedian_Problem_Python . Letzter Abruf am 26.06.2019.

Dolnicar, S. (2002). A Review of Unquestioned Standards in Using Cluster Analysis for Data-Driven Market Segmentation. *CD Conference Proceedings of the Australian and New Zealand Marketing Academy Conference (ANZMAC).*

Efthymiou, D., Chrysostomou, K., Morfoulaki, M., and Aifantopoulou, G. (2017). Electric vehicles charging infrastructure location: a genetic algorithm approach. *European Transport Research Review.*, 9(2).

Gnann, T., Plötz, P., Globisch, J., Schneider, U., Dütschke, E., Funke, S., Wietschel, M., Jochem, P., Heilig, M., Kagerbauer, M., and Reuter-Oppermann, M. (2017). *Profilregion Mobilitätssysteme Karlsruhe. Öffentliche Ladeinfrastruktur für Elektrofahrzeuge. Ergebnisse der Profilregion Mobilitätssysteme Karlsruhe.* Fraunhofer-Institut für System- und Innovationsforschung ISI, Karlsruhe.

Jain, A. K. (2009). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666.

Jaramillo, J. H., Bhadury, J., and Batta, R. (2002). On the Use of Genetic Algorithms to Solve Location Problems. *Computers and Operations Research - Location analysis*, 29(6):761–779.

Jin, X. and Han, J. (2010). K-Means Clustering. In *Encyclopedia of Machine Learning*, pages 695–697.

Kuhn, F. (2013). Optimierung. Vorlesung 9. Lineare Programmierung und Kombinatorische Optimierung. Vorlesungsfolien. URL: http://ac.informatik.uni-freiburg.de/teaching/ss_13/opti/Slides/pdf/Optimierung09.pdf. Letzter Abruf am 25.06.2019.

Mangla, M., Akhare, R., and Ambarkar, S. (2015). Implementing Genetic Algorithm to solve Facility Location Problem. *International Research Journal of Engineering and Technology (IRJET)*, 2(5):781–784.

Manning, C. D., Raghavan, P., and Schütze, H. (2009). *An Introduction to Information Retrieval.* Cambridge University Press (Draft)., Cambridge.

Marsden, R. (2009). Technical Overview: GeoJSON. Geoweb Guru (Hrsg.). URL: https://web.archive.org/web/20090221220320/http://www.geowebguru.com/articles/97-technical-overview-geojson . Letzter Abruf am 22.06.2019.

Mehar, S. and Senouci, S. M. (2013). An optimization location scheme for electric charging stations. International Conference on Smart Communications in Network Technologies (SaCoNeT).

Nationale Plattform Elektromobilität NPE (2018). *Fortschrittsbericht 2018 – Markthochlaufphase.* Gemeinsame Geschäftsstelle Elektromobilität der Bundesregierung (GGEMO) (Hrsg.), Berlin.

NetworkX Developers (Hrsg.) (2018). NetworkX Algorithms. Centrality. Betweenness Centrality. URL: https://networkx.github.io/documentation/networkx-2.2/reference/algorithms/generated/networkx.algorithms.centrality.betweenness_centrality.html. Letzter Abruf am 22.06.2019.

Nicholas, M. and Hall, D. (2018). LESSONS LEARNED ON EARLY ELECTRIC VEHICLE FAST-CHARGING DEPLOYMENTS. White Paper.

OpenStreetMap Foundation (Hrsg.) (2019). Routing. URL: https://wiki.openstreetmap.org/wiki/Routing. Letzter Abruf am 22.06.2019.

OpenStreetMap-Wikipedia (2019). Key:amenity. URL: https://wiki.openstreetmap.org/wiki/Key:amenity. Letzter Abruf am 22.06.2019.

Pagany, R., Camargo, R. L., and Dorner, W. (2018). A review of spatial localization

methodologies for the electric vehicle charging infrastructure. *International Journal of Sustainable Transportation*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. URL:https://scikit-learn.org/stable/modules/clustering.html . Letzter Abruf am 22.06.2019.

Philipsen, R., Schmidt, T., van Heek, J., and Ziefle, M. (2016). Fast-charging station here, please! User criteria for electric vehicle fast-charging locations. *Transportation Research Part F*, 40(40):119–129.

Rahman, I., M., V. P., Singh, B. M. S., Abdullah-Al-Wadud, M., and Adnan, N. (2016). Review of recent trends in optimization techniques for plug-in hybrid, and electric vehicle charging infrastructures. *Renewable and Sustainable Energy Reviews*, 58:1039–1047.

Samatova, N. (2013). Introduction to Graph Cluster Analysis. NC State University (Hrsg.). Vorlesungsfolien. URL: https://www.csc2.ncsu.edu/faculty/nfsamato/practical-graph-mining-with-R/slides/pdf/Graph_Cluster_Analysis.pdf. Letzter Abruf am 25.06.2019.

Sathaye, N. and Kelley, S. (2013). An approach for the optimal planning of electric vehicle infrastructure for highway corridors. *Transportation Research Part E*, 59:15–33.

Schild, K. and Mochol, M. (2007). Aufbau von XML-Dokumenten. Präsentation. Freie Universität Berlin. URL: http://www.ag-nbi.de/lehre/07/V_XML/Folien/02_XML-Dokumente.pdf. Letzter Abruf am 22.06.2019.

SciPy community (Hrsg.) (2019). SciPy Reference Guide. Release 1.3.0. URL: https://docs.scipy.org/doc/scipy/scipy-ref-1.3.0.pdf. Letzter Abruf am 25.06.2019.

Shareef, H., Islam, M. M., and Mohamed, A. (2016). A review of the stage-of-the-art charging technologies, placement methodologies, and impacts of electric vehicles. *Renewable and Sustainable Energy Reviews*, 64:403–420.

Shukla, A., Verma, K., and Kumar, R. (2016). Consumer Perspective based Placement of Electric Vehicle Charging Stations by Clustering Techniques. 2016 National Power Systems Conference (NPSC).

Spiegel ONLINE (Hrsg.) (2018). Regierungsberater kippen Elektroauto-Ziel. URL: https://www.spiegel.de/auto/aktuell/elektroautos-regierungsberater-kippen-millionen-ziel-a-1228897.html. Letzter Abruf am 17.06.2019.

Srinivasan, G. (2009). Lec-16 Assignment Problem - Hungarian Algorithm. YouTube Video. URL: https://www.youtube.com/watch?v=BUGIhEecipE. Letzter Abruf am 25.06.2019.

Stadt Kaiserslautern (Hrsg.) (2019). Enstadt:Pfaff. Leuchtturmprojekt. Projektinhalte. URL: https://pfaff-reallabor.de/projekt-pfaff-reallabor/projektinhalte-2/. Letzter Abruf am 17.06.2019.

Statista GmbH (Hrsg.) (2018). Anzahl der Elektroautos in Deutschland von

2006 bis 2019. URL: https://de.statista.com/statistik/daten/studie/265995/umfrage/anzahl-der-elektroautos-in-deutschland/. Letzter Abruf am 17.06.2019.

Storandt, S. and Funke, S. (2013). Enabling E-mobility: Facility Location for Battery Loading Stations. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pages 1341–1347. AAAI Press.

Studimup.de (Hrsg.) (2019). Anzahl der Möglichkeiten/Ereignisse berechnen (Kombinatorik). URL: https://www.studimup.de/abitur/stochastik/anzahl-der-m%C3%B6glichketen-berechnen-kombinatorik/ Letzter Abruf am 25.06.2019.

Swamynathan, M. (2017). *Step 3 – Fundamentals of Machine Learning*, pages 117–208. Apress, Berkeley, CA.

Varyani, Y. (2013). Hungarian Algorithm for Assignment Problem | Set 1 (Introduction). GeeksforGeeks (Hrsg.). URL: https://www.geeksforgeeks.org/hungarian-algorithm-assignment-problem-set-1-introduction/. Letzter Abruf am 25.06.2019.

von Rueden, J. (2017). Charging Ahead - Predicting Optimal Charging Station Locations across Multiple Cities. Master Thesis. Master of Science, Business Information Management.

Wagner, D. and Willhalm, T. (2007). Speed-Up Techniques for Shortest-Path Computations. In *STACS 2007*, pages 23–36, Berlin, Heidelberg. Springer Berlin Heidelberg.

Wagner, S., Brandt, T., and Neumann, D. (2014). SMART CITY PLANNING - DEVELOPING AN URBAN CHARGING INFRASTRUCTURE FOR ELECTRIC VEHICLES. *Proceedings of the European Conference on Information Systems (ECIS)*.

Wagner, S., Götzinger, M., and Neumann, D. (2013). Optimal Location of Charging Stations in Smart Cities: A Point of Interest Based Approach. Completed Research Paper. *Thirty Fourth International Conference on Information Systems.*

Wikimedia Foundation Inc. (Hrsg.) (2019). Vollständiger Graph. URL: https://de.wikipedia.org/wiki/Vollst%C3%A4ndiger_Graph. Letzter Abruf am 22.06.2019.

Xi, R., Sioshansi, R., and Marano, V. (2013). Simulation-optimization model for location of a public electric vehicle charging infrastructure. *Transportation Research Part D*, 22:60–69.

Yagcitekin, B., Uzunoglu, M., and Karakas, A. (2016). A new deployment method for electric vehicle charging infrastructure. *Turkish Journal of Electrical Engineering and Computer Sciences*, 24:1292–1305.

Zeit ONLINE (Hrsg.) (2018). Regierung verfehlt Ziel für Elektroautos. URL: https://www.zeit.de/mobilitaet/2018-09/nationale-plattform-elektromobilitaet-bundesregierung-ziele-e-autos-2022. Letzter Abruf am 17.06.2019.

Anhang

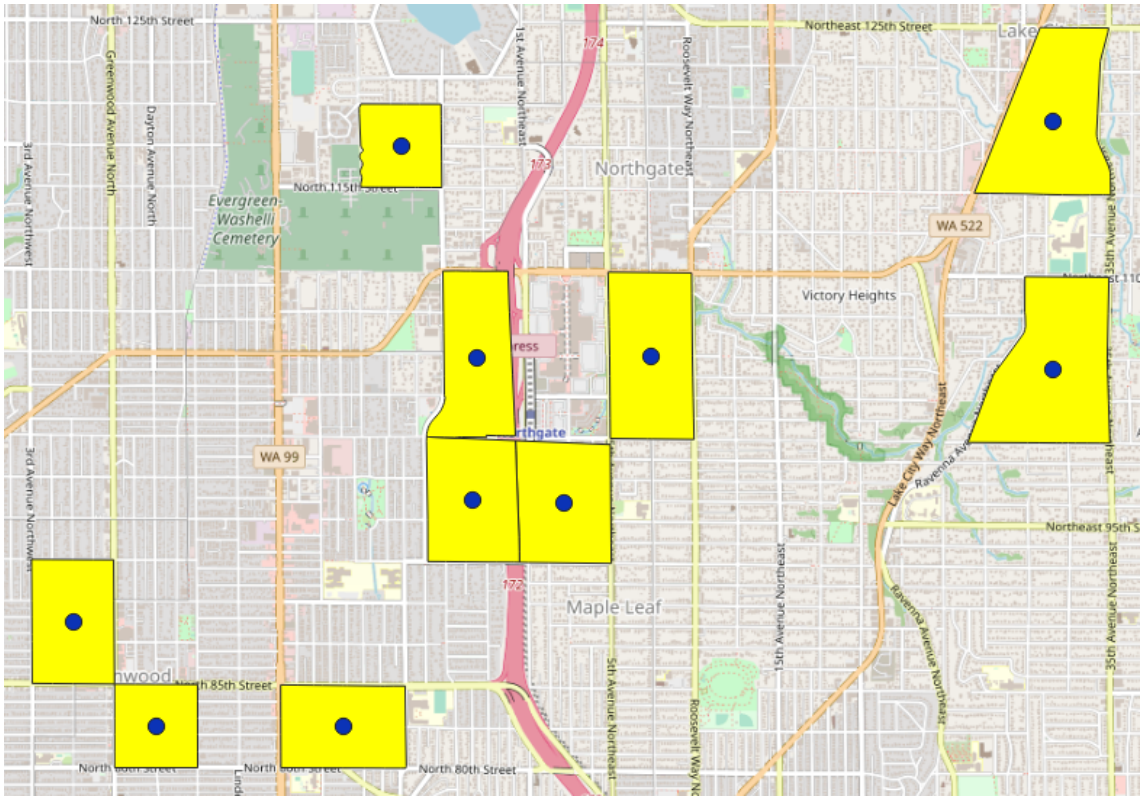


Abbildung A0.1: Umwandlung der von Chen et al. berechneten optimalen Verkehrsanalysezonen (gelb) in Punktladestandorte (blau) mittels des Software QGIS. Die blauen Punkte liegen in der Mitte der jeweiligen Polygone und werden als optimale Standorte für Nord-Seattle verwendet. Quelle: Chen et al., 2013, S. 35.

Tabelle A0.1: Die POI-Kategorien (keywords) aus OpenStreetMap, die als relevant in Bezug zu Ladestandorten eingestuft werden. Quelle: Eigene Darstellung.

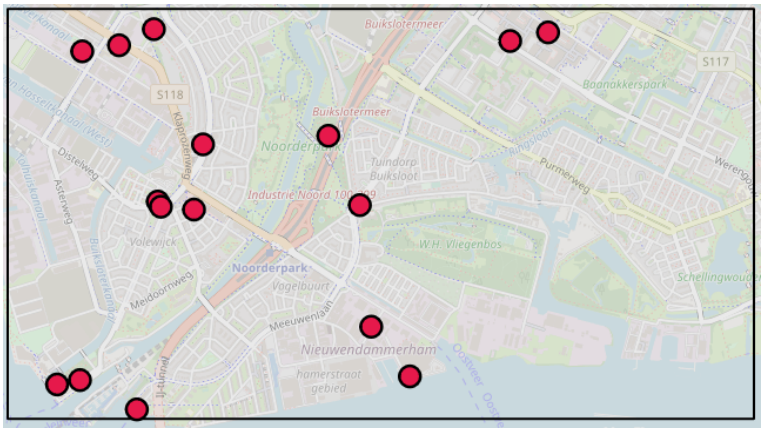
nr	keyword
0	bar
1	bbq
2	biergarten
3	cafe
4	fast_food
5	food_court
6	ice_cream
7	pub
8	restaurant
9	college
10	kindergarten
11	library
12	school
13	music_school
14	driving_school
15	language_school
16	university
17	research_institute
18	bicycle_parking
19	bicycle_repair_station
20	bicycle_rental
21	boat_rental
22	boat_sharing
23	bus_station
24	car_rental
25	car_sharing
26	car_wash
27	vehicle_inspection
28	ferry_terminal
29	motorcycle_parking
30	parking

nr	keyword
31	parking_space
32	taxi
33	bank
34	bureau_de_change
35	clinic
36	dentist
37	doctors
38	hospital
39	nursing_home
40	pharmacy
41	social_facility
42	veterinary
43	blood_donation
44	arts_centre
45	brothel
46	casino
47	cinema
48	community_centre
49	gambling
50	nightclub
51	planetarium
52	social_centre
53	stripclub
54	studio
55	swingerclub
56	theatre
57	animal_shelter
58	courthouse
59	coworking_space
60	crematorium
61	dive_centre

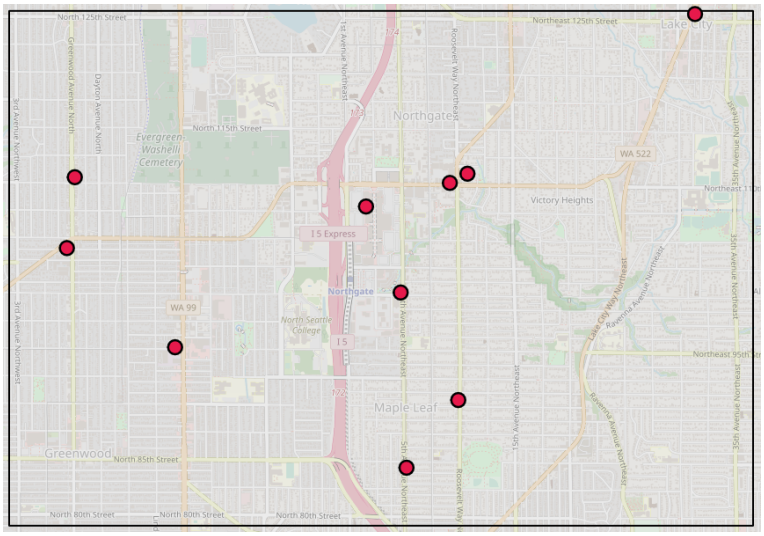
nr	keyword
62	embassy
63	fire_station
64	grave_yard
65	gym
66	hunting_stand
67	internet_cafe
68	kitchen
69	kneipp_water_cure
70	marketplace
71	place_of_worship
72	police
73	post_depot
74	post_office
75	prison
76	public_bath
77	public_building
78	ranger_station
79	rescue_station
80	rv_storage
81	sauna
82	townhall
83	waste_transfer_station
84	fitness_centre
85	swimming_pool
86	public

Tabelle A0.2: Die POI-IDs von OpenStreetMap, die nicht berücksichtigt werden, da sie im Straßennetz des Routing-Tools OSMnx aufgrund der begrenzenden Bounding Box nicht erreichbar sind. Quelle: Eigene Darstellung.

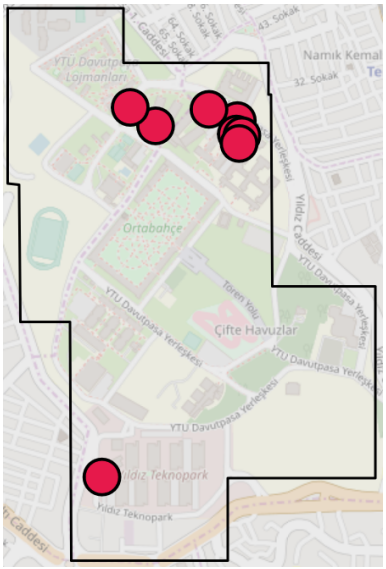
Untersuchunsgebiet	**Anzahl POIs**	**POI-IDs**
Nord-Amsterdam	3	- 2848301069, 1939968813, 2848311842
Davutpaşa-Campus	0	-
Nord-Seattle	27	356541632, 1040913036, 1564279475, 1769040321, 2420103493, 2433072446, 4766222629, 4840538767, 408571620, 462648928, 685395221, 685395346, 704929609, 944725956, 1133075436, 1133080406, 1315697757, 2293066764, 2303317876, 2303326921, 2433053922, 3452618758, 3534545354, 4709201270, 4972855524, 4972951904, 4972951908



(a) Ladestandorte des Zufalls-Algorithmus für Nord-Amsterdam.



(b) Ladestandorte des Zufalls-Algorithmus für Nord-Seattle.



(c) Ladestandorte des Zufalls-Algorithmus für den Davutpaşa-Campus.

Abbildung A0.2: Berechnete Ladestandorte des Zufalls-Algorithmus. Der schwarze Rahmen bildet die Bounding Box der Gebiete. Quelle: Eigene Darstellung.

Tabelle A0.3: Reale Laufzeiten der Algorithmen im Vergleich. Genommen wird jeweils der Durchschnitt von drei verschiedenen Durchgängen, die jeweils mit einem gewöhnlichen Laptop berechnet wurden. Alle Angaben in Stunden.

Berechnung	**Durchgang**	**Nord-Amsterdam**	**Nord-Seattle**	**Davutpaşa-Campus**
Zufalls-Algorithmus	1	0:00:00.211	0:00:00.399	0:00:00.289
	2	0:00:00.218	0:00:00.338	0:00:00.183
	3	0:00:00.227	0:00:00.274	0:00:00.198
	Durchschnitt	0:00:00.219	0:00:00.337	0:00:00.223
k-Means-Algorithmus	1	0:00:17.371	0:00:28.847	0:00:14.462
	2	0:00:11.780	0:00:51.033	0:00:06.725
	3	0:00:12.157	0:00:31.818	0:00:04.504
	Durchschnitt	0:00:13.769	0:00:37.233	0:00:08.564
k-Means-Algorithmus (rein)	1	0:00:00.345	0:00:00.260	0:00:00.232
	2	0:00:00.322	0:00:00.306	0:00:00.133
	3	0:00:00.492	0:00:00.326	0:00:00.141
	Durchschnitt	0:00:00.386	0:00:00.297	0:00:00.169
Knotenzentralitäts-werte-Algorithmus (mit DM-Berechnung)	1	0:22:53.249	3:09:04.955	0:01:06.542
	2	0:23:56.575	3:19:06.914	0:01:12.062
	3	0:22:29.285	3:24:25.846	0:00:59.814
	Durchschnitt	0:22:79.703	3:17:45.905	0:00:92.806
Knotenzentralitäts-werte-Algorithmus (ohne DM-Berechnung)	1	0:00:38.970	0:06:08.846	0:00:01.787
	2	0:00:40.887	0:06:21.489	0:00:01.826
	3	0:00:47.363	0:06:12.404	0:00:01.818
	Durchschnitt	0:00:42.407	0:06:14.246	0:00:01.810
p-Median-Algorithmus (ohne DM-Berechnung)	1	0:22:53.776	2:28:57.285	0:00:33.840
	2	0:20:30.384	2:22:57.483	0:00:30.862
	3	0:20:32.384	2:24:64.334	0:00:32.149
	Durchschnitt	0:21:05.515	2:25:26.367	0:00:32.284

Tabelle A0.4: Reale Laufzeiten der Algorithmen im Vergleich. Genommen wird jeweils der Durchschnitt von drei verschiedenen Durchgängen, die jeweils mit einem gewöhnlichen Laptop berechnet wurden. Alle Angaben in Stunden.

Berechnung	**Durchgang**	**Nord-Amsterdam**	**Nord-Seattle**	**Davutpaşa-Campus**
OSM-XML-Datei parsen	1	0:00:04.509	0:00:06.933	0:00:00.156
	2	0:00:04.133	0:00:04.919	0:00:00.156
	3	0:00:03.104	0:00:05.241	0:00:00.161
	Durchschnitt	0:00:03.915	0:00:05.698	0:00:00.158
CSV-Datei einlesen	1	0:00:01.600	0:00:01.434	0:00:04.661
	2	0:00:01.293	0:00:04.460	0:00:02.979
	3	0:00:01.302	0:00:01.395	0:00:01.310
	Durchschnitt	0:00:01.398	0:00:02.430	0:00:02.983
Distanzmatrix berechnen	1	0:21:56.188	3:02:44.139	0:01:01.801
	2	0:23:02.573	3:12:38.162	0:01:07.310
	3	0:21:36.338	3:17:36.447	0:00:55.895
	Durchschnitt	0:21:98.366	3:10:72.916	0:01:01.668
GeoJSON-Datei abspeichern	1	0:00:00.031	0:00:00.076	0:00:00.016
	2	0:00:00.040	0:00:00.083	0:00:00.016
	3	0:00:00.025	0:00:00.083	0:00:00.023
	Durchschnitt	0:00:00.032	0:00:00.081	0:00:00.018
Webseite aktualisieren	1	0:00:00.016	0:00:00.031	0:00:00.015
	2	0:00:00.015	0:00:00.054	0:00:00.000
	3	0:00:00.031	0:00:00.052	0:00:00.015
	Durchschnitt	0:00:00.021	0:00:00.046	0:00:00.010
Ungarische Methode anwenden	1	0:00:13.568	0:00:16.515	0:00:07.984
	2	0:00:18.439	0:00:12.243	0:00:07.142
	3	0:00:13.968	0:00:11.373	0:00:06.353
	Durchschnitt	0:00:15.325	0:00:13.377	0:00:07.160
Fitnesswert der Lösung berechnen	1	0:00:00.046	0:00:00.140	0:00:00.031
	2	0:00:00.048	0:00:00.134	0:00:00.015
	3	0:00:00.078	0:00:00.121	0:00:00.029
	Durchschnitt	0:00:00.057	0:00:00.132	0:00:00.025