Eva Röper

Planung von Ladevorgängen für Elektrofahrzeuge durch multikriterielle Optimierung unter nutzerorientierten Aspekten

INF FAKULTÄT FÜR INFORMATIK

Intelligente Kooperierende Systeme
Intelligente Systeme

Planung von Ladevorgängen für Elektrofahrzeuge durch multikriterielle Optimierung unter nutzerorientierten Aspekten

Bachelorarbeit

Eva Röper

28. Februar 2020

Betreuende Professorin: Prof. Dr.-Ing. habil. Sanaz Mostaghim

Betreuer: Dr.-Ing. Nikita Shchekutin

Eva Röper: *Planung von Ladevorgängen für Elektrofahrzeuge durch multikriterielle Optimierung unter nutzerorientierten Aspekten*
Otto-von-Guericke-Universität
Intelligente Kooperierende Systeme
Intelligente Systeme
Magdeburg, 2020.

Danksagungen

Zunächst bedanke ich mich bei der ITK Engineering GmbH für die Möglichkeit, mein Praktikum im Unternehmen zu absolvieren und in Kooperation meine Bachelorarbeit zu schreiben. Dadurch habe ich wertvolle Einblicke erhalten. Besonderer Dank gilt meinem dortigen Betreuer Nikita Shchekutin, der ein außerordentliches persönliches Interesse am Thema und dem Erfolg meiner Arbeit zeigte. Weiterhin fand er immer Zeit zum Korrekturlesen, gab hilfreiche Kritik und unterstützte mich bei organisatorischen und administrativen Aufgaben.
Außerdem danke ich Prof. Mostaghim für ihre Anregungen zum Thema und für die Möglichkeit, meine Bachelorarbeit an ihrem Lehrstuhl zu schreiben.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AC	Alternating current
ALNS	Adaptive Large Neighborhood Search
DC	Direct current
EVRP	Electric Vehicle Routing Problem
GVNS	General Variable Neighborhood Search
HC	Heuristic concentration
ILS	Iterated Local Search
LNS	Large Neighborhood Search
MDP	Markov decision process
MOEA	Multi-objective Evolutionary Algorithm
NSGA-II	Non-dominated sorting genetic algorithm II
SA	Simulated Annealing
SEI	Solid Electrolyte Interphase
SoC	State of Charge
SoH	State of Health
TS	Tabu Search
V2G	Vehicle-to-Grid
VNS	Variable Neighborhood Search
VNSB	Variable Neighborhood Search Branching

1 Einleitung

In den letzten Jahren haben das Problem der Klimaerwärmung und damit einhergehende Ziele zur Schonung der Umwelt immer mehr an Bedeutung gewonnen. In der Autoindustrie ist man deshalb auf der Suche nach alternativen Antriebsmöglichkeiten für Fahrzeuge, um die Emission von Treibhausgasen und Schadstoffen, vor allen Dingen von Kohlenstoffdioxid, bei der Nutzung herkömmlicher Autos zu umgehen. In Folge dessen steigt auch das Interesse an der Elektromobilität sowie die Anzahl an Elektrofahrzeugen und Ladestationen für diese [14]. Am 1. Januar 2019 betrug die Anzahl zugelassener Elektrofahrzeuge in Deutschland 83.175 und damit circa 29.000 mehr als im Vorjahr [37].
Durch die begrenzte Kapazität der Batterien der Elektrofahrzeuge ist jedoch ein Laden ähnlich wie das Tanken bei herkömmlichen Autos nötig. Allerdings ist das Laden deutlich komplexer als das Tanken. Bei Letzterem steht ein dichtes Netz an Tankstellen, die fast immer alle Arten von Kraftstoff anbieten, zur Verfügung und die Preise weichen in einem Gebiet nur gering voneinander ab. Beim Laden besteht das Problem, dass sich durch die aktuell noch selteneren Ladestationen eventuell größere Umwege von der geplanten beziehungsweise gewohnten Route ergeben. Dazu kommt, dass das Aufladen einer Batterie länger dauert als das Auffüllen eines Tanks, wodurch zusätzlich lange Wartezeiten an den Ladestationen entstehen können [48]. Außerdem sind unterschiedliche Steckertypen, die präferierte Ladetechnologie von Normalladen und Schnellladen und die gewünschte Abrechnungsart nicht an allen Ladestationen vorhanden. Je nach Ladestation und Lademenge können sich bei unterschiedlichen Kombinationen dieser Parameter merkliche Preisunterschiede ergeben. Des Weiteren ist die Lebensdauer einer Batterie begrenzt und wird durch das Fahr- und Ladeverhalten sowie die minimale und maximale Akkuladung beeinflusst [46], wodurch zusätzlich der Wunsch nach einer batterieschonenden Nutzung des Elektrofahrzeugs aufkommen kann.
Für den Fahrer eines Elektrofahrzeugs wird das Finden einer Ladestation nach seinen Vorstellungen dadurch zu komplex, um es selbst optimal zu machen.

Deshalb wird das Lösen des Problems durch einen Rechner wünschenswert. Dazu müssen die Kriterien für eine geeignete Ladestation als Zielfunktionen beschrieben.
In dieser Arbeit wird vorgeschlagen, das Problem multikriteriell mithilfe eines evolutionären Algorithmus zu lösen. Nach Beendigung des Programms stehen dem Nutzer meist mehrere Alternativrouten zu seiner jetzigen geplanten Route zur Auswahl. Die Entscheidung, welche der vorgeschlagenen Varianten schlussendlich genutzt wird, liegt also beim Nutzer. Dadurch kann er sich anhand seiner eigenen Priorisierung der Ziele eine Ladestation aussuchen, was vor allen Dingen hilfreich ist, wenn sich diese Wichtung öfters ändert. So kann es beispielsweise vorkommen, dass ein Fahrer an einem Tag möglichst schnell nach Hause und am nächsten möglichst wenig Geld ausgeben will.
Das System soll außerdem statt der einmaligen Wahl einer Ladestation bei der Planung der gesamten Route eine spontane Wahl einer Ladestation während der Fahrt ermöglichen, wofür dem Fahrer die besten, kontinuierlich aktualisierten Alternativrouten angezeigt werden. Dadurch soll eine erzwungene Umstellung auf ein neues Fahrverhalten vermieden werden, da die meisten Navigationssysteme für herkömmliche Autos Tankstellen nicht bei der Routenplanung einbeziehen und die Fahrer somit eher spontan tanken. Weiterhin kann das System dadurch auch unabhängig vom Navigationssystem genutzt werden. Viele nehmen jeden Tag dieselbe Strecke, beispielsweise zur Arbeit, ohne eine Routenplanung zu nutzen. Diese Personen könnten ebenfalls bei der Wahl einer Ladestation im Alltag unterstützt werden.
Zusammenfassen lässt sich der Beitrag dieser Arbeit somit als die Entwicklung eines Systems für multikriteriell optimierte Ladeempfehlungen inklusive Alternativrouten, das den Nutzer von Elektrofahrzeugen bei der Entscheidungsfindung unterstützt.
Der Rest der Arbeit ist wie folgt strukturiert: An dieses Kapitel schließt eine Betrachtung von multikriterieller Optimierung, den Hintergründe zur Batterie und einem verwandten Problem aus der Literatur an. In Kapitel 3 ist die detaillierte Problembeschreibung zu finden. Im Modellaufbau in Kapitel 4 werden die Zielfunktionen formuliert und die genutzten Systemparameter erklärt. Die verwendete Lösungsmethode wird in Kapitel 5 beschrieben. In Kapitel 6 werden die Generierung von Anwendungsfällen, einige Tests zur Qualität des Algorithmus und deren Ergebnisse präsentiert. Kapitel 7 fasst die Arbeit zusammen, gibt einen Ausblick und schließt die Arbeit ab.

2 Stand der Technik

In diesem Kapitel wird eine Übersicht zu multikriterieller Optimierung und zu Hintergründen zum State of Health (SoH) einer Batterie gegeben. Außerdem wird Literatur zum Electric Vehicle Routing Problem (EVRP) besprochen, welches mit meinem Problem verwandt ist.

2.1 Multikriterielle Optimierung

Eine Besonderheit des Problems dieser Arbeit ist, dass es multikriteriell optimiert wird. Das bedeutet, dass für ein Problem mit mehreren Zielfunktionen eine oder mehrere optimale Lösungen gefunden werden sollen. Diese Zielfunktionen stehen im Konflikt zueinander, sodass die Lösungen ein Kompromiss zwischen den verschiedenen Zielen darstellen. Der Wert einer Zielfunktion kann nicht verbessert werden, ohne dass ein anderer darunter leidet. Somit ist keine der Lösungen besser als eine andere. Die Menge solcher Lösungen wird Pareto-Front genannt. [18]
Im Allgemeinen kann man die Lösungsmethoden für multikriterielle Probleme in a priori und a posteriori Verfahren unterteilen [9]. Erstere transformieren das multikriterielle Problem in eins mit einer einzigen Zielfunktion. Ein Beispiel für diesen Ansatz ist die Weighted Sum Method, bei der die zu optimierende Zielfunktion aus der Summe der Zielfunktionen, multipliziert mit ihrem zugehörigen Gewicht, erstellt wird. Ein anderer Ansatz ist die ε-Constraint Method, wo nur eine der Zielfunktionen optimiert und der Rest als Nebenbedingungen behandelt wird. In den Nebenbedingungen werden für die betroffenen Zielfunktionen obere Schwellenwerte bei einer Minimierung und untere Schwellenwerte für eine Maximierung definiert. [18]
Bei a posteriori Methoden hingegen wird unter Beachtung aller Zielfunktionen eine Pareto-Front gefunden, aus der der Nutzer eine Lösung auswählt [18]. Da häufig das Problem besteht, dass Menschen ihre Präferenzen im Voraus nicht genau kennen oder formulieren können, soll in dieser Arbeit eine a posteriori

Methode zur Lösung verwendet werden.
Eine Kategorie sind interaktive Methoden, bei denen der Nutzer während des Optimierungsprozesses ab und zu Informationen zur Richtung der Suche, zur Priorisierung der Ziele oder zu anderen Faktoren bereitstellt [18]. Da der Algorithmus dieser Arbeit während der Fahrt laufen soll, können solche Methoden aus Sicherheitsgründen nicht genutzt werden.
Außerdem gibt es exakte Lösungsmethoden, die die optimale Pareto-Front finden. Ein Vertreter ist das Branch-and-Bound-Verfahren, welches aus dem Branching, dem wiederholten Aufteilen eines Problems in Teilprobleme, und dem Bounding, dem Entfernen von Teilproblemen durch die Berechnung oberer und unterer Schranken, besteht [23]. Der Nachteil an exakten Verfahren ist, dass sie zeitaufwendig sind [55]. Da mein Algorithmus in Zukunft fast in Echtzeit laufen soll, sind diese Methoden zur Lösung ungeeignet.
Deshalb wurde für diese Arbeit ein metaheuristischer Ansatz gewählt, da sie schnellere Berechnungen erlauben. Dies sind Lösungskonzepte, die näherungsweise Lösungen finden. Dadurch ist nicht garantiert, dass immer das globale Optimum gefunden wird. Der allgemeine Ablauf eines solchen Algorithmus ist, dass eine oder mehrere Startlösungen erstellt und iterativ verändert werden, bis ein Abbruchkriterium erfüllt ist. Dafür werden verschiedene Konzepte zur Exploration und Exploitation kombiniert. [22] Bei der Exploration geht es um die weitreichende Erkundung des Suchraums und bei der Exploitation um das Finden lokaler Verbesserungen [41]. Der entscheidende Vorteil von Metaheuristiken ist, dass sie, statt eine Lösung immer weiter zu verbessern, mit einer Population an Lösungskandidaten arbeiten. Dadurch ist das Ergebnis nicht eine einzige Lösung wie bei a priori Methoden sondern eine Menge von Lösungen, eine Pareto-Front. [18] Einige der am häufigsten vorkommenden Metaheuristiken werden in den folgenden Abschnitten grundlegend erklärt. Diese Algorithmen wurden ursprünglich für Probleme mit nur einer Zielfunktion entworfen und erst im Nachhinein für multikriterielle Probleme adaptiert.
Einer davon ist Tabu Search (TS), wo iterativ von der aktuellen Lösung zu der besten in einer Liste von Nachbarlösungen übergegangen wird, bis ein Abbruchkriterium erfüllt ist. Dabei werden bestimmte Aktionen von einer Tabu-Liste nicht durchgeführt, um zu vermeiden, in einem lokalen Optimum stecken zu bleiben. Bei der multikriteriellen Variante wird mit einer Menge von Lösungen gearbeitet, von denen jede ihre eigene Tabu-Liste hat. Außerdem werden für die möglichen Aktionen TS-Heuristiken eingesetzt. [33]
Beim Simulated Annealing (SA) wird eine Startlösung zufällig ein wenig verän-

dert. Wird die Lösung dadurch verbessert, wird sie angenommen. Verschlechtert sie sich, wird dies nur mit einer bestimmten Wahrscheinlichkeit akzeptiert. Diese setzt sich aus der Veränderung des Funktionswerts und einer sogenannten Temperatur, die über die Zeit abnimmt, zusammen. Dadurch werden schlechtere Lösungen mit einer geringeren Wahrscheinlichkeit angenommen, je größer die Verschlechterung ist und je länger der Algorithmus schon läuft. Wird ein Abbruchkriterium erreicht, endet der Algorithmus. Bei der multikriteriellen Variante bekommt jede Zielfunktion eine Temperatur und damit auch eine Wahrscheinlichkeit, dass die Veränderung der jeweiligen Zielfunktion akzeptiert wird. Daraus setzt sich dann die Wahrscheinlichkeit zusammen, dass eine schlechtere Lösung angenommen wird. Außerdem wird mit einer Menge von Lösungskandidaten statt nur mit einem gearbeitet. [58]
Evolutionäre Algorithmen sind von der Evolution in der Biologie inspiriert. Sie arbeiten mit einer Menge von Individuen, einer Population, und verwenden Rekombination und Mutation zu Veränderung dieser. Dafür werden für einen Lösungskandidaten entscheidende Informationen in einem Chromosom kodiert und ihm eine Fitness, also eine Güte, zugeordnet. Die Veränderungen finden iterativ von einer initialen Population ausgehend über Generationen hinweg statt, bis ein im Voraus definiertes Abbruchkriterium erreicht wurde. Überleben und fortpflanzen dürfen sich wie in der Natur nur ausgewählte, gute Individuen. Für Multi-objective Evolutionary Algorithm (MOEA) werden in Kapitel 5 dafür spezielle Selektionsmechanismen definiert, die den Vergleich von Lösungskandidaten mit Fitnessvektoren ermöglichen. [18] Von diesem grundlegenden Algorithmus gibt es inzwischen viele Abwandlungen [9].
Ein mit evolutionären Algorithmen verwandtes Verfahren ist die Particle Swarm Optimization. Dort wird mit einer Menge von Lösungskandidaten, die durch den Suchraum bewegt werden, gearbeitet. Die Bewegung ist abhängig von der besten bekannten Position eines Individuums und der global besten Position. Für die multikriterielle Optimierung werden für die Lösungskandidaten ihre nicht-dominierten Positionen gespeichert. Das sind diejenigen Stellen, für die es keine Position gibt, die für alle Zielfunktion wenigstens gleich gut und für mindestens eine echt besser war. Außerdem gibt es ein globales Archiv der nicht-dominierten Positionen, aus denen sich jedes Individuum einen Referenzpunkt als global beste Position aussucht. [16]
Eine weitere mit evolutionären Algorithmen verwandte Methode ist die Ant Colony Optimization, die auf Probleme angewandt werden kann, die sich als Suche nach dem kürzesten Pfad in einem Graphen modellieren lassen. In dem

Graphen wird eine Futtersuche von Ameisen simuliert, die beim Laufen Pheromone hinterlassen. Auf dem kürzesten Weg befindet sich nach einiger Zeit die größte Pheromonmenge, weil die Ameisen auf dem Weg schneller sind, somit in kürzerer Zeit mehr Pheromone verbreitet werden und nach und nach immer mehr Ameisen diesen Weg nehmen. Für multikriterielle Probleme gibt es für jede Zielfunktion eine eigene Ameisenkolonie und eine weitere multikriterielle Kolonie, die die Optimierung aller Zielfunktionen erreichen soll. [2]
Für keines der metaheuristischen Verfahren kann gesagt werden, dass es eindeutig besser ist als die anderen. Deswegen habe ich zur Lösung des Problems in dieser Arbeit das populärste ausgesucht, welches MOEA sind. [36]
Eine Variante eines MOEA ist der Non-dominated sorting genetic algorithm II (NSGA-II) mit kontrolliertem Elitismus. Dieser funktioniert wie ein oben beschriebener MOEA, aber verwendet zur Auswahl der Individuen, die zur nächsten Generation überleben non-dominated sorting. Dabei werden iterativ die Individuen gefunden, für die es kein Individuum gibt, das für alle Zielfunktionen mindestens gleich gut und für eine echt besser ist. Diese werden in einer Front zusammengefasst und aus der Population entfernt. Die Schritte werden solange wiederholt, bis alle Fronten bekannt sind. Bei der Variante von NSGA-II mit kontrolliertem Elitismus werden von allen Fronten Individuen übernommen. Aus der ersten, der besten, Front stammen dabei die meisten Individuen und aus jeder Front danach exponentiell weniger. Eine genauere Beschreibung des gesamten Algorithmus findet sich in Kapitel 5. Das erwähnte Verfahren weist, im Gegensatz zum elitären NSGA-II, wo immer die besten Individuen übernommen werden, sowohl eine gute Diversität als auch eine gute Konvergenz der ausgegebenen Pareto-Front auf. Die Konvergenz gibt die Nähe der ausgegebenen Pareto-Front zur echten Pareto-Front an. Die Diversität beschreibt die Verteilung der Individuen entlang der ausgegebenen Pareto-Front. Dabei ist eine gleichmäßige Verteilung über die gesamte Breite wünschenswert. Bei dem elitären NSGA-II geht die laterale Diversität über die Fronten hinweg durch die Bevorzugung der besten Individuen verloren. Dadurch kann es passieren, dass der Algorithmus bei einer lokalen Pareto-Front stecken bleibt, wodurch eine schlechte Konvergenz erzielt wird. [18] Aus diesem Grund wurde das Problem dieser Arbeit mit NSGA-II mit kontrolliertem Elitismus gelöst.

2.2 State of Health

Das Elektrofahrzeug im Problem dieser Arbeit wird von einer fest verbauten Traktionsbatterie angetrieben [48]. Diese Batterie ist eine Lithium-Ionen-Batterie, da diese von allen Typen momentan am besten ist [32]. Die Batterie hat einen State of Health (SoH), das heißt eine Schätzung der verbleibenden Lebensdauer [5]. Das Thema ist sehr umfangreich und wird aus Platzgründen in dieser Arbeit nur angeschnitten. Interessierte Leser verweise ich auf [5] und [65].
Der SoH wird durch verschiedene Alterungsprozesse verkürzt. Einer davon ist das Wachstum einer Solid Electrolyte Interphase (SEI)-Schicht an der negativen Elektrode. Dieser und andere Alterungsprozesse werden von einem hohen State of Charge (SoC) beschleunigt. Deshalb wird für eine Batterie oft ein empfohlener maximaler SoC angegeben. [5] Gleichermaßen gibt es für den SoC meist ein vom Hersteller empfohlenes Minimum. Der Grund dafür ist, dass sich die Lebensdauer mit einer größeren Tiefe der Entladung schneller verschlechtert [26].
Zum Laden der Batterie stehen verschiedene Ladetechnologien zur Verfügung. Für diese Arbeit sind davon das Normalladen und das Schnellladen relevant. Bei Ersterem wird typischerweise mit 240 Volt und bis zu 80 Ampere geladen, weswegen der Vorgang mehrere Stunden dauert. Das Schnellladen arbeitet bei 480 Volt und ermöglicht dadurch ein Aufladen in weniger als einer Stunde. [68] Eine höhere Spannung beim Laden geht allerdings mit einer schnelleren Alterung einher [5]. Somit ist das Schnellladen schädlicher für die Batterie als das Normalladen [19].

2.3 Electric Vehicle Routing Problem

Zur Optimierung von Ladevorgängen gibt es viel Literatur. Ein Großteil davon beschäftigt sich mit dem EVRP, einer Abwandlung des Vehicle Routing Problem. Als Eingabe erhält man ein Netzwerk, in dem ein Depot, mehrere Kunden und einige Ladestationen verteilt sind. Oft wird dieses Netzwerk als Graph repräsentiert. Die Ladestationen liegen dabei je nach Problemdefinition bei den Kunden oder an separaten Knoten. Eine Erweiterung des Problems beschäftigt sich gleichzeitig mit der Standortentscheidung für die Ladestationen [45]. In dieser Arbeit soll jedoch die existierende Infrastruktur genutzt

werden. Das Ziel beim EVRP ist es, möglichst optimale Routen für mehrere Elektrofahrzeuge zur Belieferung einer festgelegten Menge von Kunden zu finden. Das Depot ist dabei Startpunkt und Ziel aller Routen. Durch die begrenzte Batteriekapazität der Wagen kann dies ebenfalls das Einplanen des Ladens beinhalten. [70]
Das Problem dieser Arbeit unterscheidet sich zum EVRP dadurch, dass es keine Kunden gibt, die Optimierung nur für ein Elektrofahrzeug durchgeführt wird und der Start- und der Endpunkt einer Route nicht am selben Knoten liegen.
Eine recht umfassende und aktuelle Übersicht zu der Literatur, die sich mit dem EVRP beschäftigt, wurde von [40] erstellt. Ein Auszug dieser Tabelle, den ich selbst erweitert habe, findet sich in Tabellen 2.1, 2.2 und 2.3. Die Ergänzungen in Tabelle 2.1 wurden unterhalb des Strichs gemacht. Dafür habe ich nach Arbeiten gesucht, die EVRP multikriteriell lösen [56, 71] oder den SoH mit einbeziehen [1]. Ausgelassen wurden beispielsweise Flotten, die neben Elektrofahrzeugen auch Alternative Fuel Vehicle, Plug-in-Hybrids oder Personenkraftwagen beinhalten, da sich diese Arbeit auf Elektrofahrzeuge konzentriert. In der Literatur werden noch andere Abwandlungen des Vehicle Routing Problem betrachtet, deren Flotten komplett aus Alternative Fuel Vehicle [11] oder Plug-in-Hybrids [3] bestehen.
Die von den Arbeiten in Tabelle 2.1 betrachteten Flotten lassen sich in homogene und heterogene unterteilen. Heterogen sind diejenigen, die unterschiedliche Bauweisen eines Fahrzeugs beachten [34]. Von [48] werden beispielsweise Elektrofahrzeuge genommen, von denen ein Teil nur mit normaler Geschwindigkeit laden und der andere nur Schnellladen nutzen kann. Sind alle Fahrzeuge gleich, ist die Flotte homogen. Bei [49] können die Fahrzeuge zum Beispiel beide Ladetechnologien verwenden. Wie in Tabelle 2.1 zu sehen, sind die meisten verwendeten Flotten homogen.
Die meisten Papers beziehen nur eine Ladetechnologie zum Aufladen der Batterie ein. Einige wie zum Beispiel [25] unterscheiden aber verschiedene Ladegeschwindigkeiten.
Außerdem kommen in der Literatur noch Battery Swap Stations [7] und Wireless Charging Systems [44] vor, die in dieser Arbeit nicht betrachtet wurden. Bei Battery Swap Stations wird die Batterie im Elektrofahrzeug nicht geladen, sondern durch eine volle ausgetauscht. Durch Hochleistungsladestationen können die Batterien schneller geladen werden und die Fahrer sparen sich lange Ladezeiten. Allerdings sind die Einrichtungskosten solcher Stationen hoch,

Quelle	Flotte	Zeit-fenster	Energiever-brauchsfunktion	Auflade-funktion	Anzahl Lade-technologien	Aufladung		Zielfunktion	Lösungsmethode
						Voll	Teilweise		
Conrad and Figliozzi (2011) [17]	HO	-	L	-	1	✓	✓	1,2,3,5	Interaktive Konstruktion, Verbesserungsheuristiken
Schneider et al. (2012) [54]	HO	✓	L	L	1	✓		1,2	Hybrid-VNS und TS
Felipe et al. (2014) [25]	HO	✓	L	L	>1		✓	2,3	Mehrere Heuristiken und SA
Bruglieri et al. (2015) [13]	HO	✓	L	L	1		✓	1,2,3,4	VNSB
Ding et al. (2015) [21]	HO	✓	L	L	1		✓	2	Hybrid-VNS und TS
Keskin and Çatay (2016) [38]	HO	✓	L	L	1		✓	1,2	ALNS
Desaulniers et al. (2016) [20]	HO	✓	L	L	1	✓	✓	2	Branch-price-and-cut
Hiermann et al. (2016) [34]	HE	✓	L	L	1	✓		1,2	Branch-price, ALNS
Wen et al. (2016) [64]	HO	✓	L	L	1		✓	1,2	ALNS
Roberti and Wen (2016) [51]	HO	✓	L	L	1	✓	✓	2	GVNS, Dynamic Programming
Lin et al. (2016) [47]	HO	-	L	L	1	✓		1,2,3	Mixed Integer Programming
Grandinetti et al. (2016) [31]	HO	✓	L	L	1		✓	1,2,6	Weighted Sum Method
Sweda et al. (2017) [59]	HO	-	L	*	>1		✓	2,3,4	Verschiedenen optimale und heuristische Verfahren
Bruglieri et al. (2017) [12]	HO	✓	L	L	1		✓	1,5	VNSB basierte Matheuristik
Montoya et al. (2017) [49]	HO	-	L	NL	>1		✓	5	Hybrid-ILS und HC
Froger et al. (2017) [28]	HO	-	L	NL	1		✓	5	Zwei-Phasen-Matheuristik
Zhang et al. (2018) [70]	HO	✓	L	L	1		✓	1,2,3,4,5	TS
Breunig et al. (2018) [10]	HO	-	L	L	1	✓		1,2	LNS, exakter Algorithmus
Keskin und Çatay (2018) [39]	HO	✓	L	L	>1		✓	1,3	ALNS basierte Matheuristik
Kullman et al. (2018) [43]	HO	-	L	NL	1		✓	✓	Verschiedenen exakte und heuristische Methoden
Froger et al. (2019) [29]	HO	-	L	NL	1		✓	5	Mixed Integer Programming
Siddiqi et al. (2012) [56]	HO	-	-	-	1	✓		3,7,8	Angepasster MOEA
Zhang et al. (2016) [71]	HO	**	NL	L	1	✓		5,7,9,10,11	Multikriterielle Ant Colony Optimierung
Adulaal et al. (2017) [1]	HO	-	-	-	>1	✓	✓	3,12,13	MDP, Evolutionärer Algorithmus
Mkahl et al. (2017) [48]	HO & HE	-	L	-	>1		-	14	Primalsimplex-Verfahren
Keskin und Çatay (2019) [40]	HO	✓	L	***	1		✓	1,2,3,6	ALNS, Primalsimplex-Verfahren

L linear NL nicht-linear HO homogen HE heterogen * linear aber mit unterschiedlicher Kostenzusammensetzung für unterschiedliche Ladegeschwindigkeiten
** Einhaltung einer vorher festgelegten Ankunftszeit erforderlich *** abschnittsweise linear

Tabelle 2.1: Erweiterter Auszug aus der Literaturübersicht zu EVRP von [40]

wodurch die Infrastruktur dafür noch fehlt [45]. Bei Wireless Charging Systems kann der Wagen auf bestimmten Streckenabschnitten während der Fahrt geladen werden [44].
Weiterhin wird unterschieden, ob partial oder voll geladen wird. In manchen Papers kann auch aus beidem gewählt werden. Die Batterie nur teilweise wieder aufzuladen bietet den Vorteil, dass die Ladezeiten kürzer sind und nicht überladen werden kann. Der Nachteil ist jedoch, dass der Wagen dadurch eine geringere, eventuell nicht ausreichende Reisedistanz hat. [38]
Bei den Berechnungen nutzen die meisten Autoren eine Aufladefunktion, die zum Großteil linear und nur in seltenen Fällen nicht-linear ist. Eine andere Funktion, die fast immer aufgestellt wird, ist eine Energieverbrauchsfunktion. Wie man aus Tabelle 2.1 ablesen kann, ist diese mit einer Ausnahme immer linear.
Die Zielfunktionen beziehungsweise Komponenten der Zielfunktionen, die in der Literatur verwendet werden, sind in Tabelle 2.2 aufgelistet. Bis auf [56] und [71] betrachten alle Arbeiten nur eine Zielfunktion, die häufig aus mehreren Aspekten besteht. In den meisten Fällen werden verschiedene Kostenpunkte wie Fahrzeugkosten oder Gesamt-Fahrtkosten einbezogen. Bei [1] wird die Kostenberechnung durch die Integration des Profits von Vehicle-to-Grid (V2G), dem Zurückführen von Batterieladung an die Ladestation, verfeinert. Einige optimieren die Ladezeit beziehungsweise die Ladekosten oder die Gesamtzeit. Selten ist die Wartezeit in der Zielfunktion enthalten. Durch die Papers zu multikriteriellen EVRP sind die Reisedistanz, die Fahrzeit und der Energieverbrauch als Zielfunktionen hinzugekommen. Eine Arbeit betrachtet ein qualitatives Merkmale der Routen wie den Fahrkomfort in Form der Innentemperatur [71].
Der SoH wird von [1] als Batterielebensdauerverlust-Kosten und von [71] als durchschnittliche SEI-Wachstumsrate einbezogen. [1] hat die Berechnung der Batterielebensdauerverlust-Kosten von [66] als die Anzahl der Besuche von Ladestationen multipliziert mit den Batterieersetzungskosten geteilt durch den maximalen Lebenszyklus einer Batterie übernommen. Einen anderen Ansatz hat [25] gewählt, der annimmt, dass eine Batterie eine begrenzte Anzahl an Aufladezyklen hat. Er berechnet deshalb für jeden Zyklus einen festen Betrag beim Aufladen.
Viele Autoren bauen außerdem Zeitfenster in ihr Routing-Problem ein. Diese müssen bei der Lieferung an die Kunden und bei der Rückkehr zum Depot am Abend eingehalten werden. [40] Bei [31] wird sogar eine Strafterm vergeben,

	Minimierung der/des ...
1	Fahrzeugkosten
2	Gesamt-Fahrtkosten
3	Ladezeit/-kosten
4	Wartezeit
5	Gesamtzeit (Fahren und Laden)
6	Strafen für Verspätungen
7	Reisedistanz
8	Fahrzeit
9	Energieverbrauchs
10	durchschnittlichen SEI-Wachstumsrate
11	Differenz zwischen Temperatur im Auto und gewünschter Temperatur
12	Batterielebensdauerverlust-Kosten
	Maximierung des ...
13	Profits von V2G
14	SoC bei Erreichen der Ladestation

Tabelle 2.2: Zielfunktionen und Komponenten der Zielfunktionen aus Tabelle 2.1

wenn ein Zeitfenster verletzt wird.
Die Breite der genutzten Lösungsmethoden ist groß. Häufig werden jedoch verschiedene Varianten der Neighborhood Search, vor allen Dingen Adaptive Large Neighborhood Search (ALNS) und Variable Neighborhood Search (VNS), aber auch General Variable Neighborhood Search (GVNS) oder Variable Neighborhood Search Branching (VNSB) verwendet. Öfters sieht man ebenfalls TS. Vereinzelt wurden Dynamic Programming, Ant Colony Optimization, Weighted Sum Method, Mixed Integer Programming, SA, Iterated Local Search (ILS), Branch-price oder Branch-price-and-cut eingesetzt. Für wenige Autoren ist die Lösungsmethode nur angegeben als Matheuristiken, exakte oder heuristische Verfahren wie unter anderem Heuristic concentration (HC). [40] ist in seiner Tabelle zwar auf die Zielfunktionen eingegangen, hat aber nicht die dafür jeweils beachteten Systemparameter aufgelistet. Deshalb habe ich dazu in Tabelle 2.3 eine Übersicht, kategorisiert nach fünf Zielfunktionen, erstellt. Ein Systemparameter ist nicht in dieser Tabelle, aber bei allen als Graph repräsentierten EVRP vorhanden. Dabei handelt es sich um eine

Sys.param./ Zielfkt.	SoC bei Erreichen der Ladestation (%) → Max	Energieverbrauch (kWh) → Min	Gesamtkosten ($/£) → Min	Reisedistanz (km) → Min	Gesamt-Reisezeit (Minuten) → Min
Kartendarstellung					
Reisedistanz pro Kanten		[70], [71]	[40], [30], [17], [34], [10]	[30], [21], [38], [51], [31], [54], [56], [71]	
Reisezeit			[13], [40], [47], [59]		[12], [49], [28], [43], [29], [56], [71]
Beim Kunden benötigte Zeit			[17]		[28]
Reisekosten pro Kante			[64], [20]		
Stau (Durchschnittsgeschwindigkeit auf Kante)	[48]				
Höhenprofil	[48]		[47]		
Fahrzeug					
Leergewicht		[70], [71]	[30], [47]		
Gewicht der Ladung		[70]	[30]		
Durchschnittsgeschwindigkeit		[70], [71]	[30]		
Beschleunigung		[70], [71]	[47]		
frontale Fläche		[70], [71]	[47]		
Motoreffizienz		[70]	[47]		
Anschaffungskosten			[40], [34], [64]		
Anzahl eingesetzter Fahrzeuge			[13]	[54]	
Fahrzeugnutzungskosten			[31], [10]		
Umwelteinflüsse					
Fallbeschleunigung		[70], [71]	[47]		
Rollwiderstand		[70], [71]	[47]		
Luft-/Strömungswiderstand		[70], [71]	[30], [47]		
Fahrkomfort					
Verwendungsdauer der Klimaanlage		[71]			
Leistung der Klimaanlage		[71]			

Batterie				
Batteriekapazität			[47], [39]	
State of Charge (SoC)	[48]		[47], [39]	
Ladewirkungsgrad		[70]		
Entladewirkungsgrad		[70]	[1]	
Maximaler Lebensdauer			[1]	
Batterieersetzungskosten*			[30], [25], [1]	
Lieferung				
Stündlicher Lohn des Fahrers			[40], [30], [47]	
Bezahlung für Überstunden			[40]	
Strafen für Nichteinhaltung des Lieferfensters			[40], [31]	
Ladestation				
Position	[48]			
Freier Ladeplatz/Wartezeit	[48]		[13], [59], [1]	[12], [28], [43]
Leistung/Aufladegeschwindigkeit	[[48]]		[39],[47], [1]	
Ladekosten (Station)			[40], [30], [25], [47], [59], [39], [1]	
Ladekosten (Depot/zuhause)			[25], [39], [1]	
Laden beim Kunden			[17]	
Menge nachgeladener Energie			[39]	
Ladezeit			[13], [1]	[12], [49], [28], [43], [29], [56], [71]
Entladezeit			[1]	
Profit von V2G			[1]	
Anzahl besuchter Ladestationen			[1]	

* bei [30] fester Betrag bei jedem Aufladen (Anschaffungskosten/erwartete Anzahl Aufladezyklen),
bei [25] Batterieersetzungskosten = Anschaffungskosten/erwartete Lebensdauer in km, bei [1] nicht näher erläutert

Tabelle 2.3: Übersicht zu Zielfunktionen und dafür beachteten Systemparametern in der Literatur

binäre Variable, die angibt, ob eine Kante von einem Fahrzeug befahren wird oder nicht.
Die am häufigsten verwendete Zielfunktion ist die Minimierung der Gesamtkosten. Darin sind oft tatsächliche Kosten enthalten. Beachtet werden die Ladekosten, die Anschaffungskosten für ein Elektrofahrzug, die Fahrzeugnutzungskosten, die Batterieersetzungskosten, der stündliche Lohn des Fahrers, die Bezahlung für Überstunden und die Strafen bei Nichteinhaltung der Lieferfenster. Bei [1] konnte ein Teil der Kosten über den Profit aus V2G zurückgewonnen werden. In zwei Fällen wird für die Ladekosten erst noch der Energieverbrauch berechnet [47, 70]. Einige Autoren integrieren außerdem die Fahr-, Warte- oder Ladezeit oder die Reisedistanz in die Gesamtkosten, indem sie für diese Zielwerte eine Kostenfunktion definieren.
Andere minimieren die Gesamt-Reisezeit als eigenständige Zielfunktion und beachten dafür fast immer dieselben Parameter. Diese sind die Fahrzeit, die Wartezeit an den Ladestationen und die zum Laden benötigte Zeit. Nur [28] erweitert diese Liste um die beim Kunden verbrauchte Zeit.
Für die Minimierung der Reisedistanz wird bei allen Arbeiten außer bei [54] nur die Reisedistanz, meistens pro Kante, als Systemparameter gewählt. [54] minimiert zusätzlich die Anzahl eingesetzter Fahrzeuge.
[48] ist der Einzige, der das Hauptaugenmerk auf den SoC legt und versucht, die SoCs aller Fahrzeuge, die sie beim Erreichen der Ladestationen haben, zu maximieren.
Manche Zielfunktionen kommen nur vereinzelt vor und sind deshalb nicht in Tabelle 2.3 aufgeführt. Dazu zählt [17], der die eingesetzten Routen und Fahrzeuge minimiert. Andere, die diese Parameter einbeziehen, lösen dies indirekt über die Minimierung der Fahrzeugnutzungskosten. Eine weitere nicht aufgelistete Zielfunktion stammt von [71], der den Komfort als Differenz zwischen tatsächlicher und gewünschter Innentemperatur im Fahrzeug misst.
Der Beitrag dieser Arbeit ist das Schließen einer Lücke in der Literatur, da es nach bestem Wissen der Autorin keine Arbeiten gibt, die die Optimierung von Ladeempfehlungen a posteriori und multikriteriell lösen.
Für das EVRP beziehen viele Autoren zwar mehrere Aspekte ein, optimieren diese aber nicht gleichzeitig. Die von [40] erwähnten Paper haben entweder nur eine Zielfunktion oder mehrere, die einzeln [30] oder nacheinander [17] betrachtet werden. Bei [17] wird beispielsweise basierend auf dem Ergebnis der Optimierung der ersten Zielfunktion, die zweite minimiert. Autoren, die mehrere Aspekte beachten, kombinieren diese mithilfe von a priori Methoden

zu einer Zielfunktion. [15] und [31] nutzen beispielsweise die Weighted Sum Method, [63] einen Fuzzy Programming Approach und [52] die Adaptive ε-Constraint Method.
Es findet sich auch Literatur mit a posteriori Lösungsmethoden, allerdings nur zu EVRP ähnlichen Problemen. [35] führt eine multikriterielle Optimierung für das Vehicle Routing Problem mit konventionellen Kraftfahrzeugen durch. [6] macht dasselbe für Plugin-in-Hybrids. [57] und [69] betrachten das Problem aus Sicht des Stromnetzes und optimieren die Verteilung dessen Belastung. [67] versucht multikriteriell eine optimale Positionierung für Ladestationen zu finden.
Die Arbeiten von [56] und [71] kommen am nächsten an das Problem dieser Arbeit heran, indem sie EVRP für ein Elektrofahrzeug multikriteriell und a posteriori lösen. [56] nutzt dafür einen angepassten MOEA und [71] eine multikriterielle Ant Optimization. Der bedeutende Unterschied zu dieser Arbeit ist jedoch, dass das System nach der Optimierung eine der gefundenen Lösungen aussucht. Dem Nutzer werden also keine Ladeempfehlungen gegeben, aus denen er sich eine aussuchen kann, sondern eine festgelegte Route mit Halt zum Laden im Navigationssystem angezeigt.
Literatur, die sich mit dem Anzeigen von Ladeempfehlungen für den Fahrer beschäftigt, hat keine oder nur eine einzige Zielfunktion. [62] gibt Ladeempfehlungen für Taxis, die die kombinierte Fahr- und Wartezeit minimieren. [53] hat ein Warnsystem für Elektrofahrzeuge entwickelt, welches empfiehlt zu laden, falls der SoC bei Ankunft unter einem festgelegten Wert liegen würde. Zum Laden wird immer die Ladestation mit der geringsten Entfernung vorgeschlagen. [27] zeigt Empfehlungen zu Ladestationen basierend auf der Ladetechnologie, dem Betreiber, dem Standort, Orten von Interesse und weiteren vom Nutzer getroffene Entscheidungen an.
Zusammenfassend gesagt, gibt es viel Literatur, die sich mit zu dem Problem dieser Arbeit verwandten Themen beschäftigt. Allerdings gibt es kein Paper, welches Ladeempfehlungen inklusive Alternativrouten multikriteriell und a posteriori optimiert und die Wahl der genommenen Strecke dem Nutzer überlässt. Diese Lücke soll mit dieser Arbeit geschlossen werden.

3 Problemstellung

Das Problem ist auf einem abstrakten, automatisiert erstellten Streckennetz definiert, das durch einen ungerichteten Graphen ohne Schleifen repräsentiert wird. Die Karte eines Beispiels für ein solches Problem ist in Abbildung 3.1 zu sehen.
Knoten werden über eine aufsteigende Nummerierung mit ganzen Zahlen identifiziert und in zwei Typen unterteilt. Der erste wird als normale Knoten, wie zum Beispiel die aktuelle Position des Fahrzeugs oder sein Ziel, bezeichnet. Die zweite Art sind Ladestationen, denen eine Wartezeit und vorhandene Ladetechnologien inklusive deren Preise zugewiesen werden. Das Normalladen, auch Alternating current (AC)-Laden genannt, mit Abrechnung in ct/kWh steht immer zur Verfügung. Seltener wird Schnellladen, auch als Direct current (DC)-Laden bezeichnet, oder eine Abrechnung mit Pauschale angeboten. In der Realität unterscheidet sich bei den Ladestationen außerdem die Anzahl an Batterien, die gleichzeitig geladen werden können [48]. Diesen Parameter habe ich jedoch nicht betrachtet, da er vor allen Dingen Einfluss auf die Wartezeit hat und deshalb bereits abgedeckt ist.
Kanten sind eine Fahrzeit in Minuten und ein Energieverbrauch in vom SoC verbrauchten Prozente zugewiesen.
Auf dem beschriebenen Streckennetz fährt ein Elektrofahrzeug. Für dieses müssen die Kapazität der Batterie, der durchschnittliche Verbrauch, die Durchschnittsgeschwindigkeit und der vom Hersteller definierte Mindest-SoC bekannt sein. Außerdem ist ein SoC festgelegt, den das Fahrzeug zu Beginn der Berechnungen hat.
Weiterhin ist die geplante Route inklusive aktueller Position und Ziel, die ein Fahrer ohne Beachtung einer möglicherweise notwendigen Ladung gefahren wäre, vorgegeben. Damit kann ermittelt werden, ob eine Alternativroute, die zum Laden eventuell genommen werden muss, schlechter beziehungsweise besser ist. Für spätere Implementierungen könnte man nur die aktuelle Position und das Ziel verwenden oder Orte von Interesse bei der Auswahl der Ladestation und der Planung der Alternativroute einbeziehen. Dabei wäre es allerdings

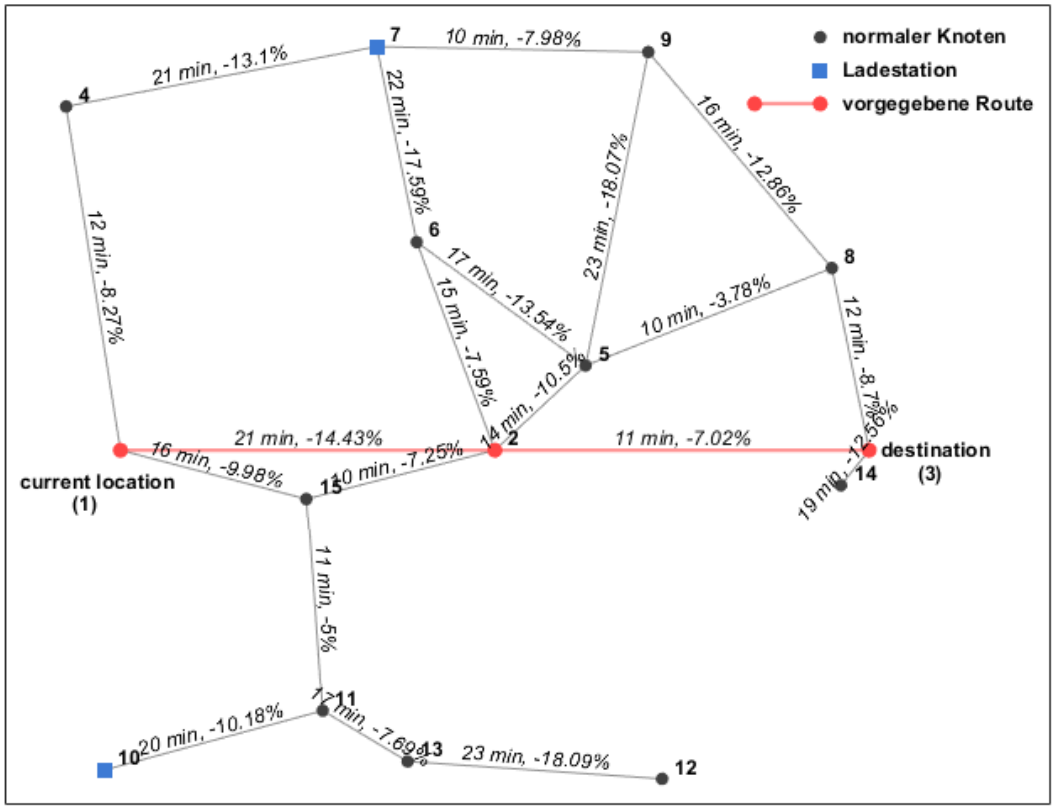


Abbildung 3.1: Beispiel-Karte bei mittlerer Verkehrsdichte

hilfreich, wenn einzelne Punkte der Strecke trotzdem bekannt wären. Da die Optimierung des Ladens für die gesamte Strecke wahrscheinlich zu rechenaufwändig ist, könnte so das Problem in kleinere Teilprobleme zerlegt werden.
Ein weitere Aspekt des Problems ist, dass angelehnt an [15] sich die Kilowattstunden-Preise je nach Tageszeit unterscheiden. Es gibt sechs Zeiträume mit drei unterschiedlichen Preisstufen: ein Normalpreis, ein Tiefpreis und ein Höchstpreis. Für den Tiefpreis wird der Normalpreis halbiert und für den Höchstpreis mit 1,5 multipliziert. Diese Faktoren werden außerdem auf die Fahrzeiten und die Wartezeiten angewandt, um die unterschiedlichen Verkehrsdichten am Tag zu simulieren. So sind beispielsweise in der Mittagspause und im Feierabendverkehr deutlich mehr Fahrzeuge unterwegs als zu anderen Tageszeiten. Dadurch braucht man nicht nur länger für dieselben Strecken sondern muss auch größere Wartezeiten an den Ladestationen einplanen. Da man bei höherer Verkehrsdichte außerdem häufiger bremst und beschleunigt, ist der Energieverbrauch ebenfalls höher [8]. Deshalb wird Letzterer an den Kanten

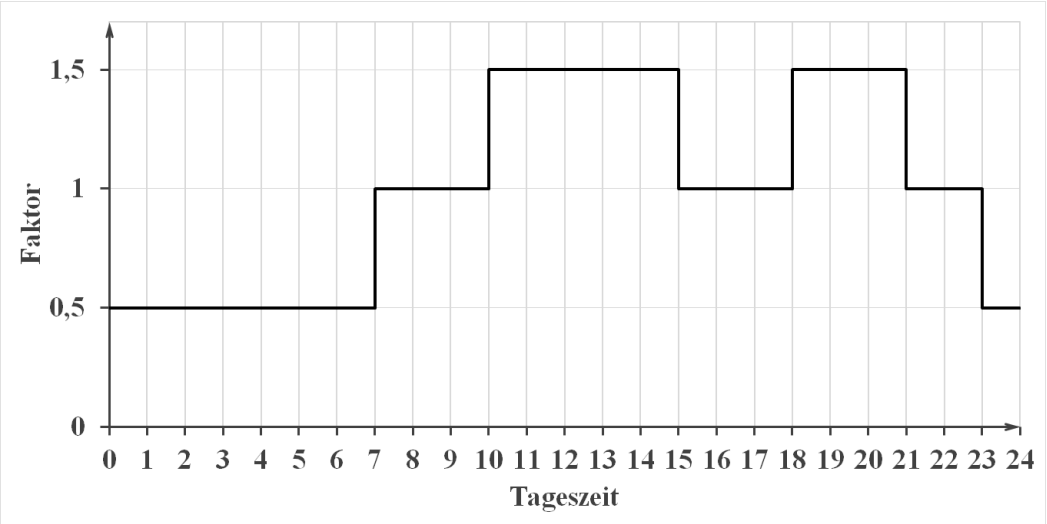


Abbildung 3.2: Preisentwicklung an einem Tag angelehnt an [15]. Der Wechsel auf einen anderen Faktor geschieht immer zur vollen Stunde, z.B. um 7 Uhr

multipliziert. Die Zeiträume und deren jeweiligen Faktoren sind in Abbildung 3.2 dargestellt. Das Beispiel-Problem in Abbildung 3.1 wurde mit eins multipliziert, wodurch die Karte bei mittlerer Verkehrsdichte dargestellt ist. Zur Berechnung wird aus diesem Grund die Angabe der Startzeit an der aktuellen Position benötigt.
Das Ziel bei der Lösung des Problems ist, dem Nutzer eines Elektrofahrzeugs während der Fahrt geeignete Ladestationen mit passenden Alternativrouten anzuzeigen. Dabei wird angenommen, dass auf einer Alternativroute maximal einmal vollgeladen wird. Welche der vorgeschlagenen Routen der Fahrer nimmt, entscheidet er selbst. Zum aktuellen Zeitpunkt ist das System nur zur Optimierung für einen Nutzer gedacht. Man könnte es erweitern zu einer Optimierung, die Informationen von der Kommunikation zu anderen Fahrzeugen nutzt, oder zu einer Optimierung der Ladeempfehlungen für mehrere Fahrzeuge. Die Eignung der Ladeempfehlungen mit den entsprechenden Alternativrouten wird anhand folgender Ziele festgestellt:

1. Minimierung des Energieverbrauchs
2. Minimierung der Ladekosten
3. Minimierung der Reisezeit
4. Minimierung des SoH-Rückgangs

Eine genauere Beschreibung der Zielfunktionen ist in Kapitel 4 zu finden. Als Zielfunktionen wurden die Werte ausgewählt, die für Fahrer von Elektrofahrzeugen am wichtigsten sind. Für viele ist, wie bei der herkömmlichen Routenplanung, die Strecke interessant, die die geringste Zeit benötigt. Ein weiterer wichtiger Faktor sind Kosten, weswegen die Ladekosten und der SoH-Rückgang beachtet wurden. Der SoH-Rückgang beeinflusst die Kosten indirekt, da das Ersetzen einer Batterie teuer ist und deshalb die Lebensdauer der Batterie so wenig wie möglich verkürzt werden sollte. Ein geringer Energieverbrauch hat für den Fahrer den Vorteil, dass er möglichst wenig und möglichst selten nachladen muss, wodurch sowohl Kosten als auch Zeit gespart wird.
Diese Ziele widersprechen sich untereinander, weswegen sie gleichzeitig betrachtet werden und es sich um ein multikriterielles Problem handelt. Wenn man beispielsweise die Strecke mit Schnellladen wählt, um Zeit zu sparen, geht damit eine geringere Schonung der Batterie einher. Ebenso hat die kürzeste Strecke nicht unbedingt den geringsten Energieverbrauch und die Ladestation, zu der man am schnellsten gelangt, nicht den niedrigsten Preis.

4 Modellaufbau

In diesem Kapitel erkläre ich die vier Zielfunktionen, die zur Optimierung genutzt werden, und gehe dabei auf die verwendeten Systemparameter ein.

4.1 Minimierung des Energieverbrauchs

Im Interesse des Fahrers liegt es, während der Fahrt möglichst wenig Energie zu verbrauchen, da er dadurch seltener oder weniger nachladen muss. Zur Ermittlung des Energieverbrauchs reicht es theoretisch nicht aus, nur die Fahrzeit oder die Distanz zu betrachten. Denn selbst wenn man zweimal dieselbe Strecke fährt, kann sich die verbrauchte Energie beispielsweise aufgrund unterschiedlicher Fahrweisen, Verkehrsdichte oder Nutzung stromverbrauchender Funktionalitäten wie Licht unterscheiden [48]. Zur Vereinfachung wurde in dieser Arbeit das Sinken des SoC für jede Kante zwar trotzdem in Relation zu dessen Fahrzeit gewählt, aber Zeiträume und zufällige Veränderungen eingebaut.
Als Erstes muss der durchschnittliche Energieverbrauch pro Minute EV/Min für ein Fahrzeug mit einer bestimmten Batteriekapazität in kWh, einem durchschnittlichen Verbrauch in kWh/km und einer Durchschnittsgeschwindigkeit in km/h wie in Formel 4.1 berechnet werden. Das Ergebnis besagt, um wie viel Prozent der SoC im Durchschnitt pro Minute bei dem angegebenen Fahrzeug sinkt. Mit EV/Min kann man für jede Kante (i, j) im Graph den Rückgang des SoC als Produkt mit der Reisezeit, wie in Formel 4.2 aufgeschrieben, ermitteln. Um abzubilden, dass zwei Strecken mit derselben Reisezeit nicht immer denselben Energieverbrauch haben, wird für EV/Min eine Unsicherheit von [-0,25, 0,25] eingebaut.

$$EV/Min = \frac{Geschwindigkeit * \frac{1}{60}h * Verbrauch}{Kapazitaet} * 100\%; \tag{4.1}$$

$$EV(i,j) = (EV/Min + random(-0.25, 0.25)) * Reisezeit(i,j) \tag{4.2}$$

Für eine Route R, die aus einer Folge von Knoten besteht, von denen jeweils zwei aufeinanderfolgende i und j durch eine Kante (i,j) verbunden sind, lässt sich der Energieverbrauch der Strecke als Summe der Rückgänge vom SoC der jeweiligen Kante multipliziert mit dem Faktor der Zeit, zu der das Fahrzeug an Knoten i war, errechnen. Da die unterschiedlichen Verkehrsdichte in den Zeiträumen Einfluss auf die verbrauchte Energie haben [8], muss deren Faktor bei der Berechnung in Formel 4.3 einbezogen werden. Falls sich der Zeitraum und damit auch der Faktor während der Fahrt auf der Kante ändern, hat dies keine Auswirkung. Dies könnte man als Erweiterung der Lösungsmethode für die Zukunft in Betracht ziehen.

$$EV(R) = \sum_{(i,j)\in R} EV(i,j) * Faktor(i) \tag{4.3}$$

4.2 Minimierung der Ladekosten

Der Fahrer will möglichst wenig beim Laden bezahlen. Dafür ist entscheidend, ob Normalladen oder Schnellladen genutzt und ob der Preis pro Kilowattstunde oder als Pauschale abgerechnet wird. Einige Ladestationen bieten auch Preise pro Minute [50], die bei meiner Berechnung der Ladekosten jedoch ungefähr dem Preis pro Kilowattstunde entsprechen und deshalb nicht betrachtet wurden. Eine Übersicht zu den Normalpreisen fürs Laden habe ich in Tabelle 4.1 zusammengestellt. Die Kilowattstunden-Preise werden von der Tageszeit beeinflusst, für die sich die Faktoren wie in Abbildung 3.2 dargestellt unterscheiden.
Für eine Elektrofahrzeug, dass an Ladestation i mit Ladetechnologie t, die pro Kilowattstunde abgerechnet wird, zu Stunde h lädt, berechnen sich die Ladekosten in Euro wie in Formel 4.5. Dazu muss vorher noch die nachzuladende Menge an Energie in Kilowattstunden ermittelt werden. Zur Vereinfachung wird angenommen, dass immer vollgeladen wird und dass das Laden linear geschieht. Wie in Formel 4.4 zu sehen, ist die benötigte Menge vom anfänglichen Ladezustand des Autos SoC_{Start}, dem Verbrauch an Batterieladung und

Ladetechnologie	Abrechnungsart	(Normal-)Preis
AC-Laden	pro kWh	29-30 ct/kWh
	Pauschale	5,95 €
DC-Laden	pro kWh	39-50 ct/kWh
	Pauschale	8,95 €

Tabelle 4.1: Übersicht zu den Ladekosten

der Kapazität der Batterie abhängig. SoC_{Start} und der Verbrauch werden in Prozent angegeben.

$$Menge = Kapizitaet * \frac{100 - (SoC_{start} - Verbrauch)}{100} \tag{4.4}$$

$$Ladekosten(i, t, s) = \frac{Kosten(i, t) * Faktor(h)}{100} * Menge \tag{4.5}$$

Für ein Elektrofahrzeug, das an Ladestation i zu einem Pauschalpreis mit Ladetechnologie t lädt, berechnen sich die Ladekosten in Euro wie in Formel 4.6.

$$Ladekosten(i, t) = Kosten(i, t) \tag{4.6}$$

4.3 Minimierung der Reisezeit

Die Alternativroute sollte möglichst wenig Zeit in Anspruch nehmen, um die Planung des Fahrers nicht negativ zu beeinflussen. Beachtet werden dazu die Verzögerung der Fahrzeit $Fahrzeit_{Alt} - Fahrzeit_{Route}$, die durch den Wechsel von der Route auf die Alternativroute entsteht, die Wartezeit und die Ladezeit bei Vollladung an der empfohlenen Ladestation. Wie in Formel 4.7 zu sehen ist, bildet die Summe der drei Bestandteile die Reisezeit. Alle Zeiten sind in Minuten angegeben.

$$Reisezeit = Fahrzeit_{Alt} - Fahrzeit_{Route} + Wartezeit + Ladezeit \tag{4.7}$$

Der Programmablauf zur Berechnung der Reisezeit ist in Abbildung 4.1 dar-

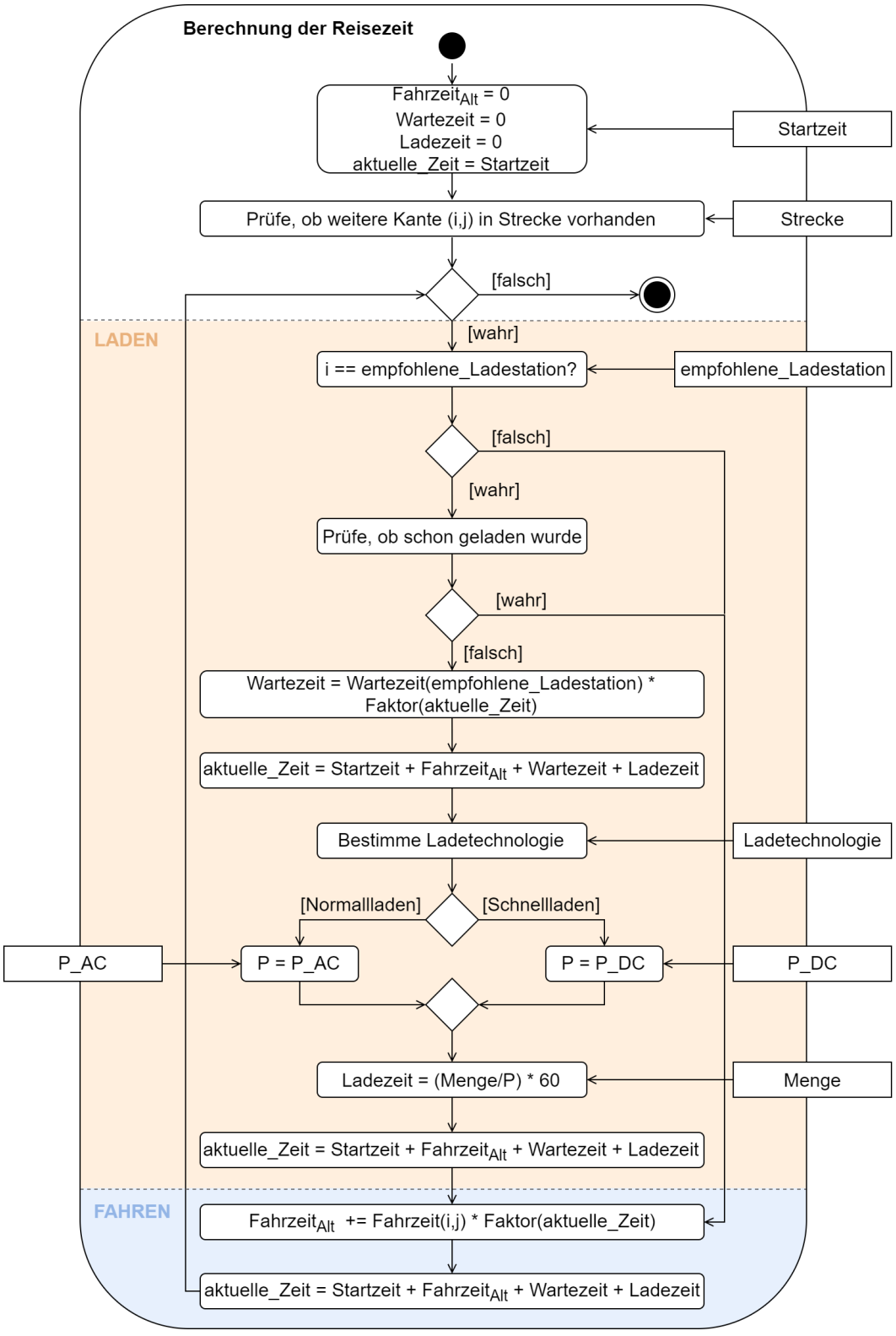


Abbildung 4.1: Programmablauf zur Berechnung der Reisezeit. Die Menge wird mithilfe von Formel 4.4 ermittelt.

gestellt. Abstrahiert beschrieben, besucht der Algorithmus nacheinander die Knoten der Alternativroute, summiert dabei die Fahrzeit auf und aktualisiert bei Erreichen der empfohlenen Ladestation die Wartezeit und mithilfe der vorgeschlagenen Ladetechnologie die Ladezeit. Je nachdem, ob Normalladen oder Schnellladen genutzt wird, unterscheidet sich die Leistung P der Ladestation. Die Art der Abrechnung hat darauf keinen Einfluss. Die Leistung beim Normalladen P_AC beträgt 15 kW und beim Schnellladen P_DC 50 kW. Als Werte habe ich die Mittelwerte von [4] genommen. Wird nicht geladen, betragen die Wartezeit und die Ladezeit null.
Bei der Ermittlung der Reisezeit muss außerdem berücksichtigt werden, dass während der Fahrt der Zeitraum wechseln kann, wodurch die Fahr- und die Wartezeiten beeinflusst werden. Deshalb wird nach dem Zurücklegen eines Routenabschnitts, dem Warten und dem Laden die aktuelle Zeit jeweils erneut festgestellt. Dazu muss die Zeit, zu der das Fahrzeug gestartet ist, angegeben werden.
Die noch benötigte Fahrzeit der Route berechnet sich wie die der Alternativroute in Abbildung 4.1.

4.4 Minimierung des SoH-Rückgangs

Da das Ersetzen einer Batterie mit hohen Kosten verbunden ist, möchte der Fahrer die Lebensdauer beim Benutzen des Elektrofahrzeugs möglichst wenig verkürzen. Dazu sollte das Unterschreiten des vom Herstellers angegebenen Mindest-SoCs und das Schnellladen vermieden werden [5, 26]. Tritt dies dennoch auf, wird ein Strafterm dafür vergeben. Wie in Formel 4.8 angegeben, setzt sich dieser aus den Straftermen für die jeweiligen Ereignisse zusammen.

$$SoH\_penalty = DC\_penalty + SoC\_penalty \tag{4.8}$$

Der Strafterm für das Schnellladen, wie in Formel 4.9 formuliert, entspricht der für eine Vollladung benötigten Ladezeit, dessen Berechnung in Abbildung 4.1 zu finden ist. Falls nicht schnellgeladen wird, beträgt der Wert null. Laut [5] ist für den Rückgang des SoHs beim Schnellladen vor allem der Unterschied zwischen dem SoC vor dem Laden und danach entscheidend. Dieser und die Ladezeit sind in meiner Arbeit direkt proportional zueinander. Zur besseren

Vergleichbarkeit mit *SoC_penalty* habe ich deshalb die Ladezeit als Strafterm genutzt.

$$DC\_penalty = \begin{cases} Ladezeit, & falls\ schnellgeladen \\ 0, & sonst \end{cases} \tag{4.9}$$

Unterschreitet der SoC den Mindest-SoC, wird für jede Kante (i, j), für die das der Fall ist, das Produkt aus der Höhe der Unterschreitung $h(i, j)$, der Fahrzeit der Kante $FZ(i, j)$ und dem Faktor des Zeitraums $F(i)$, in dem das Fahrzeug auf der Kante gestartet ist, zum Strafterm hinzugefügt. Wenn S die Menge von Kanten auf der Route ist, auf denen das Minimum unterschritten wurde, kann der Strafterm mithilfe von Formel 4.10 ermittelt werden.

$$SoC\_penalty = \begin{cases} \sum\limits_{(i,j)\in S} h(i,j) * FZ(i,j) * F(i), & falls\ S \neq \emptyset \\ 0, & sonst \end{cases} \tag{4.10}$$

5 Lösungsmethode

Das Ziel bei der Lösung des Problems ist das Finden möglichst guter Ladeempfehlungen. Dies wird in der aktuellen Implementierung für selbst erstellte Anwendungsfälle durchgeführt. Diese werden als ein Graph mit Knoten, die über ganze Zahlen identifiziert werden, und mit einer vorgegebenen Route, die das eventuell notwendige Laden nicht beachtet, repräsentiert. Dadurch ist außerdem die aktuelle Position des Fahrers und sein Ziel bekannt.
Das Problem wurde mithilfe eines MOEA gelöst. Konkret handelt es sich um NSGA-II mit kontrolliertem Elitismus nach [18], welches ich schon in Kapitel 2.1 erwähnt habe. Es wurde dieser Algorithmus gewählt, da er im Gegensatz zu anderen sowohl eine gute Konvergenz als auch eine gute Diversität verspricht [18]. Die vorgeschlagene Lösungsmethode erkläre ich an einem Beispiel-Anwendungsfall, der in Abbildung 5.1 dargestellt ist. Die Zahlen, mit denen die Knoten beschriftet sind, sind deren Indices und an jeder Kante steht zuerst die benötigte Fahrzeit und danach der verbrauchte SoC. Das erwähnte Verfahren ist bereits als Funktion *gamultiobj()* in MATLAB enthalten. Da die Lösungskandidaten meines Problems mithilfe ganzer Zahlen kodiert werden und die MATLAB-Funktion dafür ungeeignet ist, wurden die meisten Teilschritte des Algorithmus trotzdem selbst konzipiert und implementiert.
Der Ablauf des Algorithmus ist in Abbildung 5.2 veranschaulicht. Er beginnt mit der Initialisierung und der Berechnung der Fitnesswerte der daraus entstandenen initialen Population. Solange nicht eins der Abbruchkriterien erfüllt ist, werden mithilfe der Paarungsselektion Eltern für die Rekombination und Mutation aus der aktuellen Generation ausgewählt. Beide Schritte werden danach ausgeführt und erstellen insgesamt so viele Kinder wie schon Individuen in der Generation vorhanden sind. Der Anteil der Kinder, der durch die Rekombination zustande kommt, ist veränderbar. Der Rest der Kinder wird durch die Mutation erstellt. Die Rekombination und die Mutation werden dabei unabhängig voneinander ausgeführt. Das heißt, die Individuen aus der Rekombination werden nicht mutiert und mutierte Individuen werden nicht zur Rekombination genutzt. Anschließend werden alle Kinder mithilfe der Fitness-

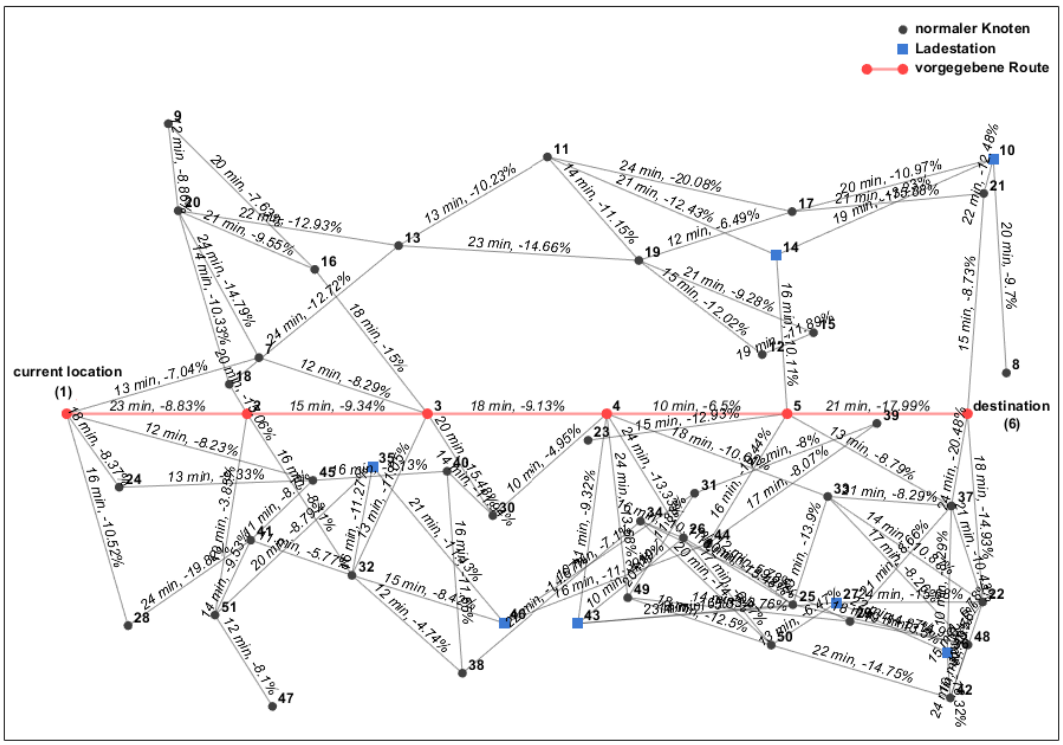


Abbildung 5.1: Karte zum Beispiel-Anwendungsfall bei mittlerer Verkehrsdichte mit der vom Navigationssystem vorgeschlagenen Route [0 0 2 3 4 5 6]

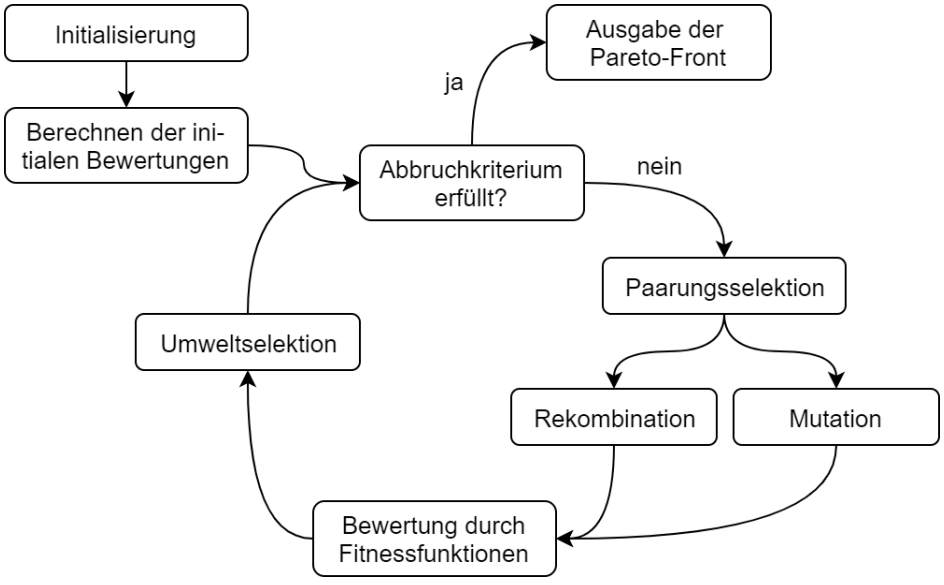


Abbildung 5.2: Ablauf des Programms

funktion bewertet. Die darauffolgende Umweltselektion dient der Verkleinerung der aktuellen Population inklusive der Kinder auf die ursprüngliche Populationsgröße. Auf die einzelnen Schritte gehe ich in den folgenden Unterkapiteln nun genauer ein.

5.1 Kodierung der Lösungskandidaten

Zur Lösung des Problems benötigt man eine einheitliche Repräsentation der Lösungskandidaten. In meiner Implementierung werden Individuen als ein Vektor fester Länge mit ganzzahligen Werten kodiert.
Die Länge des Vektors, die Chromosomlänge, muss etwas länger sein als die vorgegebene Route, um von dieser abweichen zu können und somit eine Suche im Lösungsraum zu ermöglichen. Allerdings sollte sie nicht zu lang sein, um den Suchraum und die Berechnungszeit nicht unnötig zu vergrößern. Deswegen habe ich die Chromosomlänge in Relation zur Routelänge und der Anzahl an restlichen Knoten im Graph wie in Formel 5.1 gewählt. Die Addition von zwei dient der Reservierung von Stellen für die Ladeempfehlung. Beim Beispiel-Anwendungsfall enthält die Route sechs Knoten und der Graph 45 weitere. Die Chromosomlänge ist somit $\lceil 2 + 6 + \frac{1}{4} * 45 \rceil = \lceil 19,25 \rceil = 20$.

$$Chromosomlaenge = \left\lceil 2 + Routenlaenge + \frac{1}{4}\, Knoten_{Rest} \right\rceil \tag{5.1}$$

Im Folgenden erkläre ich die Bedeutung der Positionen der Chromosome an dem Beispiel-Lösungskandidaten

[35 1 2 51 35 46 31 49 29 37 6 6 6 6 6 6 6 6 6 6],

der in Abbildung 5.3 dargestellt ist. Der erste Wert in der Repräsentation gibt den Index des Knotens an, an dem geladen werden soll. Er darf nicht dem Index der aktuellen Position des Fahrzeugs oder dem des Endknotens entsprechen, da diese laut Problemdefinition keine Ladestationen sind. An zweiter Stelle steht eine Zahl von null bis vier, die kodiert, welche Ladetechnologie an der empfohlenen Ladestation genutzt wird. Eine Übersicht zur Bedeutung der fünf Zahlen wurde in Tabelle 5.1 erstellt. Soll nicht geladen werden, sind die ersten beiden Stellen null. Beim Beispiel-Lösungskandidaten wird also an Knoten 35 mit einer Abrechnung in Kilowattstunden normalgeladen.
Der Rest des Vektors wird für den Vorschlag der alternativen Route genutzt und besteht aus einer Folge von Knoten-Indices. Der Startknoten aller Lösungskandidaten ist derselbe, da alle von der aktuellen Position des Elektrofahrzeugs

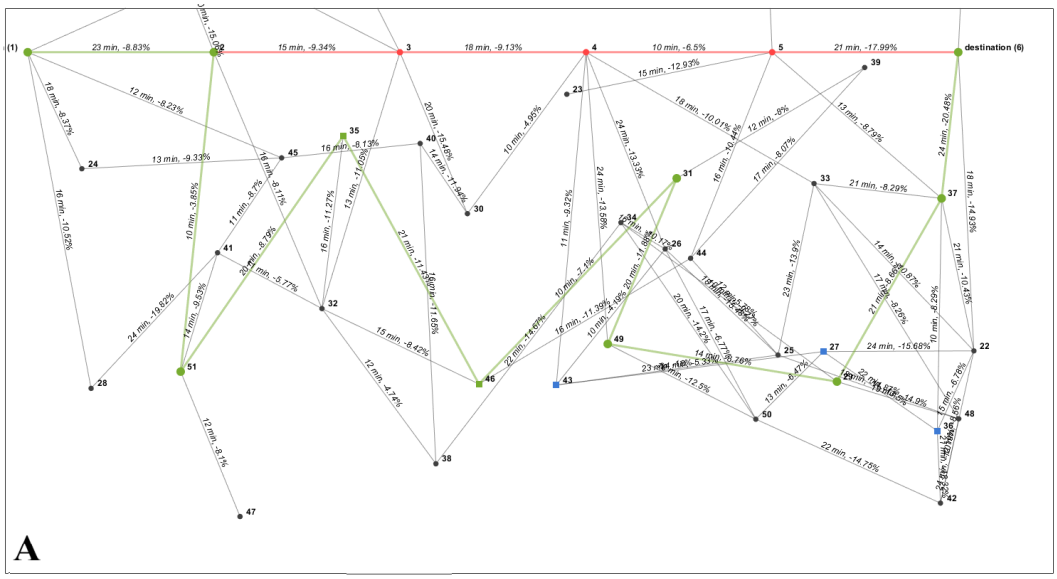
Gen	Allel
0	nicht laden
1	Normalladen pro kWh
2	Schnellladen pro kWh
3	Normalladen mit Pauschale
4	Schnellladen mit Pauschale

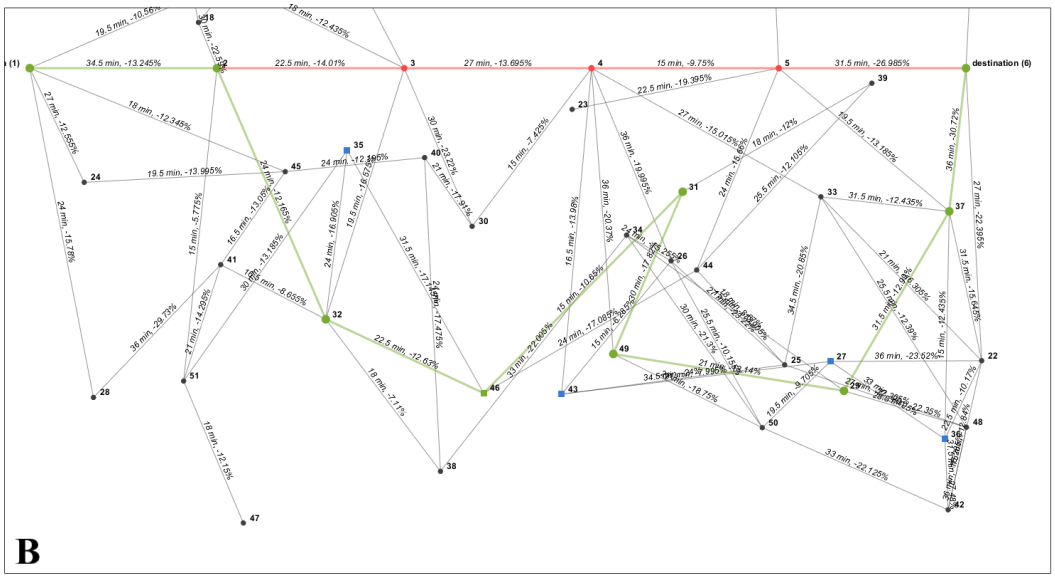
Tabelle 5.1: Kodierung der Lademöglichkeiten

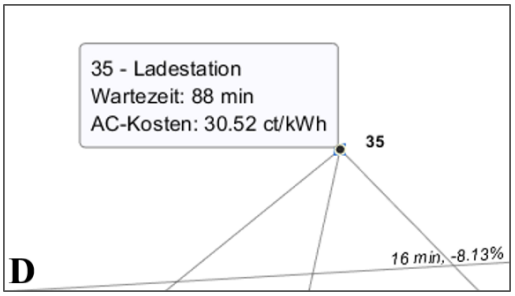
starten. Deshalb wird dieser in der Repräsentation der Lösungskandidaten ausgelassen. In meiner Implementierung hat die aktuelle Position immer den Index 1. Die im Beispiel-Lösungskandidat vorgeschlagene Alternativroute verläuft somit von Knoten 1 über Knoten 2, 51, 35, 46, 31, 49, 29 und 37 zum Endknoten, in diesem Fall Knoten 6. Der letzte Knoten muss immer dem Ziel des Fahrzeugs entsprechen. Die ursprüngliche Route des Beispiel-Anwendungsfalls sind die Knoten mit Indices 1 bis 6 in aufsteigender Reihenfolge gewesen.
Falls eine Alternativroute kürzer als die zu Verfügung stehende Länge ist, werden die übrig gebliebenen Stellen mit dem Index des Endknotens aufgefüllt. Dadurch werden beispielsweise Fehler bei der Rekombination vermieden. Die Alternativroute ohne zusätzliche Endknoten soll von hier an Routenanteil genannt werden. Für den Beispiel-Lösungskandidaten wäre dieser [2 51 35 46 31 49 29 37 6].
Bei der Interpretation eines Lösungskandidaten ist außerdem zu beachten, dass auch dann nur einmal geladen wird, wenn der Knoten mit der empfohlenen Ladestation mehrmals im Routenanteil vorkommt. In diesem Fall wird beim ersten Besuch des Knotens geladen.

5.2 Initialisierung

Bei der Initialisierung wird eine anfängliche Population mit genügend Individuen für die gewünschte Populationsgröße erstellt. Letztere wurde für diese Arbeit auf 100 gesetzt, weil dies laut [18] der für MOEA am häufigsten gewählte Wert ist und es beim Ausführen des Algorithmus damit keine offensichtlichen Probleme gab. Eine analytische Bestimmung der optimalen Populationsgröße kann in Zukunft noch durchgeführt werden.
In dieser Arbeit werden Alternativrouten für die initiale Population durch ein







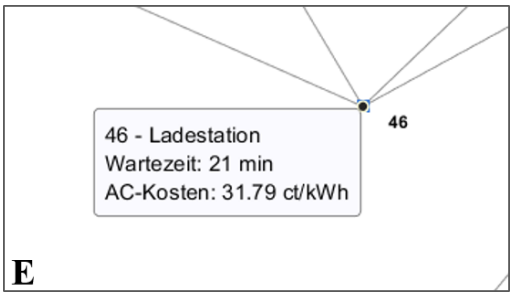


Abbildung 5.3: Verlauf der Lösungskandidaten [35 1 2 51 35 46 31 49 29 37 6] (A) und [46 1 2 32 46 31 49 29 37 6] (B) auf der Karte, deren Legende (C) und Details zu den beiden verwendeten Ladestationen (D,E) bei mittlerer Verkehrsdichte

zufälliges Durchlaufen des Graphen erstellt. Begonnen wird an der aktuellen Position des Elektrofahrzeugs. Dann wird einer der Nachbarknoten ausgewählt und besucht. Dies wird solange wiederholt, bis der Endknoten erreicht oder die Länge der Kodierung erschöpft wurde.
Die Wahrscheinlichkeit, dass ein Nachbar genommen wird, ist invers zur Anzahl der Besuche dieses Knotens. Für einen Nachbar $n \in N$ berechnet sich die Auswahlwahrscheinlichkeit $P(n)$, wie in Formel 5.2 angegeben. Sie basiert darauf, wie oft n schon besucht wurde, $visits(n)$ und was die maximale Anzahl an Besuchen $maxVisit$ von allen Nachbarn ist. Bei der Anzahl der Besuche wird ebenfalls der Startknoten beachtet, der nicht in Kodierung enthalten ist. Die Addition von eins habe ich bei $p(n)$ in Formel 5.2 hinzugefügt, damit die Wahrscheinlichkeiten nicht null sind, wenn alle Knoten gleich oft besucht wurden. $P(n)$ normiert die Summe der Wahrscheinlichkeiten dann auf eins.

$$\begin{aligned} p(n) &= maxVisit^2 + 1 - visits(i)^2 && \forall n \in N \\ P(n) &= \frac{p(n)}{\sum\limits_{i \in N} p(i)} && \forall n \in N \end{aligned} \tag{5.2}$$

Wenn man für den Beispiel-Anwendungsfall ein Individuum erstellt, fängt man an der aktuellen Position, Knoten 1, an, welche nicht im Chromosom gespeichert ist. Als nächster Knoten kommen 2, 7, 24, 28 und 45 infrage, weil sie die Nachbarn N von 1 sind. Da noch kein weiterer Knoten besucht wurde, sind $visits(n)$ und demnach auch $maxVisit$ null. Die Auswahl wird dadurch zufällig mit einer gleichen Wahrscheinlichkeit von $\frac{1}{5}$ getroffen. Angenommen, Knoten 7 wurde ausgewählt. Seine Nachbarn N setzen sich aus Knoten 1, 3, 13, 18 und 20 zusammen. Für $visits(n)$ ergeben sich dann die Werte in Tabelle 5.2 und für $maxVisit$ eins. Wenn n eins ist, dann ist $p(1) = 1^2+1-1^2 = 1$. Für alle anderen n beläuft sich $p(n) = 1^2+1-0^2 = 2$ auf zwei. Mit $\sum\limits_{i \in N} p(i) = 1+4*2 = 9$ betragen die jeweiligen $P(n)$ die in Tabelle 5.2 aufgeführten Werte. Mithilfe dieser Wahrscheinlichkeiten wird der nächste Knoten ausgewählt. Das vorgeschlagene Verfahren wird solange fortgesetzt, bis die Route des Lösungskandidaten beendet ist oder die maximale Länge der Kodierung erreicht wurde.
Eine weitere Besonderheit der Formel 5.2 ist, dass die Besuche quadriert werden. Zuerst war der Zusammenhang zwischen der Auswahlwahrscheinlichkeit und den Besuchen linear. Dadurch wurden jedoch zu häufig schon besuchte Knoten genommen oder zwischen zwei Knoten hin und her gewechselt.
Weiterhin sollen möglichst viele gültige Lösungen während der Initialisierung erstellt werden. Stehen für die Kodierung der Route nur noch maximal 20 %

n	1	3	13	18	20
visits(n)	1	0	0	0	0
p(n)	1	2	2	2	2
P(n)	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{2}{9}$	$\frac{2}{9}$	$\frac{2}{9}$

Tabelle 5.2: Besuche $visits(n)$, Auswahlwahrscheinlichkeit $p(n)$ und normierte Auswahlwahrscheinlichkeit $P(n)$ der Nachbarn für Beispiel-Lösungskandidaten mit angefangenem Routenanteil [7]

der Chromosomlänge zur Verfügung und ist der Zielknoten einer der Nachbarn des aktuell besuchten Knotens, wird die Auswahlwahrscheinlichkeit des Zielknotens deshalb auf eins und die der anderen Nachbarn auf null gesetzt.
Nachdem die Route feststeht, wird zufällig eine Ladeempfehlung aus "nicht laden" und den auf der Route passierten Ladestationen mit gleicher Wahrscheinlichkeit ausgewählt. Das bedeutet, dass nicht geladen wird, wenn kein Knoten mit einer Ladestation besucht wurde. Wird nicht geladen, bleibt auch die zweite Stelle des Chromosoms für die Ladetechnologie null. Andernfalls wird zufällig eine von den an der ausgesuchten Ladestation vorhandenen Ladetechnologien genommen. Sei der Routenanteil des oben angefangenen Lösungskandidaten als [7 20 13 11 14 10 17 21 6] vervollständigt worden. Knoten 10 und 14 sind im Beispiel-Anwendungsfall Ladestationen. Somit wird aus diesen beiden Optionen und "nicht laden" ein Wert für die erste Stelle der Kodierung ausgewählt. Sei dies Knoten 14 gewesen. Diese Ladestation bietet Normalladen in Kilowattstunden und als Pauschale abgerechnet an. Davon sei Normalladen pro Kilowattstunde genommen und wie in Tabelle 5.1 beschrieben kodiert worden. Dann würde der finale Beispiel-Lösungskandidat aus der Initialisierung [14 1 7 20 13 11 14 10 17 21 6 6 6 6 6 6 6 6 6] lauten.

5.3 Fitnessfunktion

Bei evolutionären Algorithmen wird eine Fitnessfunktion zur Bewertung von Individuen benötigt, um diese vergleichen zu können. Die Fitness ist in dieser Arbeit ein Quadrupel aus den Funktionswerten der vier Zielfunktionen. Die Reihenfolge der Zielfunktionen ist dabei immer noch dieselbe wie in Kapitel 3: Energieverbrauch in Prozent SoC, Ladekosten in Euro, Reisezeit in Minuten und Strafe für den SoH-Rückgang.

Ein Lösungskandidat eins ([35 1 2 51 35 46 31 49 29 37 6 6 6 6 6 6 6 6 6 6]) hat beispielsweise die Fitness [104.985 9.76 303.25 0]. Ein Lösungskandidat zwei ([46 1 2 32 46 31 49 29 37 6 6 6 6 6 6 6 6 6 6 6]) wurde mit [110.68 10.76 206.73 0] bewertet. Der Energieverbrauch beider Lösungskandidaten darf mehr als 100 % betragen, weil auf der Fahrt geladen wird. Der SoH-Rückgang ist in beiden Fällen null, da der minimale Ladezustand nicht unterschritten und keine Schnellladung genutzt wurde.
Die Berechnung der Fitnesswerte für jedes Individuum wird mit den in Kapitel 4 beschriebenen Formeln durchgeführt. Dazu wird die empfohlene Alternativroute durchlaufen, an jedem Knoten der SoC und die Zeit aktualisiert und wie vorgeschrieben geladen. Falls eine Lösung ungültig ist, wird ihr eine maximal schlechte Fitness von [100000 100000 100000 100000] zugeteilt. Eine Lösung verlässt den Suchraum, wenn eine Kante in der Route nicht existiert, der Endknoten nicht erreicht wird, der SoC null Prozent oder weniger beträgt oder die empfohlene Ladestation nicht im Routenanteil enthalten ist.
Zum besseren Vergleich der Funktionswerte verschiedener Lösungen, vor allem für spätere Nutzer, wird in Kapitel 5.8 eine Visualisierung vorgeschlagen.

5.4 Selektionsmechanismen

Eine Selektion von Individuen wird beim Algorithmus für die Paarungsselektion und die Umweltselektion benötigt. Erstere wählt die Eltern für die Rekombination und die Mutation aus. NSGA-II mit kontrolliertem Elitismus nutzt dafür die Crowded Tournament Selection. Durch die Umweltselektion werden diejenigen Individuen ausgesucht, die in die nächste Generation übernommen werden. Die Verfahren von NSGA-II mit kontrolliertem Elitismus verwenden dafür den Rang, die Crowding distance und eine geometrische Verteilung. [18]
Die genannten Methoden sind bereits als Standard der *gamultiobj()*-Funktion in MATLAB enthalten und auf das Problem dieser Arbeit anwendbar. Deswegen wurde für die Selektionsmechanismen nichts selbst implementiert.
Für die Methoden wird die Crowding distance benötigt, die auf den Funktionswerten der Zielfunktionen wie in Formel 5.3 angegeben berechnet wird [61]. Die Crowding distance ist definiert als die Summe der relativen Distanzen der beiden nächsten Nachbarn eines Lösungskandidaten i entlang aller Dimensionen $m = 1, ..., 4$. Letztere sind durch die, in diesem Fall vier, Zielfunktionen gegeben. Der Zweck dieses Maßes ist eine Abschätzung der Dichte von Lösungs-

kandidaten um einen bestimmten Lösungskandidaten i. Die Voraussetzung zur Berechnung ist, das der Rang aller Individuen bekannt ist. [18]

$$crowding\_distance(i) = \sum_{m} \frac{f_m(i+1) - f_m(i-1)}{f_m^{max} - f_m^{min}} \tag{5.3}$$

Der Rang eines Individuums ist eins mehr als die Anzahl an Lösungskandidaten, die das Individuum dominieren. Ein Lösungskandidat x dominiert einen Lösungskandidaten y, wenn x für alle Zielfunktionen mindestens genauso gut wie y und für wenigstens eine Zielfunktion besser ist. Je nachdem, ob eine Zielfunktion minimiert oder maximiert wird, bedeutet besser kleiner respektive größer. [18]
Die Crowding distance wird nur zwischen Individuen derselben Front, das heißt desselben Rangs, berechnet. Dazu werden die Individuen für jede Dimension $m = 1, ..., 4$ separat nach aufsteigenden Funktionswerten sortiert. Die Lösungskandidaten mit dem kleinsten und mit dem größten Funktionswert der Dimensionen, f_m^{min} und f_m^{max}, erhalten eine Crowding distance von unendlich. Anders gesagt, ist die Crowding distance die Summe über alle Dimensionen von den Differenzen der Funktionswerte der Nachbarn von i geteilt durch die maximale Differenz von Funktionswerten. [18]

5.4.1 Paarungsselektion

Bei der Paarungsselektion werden Eltern zur Rekombination und Mutation mithilfe der Crowded tournament selection gefunden. Dies ist eine binäre Turnierselektion, die den Crowded tournament selection operator nutzt. Letzterer dient zum Vergleich von zwei Lösungskandidaten. Er besagt, dass eine Lösungskandidat ein Turnier mit einem anderen Lösungskandidaten gewinnt, falls

1. der Lösungskandidat einen besseren, also kleineren, Rang hat

oder

2. sie denselben Rang haben und der Lösungskandidat die bessere, also größere, Crowding distance hat.

Die Eltern werden durch mehrere Turniere zwischen je zwei zufällig ausgewählten Lösungskandidaten ausgesucht. Das jeweils bessere Individuum wird der Menge der Eltern hinzugefügt. [18]

5.4.2 Umweltselektion

Mithilfe der Umweltselektion wird die erweiterte Population auf die Populationsgröße wie zum Anfang der Iteration begrenzt, um zu verhindern, dass die Populationsgröße exponentiell wächst. Die erweiterte Population setzt sich aus den Individuen der aktuellen Generation und deren Kindern zusammen. [18] Sie ist bei *gamultiobj()* doppelt so groß wie die Population zu Beginn der Iteration.
Die Voraussetzung für die Berechnung der Umweltselektion von NSGA-II mit kontrolliertem Elitismus ist, dass die Fitnesswerte aller Individuen berechnet wurden. Dann werden der Rang und die Crowding distance der erweiterten Population ermittelt [61]. Diese Variante von NSGA-II begrenzt die Übernahme elitärer Individuen zum Erhalt der Diversität der Population, sowohl innerhalb der Fronten als auch lateral über alle Fronten hinweg. Dadurch soll eine bessere Konvergenz zur optimalen Pareto-Front erzielt werden. Es werden also Individuen aller Fronten übernommen. Die meisten Individuen stammen aus der ersten Front. Aus denen danach kommen exponentiell immer weniger. Werden nicht alle Individuen einer Front übernommen, werden die erhalten bleibenden mit Crowded tournament selection ausgesucht. [18] Mithilfe des Parameters *ParetoFraction* kann man bestimmen, wie viel Prozent der Individuen des ersten Rangs in die nächste Generation gelangen. Die Standardeinstellung sind 35 %. [60] Hier besteht die Möglichkeit, diesen Wert für das Problem dieser Arbeit zu optimieren.

5.5 Rekombination

Bei der Rekombination werden zwei Elternteile zu einem Kind kombiniert. Dafür habe ich einen 1-Punkt-Crossover mit Reparaturmechanismus konzipiert. Die beiden Elternteile werden an einer ausgewählten Stelle jeweils geteilt. Das Kind setzt sich aus dem vorderen Stück des ersten Elternteils und dem hinteren des zweiten Elternteils zusammen. Es wurde kein Verfahren mit mehr Schnittstellen gewählt, da dadurch zu viele Reparaturen benötigt werden würden, was zu viel Rechenzeit beansprucht.
Die Anzahl der erstellten Kinder ist gleich der gewählten Populationsgröße. Standardmäßig werden 80 % davon durch Rekombination gebildet. Eine Optiminierung diese Prozentsatzes wurde aus zeitlichen Gründen nicht durchgeführt.

Zur Veranschaulichung der Vorgehensweise sollen die zwei folgenden Elternteile aus dem Beispiel-Anwendungsfall dienen:

Elternteil I) [14 1 7 20 13 11 14 5 6 6 6 6 6 6 6 6 6 6 6 6]

Elternteil II) [10 2 2 3 7 13 19 17 10 17 21 6 6 6 6 6 6 6 6 6]

Die Schnittstellen zur Rekombination werden im Folgenden mit "|" markiert. Das Verfahren in dieser Arbeit sucht den Schnittpunkt für das Crossover innerhalb des Routenanteils des ersten Elternteils. Der Grund für das erste Elternteil ist, dass eine Rekombination ohne Auswirkung vermieden werden soll. Falls das erste Elternteil kürzer als das zweite ist und der Schnittpunkt außerhalb seines Routenanteils liegt, würde das erste Elternteil inklusive auffüllender Endknoten bis zum Schnittpunkt übernommen werden. Danach würde der Routenanteil des zweiten Elternteils und dessen Endknoten ab dem Schnittpunkt übernommen werden. Für eine Rekombination von Elternteil I und II nach der neunten Stelle hieße das beispielsweise, dass [14 1 7 20 13 11 14 5 6 | 17 21 6 6 6 6 6 6 6 6 6] als Kind entstehen würde. Diese Route ist ungültig, da nach Erreichen des Endknoten 6 mit weiteren Knoten aufgefüllt werden müsste. Zur Reparatur würden die Knoten nach dem Endknoten durch den Endknoten ersetzt werden, sodass nach der Reparatur wieder Elternteil I entsteht und faktisch keine Rekombination stattgefunden hat.

Falls das erste Elternteil länger ist, wird im schlimmsten Fall der Routenanteil des zweiten Chromosoms durch den Schnittpunkt verkürzt, wodurch eine Reparatur benötigt wird. Rekombiniert man Elternteil II mit I nach der neunten Stelle, entsteht [10 2 2 3 7 13 19 17 10 | 6 6 6 6 6 6 6 6 6 6 6]. Dadurch, dass Knoten 17 und 21 von Elternteil II wegfallen, existiert keine Kante an der Schnittstelle mehr, sodass das Individuum repariert werden muss.

Sind beide Elternteile gleich lang, liegt der Schnittpunkt logischerweise innerhalb der Routenanteile beider Elternteile. Aus den aufgeführten Gründen ist es somit sinnvoll, den Schnittpunkt innerhalb des Routenanteils des ersten Elternteils zu legen.

Der Grund, warum der Schnittpunkt im Routenanteil liegen sollte, ist, dass bei einer Rekombination an der Ladeempfehlung diese in fast allen Fällen repariert werden muss und dadurch wieder eins der Elternteile entsteht. Ein Problem ist, dass die Ladestation nach einer solchen Rekombination häufig nicht mehr in der Route enthalten ist. Ein mögliche Korrektur wäre, eine Ladestation in der rekombinierten Route zu suchen, aber dann ist die Wahrscheinlichkeit

hoch, dass das zweite Elternteil entsteht. Man könnte die Ladeempfehlung immer auf "nicht laden" setzen, dies würde dann aber zu häufig vorkommen oder oft ungültige Lösungen produzieren. Ein andere Möglichkeit wäre, eine Ladestation an geeigneter Stelle in der Route einzufügen. Das Finden einer solchen Stelle ist jedoch mit zu hohem Rechenaufwand verbunden.
Außerdem ist die Ladetechnologie von einem Elternteil nicht unbedingt mit der Ladestation vom anderen Elternteil kompatibel. Eine kostengünstige Reparatur wäre, zufällig eine von den an der gewählten Ladestation vorhandenen Ladetechnologien zu nehmen. Da allerdings in den wenigsten Fällen alle fünf Ladetechnologien zur Auswahl stehen, ist auch hier die Wahrscheinlichkeit hoch, dass dieselbe Kombination wie bei den Eltern entsteht.
Eine Rekombination der Beispiel-Eltern bei der Ladeempfehlung würde nach der erste Stelle stattfinden und [14 | 2 2 3 7 13 19 17 10 17 21 6 6 6 6 6 6 6 6 6] als Kind produzieren. In diesem Fall wäre die Ladestation an Knoten 14 nicht mehr in der Route enthalten. Bei einer zufälligen Auswahl aus "nicht laden" und in der Route vorhandenen Ladestationen (Knoten 10), gibt es eine Chance von 50 %, dass wieder Elternteil II entsteht. Wird "nicht laden" genommen, ist zusätzlich die Ladetechnologie nicht mehr kompatibel.
Bei einem Crossover innerhalb des Routenanteils des ersten Elternteils kann trotzdem immer noch das Problem auftreten, dass an der Schnittstelle keine Kante existiert. Deshalb habe ich als Reparaturmechanismus eingebaut, dass in einem solchen Fall versucht wird, einen Knoten einzufügen, der die beiden problematischen Knoten verbindet. Dies funktioniert nicht immer, aber ein Versuch mit mehr Knoten zu reparieren, wäre zu aufwendig. Des Weiteren ist das Einfügen eines Knotens nur dann möglich, wenn die maximale Chromosomlänge in der Kodierung noch nicht ausgeschöpft wurde. Das heißt, es muss sowohl an der letzten als auch an der vorletzten Stelle ein Endknoten stehen. Der Mechanismus versucht außerdem, nicht den Endknoten einzufügen, um bei der Reparatur möglichst wenig am Kind zu ändern. Wird trotzdem mit dem Endknoten repariert, werden die nachfolgenden Knoten durch diesen ersetzt.
Führt man beispielsweise eine Rekombination von Elternteil I und II nach der siebten Stelle durch, lautet das Kind [14 1 7 20 13 11 14 | 17 10 17 21 6 6 6 6 6 6 6 6 6]. Zwischen Knoten 14 und 17 gibt es keine Kante, weswegen mit Knoten 10 oder 11 repariert wird. Diese Knoten verbinden die Schnittstelle der Elternteile. Nach dem Einfügen wird ein Endknoten gelöscht, um die vorgegebene Chromosomlänge

einzuhalten.
Im Anschluss an die eventuell notwendige Reparatur wird außerdem immer geprüft, ob die Ladestation, die vom ersten Elternteil übernommen wurde, noch in der Route vorhanden ist. Beim Beispiel ist dies für Knoten 14 der Fall, sodass keine Reparatur ausgeführt werden muss. Andernfalls würde zufällig "nicht laden" oder eine der in der Route enthaltenen Ladestationen mit dazu passender Ladetechnologie ausgewählt werden.

5.6 Mutation

Die Mutation ist eine zufällige kleine Veränderung eines einzelnen Gens. Durch die Mutation wird der fehlende Anteil an Kindern, der nicht durch Rekombination entstanden ist, für die erweiterte Population erstellt. Das vorgeschlagene Verfahren soll an dem Beispiel-Individuum [10 2 2 3 7 13 19 17 10 17 21 6 6 6 6 6 6 6 6 6] erklärt werden. Das Gen, welches mutiert wird, wird zufällig ausgewählt. Dieses Gen nur leicht zu verändern, ist über den Index nicht möglich. Der Grund dafür ist, dass Knoten mit ähnlichen Indices nicht unbedingt nah beieinander liegen und nicht immer Nachbarn sind.
Das Verfahren in dieser Arbeit unterscheidet zwischen der Mutation der Ladeempfehlung und der Route. Wenn bei Ersterem die Ladestation zur Mutation ausgewählt wurde, wird zufällig eine andere Ladestation inklusive einer passenden Ladetechnologie aus der Route ausgesucht. Wenn die Ladetechnologie verändert werden soll, wird eine andere Ladetechnologie von der Ladestation des Individuums genommen. Die Verwendung von "nicht laden" wurde hier ausgeschlossen, da Routen mit Ladeempfehlung dadurch oft ungültig werden. Der Erfolg der Mutationen der Ladeempfehlung ist jedoch unwahrscheinlich, da meist nur eine Ladestation auf der Route liegt und an dieser selten mehr als eine Ladetechnologie vorhanden ist. Beim Beispiel-Individuum ist keine Veränderung der Ladestation möglich, da keine weitere auf der Route liegt. Die Mutation der Ladetechnologie ist in diesem Fall jedoch möglich, da Knoten 10 neben dem Schnellladen pro kWh auch Normalladen in kWh anbietet. An der zweiten Position des Individuums würde statt einer Zwei dann eine Eins stehen. Falls die zufällig ausgewählte Stelle in der Ladeempfehlung liegt und eine Mutation nicht gelingt, wird versucht eine beliebige Position innerhalb des Routenanteils zu mutieren.
Bei der Mutation des Routenanteils wird Löschen, Tauschen und Einfügen von

Knoten in dieser Reihenfolge ausprobiert. Falls eine Variante erfolgreich ist, wird abgebrochen. Falls keine funktioniert, wird das Individuum zwangsweise unverändert zu den Kindern hinzugefügt. Die Reihenfolge wurde gewählt, um eine möglichst große Verbesserung zu erzielen. Es wird also beispielsweise angenommen, dass durch das Löschen eines Knotens im Allgemeinen eine kürzere Route mit einer kleineren Reisezeit und einem geringeren Energieverbrauch einhergeht.
Zur Mutation durch Löschen eines Knotens wird das ausgewählte Gen entfernt, falls der Knoten davor und der Knoten danach Nachbarn sind. Im Beispiel kann bei einer Mutation der zehnten Stelle, der zweiten 17, Knoten 10 und 21 durch eine Kante verbunden werden. Deswegen kann die 17 gelöscht und das Ende des Individuums mit einem Endknoten mehr aufgefüllt werden. Dabei wird nicht überprüft, ob das einzige Vorkommen der Ladestation gelöscht wird. Falls dies passiert, wird der Lösungskandidat bei der Bewertung durch die Fitnessfunktion als ungültig mit einer schlechten Fitness bewertet.
Beim Tauschen wird ein Knoten gesucht, der Nachbar vom Knoten vor der Mutationstelle und von dem danach und nicht der Knoten an der Mutationsstelle ist. Das gewählte Gen wird durch einen der gefundenen Knoten ausgewechselt. Im Beispiel-Individuum können an der vierten Stelle Knoten 2 und 7 auch über Knoten 1 oder 18 verbunden werden. Einer der beiden Knoten wird zufällig ausgewählt und ersetzt Knoten 3.
Wenn zur Mutation ein Knoten eingefügt werden soll, wird ein Knoten gesucht, der das ausgewählte Gen und den Knoten danach verbindet. Beispielsweise kann nach der fünften Stelle im Beispiel-Individuum Knoten 20 eingefügt werden, da er Nachbar von Knoten 7 und 13 ist. Außerdem ist die vorletzte Stelle des Individuums der Endknoten, sodass in der Kodierung noch Platz für mindestens einen weiteren Knoten ist. Nach dem Einfügen wird mit einem Endknoten weniger aufgefüllt. Falls der Endknoten zur Mutation eingefügt wird, werden die Knoten danach durch den Endknoten ersetzt.

5.7 Abbruchkriterien

Für die Abbruchkriterien wurden die als geeignet bewerteten Voreinstellungen der MATLAB-Funktion genutzt. Das Ausprobieren anderer Abbruchkriterien oder die Optimierung der aktuellen bleibt für zukünftige Arbeiten. Der Algorithmus endet, wenn eins der beiden Abbruchkriterien erfüllt ist.
Das erste Kriterium ist, dass die maximale Anzahl an Generationen erreicht

wurde. Der Wert dafür ist standardmäßig 200 multipliziert mit der Chromosomlänge. [60] Für den Beispiel-Anwendungsfall wären das $200 * 20 = 4000$ Generationen. Dieses Abbruchkriterium wurde während meiner Tests jedoch nie erreicht.
Bei dem zweiten Kriterium wird abgebrochen, wenn das geometrisches Mittel der relativen Veränderung des Spreads über die letzten 100 Generationen kleiner als 0,0001 und der Spread in der aktuellen Generation kleiner als der durchschnittliche Spread der letzten 100 Generationen ist. Der Spread wird für eine Generation wie in Formel 5.4 angegeben berechnet. [61]

$$spread = \frac{\mu + \sigma}{\mu + Qd} \tag{5.4}$$

Die Zeichen in der Formel bedeuten: Q die Anzahl der Individuen auf der Pareto-Front mit endlicher Crowding distance, σ die Standardabweichung der Crowding distance dieser Q Individuen, d die durchschnittliche Crowding distance der Q Individuen und μ die Summe über die vier Zielfunktionen von der Norm der Differenz zwischen dem minimalen Wert einer Zielfunktion in der aktuellen Generation und dem Minimum der Zielfunktion in der vorherigen Generation. Kleine Werte und somit der Abbruch des vorgeschlagenen Algorithmus treten auf, wenn sich die Funktionswerte zwischen den Generationen nur geringfügig ändern (kleines μ) und die Punkte auf der Pareto-Front gleichmäßig verteilt sind (kleines σ). [61]
Eine weitere Möglichkeit, die *gamultiobj()* als Abbruchkriterium bietet, ist ein Zeitlimit zu setzen [61]. Dies Option wurde jedoch nicht genutzt.

5.8 Pareto-optimale Lösungen

Nach Abbruch des Algorithmus müssen die gefundenen Lösungen in geeigneter Darstellung ausgegeben werden. Die Lösungen sind pareto-optimal, das heißt, sie werden von keiner anderen Lösung dominiert [18]. Die Berechnung der Dominanz und welche Lösungen pareto-optimal sind, ist in der MATLAB-Funktion bereits integriert.
Durch den metaheuristischen Ansatz wird eine Menge von pareto-optimalen Lösungen ausgegeben. Diese bilden die Pareto-Front [18]. Die Lösungen sind jeweils für verschiedene Zielfunktionen am besten. Im Beispiel-Anwendungsfall gibt es zwei pareto-optimale Lösungen, die auch in Abbildung 5.3 zu erkennen sind:

I: [35 1 2 51 35 46 31 49 29 37 6 ...] mit der Fitness [104.985 9.76 303.25 0]
II: [46 1 2 32 46 31 49 29 37 6 ...] mit der Fitness [110.68 10.76 206.73 0]
Von Lösung I zu Lösung II lässt sich zwar die dritte Zielfunktion, die Reisezeit, verbessern, dafür verschlechtern sich allerdings die ersten beiden Zielfunktionen, der Energieverbrauch und die Ladekosten.
Für vier Zielfunktionen lässt sich die Pareto-Front schlecht visualisieren, aber drei Achsen können noch interpretiert werden. Deshalb ist eine Variante zur Anzeige der Lösungen, die Pareto-Front mit jeder Kombination von drei der vier Zielfunktionen als Achsen auszugeben. Dies ist in Abbildung 5.4 veranschaulicht. Jedes Diagramm darin ist im Original interaktiv, sodass die Pareto-Front aus unterschiedlichen Sichten betrachtet werden kann. Nachteile sind, dass die Position und die Anzahl der Achsen nicht verändert werden können und die Lösungen untereinander schlecht vergleichbar sind.
Zur Umgehen des ersten Problems habe ich eine grafische Benutzeroberfläche programmiert, die in Abbildung 5.5 dargestellt ist. Der Nutzer kann die Pareto-Front damit nach seinen eigenen Vorstellungen anzeigen lassen. Dafür muss er eine bis drei Zielfunktionen als Achsen auswählen, auf denen die Fitnesswerte abgetragen werden. Durch diese Darstellung lässt sich sowohl die Anzahl als auch die Position der Achsen beeinflussen. Allerdings ist in dieser Darstellung ein Vergleich der Lösungen ebenfalls schlecht möglich.
Deshalb schlage ich für den Fahrer des Elektrofahrzeugs die Visualisierung mit einem Spinnennetzdiagramm wie in Abbildung 5.6 vor. Bei dieser Variante ist leicht erkennbar, dass Lösung I bessere Ladekosten und einen besseren Energieverbrauch als Lösung II hat, der SoH-Rückgang bei beiden Lösungen gleich ist und Lösung II nur bei der Reisezeit besser abschneidet.
Der Nutzer kann nun nach seinen eigenen Prioritäten entscheiden, welche der empfohlenen Alternativrouten er nimmt. Die Güte der ihm zur Verfügung stehenden Ladeempfehlungen werden im folgenden Kapitel evaluiert.

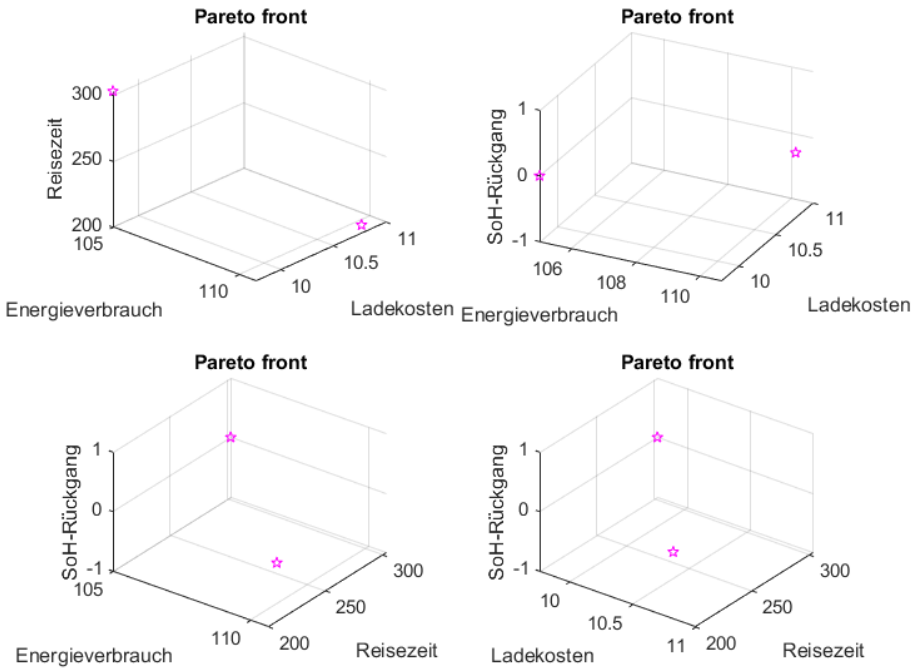


Abbildung 5.4: Pareto-Front für alle Kombination mit drei von vier Zielfunktionen

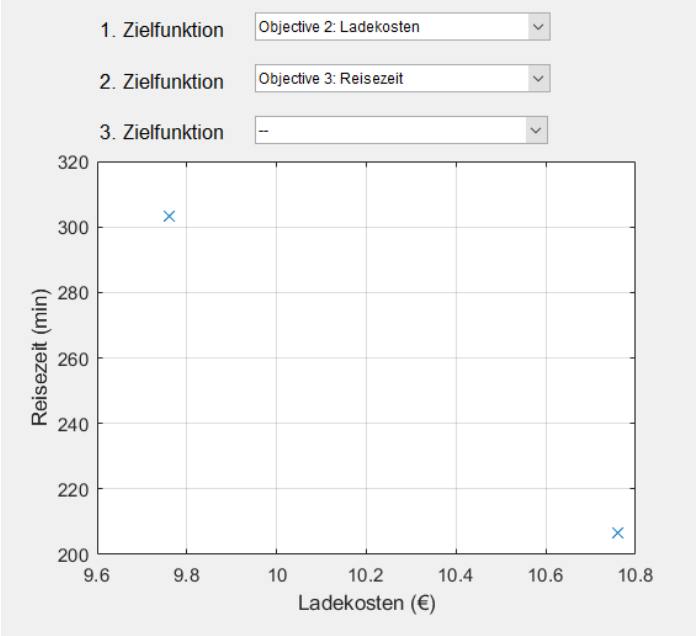


Abbildung 5.5: Benutzeroberfläche zur Darstellung der Pareto-Front mit einer bis drei beliebigen Zielfunktionen

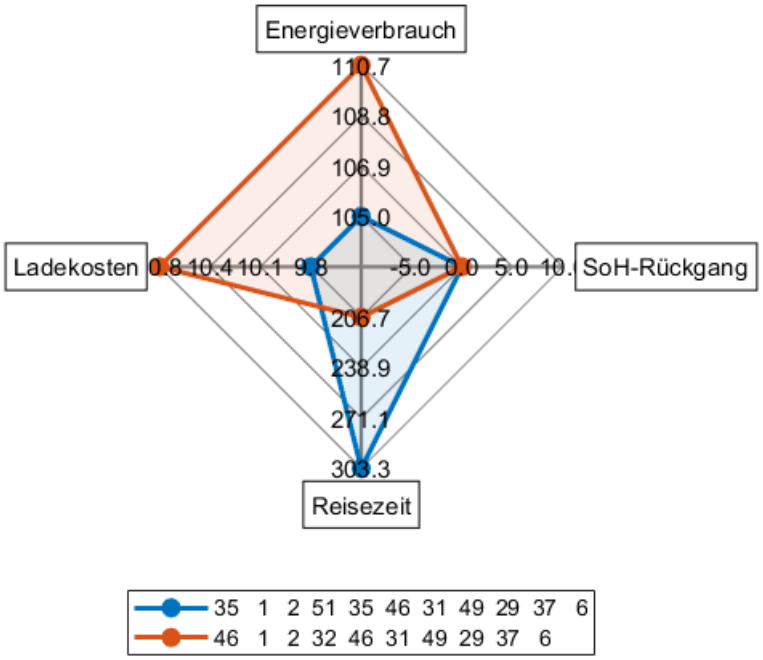


Abbildung 5.6: Darstellung der Funktionswerte der pareto-optimalen Lösungen in einem Spinnennetzdiagramm

6 Evaluierung

Mithilfe von Tests sollte die Eignung des Systems zum Einsatz in der Praxis bewertet werden. Entscheidend dafür sind die Güte der Lösungen und die Rechenzeit des Algorithmus. Für die Optimierung eines Anwendungsfalls wird eine Berechnungsdauer von maximal 60 Sekunden als Eignungskriterium eingeführt. So könnte das System aller Minute überprüfen, ob andere pareto-optimale oder sogar eine bessere Alternativroute zur Verfügung stehen. Da es für mein konkretes Problem keine Benchmark gibt und die für das EVRP vorhandene Benchmarks als ungeeignet bewertet wurden, habe ich zum Testen Anwendungsfälle mit einem selbst geschriebenen Algorithmus generiert. Dadurch soll festgestellt werden, ob die vorgeschlagene Lösungsmethode in der Lage ist, neue Anwendungsfälle unterschiedlicher Komplexität und Parametrisierung zu optimieren.

6.1 Generierung von Anwendungsfällen

Um einen Anwendungsfall zu generieren, wird zuerst der Kartenausschnitt als ungerichteter Graph erstellt. Die Grundidee basiert darauf, dass Knoten willkürlich auf einem rechteckigen Bereich verteilt und dann zufällig mit Knoten in der Nähe verbunden werden. Das Ziel ist ein Streckennetz zu generieren, das möglichst nah an echten Kartendaten ist. Der konzeptionelle Ablauf des im Folgenden vorgestellten Algorithmus ist in Abbildung 6.1 dargestellt.
Der Nutzer muss die Länge der vom Navigationssystem vorgegebenen Route und die Anzahl an außerhalb der Route liegenden Streckenknoten eingeben. Die Knoten, die zum Repräsentieren der Route benötigt werden, werden als Erstes erstellt und liegen zur besseren Übersichtlichkeit in der Mitte der Karte. Ein Viertel bis maximal drei Viertel der Knoten liegen zufällig verteilt auf der Fläche darüber. Die restlichen Streckenknoten liegen beliebig verteilt auf der Fläche darunter. Dadurch sollen Gebiete mit unterschiedlicher Dichte entstehen.

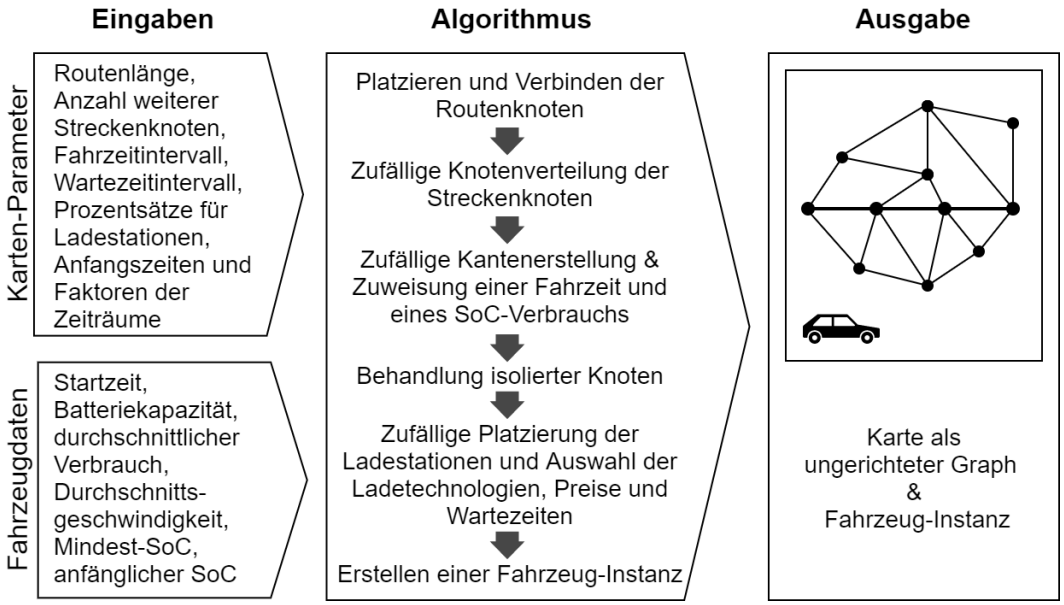


Abbildung 6.1: Konzept zur Generierung von Anwendungsfällen

Bei der Wahl der Kantenverbindungen wird zuerst die Route berücksichtigt. Für zwei in der Route aufeinanderfolgende Knoten werden jeweils Kanten in beide Richtungen erstellt. Die restlichen Knoten werden so miteinander verbunden, dass der Erwartungswert der ausgehenden beziehungsweise der eingehenden Kanten vier ist. Der Grund dafür ist, dass im realen Straßensystem von Kreuzungen meist Straßen in vier Richtungen abgehen und seltener beispielsweise T-Kreuzungen mit drei Richtungen oder Kreisverkehre mit mehr als vier Richtungen vorkommen. Außerdem wird ein Knoten nur mit Knoten in der Nähe verbunden, das heißt mit Knoten dessen euklidische Distanz zum betrachteten Knoten weniger als ein Viertel der Kartenbreite beträgt. Ein Viertel habe ich gewählt, damit die Verbindungen der Knoten nicht zu weit über die Karte reichen und die Route somit nicht mit wenigen Knoten umgangen werden kann. In dem Fall wäre die vom Navigationssystem ausgewählte Route wahrscheinlich nicht mehr die nahezu kürzeste Variante.
Wenn n die Zahl an erreichbaren Knoten von dem betrachteten Knoten sind, dann verbindet sich dieser Knoten mit einem Knoten in der Nähe mit einer Wahrscheinlichkeit von $\frac{4}{n}$. Das Problem dabei ist jedoch, dass eine Kante durch jeden der Knoten, die sie verbindet, zustande kommen kann. Dadurch ist der Durchschnitt der ausgehenden Kanten, vor allen Dingen in dichteren Bereichen, höher als vier. Deshalb habe ich bei der Verbindungswahrscheinlichkeit

ebenfalls berücksichtigt, zu wie vielen Knoten noch eine Kante aufgebaut werden kann und zu wie vielen schon eine Kante existiert. Dadurch, dass nur eine Diagonalmatrix der Adjazenzmatrix erstellt wird, kann ein Knoten immer nur Kanten zu Knoten in seiner Nähe mit einem höheren Index beeinflussen. Wenn n die Anzahl solcher Knoten zu einem betrachteten Knoten sind und e die Anzahl an schon zu diesem Knoten existierenden Kanten, haben Kanten eine Verbindungswahrscheinlichkeit von $\frac{4-e}{n}$. Falls e größer als vier ist, wird die Wahrscheinlichkeit auf null gesetzt.
Im Anschluss werden isolierte Knoten gesucht und diese mit den jeweils beiden nächsten Knoten verbunden. In seltenen Fällen passiert es, dass dann noch ein ganzer Teil des Graphen isoliert ist. Dies könnte man später noch behandeln.
Nach dem Erstellen der Adjazenzmatrix wird jeder Kante eine zufällige Fahrzeit aus einem festgelegten Intervall und basierend darauf, wie in Kapitel 4.1 beschrieben, ein SoC-Verbrauch zugewiesen.
Anschließend wird ein vom Ersteller angegebener Prozentsatz der Knoten zufällig als Ladestationen ausgewählt, die alle über Normalladen mit Abrechnung in Kilowattstunden verfügen. Von den Ladestationen bieten wiederum ein gewisser Anteil Schnellladen mit Abrechnung in Kilowattstunden und ein gewisser Anteil Pauschalpreise an. Die Bereitstellung des Pauschalbetrags und beider Ladetechnologien ist somit unabhängig voneinander. Die Preise liegen für jede Ladestation zufällig in einem bestimmten Intervall. Nach [42] ist dieser Bereich für Normalladen 29 bis 40 ct/kWh und für Schnellladen 39 bis 50 ct/kWh. Die Pauschalbeträge mit 5,95 € pro Ladung für Normalladen und 8,95 € pro Ladung für Schnellladen sind an dem Tarif von [24] ausgerichtet. Der monatliche Grundpreis wurde dabei vernachlässigt. Außerdem bekommt jede Ladestation aus einem benutzerdefinierten Bereich eine zufällige Wartezeit zugewiesen.
Weiterhin sind die Anfangszeiten und Faktoren der Zeiträume als Arrays gespeichert. Die Angaben im Anwendungsfall gelten immer für die mittlere Verkehrsdichte, selbst wenn der Wagen in einem Zeitraum mit anderer Verkehrsdichte startet. Neben den Parametern, die zur Kartenerstellung genutzt werden, kann der Ersteller außerdem einige Fahrzeugdaten für den Anwendungsfall wählen. Dies umfasst die Startzeit, die Kapazität der Batterie, der durchschnittliche Verbrauch, die Durchschnittsgeschwindigkeit, der Mindest-SoC und der anfängliche SoC.

6.2 Testaufbau

Mithilfe der Tests sollte erforscht werden, wie sich die Güte der Lösungen und die Eignung des Systems zum Einsatz in der Praxis verändern, wenn die Anwendungsfälle komplexer werden. Dazu muss man zuerst feststellen, wodurch das Problem schwieriger zu lösen wird.
Die Herausforderung bei dem Problem besteht hauptsächlich daraus, eine Route zu finden, die mindestens gleich gut wie die vorgegebene Strecke ist, und dabei gleichzeitig die optimale Ladeempfehlung auszuwählen, falls das Ziel nicht ohne zu laden oder ohne eine Verringerung des SoHs erreicht werden kann. Die Suche nach dem optimalen Pfad wird vor allen Dingen erschwert, wenn es mehr Möglichkeiten für diesen gibt, also wenn der Graph mehr Knoten enthält. In gleichem Maße sollte eine erhöhte Verfügbarkeit an Ladestationen und Ladetechnologien die Auswahl der am besten geeigneten behindern. Deshalb habe ich bei den Tests die Gesamtanzahl der Knoten im Graph und die der Ladestationen verändert.
Um die gewünschte Anzahl an Ladestationen zu erhalten, muss bei meiner Implementierung der Prozentsatz für Ladestationen in Relation zur Menge an Knoten angepasst werden. Außerdem setzt sich die Gesamtanzahl an Knoten aus der Länge der Route und den restlichen Streckenknoten zusammen. Diese beiden Parameter hätten somit auch unabhängig voneinander verändert werden können, was zur Vereinfachung allerdings nicht realisiert wurde. Stattdessen macht die Route immer 10 % der Gesamtknotenanzahl aus. Die Länge wird aufgerundet und auf zwei erhöht, falls sie kleiner als das ist.
Mit Inspiration von [70] wurden die für die Tests erstellten Anwendungsfälle in kleine und große Instanzen eingeteilt. Bei Ersteren reicht die Knotenanzahl von zehn bis 24 Knoten und es gibt immer zwei Ladestationen. Die großen Instanzen haben 25, 50, 75, 100 oder 150 Knoten und 2, 4, 6, 8 oder 10 Ladestationen.
Zum einfacheren Umgang mit den Anwendungsfällen werden diese über die veränderlichen Parameter kodiert. MS50LS6 ist beispielsweise eine Instanz mit 50 Knoten (MS, mapsize) und sechs Ladestationen (LS).
Die restlichen Parameter wurden ohne explizite Begründung wie folgt gewählt:

- Anteil der Ladestationen mit Schnellladen in kWh: 20 %
- Anteil der Ladestationen mit Pauschalpreisen: 25 %
- Intervall für Fahrzeiten: [10, 25] Minuten

- Intervall für Wartezeiten: [0, 100] Minuten
- Kapazität der Batterie: 32 kWh
- Durchschnittlicher Verbrauch: 0,2 kWh/km
- Durchschnittsgeschwindigkeit: 60 km/h
- Mindest-SoC: 20 %

In zukünftigen Tests könnten diese Parameter noch verändert werden. Weiterhin habe ich den anfänglichen SoC auf 90 % gesetzt, damit die Anzahl der Ladestationen nicht dadurch verringert wird, dass diese nicht erreichbar sind. Die Startzeit des Wagens ist immer 7.30 Uhr, damit das Fahrzeug bei mittlerer Verkehrsdichte startet und diese noch eine Weile erhalten bleibt.
Die Tests wurden auf einem Laptop mit einem Intel® Core™ i7-6600U Prozessor 2,81 GHz und 16 GB Arbeitsspeicher durchgeführt. Das Gerät war dabei immer an den Strom angeschlossen, damit die Leistung nicht vermindert wird.

6.3 Test der Güte und der Eignung des Systems zum Einsatz in der Praxis

Als Gütetest wurde gemessen, wie oft gültige Lösungen und für große Instanzen wie häufig davon dieselbe Pareto-Front gefunden wurden. Für jeden Anwendungsfall habe ich dafür zehnmal den implementierten evolutionären Algorithmus laufen lassen. Um die Eignung des Systems zum Einsatz in der Praxis zu evaluieren, wurde dabei für jeden Durchlauf die benötigte Zeit gemessen.
Dafür wollte ich zuerst herausfinden, ob eine für einen Anwendungsfall gemessene Zeit repräsentativ genug ist. Für ausgewählte Instanzen habe ich deswegen jeweils zehnmal denselben Anwendungsfall optimiert und die Standardabweichung berechnet. Das Ergebnis ist in Abbildung 6.2 dargestellt.
Wie zu sehen ist, bleibt die Standardabweichung bei den kleinen Instanzen unter einer Sekunde, sodass aus meiner Sicht eine Zeit für einen Anwendungsfall repräsentativ genug ist. Aus diesem Grund habe ich für kommende Test bei den kleinen Instanzen die Rechenzeiten für jeweils zehn verschiedene Anwendungsfälle gemessen. Dadurch wollte ich außerdem die Standardabweichung der Rechenzeiten bei unterschiedlichen Anwendungsfällen erfahren.
Für die Instanzen mit einer Kartengröße von 50 Knoten übersteigt die Standardabweichung zwölf Sekunden. Somit sind die Zeiten für die großen Instanzen

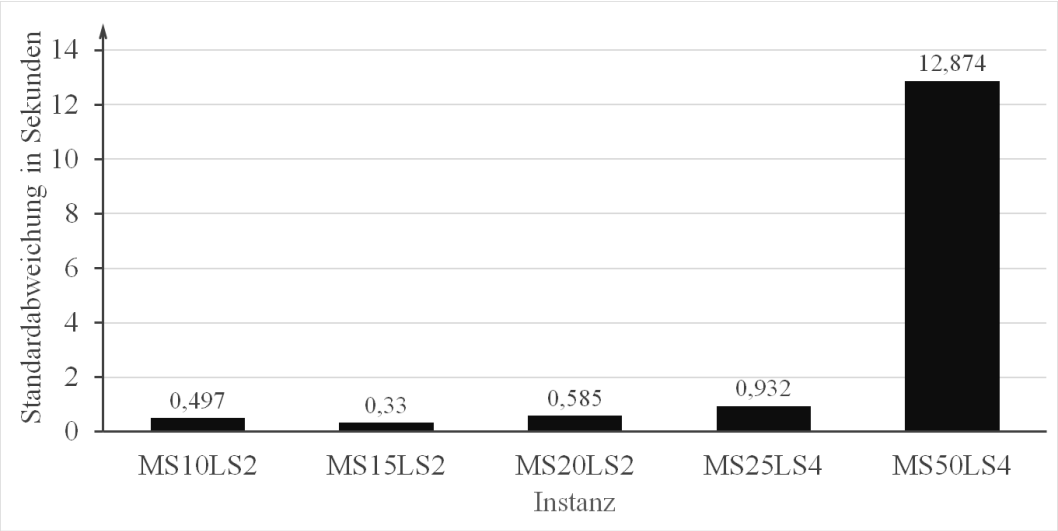


Abbildung 6.2: Standardabweichung von jeweils zehn Rechenzeiten für einige repräsentative Instanzen

nicht mehr repräsentativ. Deshalb habe ich für Letztere bei den Tests jeweils zehnmal denselben Anwendungsfall verwendet.
Für größere Instanzen mit 100 und 150 Knoten wollte ich die Standardabweichung ebenfalls berechnen. Dabei bin ich allerdings auf das Problem gestoßen, dass für diese Kartengrößen häufig keine gültigen Lösungen für einen Anwendungsfall gefunden wurden. Weitere Tests mit 100 Knoten im Graph haben ergeben, dass die Rechenzeit von der Anzahl der Durchläufe abhängt, in denen gültige Lösungen gefunden wurden. Diesen Zusammenhang erkennt man auch in Abbildung 6.3. Dadurch wurden die Standardabweichungen für diese Instanzen verfälscht, sodass ich sie nicht verwendet habe.
Bei dem Versuch, zu verhindern, dass es Durchläufe ohne gültige Lösungen gibt, habe ich zuerst die Populationsgröße bei der Optimierung für Instanzen mit 100 Streckenknoten erhöht. Für denselben Anwendungsfall unterschied sich die Anzahl an Durchläufen mit gültigen Lösungen für eine Populationsgröße von 100, 150 und 200 jedoch nur um eins. Aus diesem Grund habe ich die Populationsgröße als Ursache des Problems ausgeschlossen.
Eine andere Vermutung war, dass häufig keine gültigen Lösungen gefunden werden, weil in der initialen Population keine gültigen Lösungen vorhanden sind. Deshalb habe ich bei der Initialisierung eingebaut, dass die Route mit einer zufälligen, gültigen Ladeempfehlung ein Individuum der initialen Popu-

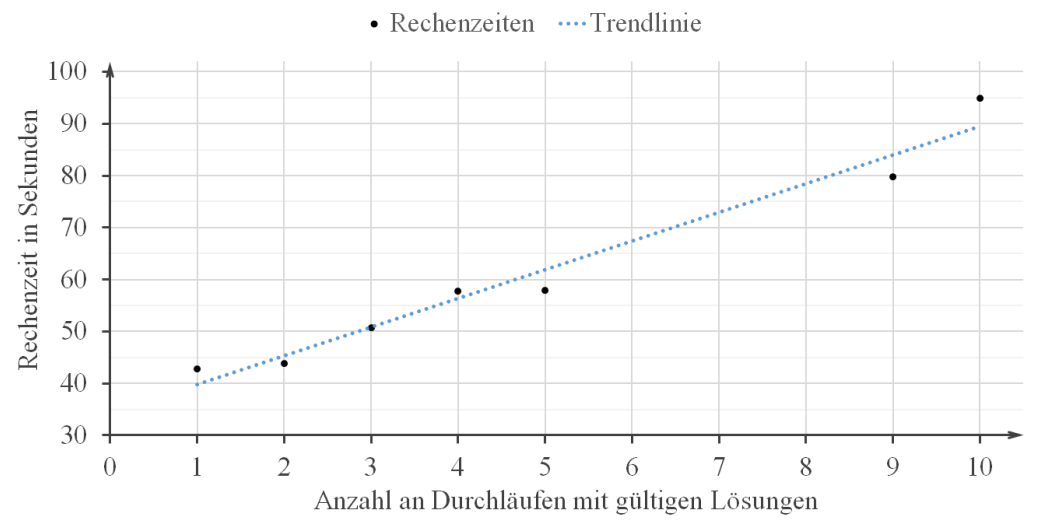


Abbildung 6.3: Rechenzeiten in Abhängigkeit der Durchläufe mit gültigen Lösungen für Graphen mit 100 Knoten

lation darstellt. Für die großen Instanzen habe ich dann für beide Varianten die Anzahl an Durchläufen mit gültigen Lösungen gemessen und in Abbildung 6.4 visualisiert. Für 50 Knoten im Graph werden bei beiden Vorgehensweisen immer in allen Durchläufen gültige Lösungen gefunden. Bei 100 Streckenknoten verbessert sich der Wert für alle Instanzen, wenn der initialen Population die Route hinzugefügt wird. Kleine Verbesserungen und Verschlechterungen zeigen sich bei 150 Knoten. Dies kommt eventuell dadurch zustande, dass die Route bei diesen Instanzen meistens keine gültige Lösung darstellt. Da sich mit dem Einbau der Route die Werte verbessern oder im schlimmsten Fall nur wenig verschlechtern, wurde diese Variante für die kommenden Tests genutzt. Da das Problem dadurch zwar ein Stück gemindert wurde, aber die Ergebnisse noch nicht zufriedenstellend waren, habe ich in Kapitel 6.4 die Auswirkung weiterer Parameter untersucht.
In Bezug auf Güte der Lösungen hat sich gezeigt, dass der Algorithmus bei Anwendungsfällen für kleine Instanzen keine Probleme hat. Es werden in allen Durchläufen gültige Lösungen gefunden. Für große Instanzen mit 25, 50 oder 75 Knoten gilt dies ebenso. Ab 100 Knoten ist dies nicht immer der Fall. Bei den großen Instanzen wurde außerdem überprüft, ob in allen Durchgängen dieselbe Pareto-Front ausgegeben wurde. Bis zu einer Kartengröße von 50 Knoten stimmt dies. Mit 100 Knoten werden nicht immer gültige Lösungen ge-

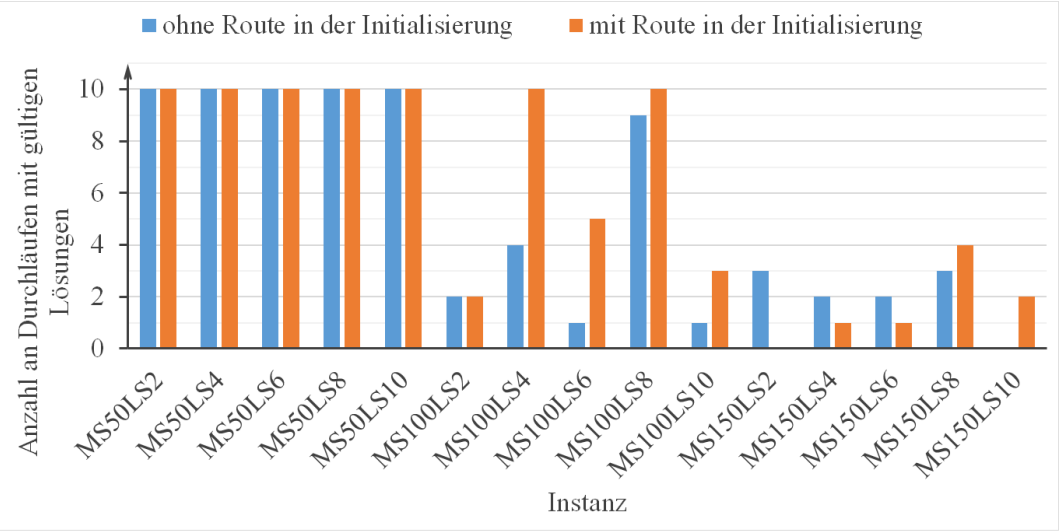


Abbildung 6.4: Vergleich der Anzahl an Durchläufen mit gültigen Lösungen mit und ohne Route in der Initialisierung

funden und die ausgegebenen Pareto-Fronten sind nicht immer dieselben. Die Werte der gefundenen Pareto-Fronten liegen in den meisten Fällen jedoch nah beieinander. Die beste Pareto-Front aus zehn Durchläufen tritt in der Regel nur einmal auf. Ab 150 wird in der Mehrzahl der Durchgängen keine gültige Lösung gefunden.
Da bis zu einer Kartengröße von 50 Knoten immer gültige Lösungen geliefert werden und die Ergebnisse deterministisch sind, ist der Algorithmus für diese Daten gut einsetzbar. Bei 100 Knoten kann immer noch eine sinnvolle Ladeempfehlung gegeben werden, falls eine gültige Lösung gefunden wird. Das System sollte aber noch verbessert werden. Ab 150 Knoten ist ein Einsatz nicht mehr empfehlenswert, da in den wenigsten Durchläufen gültige Lösungen geliefert werden und eventuell notwendige mehrmalige Neuberechnungen zu zeitaufwendig wären.
Weiterhin habe ich die Rechenzeiten der Instanzen ermittelt. Da wie oben festgestellt ab 100 Knoten die Messungen durch Durchläufe ohne gültige Lösungen verfälscht werden, habe ich für die großen Instanzen 25, 50 und 75 Knoten genommen. Zur Vergleichbarkeit sind immer zwei Ladestationen vorhanden. Wie in Abbildung 6.5 zu erkennen ist, steht die Rechenzeit für die betrachteten Anwendungsfälle annähernd in einem linearen Zusammenhang zu der Anzahl der Knoten im Graph. Basierend darauf und auf der oben gesetzten Begrenzung

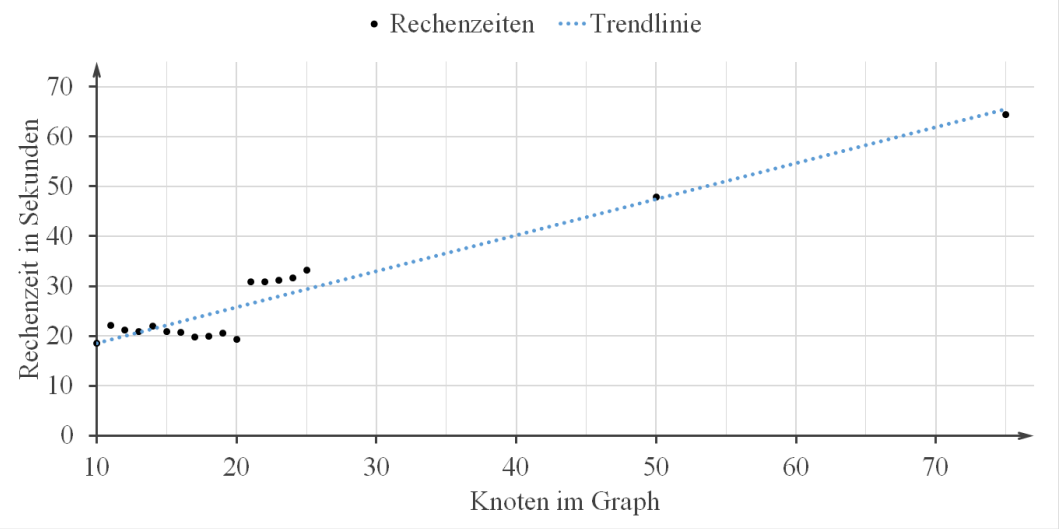


Abbildung 6.5: Durchschnittliche Rechenzeiten für Instanzen mit zwei Ladestationen in Abhängigkeit zur Anzahl an Knoten im Graph

ist der Algorithmus zum Einsatz in der Praxis geschätzt nur bis ungefähr 65 Knoten geeignet.
Weiterhin habe ich beobachtet, dass die Zeiten bei den kleinen Instanzen meist eine größere Standardabweichung bei unterschiedlichen Anwendungsfällen als bei denselben haben. Bei jeweils zehn unterschiedlichen Anwendungsfällen bleibt die Standardabweichung nur bei fünf von 15 Instanzen unter einer Sekunde. In den restlichen Fällen liegt die Standardabweichung im Bereich von circa 1,08 bis 4,98 Sekunden. Bei Verwendung derselben Anwendungsfälle lag die Standardabweichung immer unter einer Sekunde.
Außerdem unterscheiden sich die Rechenzeiten der großen Instanzen bei unterschiedlicher Anzahl an Ladestationen nicht wesentlich. Die gerundete Standardabweichung bei unterschiedlich vielen Ladestationen beträgt für 25 Knoten 1,92 Sekunden, für 50 Knoten 1,98 Sekunden und für 75 Knoten 3,05 Sekunden. Daraus könnte man schlussfolgern, dass die Anzahl an Ladestationen, im Gegensatz zu meiner Annahme, keine oder nur eine geringe Auswirkung auf die Komplexität des Problems hat. Allerdings ist die Rechenzeit als Maß der Komplexität nicht aussagekräftig genug.
Zusammenfassend lässt sich sagen, dass der vorgeschlagene Algorithmus ausgezeichnet für alle kleinen Instanzen und große Instanzen bis 50 Knoten funktioniert. Die Berechnungsdauer solcher Instanzen liegt außerdem unterhalb der

akzeptierte Grenze von 60 Sekunden. Für den Einsatz in der Praxis ist ein Kartenausschnitt dieser Größe allerdings wahrscheinlich noch nicht ausreichend. Zwar funktioniert der Algorithmus auch mit mehr Knoten, die Güte der Lösungen und die Rechenzeit müssen aber noch verbessert werden. Alternativ könnte ein anderer Algorithmus zur Lösung des Problems ausprobiert werden.

6.4 Parameterverbesserung

Wie im vorherigen Unterkapitel erwähnt, ist die Güte der Lösungen ab 100 Knoten im Graph noch nicht ausreichend. Aus diesem Grund habe ich versucht durch die Veränderung drei verschiedener Parameter für Instanzen mit 100 Knoten eine Verbesserung zu erzielen. Festgestellt wird Letztere an der Güte der Lösung, die wie oben beschrieben gemessen wird. Die drei veränderten Parameter sind die Anzahl an Zielfunktionen, die Chromosomlänge und die Abbruchkriterien.

Verringerte Anzahl an Zielfunktionen Es wird vermutet, dass das Problem einfacher wird, wenn weniger Zielfunktionen optimiert werden müssen. Dadurch könnte sich die Güte der Lösungen erhöhen. Deshalb wurden nur die beiden Zielfunktionen beachtet, die den Fahrer bei der Fahrt direkt beeinflussen: die Minimierung der Ladekosten und die Minimierung der Reisezeit.
Das Ergebnis des Tests ist in Abbildung 6.6 zu sehen. Es lässt sich keine eindeutige Verbesserung bei der Anzahl an Durchläufen mit gültigen Lösungen erkennen. Obwohl der Wert mit zwei Zielfunktionen bei drei Instanzen besser und nur bei einer schlechter ist, liegt die Summe für vier Zielfunktionen mit 30 erfolgreichen Durchläufen über der von zwei Zielfunktionen mit 28 Durchläufen.
Die Werte der Pareto-Fronten sind für beide Varianten meist ähnlich, aber bei zwei Zielfunktionen werden häufiger mehr unterschiedliche Pareto-Fronten gefunden.
Somit kann nicht bestätigt werden, dass eine Verringerung der Menge der Zielfunktionen eine Verbesserung des Algorithmus verursacht.

Veränderung der Chromosomlänge Eine andere Vermutung war, dass die Chromosomlänge entweder zu klein oder zu groß ist. Wäre sie zu klein, könnte bei der Initialisierung der Graph nicht weitläufig genug traversiert werden,

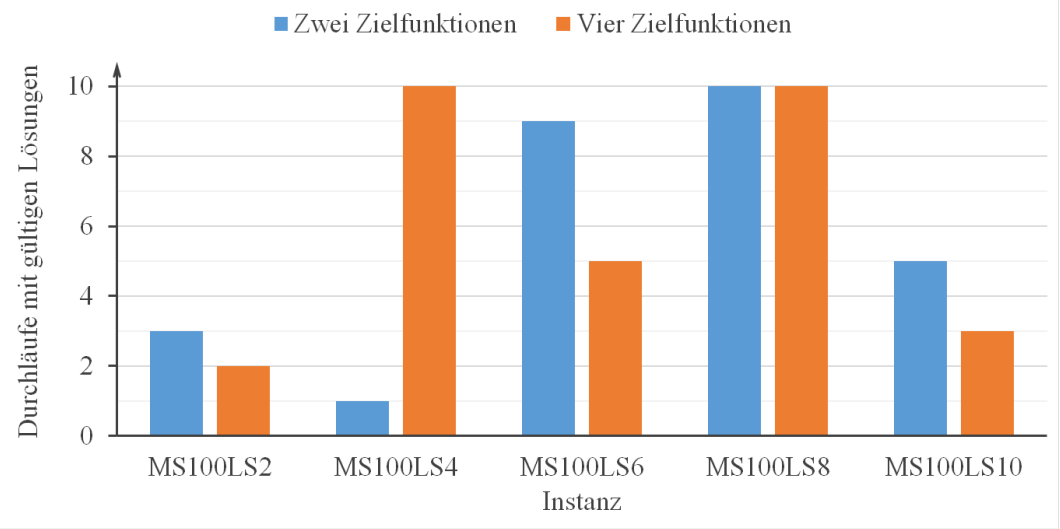


Abbildung 6.6: Vergleich der Anzahl an Durchläufen mit gültigen Lösungen für zwei und vier Zielfunktionen

wodurch die fehlende Diversität der initialen Population sich negativ auf die Güte der Lösungen auswirken würde. Wäre die Länge zu groß gewählt, wäre zu Beginn Platz für unnötig lange Umwege, wodurch es für den Algorithmus schwieriger sein könnte, von der initialen Population zu guten Lösungen zu gelangen.
Für 100 Knoten im Graph ist die Chromosomlänge nach der Berechnung aus Kapitel 5.1 35 Gene lang. Damit wurden in 30 Durchläufen gültige Lösungen gefunden. Zur Überprüfung habe ich für jede Instanz mit 100 Knoten den evolutionären Algorithmus jeweils zehnmal mit einer Chromosomlänge von 25, 30, 40 und 45 ausgeführt. Die maximale Anzahl an erfolgreichen Iterationen wäre somit 50 gewesen. Wie auch in Abbildung 6.7 zu sehen ist, erzielten die beiden höchsten Werte die kürzeste Chromosomlänge mit 46 Durchläufen und die längste Chromosomlänge mit 43 Durchläufen. Ähnlich gut war die Länge von 40, bei der es 41 erfolgreiche Durchläufe gab. Deutlich weniger konnte mit einer Chromosomlänge von 30 erreicht werden, denn dort gab es nur 35 Iterationen mit gültigen Lösungen. Am schlechtesten schnitt jedoch die ursprüngliche Chromosomlänge ab.
Wenn man die Werte für die unterschiedlichen Instanzen vergleicht, lässt sich keine klare Verbesserung erkennen. Sie vergrößern oder verkleinern sich immer für unterschiedliche Instanzen. Die jeweils gefundenen Pareto-Fronten sind bei

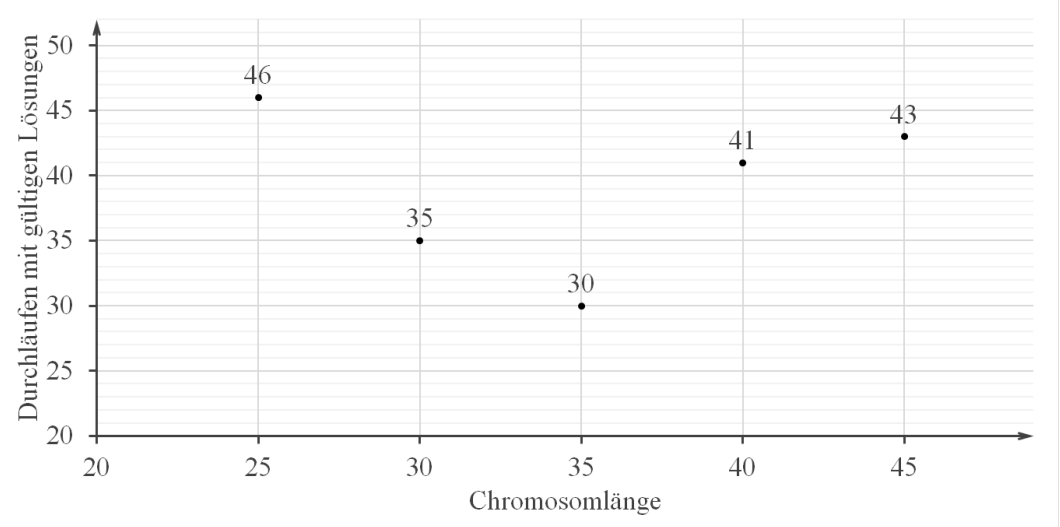


Abbildung 6.7: Vergleich der Anzahl an Durchläufen mit gültigen Lösungen für verschiedene Chromosomlängen

allen Chromosomlängen immer noch unterschiedlich.
Die Ergebnisse lassen sich durch die anfangs genannte Vermutung nicht erklären und diese sich damit weder belegen noch widerlegen. Es muss in Zukunft noch analysiert werden, warum sowohl bei kleinerer als auch bei größerer Chromosomlänge eine Verbesserung auftritt. Dazu sollte wahrscheinlich die initiale Population und deren Entwicklung über die Generationen hinweg genauer beobachtet werden.

Entnahme des Spreads bei den Abbruchkriterien Eines der Abbruchkriterien des vorgeschlagenen Algorithmus ist, dass die Berechnung endet, wenn sich die Pareto-Front für 100 Generationen nicht mehr genügend verändert. Dadurch läuft der evolutionäre Algorithmus überwiegend nur für einige Generationen mehr als 100. Selten werden mehr als 200 Generationen benötigt. Ich nehme an, dass das Ersetzen des genannten Abbruchkriteriums durch eine feste, größere maximale Anzahl an Generationen die Güte der Lösungen erhöht. Wenn der Algorithmus für mehr Generationen läuft, könnte durch eine zufällige Veränderung eine bessere, überhaupt eine gütige oder dieselbe Lösung wie in einer anderen Pareto-Front entdeckt werden.
Die maximale Anzahl an Generationen wurde auf 500 gesetzt, um laut [18] eine recht hohen Wert zu haben. Dadurch hat der Algorithmus in jedem

Durchlauf gültige Lösungen gefunden, aber die ausgegebenen Pareto-Fronten sind immer noch unterschiedlich.
Bei der Veränderung dieses Parameters muss allerdings hervorgehoben werden, dass sich die Rechenzeit ungefähr um das Vier- bis Fünffache verschlechtert. Bei 100 Knoten mit Spread in den Abbruchkriterien braucht der Algorithmus ungefähr 90 Sekunden, während die Rechendauer bei 500 Generationen zwischen 368 und 445 Sekunden beträgt. Dadurch, dass der Spread aus den Abbruchkriterien entfernt wurde, kann es nun sein, dass nach der spontanen Verbesserung noch viele Generationen ohne signifikante Verbesserung berechnet werden und die Zeit dadurch unnötig groß wird. Um die Berechnungsdauer zu verringern, ist es möglicherweise sinnvoll, den Spread in die Abbruchkriterien wieder aufzunehmen, aber die Anzahl der Generationen, über die gemittelt wird, zu vergrößern.
Daraus leite ich ab, dass durch eine größere Anzahl an Generationen der Algorithmus zuverlässiger, aber auch zeitaufwendiger wird. Das Finden der besten Anzahl an maximalen oder gemittelten Generationen, dem aus zeitlichen Gründen in dieser Arbeit nicht nachgegangen wurde, muss in Zukunft noch erledigt werden. Außerdem muss die Ursache dafür, dass die Pareto-Fronten oft unterschiedlich sind, gefunden werden.

Die einzige signifikante Verbesserung konnte durch das Anpassen der Abbruchkriterien erzielt werden. Bei der Verringerung der Anzahl an Zielfunktionen hat sich die Güte nur für einzelne Instanzen erhöht und durch die Veränderung der Chromosomlänge kam es auf bis jetzt unerklärliche Weise zu mehr Durchläufen mit gültigen Lösungen. Selbst dann werden noch verschiedene Pareto-Fronten ausgeben. Gut ist, dass deren Werte meist ähnlich sind und selten deutlich schlechtere Ladeempfehlungen vorkommen. In einigen Fällen kommt es jedoch auch vor, dass eine Kombination aus mehreren ausgegebenen Fronten eine breitere Pareto-Front ergeben würde. Aufgrund der Beobachtungen in diesem Kapitel vermute ich, dass die Exploration bei der Optimierung nicht hoch genug ist. Bei der Exploration geht um die weitreichende Erkundung des Suchraums. Dies würde erklären, warum selten dieselben oder nur einzelne Pareto-Individuen gefunden werden. Für zukünftige Implementierungen bedeutet das, dass zu einer weiteren Verbesserung wahrscheinlich vor allem die Initialisierung und die Rekombination des Algorithmus verändert werden müssen. Bei der Initialisierung könnte man

beispielsweise einbauen, dass man nicht nur Knoten, die ein Individuum selbst schon besucht hat, mit einer geringeren Wahrscheinlichkeit wählt, sondern dies auch auf von anderen Individuen häufig besuchte Knoten überträgt. Danach kann man sich weiter mit der Optimierung der Parameter beschäftigen.
Des Weiteren sollte die Konvergenz der Pareto-Fronten überprüft werden. In den trivialen Fällen sieht man, dass der Algorithmus das Optimum findet. Für größere Instanzen wurde dies aus zeitlichen Gründen nicht getan, da die Ermittlung der optimalen Pareto-Font zu lange dauern würde. Außerdem sollte die Diversität der Ergebnisse bewertet werden. Momentan wird häufig nur ein Pareto-Individuum ausgegeben, was auch an der Wahl der Parameter für den Anwendungsfall liegen kann.

7 Fazit

In dieser Arbeit wurde ein Ansatz entwickelt, welcher Nutzern eines Elektrofahrzeugs während der Fahrt Ladeempfehlungen inklusive passender Alternativrouten anzeigt und sie damit bei der Entscheidungsfindung unterstützt. Dafür wurden die Hintergründe der multikriteriellen Optimierung und des SoHs einer Batterie sowie Literatur zum verwandten EVRP besprochen. Anschließend wurde das Problem dieser Arbeit auf einem ungerichteten Graphen definiert und die nutzerorientierten Zielfunktionen formuliert. Basierend darauf wurde eine Lösungsmethode vorgeschlagen, die das Problem dieser Arbeit a posteriori und multikriteriell löst und damit eine Lücke in der Literatur schließt.
Bei dem Algorithmus handelt es sich um NSGA-II mit kontrolliertem Elitismus, für den bis auf die Selektion und Abbruchkriterien alle Teilschritte selbst konzipiert wurden. Für die Evaluierung wurde außerdem ein Algorithmus entworfen, der Anwendungsfälle generiert. Die Tests ergaben, dass die Lösungsmethode ausgezeichnet für Anwendungsfälle bis zu einer Größe von 50 Knoten funktioniert. Das gesetzte Zeitlimit von 60 Sekunden wird in diesem Fall auch eingehalten. Für 100 Knoten läuft der Algorithmus immer noch zuverlässig, wenn das Abbruchkriterium der maximalen Generationen angepasst wird. Dadurch wird jedoch das Zeitlimit überschritten. Eine abschließende Vermutung war, dass zur weiteren Verbesserung des Algorithmus die Exploration in der Lösungsmethode verändert werden muss.
Anschließend können weitere für den Algorithmus spezifische Parameter optimiert werden. Dazu zählen die Populationsgröße, das Verhältnis von rekombinierten und mutierten Kindern bei den genetischen Operatoren oder die *ParetoFraction* bei der Umweltselektion. Weiterhin könnten andere Abbruchkriterien ausprobiert oder die aktuellen verbessert werden. Alternativ ist die Anwendung eines anderen Algorithmus zur Lösung des Problems möglich.
In Zukunft wäre eine Veränderung der Problemdefinition denkbar. Bei der Route könnten Orte von Interesse in die Planung einbezogen werden. Diese Orte liegen möglicherweise nur an Ladestationen oder auch an sonstigen Kno-

ten im Streckennetz. Eine andere Variante wäre die vom Navigationssystem vorgeschlagene Route wegzulassen, um eine unabhängige Berechnung optimaler Lademöglichkeiten zu gestalten. Der Graph könnte zu einem gerichteten Graph abgeändert werden, um beispielsweise Einbahnstraßen abzubilden. Des Weiteren ist es möglich, die Zielfunktionen und die dafür verwendeten Systemparameter zu verändern. Außerdem ist eine Erweiterung um eine Kommunikation zu anderen Fahrzeugen oder die parallele Optimierung für mehrere Wagen denkbar. Für die Anwendungsfälle könnten in Zukunft echte Kartendaten genutzt werden.

Literaturverzeichnis

[1] A. Abdulaal, M. H. Cintuglu, S. Asfour, and O. A. Mohammed. Solving the multivariant ev routing problem incorporating v2g and g2v options. *IEEE Transactions on Transportation Electrification*, 3(1):238–248, März 2017.

[2] I. Alaya, C. Solnon, and K. Ghedira. Ant colony optimization for multi-objective optimization problems. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 1, pages 450–457, Oktober 2007.

[3] O. Arslan, B. Yıldız, and O. E. Karaşan. Minimum cost path problem for plug-in hybrid electric vehicles. *Transportation Research Part E: Logistics and Transportation Review*, 80:123–141, 2015.

[4] AutoScout24. Ladedauer von Elektroautos: Ladestationen im Vergleich - AutoScout24. `https://www.autoscout24.de/informieren/ratgeber/e-mobilitaet/ladedauer/`, März 2019. Zuletzt abgerufen am 12.02.2020.

[5] A. Barré, B. Deguilhem, S. Grolleau, M. Gérard, F. Suard, and D. Riu. A review on lithium-ion battery ageing mechanisms and estimations for automotive applications. *Journal of Power Sources*, 241:680–689, 2013.

[6] S. Bashash, S. J. Moura, and H. K. Fathy. Charge trajectory optimization of plug-in hybrid electric vehicles for energy cost reduction and battery health enhancement. In *Proceedings of the 2010 American Control Conference*, pages 5824–5831, Juni 2010.

[7] M. Becherif, M. Y. Ayad, D. Hissel, and R. Mkahl. Design and sizing of a stand-alone recharging point for battery electrical vehicles using photovoltaic energy. In *2011 IEEE Vehicle Power and Propulsion Conference*, pages 1–6, September 2011.

[8] C. Bingham, C. Walsh, and S. Carroll. Impact of driving characteristics on electric vehicle energy consumption and range. *IET Intelligent Transport Systems*, 6:29–35(6), März 2012.

[9] J. Branke. *MCDA and Multiobjective Evolutionary Algorithms*, pages 977–1008. Springer New York, New York, NY, 2016.

[10] U. Breunig, R. Baldacci, R. F. Hartl, and T. Vidal. The electric two-echelon vehicle routing problem. Technical report, University of Vienna, University of Bologna, Pontifícia Universidade Católica do Rio de Janeiro, 2018. Verfügbar unter: `https://arxiv.org/pdf/1803.03628.pdf`.

[11] M. Bruglieri, S. Mancini, F. Pezzella, and O. Pisacane. A path-based solution approach for the green vehicle routing problem. *Computers & Operations Research*, 103:109–122, 2019.

[12] M. Bruglieri, S. Mancini, F. Pezzella, O. Pisacane, and S. Suraci. A three-phase matheuristic for the time-effective electric vehicle routing problem with partial recharges. *Electronic Notes in Discrete Mathematics*, 58:95–102, 2017. 4th International Conference on Variable Neighborhood Search.

[13] M. Bruglieri, F. Pezzella, O. Pisacane, and S. Suraci. A variable neighborhood search branching for the electric vehicle routing problem with time windows. *Electronic Notes in Discrete Mathematics*, 47:221–228, 2015. The 3rd International Conference on Variable Neighborhood Search (VNS'14).

[14] ChargeMap.com. Anzahl der Ladestationen für Elektrofahrzeuge in Deutschland im Zeitraum 1. Quartal 2018 bis 1. Quartal 2020 (Stand: 03. Februar 2020). `https://de.statista.com/statistik/daten/studie/460234/umfrage/ladestationen-fuer-elektroautos-in-deutschland-monatlich/`, Februar 2020. Zuletzt abgerufen am 24.02.2020.

[15] T. Chen, B. Zhang, H. Pourbabak, A. Kavousi-Fard, and W. Su. Optimal routing and charging of an electric vehicle fleet for high-efficiency dynamic transit systems. *IEEE Transactions on Smart Grid*, 9(4):3563–3572, Juli 2018.

[16] C. A. Coello Coello and M. S. Lechuga. Mopso: a proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1051–1056, Mai 2002.

[17] R. Conrad and M. Figliozzi. The recharching vehicle routing problem. In T. Doolen and E. Van Aken, editors, *Proceedings of the 2011 Industrial Engineering Research Conference*, 2011.

[18] K. Deb. *Multi-Objective Optimization Using Evolutionary Algortihms*. John Wiley & Sons, Chichester, England, 2001.

[19] J. Deng, J. Shi, Y. Liu, and Y. Tang. Application of a hybrid energy storage system in the fast charging station of electric vehicles. *IET Generation, Transmission & Distribution*, 10:1092–1097(5), März 2016.

[20] G. Desaulniers, F. Errico, S. Irnich, and M. Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405, 2016.

[21] N. Ding, R. Batta, and C. Kwon. Conflict-free electric vehicle routing problem with capacitated charging stations and partial recharge. Technical report, University at Buffalo, University of South Florida, 2015. Verfügbar unter: `http://www.acsu.buffalo.edu/~batta/Nan%20Ding.pdf`.

[22] M. Ehrgott and X. Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *Top*, 12(1):1–63, Juni 2004.

[23] M. Ehrgott, X. Gandibleux, and A. Przybylski. *Exact Methods for Multi-Objective Combinatorial Optimisation*, pages 817–850. Springer New York, New York, NY, 2016.

[24] E.ON Energie Deutschland GmbH. E-Mobility - Unterwegs aufladen mit Drive Easy | E.ON. `https://www.eon.de/de/pk/e-mobility/unterwegs.html`. Zuletzt abgerufen am 06.12.2019.

[25] Á. Felipe, M. T. Ortuño, G. Righini, and G. Tirado. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, 71:111–128, 2014.

[26] I. Fernández, C. Calvillo, A. Sánchez-Miralles, and J. Boal. Capacity fade and aging models for electric batteries and optimal charging strategy for electric vehicles. *Energy*, 60:35–43, 2013.

[27] J. Ferreira, P. Pereira, P. Filipe, and J. Afonso. Recommender system for drivers of electric vehicles. In *2011 3rd International Conference on Electronics Computer Technology*, volume 5, pages 244–248, April 2011.

[28] A. Froger, J. E. Mendoza, O. Jabali, and G. Laporte. A matheuristic for the electric vehicle routing problem with capacitated charging stations. Technical report, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport (CIRRELT), 2017. hal-01559524.

[29] A. Froger, J. E. Mendoza, O. Jabali, and G. Laporte. Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. *Computers & Operations Research*, 104:256–294, 2019.

[30] D. Goeke and M. Schneider. Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81–99, 2015.

[31] L. Grandinetti, F. Guerriero, F. Pezzella, and O. Pisacane. A pick-up and delivery problem with time windows by electric vehicles. *International Journal of Productivity and Quality Management (IJPQM)*, 18(2/3), 2016.

[32] M. A. Hannan, M. M. Hoque, A. Hussain, Y. Yusof, and P. J. Ker. State-of-the-art and energy management system of lithium-ion batteries in electric vehicle applications: Issues and recommendations. *IEEE Access*, 6:19362–19378, 2018.

[33] M. Hanson. Tabu search for multiobjective optimization: Mots. Präsentiert auf MCDM '97, Cape Town, South Africa, January 6-10, 1997.

[34] G. Hiermann, J. Puchinger, S. Ropke, and R. F. Hartl. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995–1018, 2016.

[35] J. Jemai, M. Zekri, and K. Mellouli. An NSGA-II algorithm for the green vehicle routing problem. In J.-K. Hao and M. Middendorf, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 37–48, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[36] D. Jones, S. Mirrazavi, and M. Tamiz. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137(1):1–9, 2002.

[37] KBA und Aral. Anzahl der Elektroautos in Deutschland von 2006 bis 2019. `https://de.statista.com/statistik/daten/studie/`

265995/umfrage/anzahl-der-elektroautos-in-deutschland/, März 2019. Zuletzt abgerufen am 24.02.2020.

[38] M. Keskin and B. Çatay. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65:111–127, 2016.

[39] M. Keskin and B. Çatay. A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & Operations Research*, 100:172–188, 2018.

[40] M. Keskin, G. Laporte, and B. Çatay. Electric vehicle routing problem with time-dependent waiting times at recharging stations. *Computers & Operations Research*, 107:77–94, Juli 2019.

[41] M. Kirley and R. Stewart. Multiobjective evolutionary algorithms on complex networks. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization*, pages 81–95, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[42] P. Kloth. Elektroauto-Kosten: Aufladung und Verbrauch (100 km). `https://www.energieheld.de/mobilitaet/elektroauto/kosten/aufladung-verbrauch-100-kilometer`. Zuletzt abgerufen am 06.12.2019.

[43] N. Kullman, J. Goodson, and J. E. Mendoza. Dynamic electric vehicle routing: Heuristics and dual bounds. Technical report, 2018. hal-01928730.

[44] C. Li, T. Ding, X. Liu, and C. Huang. An electric vehicle routing optimization model with hybrid plug-in and wireless charging systems. *IEEE Access*, 6:27569–27578, 2018.

[45] W. Li-ying and S. Yuan-bin. Multiple charging station location-routing problem with time window of electric vehicle. *Journal of Engineering Science and Technology Review*, 8(5):190–201, 2015.

[46] H. Lin, T. Liang, and S. Chen. Estimation of battery state of health using probabilistic neural network. *IEEE Transactions on Industrial Informatics*, 9(2):679–685, Mai 2013.

[47] J. Lin, W. Zhou, and O. Wolfson. Electric vehicle routing problem. *Transportation Research Procedia*, 12:508–521, 2016. Tenth International Conference on City Logistics 17-19 June 2015, Tenerife, Spain.

[48] R. Mkahl, A. Nait-Sidi-Moh, J. Gaber, and M. Wack. An optimal solution for charging management of electric vehicles fleets. *Electric Power Systems Research*, 146:177–188, Mai 2017.

[49] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas. The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, 103:87–110, 2017. Green Urban Transportation.

[50] M. Nicholas and D. Hall. Lessons learned on early electric vehicle fast-charging deployments. Technical report, The International Council on Clean Transportation, Juli 2018.

[51] R. Roberti and M. Wen. The electric traveling salesman problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 89:32–52, 2016.

[52] A. Sarker, H. Shen, and J. A. Stankovic. Morp: Data-driven multi-objective route planning and optimization for electric vehicles. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4):162:1–162:35, Januar 2018.

[53] K. Sarrafan, K. M. Muttaqi, D. Sutanto, and G. E. Town. An intelligent driver alerting system for real-time range indicator embedded in electric vehicles. *IEEE Transactions on Industry Applications*, 53(3):1751–1760, Mai 2017.

[54] M. Schneider, A. Stenger, and D. Goeke. The electric vehicle-routing problem with time windows and recharging stations. Technical report, University of Kaiserslautern, Goethe University Frankfurt am Main, 2012.

[55] A. Setämaa-Kärkkäinen, K. Miettinen, and J. Vuori. Heuristic for a new multiobjective scheduling problem. *Optimization Letters*, 1(3):213–225, Juni 2007.

[56] U. F. Siddiqi, Y. Shiraishi, and S. M. Sait. Multi-objective optimal path selection in electric vehicles. *Artificial Life and Robotics*, 17(1):113–122, Oktober 2012.

[57] W. Su and M.-Y. Chow. An intelligent energy management system for phevs considering demand response. In *Proceedings of the 2010 FREEDM annual conference*, Mai 2010.

[58] A. Suppapitnarm, K. A. Seffen, G. T. Parks, and P. J. Clarkson. A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization*, 33(1):59–85, 2000.

[59] T. M. Sweda, I. S. Dolinskaya, and D. Klabjan. Adaptive routing and recharging policies for electric vehicles. *Transportation Science*, 51(4):1031–1386, November 2017.

[60] The MathWorks, Inc. Find pareto front of multiple fitness functions using genetic algorithm - matlab gamultiobj - mathworks deutschland. `https://de.mathworks.com/help/gads/gamultiobj.html`. (Zuletzt abgerufen am 16.12.2019).

[61] The MathWorks, Inc. gamultiobj algorithm - matlab & simulink - mathworks deutschland. `https://de.mathworks.com/help/gads/gamultiobj-algorithm.html`. (Zuletzt abgerufen am 30.01.2020).

[62] Z. Tian, T. Jung, Y. Wang, F. Zhang, L. Tu, C. Xu, C. Tian, and X. Li. Real-time charging station recommendation system for electric-vehicle taxis. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3098–3109, November 2016.

[63] Y. Wang, J. Bi, W. Guan, and X. Zhao. Optimising route choices for the travelling and charging of battery electric vehicles by considering multiple objectives. *Transportation Research Part D: Transport and Environment*, 64:246–261, 2018.

[64] M. Wen, E. Linde, S. Ropke, P. Mirchandani, and A. Larsen. An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. *Computers & Operations Research*, 76:73–83, 2016.

[65] R. Xiong, L. Li, and J. Tian. Towards a smarter battery management system: A critical review on battery state of health monitoring methods. *Journal of Power Sources*, 405:18–29, 2018.

[66] H. Yang, S. Yang, Y. Xu, E. Cao, M. Lai, and Z. Dong. Electric vehicle route optimization considering time-of-use electricity price by learnable partheno-genetic algorithm. *IEEE Transactions on Smart Grid*, 6(2):657–666, März 2015.

[67] W. Yao, J. Zhao, F. Wen, Z. Dong, Y. Xue, Y. Xu, and K. Meng. A multi-objective collaborative planning strategy for integrated power dis-

tribution and electric vehicle charging systems. *IEEE Transactions on Power Systems*, 29(4):1811–1821, Juli 2014.

[68] M. Yilmaz and P. T. Krein. Review of battery charger topologies, charging power levels, and infrastructure for plug-in electric and hybrid vehicles. *IEEE Transactions on Power Electronics*, 28(5):2151–2169, Mai 2013.

[69] A. Zakariazadeh, S. Jadid, and P. Siano. Multi-objective scheduling of electric vehicles in smart distribution system. *Energy Conversion and Management*, 79:43–53, 2014.

[70] S. Zhang, Y. Gajpal, S. Appadoo, and M. Abdulkader. Electric vehicle routing problem with recharging stations for minimizing energy consumption. *International Journal of Production Economics*, 203:404–413, Juli 2018.

[71] S. Zhang, Y. Luo, and K. Li. Multi-objective route search for electric vehicles using ant colony optimization. In *2016 American Control Conference (ACC)*, pages 637–642, Juli 2016.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Eva Röper

Magdeburg, 28. Februar 2020