# Nutzerorientierte Optimierung des Over-the-Air-Updateprozesses vernetzter Kraftfahrzeuge

Zur Erlangung des akademischen Grades eines
**Doktors der Ingenieurwissenschaften (Dr.-Ing.)**
von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)
angenommene

**Dissertation**

von

M. Sc. Roland Herberth

| | |
|---|---|
| Tag der mündlichen Prüfung: | 07.11.2023 |
| Referent: | Prof. Dr. rer. nat. Frank Gauterin |
| Korreferent: | Prof. Dr.-Ing. Hans-Christian Reuss |

# Kurzfassung

Digitalisierung und Elektrifizierung des Fahrzeugs sind die Megatrends im Automobilbereich des frühen 21. Jahrhunderts. Die rasante Entwicklung künstlicher Intelligenz ist dabei ein zentraler Treiber aktueller und zukünftiger Themen wie Emissionen, Effizienz und autonomes Fahren. Damit einhergehend ist ein stetig zunehmender Anstieg des heute schon erheblichen Entwicklungs- und Wartungsanteils an Software und Algorithmen im Fahrzeug. Somit ist neben der eigentlichen Entwicklung auch die kontinuierliche Pflege und Wartung dieser Software, aber auch das Nachladen neuer Funktionen in Zukunft essentiell. Immer kürzere Innovationszyklen und Modellpflegen verstärken diesen Bedarf zusätzlich.

Per Fernzugriff eingespielte Softwareupdates ermöglichen es, den Bedarf an Software auch über mehrere Jahre hinweg aktuell zu halten. Dabei eröffnen Mobilfunkstandards wie 5G neue Perspektiven für die technische Umsetzung. Während jedoch das Einspielen von Softwareupdates in der Unterhaltungselektronik (zum Beispiel Smartphones) ein gängiger Standard ist, ist dies bei Fahrzeugen bisher noch nicht üblich. Aus diesen Over-the-Air-Softwareupdates im Automobilbereich resultieren somit gleichermaßen neue Potentiale sowie Herausforderungen. So erfolgt das Einspielen neuer Software bis dato nur über den fahrzeuginternen Diagnosezugang, das in einer gesicherten Umgebung (Werkstatt), von qualifiziertem Personal durchgeführt wird. Dieser Prozess verlagert sich durch Over-the-Air-Updates hin zum Fahrzeughalter, was beispielsweise ein verkehrssicheres Abstellen des Fahrzeugs erfordert. Des Weiteren ist die gesamte Software im Fahrzeug dezentral verortet. Die rund 100 Millionen Zeilen Code eines Oberklasse-

fahrzeugs sind auf bis zu 100 einzelne Kleincomputer, sogenannte Steuergeräte, verteilt. Weiterhin müssen sowohl die Energieversorgung, als auch die Einschränkung der Mobilität bei der Fahrzeugnutzung während des Installationsvorgangs berücksichtigt werden.
In der vorliegenden Arbeit werden zunächst die durch Over-the-Air-Updates in Fahrzeugen entstandenen Potentiale als auch Herausforderungen ausgearbeitet und aufgezeigt. Aus diesen Erkenntnissen wird eine Reduktion der Updatedauer durch eine effiziente Parallelisierung der einzelnen Updateschritte über mehrere Steuergeräte näher betrachtet. Konkret wird dabei eine parallele Anweisung der Einzelupdates mit Hilfe mathematischer Optimierungsverfahren berechnet, wodurch das Erreichen der optimalen Zeitersparnis sichergestellt ist. Entsprechende Ergebnisse und Vorteile werden diskutiert und in die wissenschaftliche Literatur eingeordnet. Um der Mobilitätseinschränkung entgegenzuwirken, erfolgt im Anschluss die Betrachtung eines Konzepts zur automatisierten Einplanung des Installationszeitpunktes. Dazu wird das individuelle Fahrzeugnutzungsverhalten eines Fahrzeughalters erlernt und prognostiziert, auf der die Einplanung selbst aufbaut. Abschließend wird aus der Reduktion der Updatedauer und automatisierten Einplanung des Updatevorgangs ein Gesamtprozess abgeleitet und durch ein Beispiel veranschaulicht.
Hervorzuheben ist, dass bei der Ausarbeitung kein technisches Konzept für Over-the-Air-Updates im Fokus steht, sondern die Optimierung des Over-the-Air-Updatesprozesses aus Nutzersicht. Verfolgt wird somit die grundlegende Prämisse, die Akzeptanz als auch Zufriedenheit des Fahrzeughalters gegenüber Over-the-Air-Softwareupdates im Fahrzeug zu erhöhen.

# Abstract

Digitalization and electrification of vehicles are early 21st century megatrends in the automotive sector. Driven by the rapid development of artificial intelligence, topics of future mobility such as emissions, efficiency and autonomous driving are in focus. Consequently, a further increase in the already considerable development and maintenance share of software and algorithms in the vehicle takes place. Thus, in addition to the actual development, the continuous maintenance and care of this software, but also the reloading of new functions, will be essential in the future. Shorter and faster innovation cycles as well as model maintenance are further reinforcing this need.

Remote software updates keeping up the need for up-to-date software over several years. Mobile communication standards such as 5G open up new perspectives for technical implementation. However, while the import of software updates in consumer electronics (e.g. smartphones) is a common standard, it is not yet common for vehicles. In the automotive sector these over-the-air-software-updates thus offer both new potential and challenges. Up to now, new software has only been installed via the vehicle's internal diagnostic access, which is carried out by qualified personnel in a secure environment (e.g. workshop). This process is shifted to the vehicle owner by over-the-air-updates, who, for example, requires the vehicle to be parked safely. Furthermore, the entire software in the vehicle is not centrally located. Approximately 100 million lines of code which are needed for a luxury class vehicle are distributed among up to 100 individual small computers, known as electronic control units. The energy supply, as well as the limita-

tion in mobility (vehicle usage) must also be taken into account during the installation process.
This thesis elaborates and highlights the potentials and challenges resulting from over-the-air-updates in vehicles. Based on these findings, a reduction of the update time by an efficient parallelisation of individual update steps over several electronic control units will be considered in more detail. In concrete terms, a parallel instruction of individual updates is calculated with the aid of mathematical optimization procedures, which ensures that the optimum time saving is achieved. Corresponding results and advantages are discussed and classified in scientific literature. In order to react to the limitation of mobility, a concept for automated scheduling of installation time is considered. For this purpose, the individual vehicle usage behaviour of a user is learned and predicted, on which in turn the planning itself is based. Finally, an overall process is derived from the reduction of update time and automated scheduling of the update process which is illustrated by an example.
To be highlighted, the focus of this elaboration is not on a technical concept for over-the-air-updates, but rather the optimization of the over-the-air-update-process from the user's point of view. The basic premise is therefore to increase the acceptance and satisfaction of the vehicle owner with over-the-air-software-updates in the vehicle.

# Danksagung

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Fahrzeugsystemtechnik des Karlsruher Instituts für Technologie sowie als Doktorand im Bereich 'Over-the-Air Diagnose' bei der Dr. Ing. h. c. Porsche AG.
Herzlich möchte ich mich bei meinem Doktorvater Herrn Prof. Dr. rer. nat. Frank Gauterin für die wissenschaftliche Betreuung, das entgegengebrachte Vertrauen und nicht zuletzt die anregenden Diskussionen bedanken. Herrn Prof. Dr.-Ing. Hans-Christian Reuss danke ich für das Interesse an meiner Arbeit sowie die Übernahme des Korreferats. Des Weiteren gilt mein Dank auch Herrn Dr.-Ing. Michael Frey für den stets hilfreichen wissenschaftlichen Austausch. Für die Übernahme des Prüfungsvorsitzes danke ich ebenfalls Frau Prof. Dr. Dr.-Ing. Dr. h. c. Jivka Ovtcharova.
Bei der Porsche AG gilt mein aufrichtigster Dank Herrn Dr. Markus Koch für die Ermöglichung und Förderung meines Promotionsvorhabens. Den Arbeitskollegen und Freunden aus der Diagnosefachabteilung, insbesondere Sidney Körper, Leonhard Menz, Tim Stiesch, Gerald Zoppelt und Sven Conradi möchte ich für unzählige inspirative Anregungen, kritische Anmerkungen und die tolle Zusammenarbeit danken. Besonderer Dank gilt auch meinen Eltern Erika und Martin Herberth für die jahrelange Förderung und Unterstützung als Studierender sowie beim Einstieg in das Berufsleben. Der größte Dank geht an meine Frau Helena Herberth für ihre unermüdliche Motivation, Unterstützung und Fürsorge.

Karlsruhe, im September 2019 *Roland Herberth*

# Inhaltsverzeichnis

# 1 Einleitung und Zielsetzung

Viele der heutigen Alltagsgegenstände realisieren ihre Funktionen auf Basis zahlreicher Algorithmen und Softwarekomponenten. Gegenwärtige Unterhaltungselektronik[1] wie Smartphones, Computer, Fernseher, Spielekonsolen, Saugroboter, Smartwatches, aber auch Heimautomatisierungen[2] verfügen dementsprechend über eine Vielzahl softwarebasierter Funktionsumfänge. Aufgrund des täglichen Umgangs ist auch die kontinuierliche Pflege und Wartung dieser Geräte in Form von Softwareupdates und -upgrades beim Anwender etabliert. Eine stetige Aufwertung neuer Funktionen und Features zur Erweiterung des Produktumfangs erfolgen durch *Upgrades*. Im Gegenzug ermöglichen *Updates* Fehlerbehebungen, Optimierungen der Performance, sowie das Schließen von Sicherheitslücken. Sowohl Upgrades als auch Updates sind bereits über Jahrzehnte hinweg elementarer Bestandteil der Qualitätssicherung von Produkten. Da im allgemeinen Sprachgebrauch meist keine Unterscheidung zwischen den beiden Begriffen erfolgt, werden Upgrades und Updates im weiteren Verlauf unter *Softwareupdates* zusammengefasst. Im Gegensatz zur Unterhaltungselektronik sind per Fernzugriff eingespielte Softwareupdates im Automobilbereich noch kein gängiger Standard. Dies wird sich jedoch mit den kommenden Fahrzeuggenerationen aufgrund der Vernetzung mit dem Internet[3] zunehmend ändern.

[1] Consumer Electronics
[2] Smart Home
[3] Connected Car

## 1.1 Software in Kraftfahrzeugen

Speziell in Kraftfahrzeugen ist der Softwareanteil in den letzten Dekaden signifikant gestiegen. Bis zu 100 Millionen Zeilen Quellcode [28] sind in Oberklassefahrzeugen auf bis zu 100 softwarebasierten Kleincomputern, sogenannten Steuergeräten[4], verteilt [44, 58]. Sie berechnen und steuern eine Vielzahl elementarer Fahrzeugfunktionen, die in früheren Fahrzeuggenerationen mechanisch gelöst wurden. Diese Elektrifizierung - in Kombination mit Software - war und ist ein wesentlicher Treiber in puncto Verbrauchsreduktion durch einen optimierten Verbrennungsprozess, Verkehrssicherheit aufgrund Fahrerassistenzsystemen oder Benutzerfreundlichkeit wie beispielsweise Sprachsteuerung.
Auch die zukünftige Entwicklung der Automobilbranche ist durch einen zunehmenden Digitalisierungsgrad und den gleichzeitigen Wandel des Mobilitätsverhaltens geprägt. Digitalisierungs- und Mobilitätstrends werden hierbei zukünftig verstärkt als Differenzierungsmerkmal innerhalb der Fahrzeugklassen und im Konkurrenzumfeld eingesetzt. Die relevantesten *Digitalisierungstrends* (1-2) und *Mobilitätstrends* (3-4) sind unter anderem:

1. *Konnektivität:* Die zunehmende Vernetzung des Automobils mit seiner Umwelt dient zur Interaktion mit anderen Verkehrsteilnehmern (C2C)[5] oder IT-Infrastrukturen (C2I)[6]. Damit einhergehend sind die auf Big Data basierenden digitalen Dienste, wie Echtzeit-Navigationsdaten oder die Aggregation beziehungsweise Verteilung von Schwarmdaten. Letzteres kann beispielsweise zur Warnung vor Aquaplaning genutzt werden. Softwareupdates können über diesen Kommunikationsweg via IT-Backendsystemen per Luftschnittstelle ortsunabhängig eingespielt werden.

[4] Electronic Control Unit (ECU)
[5] Car-to-Car
[6] Car-to-Infrastructure

2. *Autonomes Fahren:* Heutige Fahrerassistenzsysteme, beispielsweise Spurhalteassistenten oder automatische Bremsassistenten, werden schrittweise zum (hoch)automatisierten Fahren aggregiert und ausgebaut. Die langfristige Zielsetzung umfasst das vollständig autonome Fahren. Methoden der *künstlichen Intelligenz* (KI), wie zum Beispiel *maschinelles Lernen* (ML), werden vermehrt zur Realisierung der Funktionen eingesetzt.
3. *Elektromobilität:* Die kontinuierliche Elektrifizierung des Antriebsstrangs in Form von Plug-in-Hybriden[7] und rein batteriebetriebenen Elektrofahrzeugen[8], erfordert intelligente Betriebsstrategien zur Maximierung der Reichweite unter Berücksichtigung von Route, Fahrweise, Rekuperation und Ladevorgängen. Die neuen Antriebskonzepte ermöglichen gleichzeitig flexiblere Regelungsstrategien zur dynamischen Lastverteilung der Antriebsräder[9].
4. *Car Sharing:* Car Sharing findet insbesondere in innerstädtischen Gebieten mit kritischer Parkplatzsituation verstärkt Anwendung. Insgesamt ist ein stetiger Zuwachs beziehungsweise Wandel zu Mobilitätsdienstleistungen ersichtlich. Damit einhergehend etablieren sich neuartige Vertriebsmöglichkeiten durch Abo-Modelle und App-basierte[10] Steuerungen sowie Bezahlsysteme.

Die aufgezählten Aspekte verdeutlichen, dass zukünftige Innovationen im Automobilbereich noch intensiver aufgrund elektronischer und softwarebasierter Systeme realisiert werden. Folglich ist zu erwarten, dass der Softwareanteil in Zukunft weiterhin an Relevanz und Umfang zunehmen wird. Konsequenterweise weckt dieser Trend im Fahrzeug den selben Bedarf an Wartung, Funktionserweiterung und Pflege der Softwarekomponenten analog zur bereits üblichen Unterhaltungselektronik. Im Zuge der Einfüh-

[7] Plug-in Hybrid Electric Vehicle (PHEV)
[8] Battery Electric Vehicle (BEV)
[9] Torque Vectoring
[10] Application (Anwendungssoftware auf dem Smartphone)

rung von Connected Car ist daher das Einspielen von *Over-the-Air-Updates* (OTA-Updates) in Kraftfahrzeugen erforderlich. Folglich ist der Entwurf von Konzepten zur Realisierung dieser OTA-Updates in der Automobilbranche Forschungsgegenstand der letzten Jahre [74, 91, 99, 119].

## 1.2 Bedarf an OTA-Softwareupdates

OTA-Updates sind nicht ausschließlich ein Mittel zur Erfüllung des steigenden Wartungsbedarfs oder ein Werkzeug zur Bewältigung der zunehmenden Komplexität. Durch das breite Spektrum an Anwendungsfällen lassen sich Vorteile und Optimierungen entlang des gesamten Fahrzeugentstehungs- und Fahrzeuglebenszyklus ableiten. Diese erstrecken sich dementsprechend über alle Ressorts eines Fahrzeugherstellers[11]. So sind insbesondere Kosten- und Zeiteinsparungen in der Produktion, als auch ein flexiblerer Kundenservice im After-Sales realisierbar. Einige der wichtigsten Vorzüge durch OTA-Updates entlang des Fahrzeugentstehungs- und Fahrzeuglebenszyklus zeigt Abbildung 1.1.
Grundsätzlich wird eine *Harmonisierung der Fahrzeug- und Digitalen-Updatezyklen* ermöglicht. Megatrends, wie (hoch)automatisiertes Fahren, wecken den Bedarf an kontinuierlicher sowie zeitnaher Optimierung von Algorithmen und Methoden. Auch das Kartenmaterial von Navigationsgeräten ist bereits nach mehreren Monaten teilweise veraltet und bedarf einer Aktualisierung. Beides verdeutlicht die Signifikanz, sich zukünftigen Anforderungen bezüglich Flexibilität, Verfügbarkeit und Funktionalität anzupassen. Mit den in der Automobilindustrie gängigen Modellpflegezyklen von 4 bis 7 Jahren kann dieser Bedarf nicht mehr gedeckt werden [39]. Per Fernzugriff eingespielte Softwareupdates ermöglichen bedarfsgerechte Updatezyklen, angepasst an die Entwicklungszyklen der heutigen Unterhaltungselektronik, die etwa 6 bis 12 Monate betragen.

[11] Original Equipment Manufacturer (OEM)

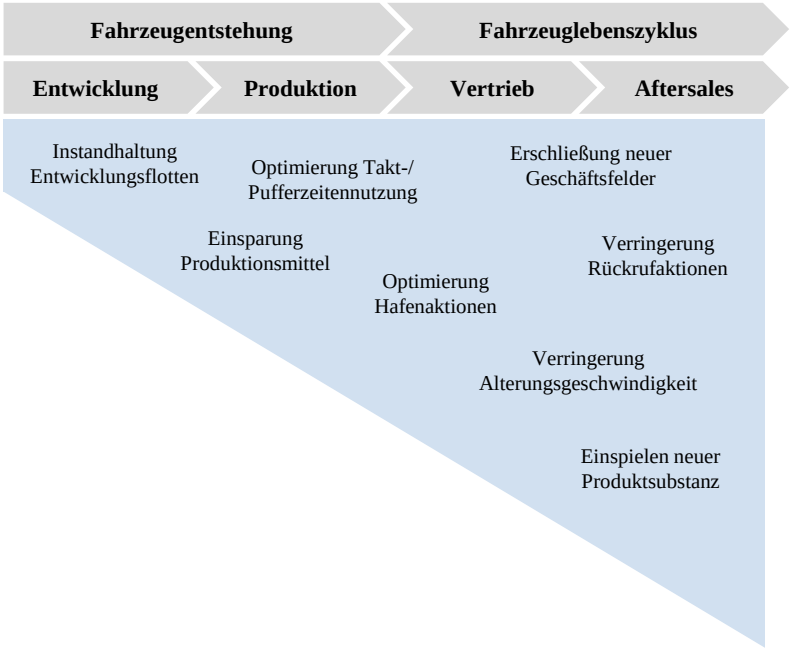



Abbildung 1.1: Vorteile von OTA-Softwareupdates über den gesamten Fahrzeugentstehungs- und Fahrzeuglebenszyklus. Eine Hafenaktion umfasst das letztmalige Softwareupdate bereits produzierter Fahrzeugflotten vor Übergabe an den Vertrieb. Der Hafen symbolisiert dabei den Abstellort der Fahrzeugflotte.

Des Weiteren kann eine *Reduktion von Rückrufaktionen* erzielt werden. Häufige Rückrufaktionen aufgrund technischer Fehlfunktionen verursachen bei Fahrzeugherstellern hohe logistische Aufwände, enorme Kosten und nachhaltige Imageschäden. Für den Fahrzeughalter bedeuten Rückrufaktionen einen ein- bis mehrtägigen Verzicht auf das Fahrzeug sowie organisatorischer Aufwand bezüglich Übergabe- und Abholterminen mit der Werkstatt. Aus Statistiken der letzten Jahre lässt sich ein genereller Zuwachs an Rückrufaktionen entnehmen, was unter anderem auf eine steigende Fahrzeugkomplexität und kürzere Entwicklungszyklen zurückzuführen

ist. Insbesondere Fehler im Zusammenhang mit Software treten verstärkt auf [8, 25]. Durch die Standortunabhängigkeit von OTA-Updates können rein softwarebedingte Rückrufaktionen zukünftig gänzlich vermieden werden.

OTA-Updates ermöglichen auch die *Erschließung neuer Geschäftsfelder*. Neue Dienstleistungsangebote, wie zum Beispiel das Nachladen von temporär oder dauerhaft nutzbaren Funktionen[12], erschließen zukünftige Ertragsquellen für Fahrzeughersteller über den gesamten Fahrzeuglebenszyklus hinweg. Gleichzeitig erfüllen sie die gestiegene Erwartungshaltung des Nutzers bezüglich Digitalisierung und Individualisierung des Fahrzeugs. Damit einhergehend sind sowohl eine Steigerung der Kundenbindung, als auch eine erhöhte Werterhaltung durch die Verringerung der Alterungsgeschwindigkeit von Fahrzeugen.

Weitere Vorteile ergeben sich durch die *Optimierung von Produktionsprozessen*. Im Bereich der Produktionsplanung und -logistik ermöglicht das kabellose Einspielen von Software einen erhöhten Grad an Flexibilität und Effizienz. Dieser kann unter anderem durch den Entfall von Taktzeiten (Pufferzeiten) für den Updateprozess oder eine Parallelisierung von Arbeitsschritten erzielt werden. Die Einsparung von Produktionsmitteln dank zentralen IT-Servern, statt lokaler Testgeräte und Kommunikationsinterfaces[13], senkt zudem dauerhaft Kosten.

## 1.3 Problembeschreibung und Aufgabenstellung

Die aufgezählten Vorteile verdeutlichen den grundlegenden Bedarf an OTA-Updates in der Automobilindustrie. Im Gegensatz zur Unterhaltungselektronik, in der Softwareupdates fest etabliert sind, steht die Automobilindustrie, inklusive Fahrzeughalter, dem Thema Softwareupdates im Fahrzeug

[12] Function on Demand (FoD)
[13] Vehicle Communication Interface (VCI)

weitestgehend ohne Erfahrung gegenüber. Damit einhergehend entsteht der Wunsch beziehungsweise die Erwartungshaltung Mechanismen der Unterhaltungselektronik auf das Automobil zu übertragen. Dies gilt insbesondere für Zuverlässigkeit und intuitive Benutzerfreundlichkeit.
Fahrzeugverfügbarkeit, Nutzerinteraktion, Nutzerakzeptanz und Prozesssicherheit sind sowohl Herausforderungen als auch Potentiale, die sich zwangsläufig durch OTA-Updates im Automobilbereich ergeben. So steigt beispielsweise die Fahrzeugverfügbarkeit, da für ein Softwareupdate kein Werkstatttermin mehr erforderlich ist. Da die Fahrzeugfunktionen jedoch während des Updatevorgangs nur teilweise oder gar nicht zur Verfügung stehen, verringert sich diese gewonnene Fahrzeugverfügbarkeit wiederum. Gleichzeitig wird der Updateprozess durch die Verlagerung zum Fahrzeughalter Teil der Benutzererfahrung[14].
Es gilt somit primär die Potentiale und Herausforderungen, die im Zusammenhang mit der Einführung von OTA-Updates im Kraftfahrzeug entstehen, zu identifizieren. Auf diesem Fundament basierend sollen zum einen sowohl Maßnahmen zur Bewältigung der Herausforderungen, als auch Sicherstellung der Potentiale systematisch erarbeitet werden. Abschließend müssen diese Maßnahmen in Konzepte überführt und ausgearbeitet werden. Konkret sollen dabei die Updatedauer reduziert und der Updatezeitpunkt automatisiert ermittelt werden.
Der Fokus dieser Dissertation liegt somit nicht auf der Realisierung eines technischen Systementwurfs zur Ermöglichung von OTA-Updates. Vielmehr sollen Möglichkeiten zur Optimierung des OTA-Updateprozesses im Automobilbereich ausgearbeitet werden, mit der übergeordneten Prämisse die Nutzerakzeptanz zu steigern.

[14] User-Experience

## 1.4 Struktur der Arbeit

Kapitel 1 zeigt die Motivation und Problemstellung der OTA-Softwareupdates in Kraftfahrzeugen auf. Darauffolgend vermittelt Kapitel 2 die Hintergrundinformationen zu Elektronik und Software in Kraftfahrzeugen im Kontext der Fahrzeugdiagnose. Hierbei werden die Elektrik/Elektronik-Architektur, gängige Diagnosestandards, sowie der Updatevorgang von Steuergeräten vertiefend betrachtet. Beschrieben wird sowohl die zentrale Business-Logik, als auch die daraus resultierende gesamtheitliche Wirkkette des OTA-Updateprozesses.
Kapitel 3 behandelt die strukturierte Herleitung potentieller Optimierungsparameter des OTA-Updateprozesses. Die daraus abgeleiteten Maßnahmen und Konzepte werden anschließend in die aktuelle wissenschaftliche Literatur eingeordnet. Etwaige Lücken und offene Fragestellungen aus der Literatur werden dabei identifiziert und bilden den Rahmen dieser Dissertation. Einige Inhalte dieses Kapitels konnten bereits in verkürzter Form erfolgreich publiziert werden [55]:

> R. Herberth, M. Koch, F. Gauterin, "Kundenorientierte Selbstoptimierung des OTA-Updateprozesses vernetzter Kraftfahrzeuge", Diagnose in mechatronischen Fahrzeugsystemen (IAD), May 16th-17th 2017, Dresden, SN, Germany, 2017, Seiten 13-30.

Die ausgewählten Optimierungspotentiale werden jeweils in Kapitel 4 und Kapitel 5 ausgearbeitet und anschließend evaluiert. Eine Reduktion der Updatedauer aufgrund einer effizienten und automatisierten Parallelisierung einzelner Steuergeräteupdates pro Fahrzeug ist Bestandteil des Kapitels 4. Konkret erfolgt hierbei eine Abstrahierung des Updatevorgangs, woraufhin zwei Verfahren zur Lösung des parallelen Scheduling-Problems verglichen werden. Einige Inhalte des Kapitels 4 wurden bereits in verkürzter Form erfolgreich veröffentlicht [56]:

> R. Herberth, S. Körper, T. Stiesch, F. Gauterin, O. Bringmann, "Automated Scheduling for Optimal Parallelization to Reduce the Duration of Vehicle Software Updates", IEEE Transactions on Vehicular Technology (TVT), March 2019, Volume 68, Issue 3, Pages 2921-2933.

Eine automatisierte Einplanung des OTA-Updates auf Basis des gelernten und prognostizierten Fahrzeugnutzungsverhaltens wird in Kapitel 5 beschrieben. Auch hier konnten bereits Inhalte des Kapitels 5 in verkürzter Form erfolgreich veröffentlicht werden [57, 93]:

> R. Herberth, L. Menz, S. Körper, C. Luo, F. Gauterin, A. Gerlicher, Q. Wang, "Identifying Atypical Travel Patterns for Improved Medium-Term Mobility Prediction", IEEE Transactions on Intelligent Transportation Systems (T-ITS), December 2020, Volume 21, Issue 12, Pages 5010-5021.

> L. Menz, R. Herberth, C. Luo, F. Gauterin, A. Gerlicher, Q. Wang, "An improved method for mobility prediction using a Markov model and density estimation", IEEE Wireless Communications and Networking Conference (WCNC), April 15th-18th, Barcelona, BA, Spain, 2018.

In Kapitel 6 erfolgt die Überführung der bisher autark betrachteten Konzepte in einen durchgängigen Gesamtprozess. Die zusammenhängende Funktionsweise wird anhand eines konkreten Beispiels verdeutlicht. Die abschließende Zusammenfassung inklusive Ausblick behandelt Kapitel 7. Einen Überblick über die Struktur und sowie Zusammenhänge der einzelnen Kapitel fasst Abbildung 1.2 zusammen.

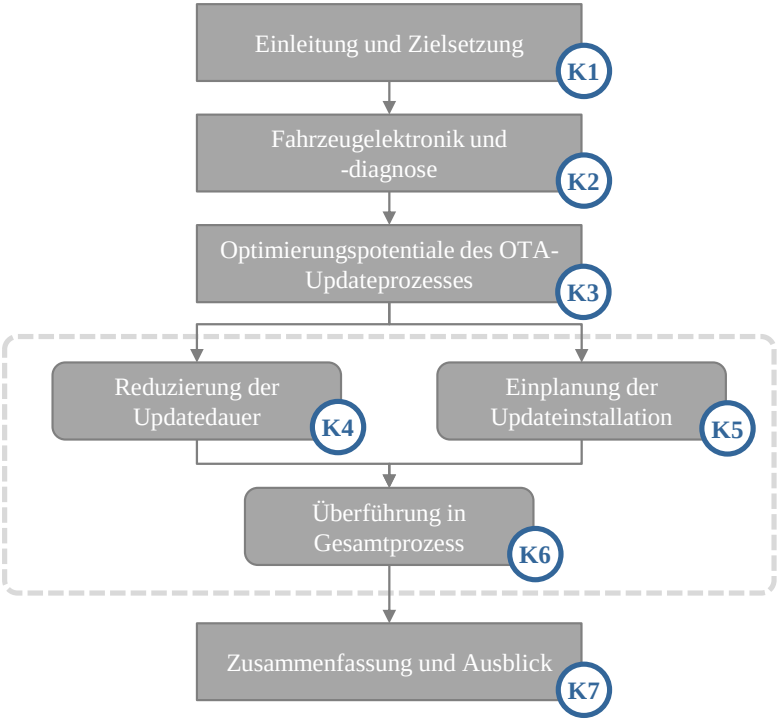



Abbildung 1.2: Gliederung der Arbeit. Die Nummerierungen in den Kreisen beziehen sich auf das jeweilige Kapitel.

# 2 Fahrzeugelektronik und -diagnose

Dieses Kapitel vermittelt die technologischen Hintergrundinformationen und Grundlagen zum allgemeinen Verständnis von Fahrzeugelektronik, Fahrzeugdiagnose und OTA-Softwareupdates in vernetzten Kraftfahrzeugen. In Abschnitt 2.1 erfolgt die Beschreibung des Aufbaus heutiger und zukünftiger Elektrik/Elektronik-Architekturen. Grundlagen zum technischen und funktionalen Hintergrund der Fahrzeugdiagnose sowie dem Updateprozess von Steuergeräten werden in Abschnitt 2.2 vermittelt. Ein Systementwurf und die gesamtheitliche Wirkkette von OTA-Updates werden letztlich in Abschnitt 2.3 beschrieben.

## 2.1 Fahrzeugelektronik und Steuergeräte

Der Großteil heutiger Fahrzeugfunktionen wird mit Hilfe von Softwareapplikationen realisiert. Diese Software ist über ein fahrzeuginternes Netzwerk, auch *Elektrik/Elektronik-Architektur* (E/E-Architektur)[1] genannt, mit bis zu 100 Steuergeräten zur Steuerung und Regelung aller Fahrzeugfunktionen verteilt. Bei einem Steuergerät handelt es sich um einen auf einen spezifischen technischen Anwendungsfall optimierten Mikrocomputer. Das *Eingabe-Verarbeitung-Ausgabe-Prinzip* (EVA-Prinzip) stellt vereinfacht die grundlegende Funktionsweise eines Steuergeräts dar. Darüber hinaus muss es Anforderungen im Kontext der Fahrzeugdiagnose erfüllen. Zusammengefasst hat ein Steuergerät somit folgende Funktionen [92]:

[1] In-Vehicle Network (IVN)

1. *EVA-Prinzip*: Das EVA-Prinzip beschreibt die Verarbeitung eingehender Sensor- und Kommunikationsdaten, Steuerung beziehungsweise Regelung der Aktoren sowie Senden generierter Daten an das verbundene Bussystem.
2. *Onboard-Diagnose*: Die Prüfung der fahrzeuginternen Sensoren beziehungsweise Aktoren hinsichtlich Funktionalität sowie Plausibilität wird Onboard-Diagnose genannt. Die Prüfung der Sensoren erfolgt beispielsweise anhand des gemessenen Spannungsbereichs. Im Falle eines detektierten Fehlers wird ein Eintrag in den Fehlerspeicher geschrieben.
3. *Offboard-Diagnose*: Die Offboard-Diagnose beschreibt die Interaktion eines Diagnosetesters mit dem Fahrzeug beziehungsweise den darin verbauten Steuergeräten. Sie dient zum aktiven Abfragen von Informationen oder Schreiben von Daten in die Steuergeräte.

Sämtliche im Fahrzeug verbauten Steuergeräte kommunizieren über diverse Bussysteme miteinander. Diese Bussysteme unterscheiden sich hinsichtlich ihrer Übertragungsraten und Echtzeitanforderungen.

### 2.1.1 Struktur heutiger E/E-Architekturen

In heutigen Fahrzeuggenerationen basiert die E/E-Architektur auf einer klassischen Sterntopolgie. Als zentrale Komponente verbindet das *Gateway* (GW) alle Steuergeräte über Bussysteme miteinander. Neben der physikalischen Anbindung aller Fahrzeugsteuergeräte fungiert das Gateway zusätzlich als Übersetzer zwischen den verschiedenen Busprotokollen [58, 111]. Gängige Bussysteme sind CAN (weit verbreitet; Kompromiss zwischen Kosten und Nutzen), FlexRay (geeignet für Echtzeitanwendungen beispielsweise für sicherheitskritische Systeme), LIN (geringe Busbandbreite und Kosten) und Ethernet (hohe Busbandbreite) [127]. Die einzelnen Subbusse entsprechen dabei in den meisten Fällen den Kategorien *Fahrwerk*, *Antriebsstrang* oder *Infotainment*. Das passende Bussystem für den jeweili-

gen Subbus wird anhand der Anforderungen der Kategorie ausgewählt. Eine Kurzübersicht der Eigenschaften und Einsatzgebiete der verschiedenen fahrzeuginternen Bussysteme liefert Anhang C.1.
Ein externer Diagnosetester kann üblicherweise über eine Diagnoseschnittstelle am Gateway angeschlossen werden. In modernen Fahrzeugen ist das Bussystem für diese Diagnoseschnittstelle Ethernet mit dem Protokoll *Diagnostic over Internet Protocol* (DoIP) [4]. Zusätzlich verfügen moderne Fahrzeuge über eine Mobilfunkverbindung oder Wireless-LAN Schnittstelle für den Fernzugriff. Generell können über diese beiden Schnittstellen Softwareupdates eingespielt werden. Eine beispielhafte Konfiguration einer heute klassischen E/E-Architektur ist in Abbildung 2.1 dargestellt.

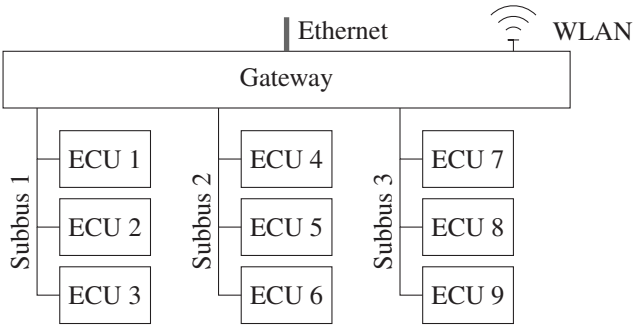



Abbildung 2.1: Heutige E/E-Architektur bestehend aus einem zentralen Gateway mit verschiedenen Sub-Bussystemen, wie zum Beispiel CAN oder FlexRay. Ein externer Diagnosetester kann per Wireless-LAN oder Ethernet angeschlossen werden (nach [56]).

### 2.1.2 Struktur zukünftiger E/E-Architekturen

Aufgrund des Digitalisierungstrends steigen auch die Anforderungen an bestehende E/E-Architekturen. Diese Anforderungen betreffen sowohl die Datenübertragungsraten der fahrzeuginternen Bussysteme, als auch die erforderlichen Rechenleistungen und Speicherbedarfe der Steuergeräte. Daher werden heutige E/E-Architekturen mit einem zentralen Gateway durch

Ethernet-Backbone-basierte Systeme ersetzt [53, 68, 120]. Mehrere Domänenrechner[2] werden dabei für rechen- und speicherintensive Anwendungen, wie künstliche Intelligenz, benötigt. Analog zu den Subbussen bei Gateway-basierten E/E-Architekturen entsprechen die Domänen dabei in den meisten Fällen den Kategorien *Fahrwerk*, *Antriebsstrang* oder *Infotainment*. Eine beispielhafte Architektur ist in Abbildung 2.2 dargestellt.

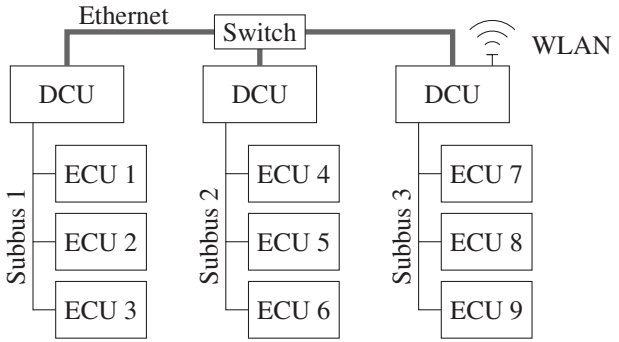



Abbildung 2.2: Zukünftige E/E-Architektur bestehend aus mehreren DCUs, welche über ein Ethernet-Backbone miteinander verbunden sind. An jede DCU sind wiederum mehrere Steuergeräte über Sub-Bussysteme, wie zum Beispiel CAN oder FlexRay, angeschlossen (nach [56]).

Ethernet mit höheren Datenraten wird für die Übertragung großer Datenmengen zwischen diesen DCUs benötigt (entspricht dem Ethernet-Backbone). Es kombiniert eine hohe Busbandbreite mit verhältnismäßig niedrigen Kosten und wird voraussichtlich das dominierende Bussystem in zukünftigen E/E-Architekturen sein [19, 67, 80]. Eine große Anzahl echtzeitfähiger Steuergeräte sowie intelligente Sensoren und Aktoren werden weiterhin Regelungs- und Steuerungsaufgaben übernehmen. Sie werden nach wie vor mit den klassischen Bussystemen wie CAN oder FlexRay miteinander verbunden sein.

[2] Domain Control Units (DCU)

## 2.2 Fahrzeugdiagnose und Updateprozess

Bereits seit Ende der 80er Jahre ist die elektronische Fahrzeugdiagnose für die Überwachung emissionsrelevanter Komponenten im Fahrzeug unter dem Begriff Onboard-Diagnose vorgeschrieben. Diese gesetzesrelevanten Anforderungen gelten für alle Fahrzeughersteller gleichermaßen, wodurch gemeinsame Standards im Laufe der Jahre herstellerübergreifend konsolidiert und etabliert wurden. Diese Standards lassen sich vereinfacht in *Standardsoftwarekomponenten* wie MCD-Kernel[3], *Open Test Sequence Exchange* (OTX) oder *Open Diagnostic Data Exchange* (ODX) [1, 2, 5] und *Kommunikationsprotokolle* wie *Unified Diagnostic Services* (UDS), *On-Board Diagnostics* (OBD), *Diagnostics over CAN* (DoCAN) oder *Diagnostics over Internet Protocol* (DoIP) [3, 4, 6, 7] untergliedern.

### 2.2.1 Diagnose-Toolkette und -Kommunikation

Die Fahrzeugdiagnose unterscheidet generell zwischen *On-* und *Offboard-Diagnose*. Die Onboard-Diagnose, auch Eigendiagnose genannt, läuft stets autark und kontinuierlich im Hintergrund der gängigen Fahrzeugoperation ab. Dabei erfolgt eine stetige Plausibilitätsprüfung der Steuergerätekommunikation sowie übermittelter Sensor- und Aktorsignale. Identifiziert die Plausibilitätsprüfung der Ein- und Ausgangssignale Abweichungen gegenüber dem Sollzustand, erfolgt ein Eintrag in den Fehlerspeicher im betroffenen Steuergerät. Diese können als Fehlerspeichereintrag[4] ausgelesen werden. Fehlerspeichereinträge, die als kritisch anzusehen sind, werden ergänzend dazu im Kombiinstrument per Warnleuchte visualisiert und somit letztendlich an den Fahrer kommuniziert.
Im Gegenzug stellt die Offboard-Diagnose eine Kommunikation über die interne Fahrzeugperipherie hinaus dar. Durchgeführt wird sie typischerweise von einem Diagnoseexperten mit Hilfe eines extern angeschlossenen Dia-

[3] Measurement, Calibration und Diagnostic-Kernel
[4] Diagnostic Trouble Code (DTC)

gnosetesters. Dieser externe Diagnosetester setzt sich aus diversen standardisierten Softwarekomponenten und Datenformaten zusammen. Als zentrale Instanz dient der MCD-Kernel nach ISO 22900 [1] mit drei Schnittstellen zur *Diagnoseapplikation* sowie zu *Diagnosedaten* und *Diagnoseprotokollen*. Auf Seiten der Diagnoseapplikation werden vermehrt austauschbare Diagnoseabläufe (Skripte) mit dem Standard OTX nach ISO 13209 [5] eingesetzt. Sie ermöglichen das Erstellen von Diagnosesequenzen auf Spezifikationsniveau bei gleichzeitiger direkter Ausführbarkeit durch den Diagnosetester. In den Diagnosedaten, beschrieben durch den Standard ODX nach ISO 22901 [2], sind sämtliche verfügbaren Diagnosedienste hinterlegt. ODX wurde als einheitliches Austauschformat eingeführt und beschreibt die Umwandlung von durch Menschen interpretierbaren Werten (wie zum Beispiel elektrische Spannung in Volt) zu Diagnoseprotokoll-Botschaften (hex) und umgekehrt. Die Schnittstellen und das Zusammenspiel der beschriebenen Softwarekomponenten innerhalb der Diagnose-Toolkette sowie die hardwareseitige Verteilung veranschaulicht Abbildung 2.3.

Bei der Offboard-Diagnose sendet der extern am Fahrzeug angeschlossene Diagnosetester seine Abfrage gezielt an ein zu diagnostizierendes Steuergerät. Über die in den Diagnosedaten hinterlegten Diagnosedienste können dabei beispielsweise Fehlerspeichereinträge (die per Eigendiagnose im Steuergerät gespeichert wurden) ausgelesen oder gelöscht werden. Konkret erfolgt der Abfrageprozess zwischen Diagnosetester und Steuergerät durch das sogenannte Request-/Response-Prinzip. Der Diagnosetester sendet die *Request-Botschaft* an das spezifische Steuergerät, woraufhin das Steuergerät den Request verarbeitet, um mit einer definierten *Response-Botschaft* zu antworten. Dies erfolgt je nach Bussystem über die verschiedenen im OSI-Schichtenmodell [116] beschriebenen Protokollebenen.

Vom Bussystem entkoppelt lassen sich die Diagnoseprotokolle zwischen herstellerspezifisch (UDS nach ISO 14229 [7]) und gesetzesrelevant (OBD nach ISO 27145 [6]) unterscheiden. Vom Bussystem abhängig sind die Transportprotokolle ISO-TP bei DoCAN nach ISO 15765 [3] auf dem CAN-

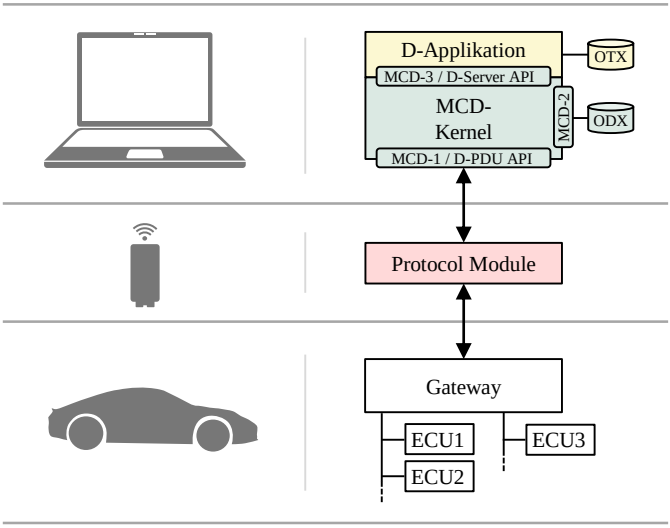



Abbildung 2.3: Komponenten, Verortung und Prinzip der klassischen Fahrzeugdiagnose mit Detailansicht der standardisierten Diagnose-Softwarekomponenten.

Bussystem und TCP bei DoIP nach ISO 13400 [4] auf dem Ethernet-Bussystem. Den kompletten Kommunikationspfad zwischen Diagnosetester und einem exemplarischen Steuergerät über die Protokollschichten hinweg zeigt Abbildung 2.4.

Die heutige Fahrzeugdiagnose umfasst jedoch weitaus mehr als nur das Setzen und Auslesen von Fehlerspeichereinträgen. Auch das Updaten (Flashen) von Steuergeräten wird durch die hier beschriebenen Diagnosemechanismen und -protokolle umgesetzt und ist daher thematisch der Fahrzeugdiagnose zugeordnet. Folglich umfasst die Fahrzeugdiagnose *lesende* (Messwerte, Fehlerspeicher), *steuernde* (Routinen) und *schreibende* (Kodieren, Flashen) Funktionsumfänge.

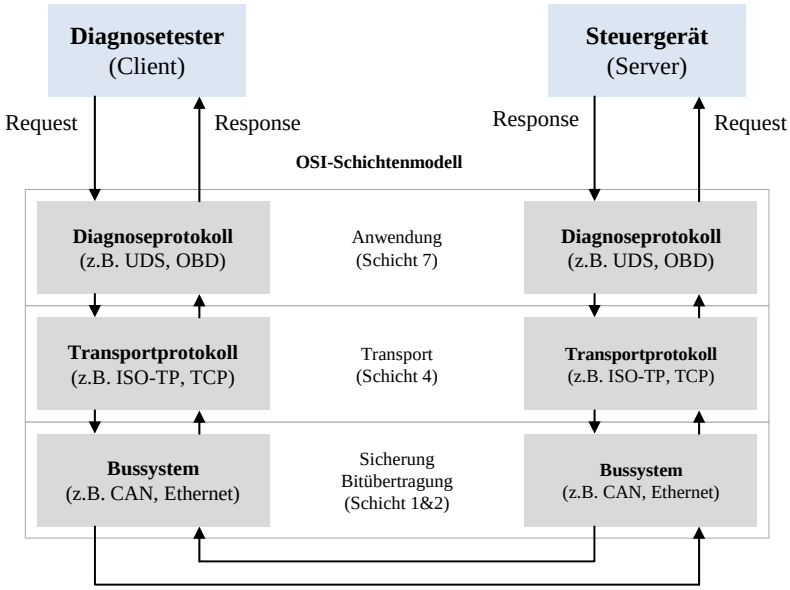



Abbildung 2.4: Prinzip der Diagnosekommunikation inklusive der Übertragungsprotokolle über das OSI-Schichtenmodell.

### 2.2.2 Updateprozess und Flashprogrammierung

Die Aktualisierung der Steuergeräte-Software ist ein dreistufiges Verfahren, das aus *Flashprogrammierung*, *Kodieren* und *Inbetriebnahme* besteht. Die Flashprogrammierung selbst (auch Flashen genannt), lässt sich wiederum in die drei Prozessschritte *Programmier-Einleitung*, *Programmier-Ausführung* und *Programmier-Ausleitung* unterteilen, was in Abbildung 2.5 verdeutlicht ist.

Während bei der Programmier-Einleitung Vorbedingungen wie *stehendes Fahrzeug* oder *Zündung aus* geprüft werden, erfolgt bei der Programmier-Ausführung das abwechselnde Löschen und Beschreiben der logischen Speicherblöcke. Dies entspricht dem eigentlichen Überschreiben der Software und unterliegt auch dem in Abschnitt 2.2.1 beschriebenen Request-

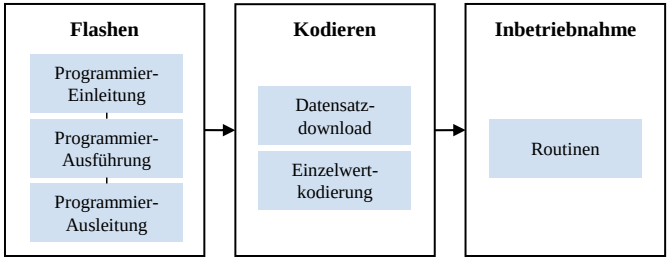



Abbildung 2.5: Teilschritte des Updateprozess mit Detailansicht der Flashprogrammierung (in Anlehnung an [126]).

/Response-Prinzip. Dieser Prozess ist der kritischste Schritt innerhalb eines Steuergeräteupdates. Im Falle eines unerwarteten Fehlers oder (ungewollten) Abbruchs enthält das Steuergerät teilweise neue und alte Softwareanteile. Diese Dateninkonsistenz/-inkompatibilität kann zu einer Funktionsunfähigkeit des Steuergeräts führen. Eine unter Umständen vorliegende Dateninkonsistenz wird unter anderem im Schritt Programmier-Ausleitung geprüft.

Nach der eigentlichen Flashprogrammierung muss das Steuergerät *kodiert* werden. Dies ist notwendig, um ein Fahrzeug an die kundenspezifische Fahrzeugkonfiguration oder länderspezifische Anforderungen anzupassen. Abschließend wird die Steuerung durch sogenannte *Inbetriebnahme-Routinen* in Betrieb genommen. Je nach erforderlicher Routine werden diverse Aktoren im Fahrzeug angesteuert. Die Teilschritte Kodieren und Inbetriebnahme sind nicht zwangsläufig bei jedem Steuergeräteupdate durchzuführen und somit optional.

Anzumerken ist, dass heutige Steuergeräte noch weitestgehend speicherbasiert (flashbasiert) sind (vgl. Abschnitt 2.1.1) und zukünftige Steuergeräte, wie Domänenrechner (vgl. Abschnitt 2.1.2), vermehrt dateibasiert (filebasiert) sein werden. Das Grundprinzip des hier beschriebenen dreistufigen Verfahrens bleibt dennoch erhalten.

## 2.3 OTA-Diagnose und OTA-Updates in vernetzten Kraftfahrzeugen

In Zeiten stetig steigender Busbandbreiten und dem Ausbau der Kommunikationsinfrastrukturen zur 5G-Konnektivität wird auch das Automobil Teil des Internet der Dinge[5] [10, 79]. Ein Fahrzeug, das über eine oder mehrere Funkschnittstellen mit seiner Umwelt verknüpft ist, wird als *Connected Car* bezeichnet. Die fahrzeugseitigen Funktionen können dank dieser Konnektivität um Over-the-Air-Dienste (OTA-Dienste) erweitert werden. Innerhalb der OTA-Dienste wird zwischen *bunten* und *grauen Diensten* unterschieden. Bunte Dienste, wie Echtzeitnavigation oder Wettervorhersage, sind für den Fahrzeughalter durch direkte Interaktion wahrnehmbar. Im Gegensatz dazu werden graue Dienste lediglich indirekt wahrgenommen, wodurch das Gebiet der Fahrzeugdiagnose überwiegend den grauen Diensten zugeordnet ist.

### 2.3.1 Abgrenzung OTA-Diagnose und OTA-Update

Zahlreiche Funktionen sind als OTA-Dienst realisierbar. Sie stellen gleichzeitig einzelne Business-Cases dar. So kann beispielsweise mit dem regelmäßigen Auslesen und Aggregieren von Fahrzeugdaten, eine Abweichung des Sollzustandes im Fahrzeug (*Realtime Service*) oder eine Vorhersage des zukünftigen Verschleißes von Systemkomponenten (*Predictive Maintenance*) getroffen werden. In beiden Fällen kann sowohl die Verarbeitung als auch die Auswertung der Informationen direkt im Fahrzeug oder bei komplexeren Rechenoperationen auf einem IT-Backend stattfinden. Insbesondere durch die Aggregation von Informationen in einem Backendsystem entsteht ein wesentlicher Vorteil, da sich Fehler mit einem ähnlichen Muster über die lokalen Grenzen des Fahrzeugs hinweg zu einem Gesamtbild einer Fahrzeugflotte zusammenfügen lassen.

[5] Internet of Things (IoT)

Tritt hingegen eine Fahrzeugpanne ein, kann dank Mobilfunkzugriff eine umgehende Analyse des Fehlerbildes (*Live-Diagnose*) erfolgen, um im gleichen Zuge eine Empfehlung zum weiteren Handeln zu geben. Diese Handlungsempfehlung kann zu einer Sofortmaßnahme, wie beispielsweise dem Neustarten eines Steuergeräts, führen. Alternativ erfolgt die Weiterleitung des Fehlerbildes an eine Werkstatt, die sich anschließend um das Pannenfahrzeug kümmert.
Schlussendlich können per Softwaredownload neue Fahrzeugfunktionen (*Function on Demand*), aber auch die Wartung und Pflege der Bestandsfunktionen (*Software Maintenance*) initiiert werden. Neue oder temporäre Funktionen (Betriebsmodi, Leistungssteigerung, etc.) dienen dabei der individuellen Personalisierung und Produktaufwertung, während die Wartung und Pflege maßgeblich zur Aktualisierung der bereits existierenden Softwarekomponenten dient.
Die beschriebenen Online-Dienste lassen sich in die beiden Begriffe *OTA-Diagnose* und *OTA-Update* einteilen. Die jeweilige Zuordnung wird in Abbildung 2.6 verdeutlicht.

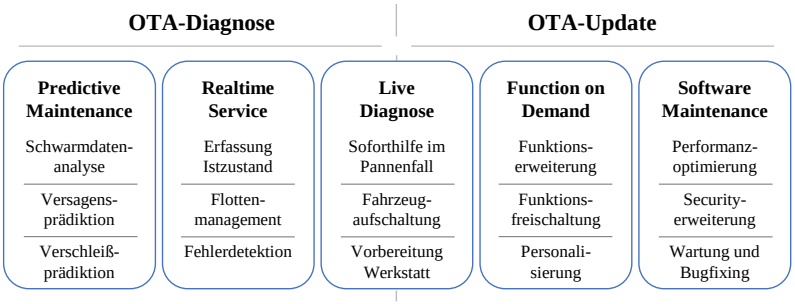



Abbildung 2.6: Gegenüberstellung und Abgrenzung von OTA-Diagnose und OTA-Update anhand der Business-Cases beziehungsweise OTA-Dienste (nach [55]).

Die OTA-Diagnose beschreibt vereinfacht ausgedrückt alle lesenden Zugriffe auf steuergeräteinterne Daten, wodurch sie sowohl *Predictive Maintenan-*

*ce* als auch *Realtime Service* umfasst. OTA-Updates entsprechen hingegen einem schreibenden Eingriff auf Steuergeräteapplikation und -daten, weshalb dieser Klassifizierung *Function on Demand* sowie *Software Maintenance* unterliegen. Ein Sonderfall ist die Live Diagnose. Aufgrund möglicher lesender und schreibender Zugriffe erfolgt die Klassifizierung sowohl zur OTA-Diagnose als auch zu den OTA-Updates. Basierend auf den Beschreibungen der OTA-Dienste im Kontext der OTA-Updates lassen sich unterschiedliche Updatecharakteristiken ableiten. Relevant sind unter anderem Anzahl der Zielsteuergeräte, Reaktionszeit, Priorität sowie die durchschnittliche Updatedauer. Tabelle 2.1 beschreibt die individuellen Merkmale und Eigenschaften der OTA-Updates bezüglich ihrer spezifischen OTA-Dienste. Das volle Spektrum der Varianz von OTA-Softwareupdates in Kraftfahrzeugen lässt sich daraus ableiten.

Tabelle 2.1: OTA-Dienst spezifische Merkmale der OTA-Updates.

| | **Live Diagnose** | **Function on Demand** | **Software Maintenance** |
|---|---|---|---|
| Umfang | 1-2 ECUs | 1-10 ECUs | 1-100 ECUs |
| Dauer | Minuten | Minuten | Stunden |
| Priorität | Hoch | Mittel | Niedrig-Hoch |
| Reaktionszeit | Minuten | Stunden-Tage | Stunden-Tage |

### 2.3.2 Systementwurf für OTA-Updates

Die Integration einer OTA-Diagnosetechnologie beziehungsweise OTA-Updatetechnologie in Kraftfahrzeuge ist seit Anfang der 2000er Jahre Gegenstand der Forschung und Entwicklung. Dabei spezialisieren sich viele der erbrachten Konzepte entweder auf die Teilumfänge OTA-Diagnose oder OTA-Updates und sind meist proprietär umgesetzt [81, 82, 85, 139]. Dadurch er-

folgt meist nur eine Teilbetrachtung der Diagnoseumfänge im Kontext *Over-the-Air*. Auf der anderen Seite ist die onboard- und offboardseitige Partitionierung der Standardsoftwarekomponenten der Diagnose-Toolkette (vgl. Abbildung 2.3) ein vielfach betrachteter Ansatz [37, 106, 125]. Dieser Ansatz wird nachfolgend näher erläutert, da er das Potential bietet, die vollen Diagnoseumfänge (sowohl OTA-Diagnose als auch OTA-Update) mit einem Konzeptentwurf abzudecken.

In der Literatur wird oftmals zwischen *synchroner* und *asynchroner* OTA-Diagnose [64, 107] unterschieden. Diese Unterscheidung wird auch in den vier nachfolgend erläuterten Partitionierungskonzepten getroffen und ist in Abbildung 2.7 verdeutlicht. Die Partitionierung der Diagnose-Standardsoftwarekomponenten erfolgt dabei im Wesentlichen zwischen *offboard* und *onboard*. Offboard entspricht einer Verortung außerhalb des Fahrzeugs, wobei die Komponenten typischerweise in ein IT-Backendsystem integriert sind. Onboard entspricht wiederum einer Integration der Standardsoftwarekomponenten innerhalb des Fahrzeugs, wobei sie typischerweise im Gateway oder einem Domänenrechner verortet sind.

*Synchrone Konzepte:* Die beiden synchronen Konzepte aus Abbildung 2.7 entsprechen einer Trennung innerhalb der standardisierten Schnittstellen der Diagnose-Toolkette. Der Ansatz *Request/Response* entspricht einer Partitionierung an der MCD-1 (D-PDU-API) Schnittstelle. Somit werden verpackte Diagnosebotschaften[6] nach dem klassischen Request/Response-Prinzip über die Funkschnittstelle verschickt. Im Ansatz *Remote Procedure* hingegen erfolgt die Trennung an der MCD-3 (D-Server API) Schnittstelle. Somit werden entfernte Methodenaufrufe, sogenannte *Remote Procedure Calls*, über die Funkschnittstelle versandt.

*Asynchrone Konzepte:* Das Konzept *Onboard-Toolchain* sieht die komplette Integration der Diagnose-Toolkette im Fahrzeug vor. Der Austausch von Daten erfolgt mittels im IT-Backend prozessierter Datenpakete (hier Job

[6] Protocol Data Units (PDUs)

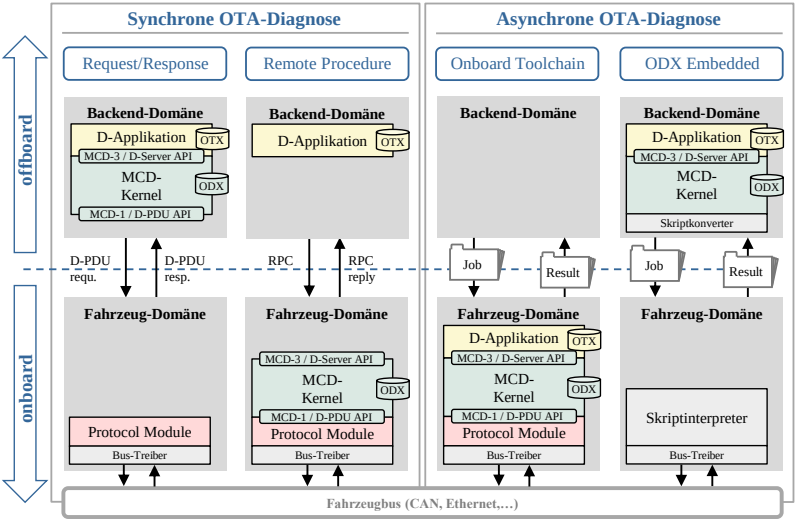



Abbildung 2.7: Vergleich von onboard und offboard Partitionierungskonzepten von Diagnose-Standardsoftwarekomponenten zur Realisierung von OTA-Diagnose und -Updates in Kraftfahrzeugen. Allgemein kann zwischen synchronen und asynchronen Konzeptansätzen unterschieden werden (in Anlehnung an [37, 106]).

genannt), welche die Installationsdaten und Aufträge zur Ausführung von Anweisungen beinhalten. Sämtliche benötigten Diagnosedienste liegen über die Diagnosedaten (ODX) vollständig im Fahrzeug vor. Das Ergebnis der Ausführung wird in einem entsprechenden Datenpaket (hier Result) rückgemeldet. Im Ansatz *ODX Embedded* ist die komplette Diagnose-Toolchain offboardseitig verortet. Durch den im Standard OTX beschriebenen Mechanismus *DiagComRaw* [5] werden die Diagnosedienste aus den Diagnosedaten (ODX) direkt in ein ausführbares Skript integriert. Im Fahrzeug selbst genügt dadurch lediglich ein schlanker, wenn auch proprietärer Skript-Interpreter. Auch hier erfolgt die Kommunikation vollständig über Datenpakete. Weitere Details zu den Partitionierungskonzepten können den Quellen [37, 106, 125] entnommen werden.

Die vier vorgestellten Ansätze bringen diverse Vor- und Nachteile mit sich, insbesondere bezüglich Integrationsaufwand, Hardwareanforderungen im Fahrzeug und Umsetzbarkeit der Diagnoseumfänge. Grundsätzlich unterliegen beide synchronen Ansätze der Problematik, eine stabile und kontinuierliche Funkverbindung vorauszusetzen, da die Kommunikation dem Prinzip der sequenzierten Anfrage und Antwort unterliegt. Eine wesentliche Herausforderung dieses Prinzips besteht folglich im Umgang mit Latenzzeiten und Verbindungsabbrüchen (Timeouts) der Übertragungsstrecke, da diese einen entscheidenden Einfluss auf die Leistungsfähigkeit und Prozesssicherheit haben. Bei den asynchronen Ansätzen hingegen ist die Kommunikation durch das Übertragen von Datenpaketen weitestgehend von Latenzen der Übertragungsstrecke unabhängig. Die Übermittlung der Informationen ins Fahrzeug und die anschließende Ausführung der Diagnoseoperation sind dadurch voneinander entkoppelt. Aus den beschriebenen Aspekten lässt sich ableiten, dass nicht alle Partitionierungskonzepte den vollständigen Funktionsumfang, im Sinne von lesender, steuernder und schreibender Diagnose, im Kontext Over-the-Air prozesssicher umsetzen können. Tabelle 2.2 liefert daher einen Überblick der Umsetzbarkeit der Diagnoseumfänge in Abhängigkeit der vorgestellten Partitionierungskonzepte aus Abbildung 2.7.

Tabelle 2.2: Gegenüberstellung von Vor- und Nachteilen der Partitionierungskonzepte bezüglich lesender, steuernder und schreibender Diagnose. Ein Plus symbolisiert einen Vorteil. Ein Minus symbolisiert einen Nachteil. Ein Kreis ist als neutral anzusehen.

| | **Request/ Response** | **Remote Procedure** | **Onboard Toolchain** | **ODX Embedded** |
|---|---|---|---|---|
| lesend | + | + | + | + |
| steuernd | o | o | + | o |
| schreibend | - | - | + | o |

Es zeigt sich, dass zur vollumfänglichen Umsetzung der Diagnosemechanismen ein asynchrones Konzept klar zu favorisieren ist. Insbesondere für schreibende Diagnoseumfänge (und damit OTA-Updates), welche im Fokus dieser Arbeit liegen, ist ein asynchroner Konzeptansatz Grundvoraussetzung. Da das Konzept *Onboard Toolchain* rein auf Standardsoftwarekomponenten beruht und im Vergleich zum Konzept *OTX Embedded* keine proprietäre Umsetzung für den vollen Funktionsumfang benötigt, ist für die weitere Betrachtung nur noch das dritte Partitionierungskonzept aus Abbildung 2.7 relevant.

### 2.3.3 Business-Logik des OTA-Updateprozesses

Abschließend soll das ganzheitliche Funktionsprinzip inklusive der einzelnen Prozessschritte von OTA-Updates in Kraftfahrzeugen erläutert werden. Der exemplarische Systementwurf basiert auf dem grundlegenden Ansatz des asynchronen Partitionierungskonzepts *Onboard Toolchain* aus Abbildung 2.7 und ist in Abbildung 2.8 dargestellt.
Der OTA-Updateprozess kann vereinfacht in drei Phasen eingeteilt werden. In der ersten Phase, dem *Softwaredownload*, werden per drahtloser Internetkonnektivität die für das Update relevanten Daten in das Zielfahrzeug übertragen. Vorgelagert erfolgt in der Backend-Domäne die Konsolidierung des Datenpakets im Paket-Management, bestehend aus OTA-Auftrag (Ablaufsteuerung), OTX-Skript (Installationsanweisung) und den eigentlichen Flashdaten. Die für das Fahrzeug relevanten Daten werden dazu per Soll-/Ist-Vergleich der Softwareversionen (Verbauzustand) im Flotten-Management ermittelt. Das Datenpaket wird anschließend von der Backend- zur Fahrzeug-Domäne übertragen. Dies geschieht ohne Einschränkung der Fahrzeugfunktionen und kann daher auch während der Fahrt über eine Mobilfunkverbindung erfolgen. Bei großen Datenmengen und zur Einsparung von Mobilfunkkosten können die Daten alternativ über eine Wireless-LAN Verbindung oder im Falle eines Elektrofahrzeugs über eine internetfähige

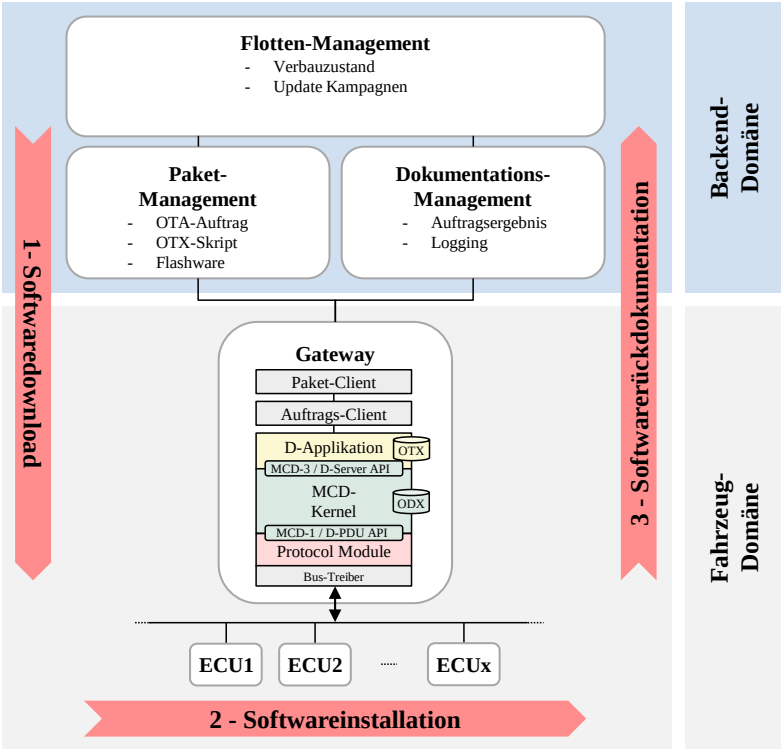



Abbildung 2.8: Fahrzeug- und backendseitige Businesslogik des OTA-Sotwareupdates.

Ladeinfrastruktur[7] bezogen werden. Die beiden letztgenannten Fälle sind jedoch ausschließlich bei einem stehenden Fahrzeug möglich.

Nach dem vollständigen Bereitstellen des Datenpakets im Fahrzeug erfolgt in Phase zwei die *Softwareinstallation* in den Zielsteuergeräten. Wie viele und welche Steuergeräte eine neue Software erhalten, hängt vom Ergebnis des Soll-/Ist-Vergleichs in der Backend-Domäne ab. Die Softwareinstallation in jedem einzelnen Steuergerät entspricht dabei wiederum den drei Teilschritten *Flashen*, *Kodieren* und *Inbetriebnahme* aus Abschnitt 2.2.2.

[7] Power Line Communication (PLC)

In dieser Phase muss das Fahrzeug vom Fahrzeughalter bei ausgeschalteter Zündung sicher abgestellt worden sein.
In der dritten und letzten Phase, der *Softwarerückdokumentation*, erfolgt eine automatisierte Konsistenzprüfung (Systemprüfung) inklusive Dokumentation der neuen Softwarestände. Dabei werden die aktuellen Software- und Hardwareversionen über das gesamte Fahrzeug konsolidiert, in einem Verbauzustandsdokument abgespeichert und anschließend in das entsprechende Backend hochgeladen. Dadurch werden die aktuellen Softwarestände der gesamten Flotte im Flotten-Management (Backend-Domäne) gespiegelt und gepflegt.
Dieser exemplarische Systementwurf beschreibt das Einspielen von OTA-Updates in Kraftfahrzeuge und ist Grundvoraussetzung für die eigentliche Optimierung des OTA-Updateprozesses.

# 3 Optimierungspotentiale des OTA-Updateprozesses

In diesem Kapitel werden die Zielgrößen, nach denen der OTA-Updateprozess optimiert werden soll, systematisch hergeleitet. Dazu erfolgt zunächst in Abschnitt 3.1 die Identifikation der Potentiale und Herausforderungen, die durch den OTA-Updateprozess aus Sicht des Fahrzeugherstellers sowie des Fahrzeughalters entstehen. Basierend auf den Herausforderungen werden in Abschnitt 3.2 die eigentlichen Prämissen sowie Anforderungen abgeleitet. Die Abschnitte 3.3 beziehungsweise 3.4 erläutern den aktuellen Stand der Forschung und Technik bezüglich der gesetzten Schwerpunkte. Schlussendlich fasst Abschnitt 3.5 den Beitrag dieser Arbeit zusammen. Einige Inhalte dieses Kapitels wurden bereits in verkürzter Form in [55, 56, 57, 93] veröffentlicht.

## 3.1 Identifizierung der Potentiale und Herausforderungen durch den OTA-Updateprozess

OTA-Updates haben in der Automobilindustrie einen gravierenden Einfluss auf den heutigen After-Sales-Prozess. Bisher finden Softwareupdates im Fahrzeug üblicherweise in vorgegebenen Serviceintervallen (kilometer- beziehungsweise zeitabhängig) oder durch explizite Rückrufaktionen statt. Sobald ein Update erforderlich ist, durchläuft das Fahrzeug in der Servicewerkstatt einen definierten (After-Sales-)Prozess: Terminvereinbarung, Übergabe des Fahrzeugs, Wartung und Instandsetzung sowie die Rückgabe des Fahr-

zeugs. Das Softwareupdate wird häufig ohne bewusste Wahrnehmung des Fahrzeughalters per externen Diagnosetester über die OBD-Schnittstelle auf die im Fahrzeug verbauten Steuergeräte eingespielt.
Für den Fall eines reinen Softwareupdates wird dieser etablierte Prozess zukünftig durch OTA-Updates ergänzt. Der eigentliche Softwareupdateprozess verlagert sich dabei von der Autowerkstatt zum Fahrzeughalter, der dadurch Teil des Updateprozesses wird. Abbildung 3.1 verdeutlicht in einer Gegenüberstellung die essentiellen Unterschiede des heutigen After-Sales-Prozess und des zukünftigen OTA-Updateprozesses.

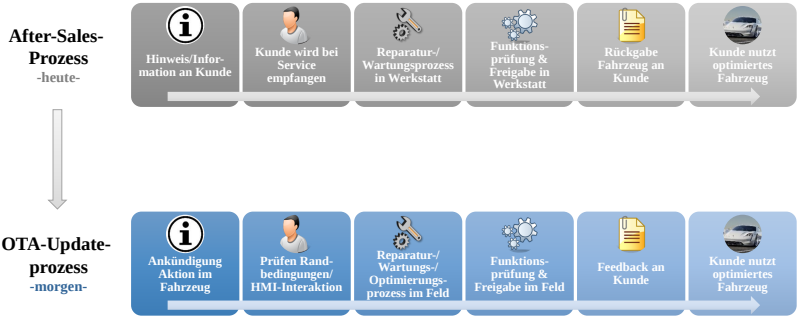



Abbildung 3.1: After-Sales-Prozess vs. OTA-Updateprozess (nach [55]).

Aufgrund dieser Differenzen ergeben sich zukünftig neben Potentialen (für Fahrzeughersteller und -halter) auch Herausforderungen, die ausschließlich für den Fahrzeughersteller relevant sind. Als offensichtlichster und gleichzeitig entscheidenster Vorteil ist die ortsunabhängige Flexibilität zu nennen. Die hinzugewonnene Flexibilität äußert sich durch den Wegfall der Terminvereinbarung sowie Fahrzeugüberführung. Letztendlich kündigt zukünftig nur noch das *Human Machine Interface* (HMI) ein Softwareupdate an. Gleichzeitig kann die Quantität der Updateintervalle optimiert werden, sodass nur minimale Einschränkungen bei der Fahrzeugverfügbarkeit entstehen. Kostenreduktion aufgrund der Zeitersparnis und automatisierter IT-Backendprozesse sind ebenfalls positive Effekte. Nichtsdestotrotz entstehen

bei OTA-Updates auch Einschränkungen im Vergleich zum After-Sales-Prozess. Diese Einschränkungen sind als Herausforderungen für die Fahrzeughersteller zu sehen. Während heutzutage eine Dialogannahme und -abgabe bei der Servicewerkstatt stattfindet, wird sich dieser Schritt zukünftig auf eine HMI-Interaktion beschränken. Dementsprechend ist ein intuitives Bedienkonzept unabdingbar, das dem Standard der aktuellen Unterhaltungselektronik entspricht. Aufgrund der Verlagerung des Updatevorgangs von der Servicewerkstatt zum Fahrzeughalter ergeben sich zwangsläufig variierende Randbedingungen. So entfällt beispielsweise die externe Spannungsversorgung, die beim Updatevorgang in der Servicewerkstatt die Fahrzeugbatterie stützt. Bei Elektrofahrzeugen könnte diese externe Spannungsversorgung durch die höhere Kapazität oder Laden der *Hochvolt-Batterie* (HV-Batterie) kompensiert werden. Allerdings ist zu beachten, dass eine HV-Batterie beim Update einer HV-Komponente aus Sicherheitsgründen nicht zur Verfügung steht. Folglich bleibt die Problematik der fehlenden externen Spannungsversorgung bestehen. Nach erfolgtem OTA-Updatevorgang müssen Prüfung und Freigabe der Funktionen zukünftig automatisiert werden. Entsprechende Einschränkungen sind durch das Fehlen des geschulten Fachpersonals zu erwarten. Neben variierenden Randbedingungen haben auch automatisierte Funktionsprüfungen teilweise gravierende Auswirkungen auf die Prozesssicherheit. Diese Sicherheit ist essentiell, da ein fehlgeschlagenes Softwareupdate ein Sicherheitsrisiko darstellen kann und beispielsweise zum Liegenbleiben des Fahrzeugs führt. Ebenfalls zu berücksichtigen ist, dass durch die Softwareinstallation eine kurzzeitige Mobilitätseinschränkung erfolgt, da alle grundlegenden Fahrzeugfunktionen während der Installationsdauer (ausgeschaltete Zündung) nur eingeschränkt oder gar nicht verfügbar sind.

## 3.2 Ableitung von Prämissen und Anforderungen

Kapitel 3.1 zeigt, dass Over-the-Air-Softwareupdates eine zeitgemäße Ergänzung des gängigen After-Sales-Prozesses sind. Allerdings müssen die bestehenden hohen Erwartungshaltungen in Bezug auf Qualität und Service beibehalten werden. Weiterführend ist daher die Aufgabe, die identifizierten Herausforderungen als Zielgrößen abzuleiten, um final Parameter zur Verbesserung für den OTA-Updateprozess zu entwickeln. Diese Parameter sind sowohl Hilfsmittel zur Bewältigung aller Herausforderungen als auch Sicherstellung des Potentials der OTA-Updatetechnologie.
Für die Mobilität sind *Installationszeitpunkt*, *Installationsdauer* und *Installationshäufigkeit* maßgeblich. Unter Berücksichtigung aller relevanten Randbedingungen kann ein individueller Zeitpunkt für ein Softwareupdate (Softwareinstallation) ermittelt und via HMI dem Fahrzeughalter vorgeschlagen werden. Für die Installationsdauer sowie Installationshäufigkeit ist der *Ressourcenbedarf*, beispielsweise der Energieverbrauch sowie die Auslastung der im Fahrzeug verbauten Bussysteme, nennenswert.
Gravierende Auswirkungen für den Fahrzeughalter können fehlgeschlagene Updates darstellen, da diese Situation gegebenenfalls das Vertrauen in die Fahrzeugqualität verringert. Vergleichbar ist das Empfinden bei einer eingetretenen Fahrzeugpanne. Daher ist die *Prozesssicherheit* ein wichtiger Aspekt, der mit einer möglichen Rückführung auf den Ursprungszustand (Rollbackstrategie) sicherzustellen ist. Schlussendlich ist der Einfluss aller äußeren, individuellen Randbedingungen, die sich auf den OTA-Updateprozess auswirken, zu erkennen und in den Gesamtprozess einzubeziehen. Mögliche Randbedingungen können fahrzeugspezifische Zustände (Parkbremse, Position Wählhebel) oder auch umgebungsbedingte Einflüsse (Mobilfunkkonnektivität, Ladeinfrastruktur) sein.
Abschließend sollen die identifizierten Parameter in konkrete Anforderungen übertragen werden. Die resultierenden Anforderungen sind in Abbil-

dung 3.2 ausformuliert. Anzumerken ist, dass einige Anforderungen in Korrelation zueinander stehen. Der Installationszeitpunkt hängt von der Installationsdauer ab. Der Ressourcenbedarf ist von der Installationshäufigkeit und der Installationsdauer abhängig. Eine Reduktion der Installationsdauer steht im Kontrast zur Reduktion der Installationshäufigkeit.

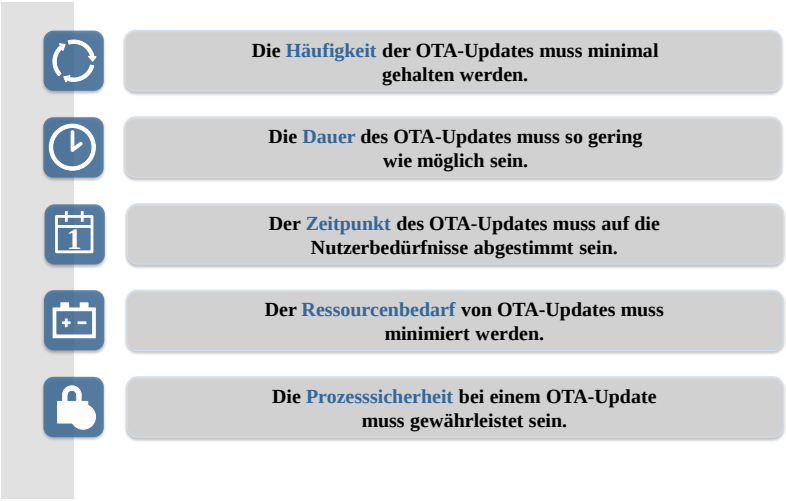



Abbildung 3.2: Abgeleitete Anforderungen an den OTA-Updateprozess.

Die beiden Schwerpunkte *Reduzierung Installationsdauer* und *Ermittlung Installationszeitpunkt* stehen in den folgenden Abschnitten, mit jeweils entsprechendem Stand der Forschung und Technik, im Fokus.

## 3.3 Stand der Forschung: Reduzierung der Dauer von Steuergeräte-Softwareupdates

Aus der Literatur lassen sich diverse Möglichkeiten zur Reduzierung der Dauer von Steuergeräte-Softwareupdates in Kraftfahrzeugen erschließen. Die gängigsten Verfahren sind dabei:

*Code Compression* (CC) basiert auf Algorithmen, die Muster und Wiederholungen in den Datenstrukturen finden und ersetzen. Das Ziel ist die Reduktion der Datenmenge, welche übertragen und neu installiert werden muss. In Abhängigkeit der Datenstrukturen (Vorhandensein an Mustern) und je nach angewendeten Ersetzungsregeln, kann der erzielte Effekt deutlich variieren. Diverse fahrzeugspezifische Algorithmen und Konzepte wurden hierzu bereits veröffentlicht [101, 102, 115]. Die erzielte Kompression erstreckt sich im Bereich von 30 % bis 50 % [102, 115]. Allerdings wurden keine konkreten Angaben zur eigentlich möglichen Reduktion der Updatedauer getroffen.

Das Konzept von *Delta Updates* (DU) entspricht der Idee, nur die Änderungen beziehungsweise Differenzen von einer Softwareversion 'X' zu einer Softwareversion 'Y' zu aktualisieren. Dadurch wird die zu installierende Datenmenge und Updatedauer reduziert, weil nicht der gesamte Speicherbereich neu beschrieben werden muss. Die bisher veröffentlichten Ergebnisse zeigen ein zeitliches Einsparungspotential von bis zu 90 % [23, 115, 128]. Allerdings hängt der erzielte Effekt bei Delta Updates stark von den Änderungen zwischen den Softwareversionen ab. Im Falle einer kompletten Softwareänderung kann folglich gar kein Vorteil in der Reduktion der Updatedauer erzielt werden.

*Parallele Updates* (PU) der einzelnen Steuergeräte sind ein weiterer effektiver Weg, um die gesamte Updatedauer in Kraftfahrzeugen zu verkürzen. Da die Softwarekomponenten auf bis zu 100 Steuergeräte pro Fahrzeug verteilt sind, zeigte die effiziente Parallelisierung von Steuergeräte-Softwareupdates ein hohes Potential zur Einsparung der gesamten Aktualisierungszeit [68, 77, 78, 115, 122, 140]. Hintergrund dieses Konzepts ist, dass die Verarbeitungsgeschwindigkeit der Steuergeräte in der Regel den Flaschenhals des Installationsprozesses darstellen und nicht die Datenrate der Bussysteme.

Die beiden Verfahren CC und DU sind unter anderem aus dem Bereich der Unterhaltungselektronik bekannt und werden hier bereits großflächig angewendet [70, 71]. Anzumerken ist, dass sich beide Verfahren gleichermaßen

auf den Softwaredownload (die zu übertragenden Datenmengen) und die Softwareinstallation (eigentliche Updatedauer) positiv auswirken (vgl. Abschnitt 2.3.3). Im Falle eines OTA-Updates kann somit von beiden Aspekten profitiert werden. Dabei hängt die mögliche Zeitersparnis von CC im Wesentlichen von den Datenstrukturen ab, während DU nur bei geringen Änderungen der Softwareanteile Vorteile erzielen kann. Somit ist im ungünstigsten Fall bei beiden Verfahren auch keine Zeiteinsparung möglich. PU hingegen ist ein rein fahrzeugspezifischer Ansatz, da hierbei eine Verteilung der Softwarekomponenten auf bis zu 100 Steuergeräte vorliegt. Schmidgall [115] vergleicht und klassifiziert gleich mehrere Verfahren (unter anderem auch CC und DU) zur Zeitreduktion von Steuergeräte-Softwareupdates. Dabei wurde das PU Verfahren mit einem hohen Potential und vergleichsweise geringem Implementierungsaufwand sowie geringen Kosten bewertet.
Die Idee ist es, diesen Ansatz der Parallelisierung aufzugreifen und so zu erweitern, dass abhängig von der ausgewählten Konstellation an Zielsteuergeräten eine automatisierte Updateanweisung (Scheduling) berechnet wird. Dabei soll bei gleichzeitiger Berücksichtigung definierter Nebenbedingungen eine optimale Anordnung der Parallelität bezüglich der Zeitersparnis bestimmt werden. Neben der eigentlichen Zeitersparnis löst dieser Ansatz gleichzeitig das Problem, dass die Updateanweisung bei einem Fahrzeugupdate über unterschiedliche Steuergerätekonstellationen ansonsten erneut und manuell bestimmt werden müsste. Die grundlegenden Schritte dieser Idee sind in Abbildung 3.3 veranschaulicht.
Dabei erfolgt zunächst die Ermittlung und Auswahl der für das Update relevanten Steuergeräte. Basierend darauf müssen alle Steuergeräteabhängigkeiten (als Nebenbedingungen) der zuvor erfolgten Auswahl identifiziert und berücksichtigt werden. Essentiell sind vor allem Reihenfolge und Exklusivität zwischen den Steuergeräte-Softwareupdates. Abschließend wird die eigentliche parallele Updateanweisung (Scheduling) berechnet und angewendet. Insgesamt liegt die zentrale Herausforderung in der optimalen parallelen Anordnung (Scheduling) aller Steuergeräte unter Einhal-

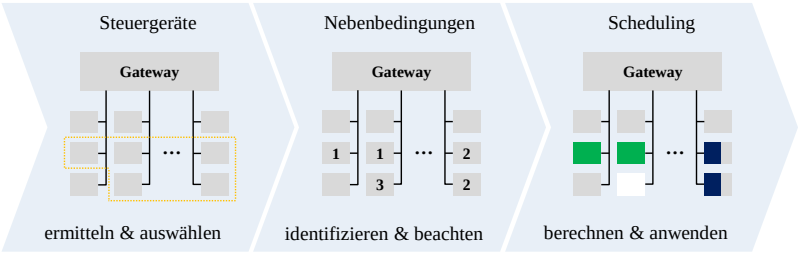



Abbildung 3.3: Idee zur Reduzierung der Updatedauer durch automatisierte Parallelisierung der Steuergeräte-Softwareupdates. Links: Die Umrandung markiert alle für das Softwareupdate relevanten Steuergeräte. Mitte: Die Zahlen geben Reihenfolge beziehungsweise Parallelität der Softwareupdates an. Rechts: Farbe und Füllgrad veranschaulicht den aktuellen Fortschritt der Softwareupdates.

tung sämtlicher Randbedingungen, was wiederum auf ein Optimierungsproblem zurückzuführen ist. Diesbezüglich bereits betrachtete Lösungsansätze (Scheduling-Algorithmen) werden daher nachfolgend näher analysiert, sowie die noch existierenden Defizite abschließend offengelegt.

Lee et al. [77] präsentieren eine parallele Methode für Steuergeräte-Softwareupdates im Gateway des Fahrzeugs. Mit drei parallelen Steuergeräten auf CAN und FlexRay wird eine Zeitersparnis von bis zu 61 % erreicht. Eine weitere parallele Methode für eine Ethernet-Backbone-basierte E/E-Architektur (vgl. Abschnitt 2.1.2) ist in [78] vorgestellt. Experimentelle Ergebnisse zeigen, dass der vorgeschlagene parallele Ansatz die gesamte Updatedauer im Vergleich zum sequentiellen Steuergeräte-Softwareupdate um mindestens 39 % reduzieren kann. Sowohl in [77] als auch [78] wird jedoch kein automatisiertes Scheduling der Parallelisierung betrachtet und angewendet. Das Scheduling erfolgt somit jeweils manuell für eine kleine Anzahl Steuergeräte. Sommer et al. [122] präsentieren und vergleichen verschiedene Scheduling-Algorithmen auf zwei Protokollschichten für den *Media Oriented Systems Transport* (MOST) Bus. Der Ansatz verwendet ein Petri-Netz, um die Softwareupdate-Aktivitäten zu synchronisieren. Die Ergebnisse zeigen ein Zeiteinsparungspotential von etwa 43 % bis 58 % im

Vergleich zur sequentiellen Update-Verarbeitung. Die Auswertung enthält jedoch nur fünf Steuergeräte auf dem MOST-Bus. Eine Aussage über die Gesamtleistung, insbesondere für verschiedene Bussysteme, ist daher nur schwer möglich. Schmidgall [115] präsentiert unter anderem ein rudimentäres Schedulingverfahren zur Parallelisierung mehrerer Steuergeräte. Der Algorithmus basiert darauf, die Busbandbreite der Bussysteme möglichst effizient beziehungsweise vollständig auszunutzen. Dafür werden die relevanten Steuergeräte anhand ihrer erwarteten Updatedauer sortiert (priorisiert) und jeweils parallelisiert, sofern noch Busbandbreite verfügbar ist. Dabei wird eine flexible beziehungsweise dynamische Datenübertragungsrate während der Flashprogrammierung pro Steuergerät vorausgesetzt. Das Prinzip wird beispielhaft für vier fiktive Steuergeräte demonstriert. In Summe wird eine theoretische Zeitersparnis von 53 % erzielt, ohne jedoch den Scheduling-Algorithmus mit realen Steuergeräten und Messergebnissen zu verifizieren. Eine Aussage über die Berechnungsdauer des Scheduling-Algorithmus selbst wird ebenfalls nicht getroffen.

Zoppelt [140, 141] hat die Parallelisierung erstmals als Optimierungsproblem aufgefasst. Ziel ist auch hier die maximale Auslastung der verfügbaren Busbandbreiten, was durch einen einfachen Scheduling-Algorithmus sichergestellt werden soll. Dazu wird die Flashprogrammierung eines Steuergerätes als rechteckige Fläche interpretiert, wobei eine Seitenlänge der Flashdauer und die andere Seitenlänge der Flashdatenmenge entspricht. Anschließend sortiert und ordnet ein Scheduling-Algorithmus die Flächen der Größe nach ein. Mit dem vorgestellten Verfahren kann jedoch keine optimale Anordnung garantiert werden. Weitere Randbedingungen, wie Updatereihenfolge der Steuergeräte oder Busbandbreiten der Sub-Bussysteme, werden ebenfalls nicht beachtet.

Die vorliegende Literatur belegt das grundlegende Potential der PUs. Allerdings ist bisher noch kein Scheduling-Algorithmus eingeführt, der eine optimale und automatisierte parallele Flashprogrammierung mehrerer Steuergeräte-Softwareupdates unter Berücksichtigung definierter Randbedingungen

ermöglicht. Bestehende Lösungen berücksichtigen zudem nur bis zu sechs Steuergeräte in ihrem parallelen Scheduling, was für kommende OTA-Updates als zu gering eingeschätzt wird. Darüber hinaus beinhaltet ein Steuergeräte-Softwareupdate nicht nur die Umprogrammierung des Flash-Speichers (Flashen), sondern auch die Kodierung und Inbetriebnahme der Software-Komponenten/Steuergeräte (siehe Abschnitt 2.2.2). Diese Teilschritte können grundsätzlich auch parallelisiert werden, um die gesamte Aktualisierungszeit zu reduzieren. Eine optimale und automatisierte Scheduling-Methodik für verschiedene Bussysteme ist daher eine Aufgabe mit höherer Komplexität sowie geringerer Aufmerksamkeit und somit Fokus der weiteren Ausarbeitung. Abschließend werden die identifizierten Defizite aus der Literatur zusammengefasst, auf denen das weitere Vorgehen basiert:

- Es ist bisher kein automatisiertes und optimales Scheduling paralleler Updateschritte bekannt.
- Existierende Betrachtungen beziehen sich auf eine geringe Anzahl an Steuergeräten im Scheduling (drei bis sechs).
- Bisher fehlen Berücksichtigungen bezüglich mehrerer Randbedingungen wie Übertragungsraten, Reihenfolgen oder Abhängigkeiten.
- Existierende Lösungen beschränken sich auf den Flashprozess. Somit ist aktuell keine Kodierung oder Inbetriebnahme im Gesamtprozess integriert.

## 3.4 Stand der Forschung: Örtliche und zeitliche Mobilitätsvorhersage

Wie bereits in Kapitel 1 erläutert, sind per Fernzugriff eingespielte Softwareupdates in vielen Bereichen der Unterhaltungselektronik alltäglicher Standard. Die Einplanung der Updateinstallation erfolgt meist statisch durch einen fest vorgegebenen, beziehungsweise durch den Benutzer manuell änderbaren Installationszeitpunkt (Datum und Uhrzeit). Nutzerindividuelle

Verhaltensmuster des zu aktualisierenden Systems fließen dabei im Regelfall nicht in die Einplanung des Updates ein.
Durch die neue Schlüsseltechnologie des maschinellen Lernens lässt sich jedoch ein vermehrtes Aufkommen von intelligenten Lernfunktionen feststellen, die auf Basis der Analyse des systemseitigen Nutzerverhaltens automatisiert Vorschläge oder Optimierungen vornehmen. Diese Helferfunktionen lassen sich in diversen Technologiefeldern, jedoch insbesondere in der Unterhaltungselektronik, finden. Beispielweise können Android-Smartphones seit der Version 9 oder höher nach Zustimmung des Nutzers die eigene Smartphone-Nutzung erfassen und daraufhin den Akkuverbrauch kontinuierlich optimieren. Konkret erfolgt entweder eine automatische Anpassung der Bildschirmhelligkeit oder Hintergrunddienste werden in Abhängigkeit der Nutzungsanalyse in den Ruhezustand versetzt [50]. Das Betriebssystem Windows 10 für Heimcomputer hingegen bietet mit dem Build 17723 seit 2018 eine Verbesserung der Update-Bereitstellung. Installation und Neustarts sollen dabei auf Grundlage der gelernten individuellen Computer-Nutzung zu einem möglichst günstigen Zeitpunkt automatisiert eingeplant und durchgeführt werden [94].
Da die Softwareinstallation eine Restriktion der Fahrzeugfunktionen und somit Mobilitätseinschränkung für den Fahrzeughalter bedeutet, ist die Idee den Ansatz der automatisierten Update-Einplanung (wie bei Windows 10) aufzugreifen und für OTA-Updates in Fahrzeugen zu adaptieren. Konkret bedeutet dies, das fahrzeugspezifische Nutzungsverhalten aufzuzeichnen, um bei Bedarf ein zukünftiges Mobilitätsverhalten vorherzusagen. Sich wiederholende Muster, wann ein Fahrzeug wo und wie lange steht, sollen dabei Rückschlüsse auf ein mögliches zukünftiges Verhalten geben. Auf Basis dieser Mobilitätsprognose kann anschließend eine individuelle Einplanung des OTA-Updates für den Fahrzeughalter erfolgen. Hierbei sind für die Einplanung von Softwareupdates wiederkehrende Zeiträume mit längeren Standzeiten von hohem Interesse. Abbildung 3.4 veranschaulicht die drei Teilschritte der Idee.

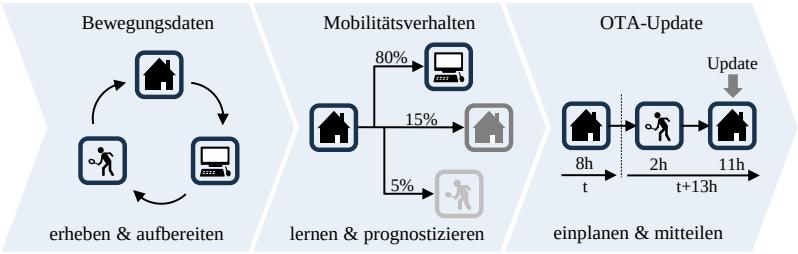



Abbildung 3.4: Idee der Zeitpunktermittlung zur automatisierten Einplanung eines OTA-Updates in Kraftfahrzeugen auf Basis des Fahrzeugnutzungsverhaltens (nach [55]

Dabei werden im ersten Schritt die Fahrzeugbewegungsdaten des oder der jeweiligen Fahrzeugnutzer(s) über mehrere Wochen erhoben und zur weiteren Verarbeitung aufbereitet. Das Mobilitätsverhalten muss anschließend auf Grundlage der aufbereiteten Bewegungsdaten mit Hilfe eines geeigneten Verfahrens gelernt und prognostiziert werden. Basierend auf dieser Prognose erfolgt im abschließenden dritten Schritt die Einplanung des OTA-Updates. Der vorgeschlagene Installationszeitpunkt soll hierbei konkret mit Datum und Uhrzeit an den Fahrzeughalter übermittelt werden. Für eine möglichst optimale und präzise Einplanung ist somit die korrekte Vorhersage des Mobilitätsverhaltens entscheidend. Damit liegt die zentrale Herausforderung weniger in der Einplanung der Installation selbst (Schritt 3), als vielmehr in einer möglichst präzisen örtlichen und zeitlichen Prognose des Fahrzeugnutzungsverhaltens (Schritt 2), auf der die Einplanung basiert. Diesbezüglich betrachtete Lösungsansätze in der Literatur (Mobilitätsvorhersagen) werden daher nachfolgend näher analysiert sowie die noch vorhandenen Defizite offengelegt.

Während die Mobilitätsvorhersage seit langem ein aktives Forschungsgebiet darstellt, ist das Anwendungsspektrum durch die Verfügbarkeit hochauflösender Mobilitätsdaten kontinuierlich gewachsen. Dies ist vor allem auf die große Menge an Smartphone-Daten [75] zurückzuführen, was wiederum

viele Forschungsprojekte [31, 123] in diesem Bereich unterstützt. Wissen über das zukünftige individuelle Mobilitätsverhalten ist für eine Vielzahl von Anwendungen und Disziplinen nützlich und dient als technischer *Enabler* für eine Vielzahl darauf aufbauender Funktionen.

Beispielsweise wird in [17] die Mobilitätsvorhersage im Bereich der Städteplanung zur Auslegung der Auslastung von Verkehrsnetzen oder ortsbezogenen Diensten verwendet. Gambs et al. [47] listen ebenfalls ortsbezogene Dienste und Mechanismen für Geodatenschutz als mögliche Anwendungsfälle für die Vorhersage des nächsten Aufenthaltsortes auf. Auch bei der Verbindungsübergabe[1] in drahtlosen Netzwerken, wie beispielsweise Mobilfunk oder Fahrzeug-Ad-hoc-Netzwerken[2], findet die Mobilitätsvorhersage Anwendung [15, 69, 90, 97]. Die Vorhersage wird unter anderem dazu genutzt, um einen möglichst planbaren und reibungslosen Übergang zwischen Funkzellen zu realisieren. Jenelius et al. [63] schlagen unter Verwendung von Taxi-Bewegungsdaten eine Methode zur Reisezeitvorhersage vor, die Zeiträume von mehreren Minuten bis zu etwa einer Stunde abdeckt. Do et al. [35] verwendeten GPS-Daten, die auf Smartphones gesammelt wurden, um den Standort eines Benutzers für ein bis drei Stunden vorherzusagen. Ein umfassender Überblick über die verschiedenen Bereiche der menschlichen Mobilitätsmodelle und -anwendungen sind ebenfalls in [16, 136] enthalten.

Die große Vielfalt der Anwendungen macht es notwendig, zwischen verschiedenen Arten von Mobilitätsvorhersagen zu unterscheiden. Ausschlaggebende Differenzierungsmerkmale sind sowohl der Prognosezeitraum, die erforderliche örtliche und zeitliche Auflösung der Vorhersage als auch die Bestimmung des richtigen Formats sowie die Darstellung der Daten. Dementsprechend lassen sich aus dem breiten Anwendungsspektrum diverse Klassifizierungen bezüglich Mobilitätsvorhersagen ableiten. Nachfolgend werden drei Klassen definiert, um die Herausforderung der Mobilitätsvor-

[1] Handover
[2] Vehicular Ad Hoc Network (VANET)

hersage im hier vorliegenden Anwendungsfall, der Einplanung von OTA-Updates für Kraftfahrzeuge, besser einordnen zu können.
Eine erste Unterscheidung bei Mobilitätsvorhersagen kann bezüglich des *Personenbezugs* des jeweiligen Anwendungsfalls getroffen werden. Konkret ist zu betrachten, ob es sich um eine *nutzerspezifische* oder *nutzerübergreifende* Vorhersage handelt. Während bei der Planung und Auslegung von Verkehrsnetzen das zukünftige Schwarmverhalten aller Verkehrsteilnehmer ausschlaggebend ist, hat die Schwarminformation wenig Aussagekraft über das Reiseverhalten eines Einzelnen. Folglich wird die Beschreibung von Bewegungsmustern einer großen Anzahl von Individuen in den meisten Fällen nicht helfen, die Bewegungsmerkmale eines Individuums zu identifizieren. Auch die Betrachtung der Optionen möglicher *Fortbewegungsmittel* spielt für die Mobilitätsvorhersage eine Rolle. Hierbei kann zwischen *unimodaler* und *multimodaler* Mobilität, beziehungsweise Fortbewegung unterschieden werden. Unimodal bedeutet dabei die ausschließliche Nutzung eines Verkehrsmittels, während multimodal das Reisen mittels mindestens zwei oder mehreren Verkehrsmitteln bezeichnet. Anhand dieser ersten beiden Klassifizierungen lassen sich die Anwendungsfälle der Mobilitätsvorhersage bereits einordnen. Tabelle 3.1 liefert hierzu einen exemplarischen Überblick. Da die Einplanung von OTA-Updates individuell pro Fahrzeug betrachtet wird, ordnet es sich in die unimodale und nutzerspezifische Kategorien ein.
Aufbauend auf den ersten beiden Klassifizierungen variiert auch der erforderliche *Prognosehorizont* bezüglich der Anwendungsfälle und spannt eine dritte Dimension der Darstellung aus Tabelle 3.1 auf. Er beschreibt, wie weit in die Zukunft das Mobilitätsverhalten prognostiziert werden muss. Dabei bezieht sich eine signifikante Anzahl von Anwendungen auf die Mobilitätsvorhersage im Kontext der *Next Location*, bei dem der nächste Standort eines Benutzers vorhergesagt wird [89, 114, 137]. Nachfolgend soll diese Art der Vorhersage als *kurzfristige Mobilitätsvorhersage* bezeichnet werden. Dabei beschränkt sie sich nicht nur auf den nächsten Standort, sondern auch auf den Zeithorizont der nächsten Stunden. Auf der anderen Seite des

Tabelle 3.1: Beispielhafte Anwendungsfälle von Mobilitätsvorhersagen klassifiziert nach Personenbezug und Fortbewegungsmittel.

| | **nutzerspezifisch** | **nutzerübergreifend** |
|---|---|---|
| **uni-modal** | Ladestrategie für Elektrofahrzeuge, Fahrzeugvorkonditionierung, Einplanung OTA-Update | Logistikplanung in Speditionen (Wartungsmanagement) |
| **multi-modal** | Steuerung Heimautomatisierung, Handover in drahtlosen Netzwerken | Städteplanung zur Verkehrsregulierung, Auslastung Mobilfunknetz (Service Provider) |

Spektrums gibt es *langfristige Mobilitätsvorhersagen*, die auf eine Mobilitätsvorhersage im Bereich von Monaten bis Jahren abzielen. Sadilek et al. [113] weisen darauf hin, dass auf *Markov-Modellen* (MM) und *Random Walk* (RW) basierende Formalismen, die für die Vorhersage der Kurzzeit-Mobilität gut geeignet sind, im Kontext der langfristigen Mobilität wenig hilfreich sind. Zwischen der kurzfristigen- und langfristigen Mobilitätsvorhersage liegt die *mittelfristige Mobilitätsvorhersage*, die typischerweise einen Zeitraum von einem Tag bis zu einer Woche abdeckt. Diese mittelfristige Mobilitätsvorhersage definiert einen geeigneten Einplanungszeitraum für OTA-Updates. Die automatisierte Ermittlung des Installationszeitpunktes von OTA-Updates für Kraftfahrzeuge lässt sich somit in die Kategorien *unimodal*, *nutzerspezifisch* und *mittelfristige Mobilitätsvorhersage* einordnen. Diese Einordnung hilft wiederum ein geeignetes Vorhersagemodell abzuleiten.

Viele der durchgeführten Forschungsprojekte konzentrierten sich auf die kurzfristige Mobilitätsvorhersage und beantworteten die Frage: 'Wo wird die Person zum Zeitpunkt X des Tages Y sein?'. Allerdings bleibt die Frage: 'Wann verlässt die Person den Ort A, um den Ort B zu erreichen?' oftmals

unbeantwortet. Die Komplexität der Antwort auf diese Frage lässt sich an den Ergebnissen der Studien [31, 123] verdeutlichen. Es wurde gezeigt, dass sich die Mehrheit der Individuen zwischen wenigen signifikanten *Points of Interest* (POIs) bewegen. Das Wissen um die wichtigsten Standorte einer Person reduziert die Anzahl der wahrscheinlichen Antworten auf die erste Frage drastisch. Das Gleiche kann jedoch nicht über die Abfahrtszeiten (Verlassen eines Standortes) gesagt werden. Je nach zeitlicher Auflösung sind Aussagen über Abfahrtszeiten aufgrund der nahezu unbegrenzten Antwortmöglichkeiten entweder ungenau oder nicht sinnvoll. Burbey [26] wies erstmals darauf hin, dass es eine andere und komplexere Aufgabe ist, *wann* sich eine Person an einem bestimmten Ort befindet, als *wo* jemand zu einem bestimmten Zeitpunkt sein wird. Baumann [17] zeigte ebenfalls, dass Örtlichkeiten vorhersehbarer sind als Aufenthalts- und Ankunftszeiten. Eine zentrale Herausforderung bei der Vorhersage eines detaillierten Mobilitätsverhaltens ist folglich die Kombination von örtlichen und zeitlichen (spatiotemporal) Daten.

Aktuelle Forschungsergebnisse zeigen diverse Methoden und Algorithmen zur Mobilitätsvorhersage. Eine umfassende Übersicht liefert [136]. Die meisten lassen sich den Ansätzen der *Informationstheorie* wie Markov-Modelle [13, 17, 30, 34, 46, 54, 66, 83, 88] und Kompressionsalgorithmen [21, 51, 109] oder dem *maschinellen Lernen* wie Bayes'sche Netze [36, 48], Künstliche Neuronale Netze [40, 84, 89] und Pattern-Mining-Techniken [33, 76] zuordnen. MM und Kompressionsalgorithmen (LZ-Familie) sind gängige Ansätze, da sie aufgrund ihrer geringen Komplexität und ihres niedrigen Ressourcenbedarfs häufig eingesetzt werden. MM erster beziehungsweise n-ter Ordnung sind einfache und effektive Werkzeuge für Standortvorhersagen. Einige Projekte, die auf MM basieren, erreichen eine Genauigkeit von über 90 % bei der Standortprognose. Die höchste Vorhersagegenauigkeit für die Standortvorhersage bei einer zeitlichen Auflösung von 20 Minuten wurde mit einem MM erster Ordnung in [26] erreicht. Abhängig von verschiedenen Faktoren wie der Zufälligkeit im Mobilitätsverhalten

und der zeitlichen Auflösung, schwankt diese Vorhersagegenauigkeit jedoch teils stark (33 % bis 91 %) [26, 34]. MM erster Ordnung sind dadurch gekennzeichnet, dass der aktuelle Zustand (Standort) ausschließlich vom vorherigen Zustand (Standort) abhängig ist. Der Einsatz einer höheren Ordnung ($\geq 2$) führt ausschließlich zur geringfügigen Steigerung der Vorhersagegenauigkeit, bei gleichzeitig deutlich erhöhtem Rechenaufwand [86]. Die so genannten *Mobility Markov-Chains* [45] sind somit ausreichend um Orte vorherzusagen, scheitern jedoch, wenn sie zur Kombination von örtlichen und zeitlichen Daten verwendet werden. Neueste Arbeiten kombinieren dennoch Markov-Ketten, unter dem Begriff *Time-Based Markov* (TBM), mit zeitlichen Informationen in Form von Zeitstempeln [17, 30, 46, 54]. Auf der einen Seite zeigen die Ergebnisse von [54] eine minimale Erhöhung der Vorhersagegenauigkeit von 38,2 % auf 39,2 %, wenn Zeitinformationen in den Vorhersageprozess einbezogen werden. Andererseits ist ihr Zeitbereich von vier Stunden (für die ihre Vorhersage am genauesten war) für die Vorhersage der Abfahrts- und Ankunftszeit recht ungenau. Cheng et al. [30] verbesserten die Vorhersagegenauigkeit unter Berücksichtigung von Zeitinformationen um 6 % im Vergleich zu den ursprünglichen Markov-Algorithmen. In diesem Fall beträgt der gewählte Zeitraum eine Stunde, was bedeutet, dass jeder Tag in 24 Zeitintervalle unterteilt ist. Tatsächlich werden für beide TBMs zeitliche Informationen nur verwendet, um die Leistung für die Standortvorhersage zu verbessern. Ein Mangel an Vorhersagen, zum Beispiel wann jemand ankommt, bleibt dadurch immer noch bestehen. Die Effizienz von Markov-Ketten leidet grundsätzlich systemimmanent unter zeitlichen Informationen insbesondere bei geringen Datenmengen. Die Nachteile überwiegen, wenn eine hohe zeitliche Auflösung (minütlich) gewünscht wird. Ohne ausreichend historische Daten des Mobilitätsprofils (mehrere Monate) ist die Übergangsmatrix zu dünn besetzt, um aussagekräftige Vorhersagen über Abfahrts- und Ankunftszeiten treffen zu können. Des Weiteren erfolgt in vielen Untersuchungen eine Vorhersage auf Basis des (vorherigen) Wochentages [17, 35, 108]. Alternativ wird zur Verein-

fachung in einigen Fällen zwischen Arbeitstagen und Wochenenden unterschieden [17, 35, 118, 131]. Jedoch ist solch eine Vereinfachung nicht förderlich, um ein atypisches Mobilitätsverhalten zu erfassen. So kann beispielsweise bei einem Schichtarbeiter mit Dreischichtbetrieb außerhalb eines Wochenzyklus eine wochentagsspezifische Betrachtung zu gänzlich falschen Ergebnissen führen. Speziell für die mittelfristige Mobilitätsvorhersage (mehrere Tage bis eine Woche) ist das Erkennen von atypischen Bewegungsmustern bedeutend. Das Phänomen des atypischen Fahrverhaltens, das sich durch ein Mobilitätsprofil auszeichnet, welches sich nicht innerhalb einer wöchentlichen Periodizität wiederholt, wurde in der Studie [52] behandelt. Beispiele für atypische Bewegungsmuster sind ein periodischer Standortbesuch alle zwei Wochen oder ein Standortbesuch alle zwei Tage.
Die Analyse der Literatur führt zu der Schlussfolgerung, dass zeitliche Informationen oft nur zur Verbesserung der Standortvorhersage verwendet werden. Die Kombination von Orts- und Zeitvorhersage (wann und wo) ist ein bisher nur geringfügig untersuchtes Forschungsgebiet. Gleichzeitig werden meist niedrige zeitliche Auflösungen $\geq 20$ Minuten betrachtet. Die vorgestellten Lösungen, insbesondere MM, sind im Allgemeinen unabhängig von zeitlichen Informationen. Neueste Arbeiten wie [17, 30, 46, 54] überwinden diesen Nachteil, indem sie zeitliche und örtliche Informationen im MM kombinieren. Allerdings sind hohe zeitliche Auflösungen (zum Beispiel eine Minute) hierfür nicht geeignet. Zudem unterliegt die Prognose von atypischen Fahrverhalten (Bewegungsmustern) nur geringer Betrachtung, welches jedoch für die mittelfristige Mobilitätsvorhersage von höherer Bedeutung ist. Abschließend werden die identifizierten Defizite aus der Literatur zusammengefasst, auf denen das weitere Vorgehen basiert:

- Ein häufiger Fokus auf kurzfristige Mobilitätsvorhersagen und damit geringe Betrachtung von mittelfristigen Prognosehorizonten.
- Meist mangelnde Verknüpfung von zeitlicher und örtlicher Information zur Mobilitätsvorhersage.

- Geringe Berücksichtigung von tagesspezifischen und tagesübergreifenden Mobilitätsmustern (typische und atypische Muster).
- Ein häufiger Schwerpunkt auf die Prognose der *Next Location*.

## 3.5 Beitrag der Arbeit

Nachdem in diesem Abschnitt der Stand der Forschung und Technik bezüglich der beiden Optimierungspotentiale *Reduzierung der Installationsdauer* und *Ermittlung des Installationszeitpunktes* in die wissenschaftliche Literatur eingeordnet wurde, besteht der Beitrag dieser Arbeit darin, entsprechende Maßnahmen und Konzepte zur Bewältigung der identifizierten Defizite auszuarbeiten.
Die Herausforderungen bezüglich der Parallelisierung von Steuergeräte-Softwareupdates zur Reduzierung der Updatedauer (Abschnitt 3.3), werden entsprechend in Kapitel 4 aufgegriffen und durch ein optimales Scheduling-Verfahren gelöst. Zu berücksichtigen sind dabei steuergerätespezifische Randbedingungen, als auch der Kodier- und Inbetriebnahmeprozess.
Gleichermaßen werden die Defizite bezüglich der Einplanung von Steuergeräte-Softwareupdates (Abschnitt 3.4) in Kapitel 5 aufgegriffen und mit Hilfe eines praktikablen Vorhersagemodells eruiert. Dies bezieht sich insbesondere auf die mittelfristige Mobilitätsvorhersage von mehreren Tagen bis einer Woche, auf der die Einplanung des Installationszeitpunktes aufbaut. Abschließend sollen in Kapitel 6 die zunächst einzeln betrachteten Konzepte in einen durchgängigen Gesamtprozess überführt werden, wobei zuerst die Updatedauer reduziert wird und anschließend die Einplanung erfolgt. Über allen konzeptionellen Anforderungen steht das Ziel, die Nutzerzufriedenheit des Fahrzeughalters zu erhöhen.

# 4 Reduzierung der Update-Installationsdauer

Ziel des Kapitels ist es, alle identifizierten Defizite bezüglich paralleler Steuergeräte-Softwareupdates aus Abschnitt 3.3 aufzugreifen und entsprechend ausgearbeitete Lösungsansätze beziehungsweise Maßnahmen vorzustellen. Die zentrale Aufgabe ist dabei, einen Scheduling-Algorithmus herzuleiten, der eine optimale parallele Reprogrammierung mehrerer Steuergeräte ermittelt. Diese optimale Anordnung hat zum Ziel, die Busbandbreiten der einzelnen Bussysteme vollständig auszunutzen und somit alle Steuergeräte in einer möglichst kurzen Zeit zu aktualisieren. Dabei ist zu beachten, dass keine echte Parallelisierung auf den Bussystemen stattfinden kann, da pro diskretem Zeitpunkt immer nur ein Bit übertragen wird. Eine echt konkurrierende Kommunikation auf dem Bus ist somit nicht möglich. Die Parallelisierung soll vielmehr dazu dienen, nicht genutzte (diskrete) Zeitschlitze bei einem exklusiven Steuergeräte-Softwareupdate, was einem nicht voll ausgelasteten Bussystem entspricht, durch weitere gleichzeitig laufende Steuergeräte-Softwareupdates aufzufüllen.

Zur Sicherstellung aller geforderten Funktionsumfänge werden in Abschnitt 4.1 zunächst alle nötigen Anforderungen an den Scheduling-Algorithmus formuliert. In Abschnitt 4.2 folgt die eigentliche Modellbildung, wobei zunächst die Abstraktion eines Updatesschrittes erfolgt, bevor anschließend zwei alternative Scheduling-Algorithmen ausgearbeitet werden. Eine theoretische und experimentelle Gegenüberstellung beider vorgestellten Algorithmen wird in Abschnitt 4.3 behandelt. Diverse Ausrichtungen der Zielgrößen des Schedulings werden in Abschnitt 4.4 verglichen, bevor eine ab-

schließende Diskussion in Abschnitt 4.5 geführt wird. Einige Inhalte dieses Kapitels wurden bereits in verkürzter Form in [56] veröffentlicht.

## 4.1 Anforderungen an das parallele Scheduling

Die Parallelisierung mehrerer Steuergeräte-Softwareupdates unterliegt diversen Einschränkungen, die als konkrete Anforderungen an den Scheduling-Algorithmus zu definieren sind. Sämtliche Anforderungen müssen bei der Modellierung des Scheduling-Algorithmus berücksichtigt werden. Einige ergeben sich direkt aus der E/E-Architektur, aufgrund der Topologie sowie den verbauten Bussystemen und deren Datenübertragungsraten. Weitere Anforderungen erschließen sich aus den Kommunikationsbeziehungen oder Datenverarbeitungsgeschwindigkeiten der einzelnen Steuergeräte. Zusammenfassend sind die folgenden vier Anforderungen als Parameter in den Scheduling-Algorithmus einzubeziehen:

i) *Einhaltung maximaler Bandbreiten aller Bussysteme:* Eine Überlastung der Busbandbreiten durch die Parallelisierung kann zu Verlusten von Botschaften (Datenpakete) führen. Dieser Datenverlust kann wiederum inkorrekte oder fehlgeschlagene Softwareupdates hervorrufen. Folglich darf zu keinem Zeitpunkt eine Überschreitung der maximal zulässigen Busauslastung erfolgen.

ii) *Einhaltung einer definierten Updatereihenfolge:* Bei einem Softwareupdate muss die Reihenfolge der Teilschritte: Flashen, Kodieren und Inbetriebnahme pro Steuergerät eingehalten werden. Ein Vertauschen ist zu keinem Zeitpunkt zulässig. Zusätzlich muss bei Steuergeräten mit einer Master-Slave-Beziehung immer zuerst das Slave-Steuergerät aktualisiert werden. Dies hat den Hintergrund, dass ein Slave-Steuergerät bei einem Softwareupdate eine neue Netzwerk-Adresse erhalten kann. Würde das Master-Steuergerät zuerst aktualisiert werden,

könnten folglich keine Update-Pakete mehr auf die alte Netzwerk-Adresse geroutet werden.

iii) *Einhaltung einer maximalen Anzahl paralleler Updateschritte:* Das Gateway oder die DCU muss als zentrale Schnittstelle Kommunikationspakete an mehrere verschiedene Bussysteme weiterleiten (Routing). Neben der Datenübertragung selbst muss das Gateway die Datenpakete in spezifische Datenframes des Bussystems packen. Werden zu viele Anfragen parallel gestellt, kann dies zu einer Überlastung des Gateways führen. Konsequenz ist der Verlust von Datenpaketen, was wiederum zu Updateabbrüchen führen kann. Um den Scheduling-Algorithmus an die Routing-Leistungen verschiedener Gateways anzupassen und gleichzeitig die Prozesssicherheit zu erhöhen, muss die Anzahl der parallel zu verarbeitenden Schritte manuell begrenzt werden.

iv) *Einhaltung exklusiv laufender Updateschritte:* Eine Parallelisierung der Updatevorgänge ist nicht immer möglich. Bestimmte Updateschritte erfordern beispielsweise einen Zündungswechsel (Klemme 15). Dieser Zündungswechsel führt dazu, dass gleichzeitig laufende Prozesse abgebrochen werden. Auch Steuergeräte in einer Master-Slave-Beziehung können nicht parallel aktualisiert werden. Folglich muss gewährleistet sein, dass bestimmte Updateschritte exklusiv eingeplant werden können.

Zusammengefasst unterliegt das Konzept PU in vielerlei Hinsicht den Randbedingungen, welche sich aus den Gegebenheiten der Topologie und verbauten Bussystemen der E/E-Architektur ergeben. Daher soll der Scheduling-Algorithmus sowohl für die heutigen als auch die zukünftigen E/E-Architekturen (siehe Abbildungen 2.1 und 2.2) kompatibel modelliert werden.

## 4.2 Modellbildung: Optimales Scheduling paralleler Steuergeräte-Softwareupdates

In diesem Kapitel erfolgt die Herleitung des Scheduling-Algorithmus zur Parallelisierung mehrerer Steuergeräte-Softwareupdates. Hierzu wird zunächst in Abschnitt 4.2.1 die Abstraktion eines Updateschrittes eingeführt, bevor die zum Vergleich herangezogenen Scheduling-Algorithmen in Abschnitt 4.2.2, respektive Abschnitt 4.2.3, vorgestellt werden.

### 4.2.1 Modellierung eines Updateschrittes

In dieser Sektion wird ein abstraktes Modell für das Parallelisierungsproblem erstellt. Es dient als Grundlage sowohl für die mathematische Formulierung als *Gemischt-Ganzzahlige Lineare Programmierung*[1] (MILP), als auch für den nachfolgend vorgestellten *Hybrid-Algorithmus* (HA).
Bisherige Untersuchungen zu parallelen Steuergeräte-Softwareupdates haben nur den Teil des Flashens berücksichtigt [68, 77, 78, 115, 122, 140]. Wie bereits in Kapitel 2.2.2 beschrieben, ist ein Steuergeräte-Softwareupdate jedoch ein dreistufiges Verfahren. Das Flashen benötigt in der Regel die meiste Zeit im Vergleich zu Kodierung und Inbetriebnahme. Grundsätzlich können jedoch alle drei Schritte steuergeräteübergreifend parallelisiert werden, um eine optimale Zeitersparnis zu erzielen.
Im Folgenden werden diese als *Abstrakte Updateschritte* (AUS) bezeichnet. Jeder Aktualisierungsschritt (Flashen, Kodieren und Inbetriebnahme) wird daher als individueller AUS für jedes Steuergerät und jede Softwareversion definiert. Die in Abschnitt 4.1 festgelegten Anforderungen sollen verwendet werden, um sicherzustellen, dass diese Schritte in der richtigen Abhängigkeit zueinander ausgeführt werden.
Jeder AUS umfasst die beiden Dimensionen: *Dauer* und *Allokierte Busbandbreite* (ABB) (nach [140]). Die *Dauer* eines AUS beschreibt, wie viel

[1] Mixed-Integer Linear Programming (MILP)

Zeit die Ausführung des Schrittes in Anspruch nimmt. Diese lässt sich leicht bei der exklusiven Ausführung eines AUS messen. Wie viel Bandbreite eines Bussystems ein AUS beansprucht wird *ABB* genannt. Da jede Phase des Softwareupdates als separater AUS definiert ist, haben die Phasen in der Regel eine unterschiedliche ABB und Dauer. Selbst bei AUS, die auf demselben Steuergerät durchgeführt werden, können diese Werte je nach Aktualisierungsschritt (Flashen, Kodieren oder Inbetriebnahme) unterschiedlich sein. Die ABB und die Dauer eines einzelnen AUS werden als nicht veränderbar angenommen. Das bedeutet, dass ein Wert nicht erhöht werden kann um den anderen zu verringern und umgekehrt. Der Grund dafür ist, dass bestimmte Updateschritte, wie zum Beispiel die Inbetriebnahme, eine feste Zeit benötigen, um bestimmte Stellglieder anzusteuern. Außerdem muss eine frei konfigurierbare ABB sowohl durch die Hardware-Spezifikationen des Steuergeräts als auch durch das Busprotokoll unterstützt werden, was nicht zwangsläufig gegeben ist. Abbildung 4.1 zeigt, wie ein AUS grafisch als Rechteck dargestellt werden kann.

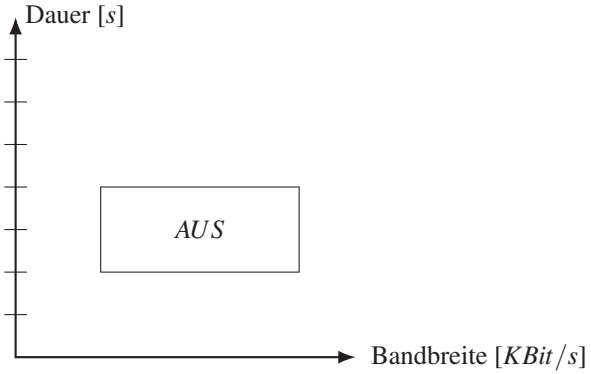



Abbildung 4.1: Grafische Darstellung eines AUS. Der AUS besitzt eine Bandbreite und Dauer (nach [140]).

Typischerweise werden die Dauer und ABB eines AUS während der Test- und Absicherungsphase in der Entwicklung bestimmt. Während die Dauer

eines AUS einfach durch Ausführen des AUS gemessen werden kann, ist das Finden der richtigen ABB nicht trivial. Eine einfache Möglichkeit ist die Berechnung der durchschnittlich benötigten Bandbreite, ausgedrückt in Gleichung (4.1) (nach [140]).

$$\text{ABB [KBit/s]} = \frac{\text{Datenmenge [KBit]}}{\text{Dauer [s]}} \tag{4.1}$$

Dies setzt voraus, dass die benötigte Bandbreite über die Dauer des AUS konstant bleibt. Eine zu niedrige Einschätzung der ABB führt zu einer Verletzung der maximalen Busbandbreite, das wiederum zu einem Abbruch oder fehlerhaften Update führen kann. Der ABB eines AUS sollte daher eher überschätzt werden, um einen sicheren Updateprozess zu gewährleisten. Dies kann zum Beispiel durch die Verwendung eines aufgeschlagenen Puffers (Offset) zum tatsächlich ermittelten Wert erfolgen. Eine präzisere Methode zur Messung der ABB ist die reale Analyse, beziehungsweise Messung der Buslast pro AUS bei exklusiver Ausführung. Seine Fläche entspricht der maximal möglichen Datenmenge des zu übertragenden AUS. Mit diesen Abstraktionen kann das Problem nachfolgend modelliert werden.
Beim Update eines Steuergeräts wird jedes Datenpaket über das Eingangsbussystem, zum Beispiel Ethernet mit dem Protokoll DoIP (vgl. Kapitel 2.1), an das Gateway (oder DCU) gesendet. Das Gateway verarbeitet diese Datenpakete und sendet sie über den entsprechenden Subbus an das Zielsteuergerät. Sowohl das Bussystem zum Gateway als auch der Subbus zum Zielsteuergerät haben eine maximal verfügbare Bandbreite, die mit $BW_{input}$ beziehungsweise $BW_{SB}$ bezeichnet wird. Der Prozess der Datenübertragung vom Eingangsbussystem zum Ziel-Subbus ist in Abbildung 4.2 exemplarisch dargestellt.
Ziel der Parallelisierung ist es, ein Scheduling zu finden, in dem die Bandbreiten aller Bussysteme optimal ausgenutzt werden, ohne dass es zu einer Überschreitung der Bandbreiten kommt. Dadurch kann die gesamte Updatedauer minimiert werden. Durch das Scheduling der AUS, welche über

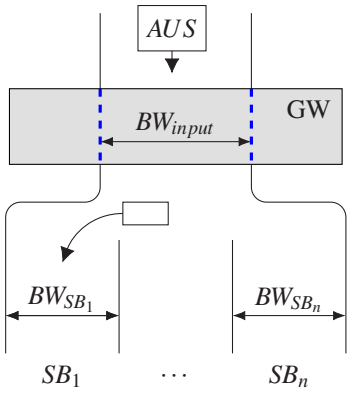



Abbildung 4.2: Alle Updatedaten müssen das Gateway (oder DCU) passieren und werden dann auf den entsprechenden Subbus geroutet. Die Bandbreiten jedes Bussystems müssen bei der Parallelisierung der Steuergeräte-Softwareupdates zu jeder Zeit eingehalten werden (nach [56]).

das Eingangsbussystem an das Gateway gesendet werden, wird auch das Scheduling der folgenden Subbusse festgelegt. Daher erfolgt die Optimierung auf dem Eingangsbus, wobei die Anforderungen für alle Subbusse zusätzlich berücksichtigt werden müssen. Abbildung 4.3 zeigt ein Scheduling von drei AUS, wenn jeder AUS sequentiell durch den Eingangsbus gesendet wird. Jeder AUS hat eine ABB und eine Dauer, $BW_{input}$ markiert die maximale Bandbreite dieses Busses. Der blau schraffierte Bereich in Abbildung 4.3 zeigt das vorhandene Optimierungspotential.
Abbildung 4.4 hingegen veranschaulicht, wie eine optimale Parallelisierung aussehen könnte. Da die kombinierte ABB von $AUS_1$ und $AUS_3$ das zulässige $BW_{input}$ nicht überschreitet, können diese beiden AUS parallelisiert werden. Der grün schraffierte Bereich zeigt die eingesparte Zeit an.
Das hier modellierte Problem ist ähnlich zu dem zweidimensionalen Zuschnittproblem rechteckiger Objekte[2]. Dabei ist die heuristische Lösung des zweidimensionalen Zuschnittproblems ein weit verbreitetes Forschungsge-

[2] Rectangular Nesting Problem

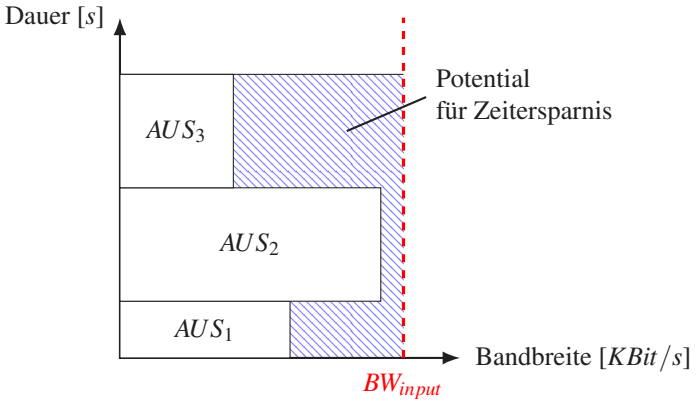



Abbildung 4.3: AUS werden sequentiell über das Eingangsbussystem gesendet. Die rote, gestrichelte Linie zeigt die maximale Bandbreite des Eingangsbusses an. Der blaue Bereich zeigt das Optimierungspotential (nach [56]).

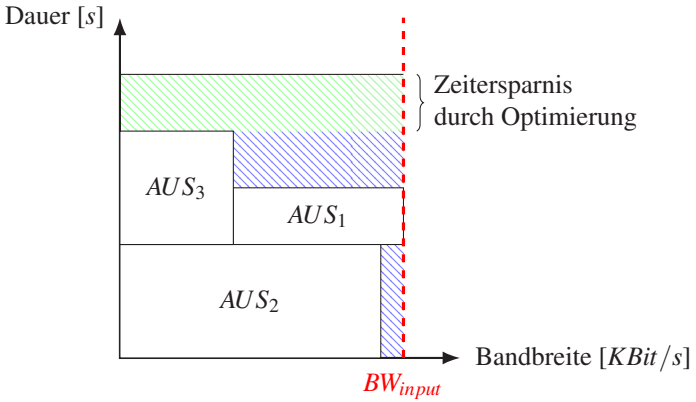



Abbildung 4.4: Durch eine optimale Parallelisierung wird die volle Bandbreite des Bussystems ausgenutzt. Die grüne Fläche symbolisiert die erzielte Zeitersparnis (nach [56]).

biet [59, 100, 129, 130]. Ein vergleichbarer Ansatz wird in [130] untersucht. Auch wenn im hier vorliegenden Parallelisierungsproblem versucht wird, eine möglichst dichte Anordnung von Elementen mit zwei Dimensio-

nen (ABB und Dauer) zu finden, darf das hier beschriebene Optimierungsproblem jedoch nicht mit einem zweidimensionalen Zuschnittproblem von rechteckigen Objekten verwechselt werden. Dies liegt daran, dass ein AUS entlang der Bandbreitenachse nicht zusammenhängend sein muss. Abbildung 4.5 zeigt hierzu ein exemplarisches Scheduling von drei weiteren AUS. $AUS_5$ wird horizontal geschnitten, da der AUS aber zeitlich zusammenhängend ist, ergibt dieses Scheduling eine valide Anordnung.

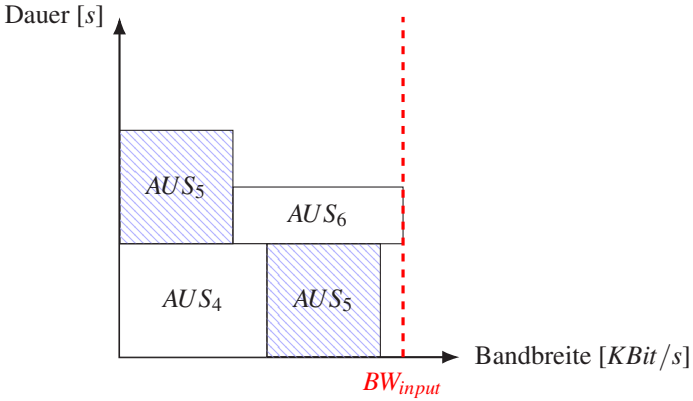



Abbildung 4.5: Bei einem zulässigen Scheduling kann jeder AUS entlang der Bandbreitenachse diskontinuierlich sein, solange er zeitlich kohärent ist (nach [56]).

Durch diesen zusätzlichen Freiheitsgrad muss die Lösung des Optimierungsproblems neu betrachtet werden. Aufgrund der getroffenen Abstraktionen können nachfolgend sowohl die Gemischt-Ganzzahlige Lineare Programmierung als auch der Hybrid-Algorithmus in den nächsten Abschnitten formuliert werden.

### 4.2.2 Scheduling-Algorithmus I: MILP

In diesem Abschnitt wird das parallele Scheduling als Optimierungsproblem der *Gemischt-Ganzzahligen Linearen Optimierung* formuliert. Ziel eines Optimierungsproblems ist die Bestimmung der optimalen Lösung ei-

ner Zielfunktion $f$, wobei die Lösungsmenge (Lösungsraum) meist durch zusätzliche Nebenbedingungen eingeschränkt ist. Je nach Anwendungsfall kann hierbei das Maximum oder Minimum der Funktion gesucht sein. Das Maximierungsproblem kann durch Multiplizieren mit $-1$ in ein Minimierungsproblem transformiert werden. Deshalb wird im Folgenden nur das Minimierungsproblem betrachtet. Im Allgemeinen lässt sich ein Optimierungsproblem analog zu [104] wie folgt formulieren:

$$\min_{x,y} f(x,y), \quad s.t.(x,y) \in M, \tag{4.2}$$

wobei

$$\begin{aligned} M = \{ & x \in X \subseteq \mathbb{R}^{n_i}, \quad y \in Y \subseteq \mathbb{Z}^{n_j} \mid \\ & g(x,y) \leq 0, \quad g : X \times Y \rightarrow \mathbb{R}^{n_l} \} \end{aligned}$$

Die zulässige Lösungsmenge $M$ ist durch die Ungleichungen $g$ definiert und kann grafisch wie in Abbildung 4.6 veranschaulicht werden. Die in Abbildung 4.6 gezeigten Nebenbedingungen sind ausschließlich linear.

**Definition 4.1.** *Die Lineare Programmierung (LP) bestimmt die optimale Lösung einer linearen Zielfunktion $f$, welche ausschließlich durch lineare Nebenbedingungen beschränkt ist, wobei $n_j = 0$ und $n_i > 0$.*

In der Praxis hat sich das Simplex-Verfahren [32] zur Bestimmung der exakten Lösung des LPs in polynomialer Zeit bewährt. In polynomialer Zeit lösbare Probleme werden allgemein als handhabbar betrachtet, da bezogen auf die Problemgröße die Rechenzeit zur Lösungsfindung mit einer deterministischen Maschine nicht stärker als gemäß einer Polynomfunktion wächst [32, 104]. Falls die Lösungsmenge $M$ einem Polyeder wie in Abbildung 4.6 entspricht, liegt das gesuchte Optimum der Zielfunktion auf den Punkten einer Kante, einem Eckpunkt oder im Inneren der Lösungsmenge. Das Simplex-Verfahren wird dabei beginnend von einer zulässigen Startlösung

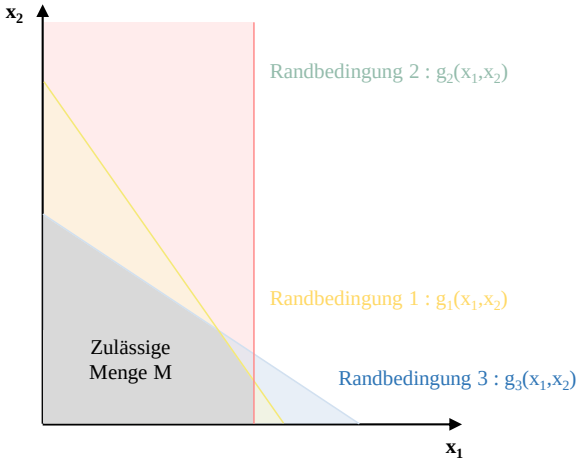



Abbildung 4.6: Exemplarische zweidimensionale lineare Optimierung mit drei Nebenbedingungen ($g_1$, $g_2$ und $g_3$). Das sich daraus ergebende Polyeder definiert die zulässige Lösungsmenge $M$.

entlang der Kanten kontinuierlich nach einem besseren Zielfunktionswert suchen, bis kein besserer Zielfunktionswert mehr in $M$ existiert. Im Falle einer unbeschränkten oder leeren Lösungsmenge $M$ wird die Unlösbarkeit des linearen Optimierungsproblems vom Simplex-Verfahren erkannt. Alternativ zum Simplex-Verfahren existiert das Innere-Punkte-Verfahren [65, 103] zur Lösung eines linearen Problems.

**Definition 4.2.** *Eine Gemischt-Ganzzahlige Lineare Programmierung (MILP) bestimmt die optimale Lösung einer linearen Zielfunktion $f$, welche ausschließlich durch lineare Nebenbedingungen beschränkt ist. Zusätzlich nehmen einige Variablen nur ganzzahlige Werte an ($n_j > 0$ und $n_i > 0$).*

Die MILP unterscheidet sich zu einer LP dahingehend, dass zusätzlich einige Variablen nur ganzzahlige Werte annehmen dürfen. Die Zielfunktion $f$ ist somit von reellen und ganzzahligen Variablen abhängig. Im Vergleich zu LP ist die Lösungsmenge $M$ bei MILP nicht zusammenhängend, wodurch

das Problem NP-schwer ist [135]. Die Problemklassifizierung NP-schwer stammt aus der Komplexitätstheorie, welche sich mit der Klassifizierung von Problemen bezüglich ihrer Komplexität beschäftigt. NP steht dabei für *nichtdeterministische Polynomialzeit*, wobei ein NP-schweres Problem entweder gar nicht gelöst werden kann oder eines der schwierigsten Probleme in NP ist.

Das gemischt-ganzzahlige Optimierungsproblem lässt sich jedoch mit Hilfe *relaxierter* Nebenbedingungen effizienter lösen. Konkret wird bei der sogenannten LP-Relaxierung die Bedingung der Ganzzahligkeit aufgehoben. Folglich entsteht ein Lineares Problem ($LP_2$) mit reellen Nebenbedingungen (Lösungsmenge $M_2$), welches als Teilmenge die ursprünglich gesuchte ganzzahlige Lösungsmenge ($M_1$) enthält. Dies ist in Abbildung 4.7 veranschaulicht.

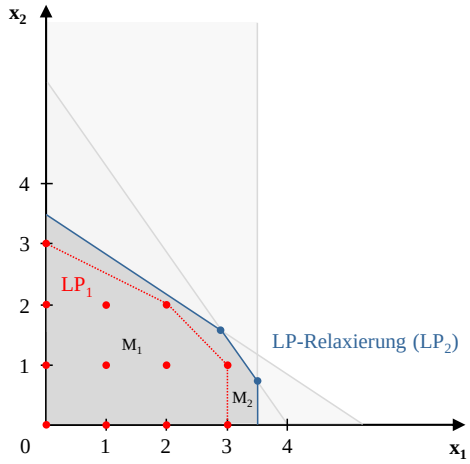



Abbildung 4.7: Die roten Punkte symbolisieren die zulässigen ganzzahligen Lösungen, wobei die roten Linie die kleinste zulässige Lösungsmenge ($M_1$) als Polyeder definiert. Die blaue Linie veranschaulicht die dazugehörige LP-Relaxierung ($M_2$) mit reellen Nebenbedingungen.

Konkret erfolgt durch die Relaxierung eine Transferierung des NP-schweren gemischt-ganzzahligen Optimierungsproblems in ein reelles Problem. Dadurch lässt sich das Optimum von $LP_2$ wieder mit Hilfe eines Lösungsverfahrens der Linearen Programmierung, wie dem Simplex-Verfahren, in polynomialer Zeit lösen. Die Lösung des relaxierten Problems erfüllt jedoch im Allgemeinen nicht die Ganzzahligkeitsbedingung, kann anschließend aber als Näherungslösung für einen Algorithmus zur exakten Lösung der ganzzahligen Optimierung verwendet werden. Ein exaktes Lösungsverfahren für die MILP ist das *Branch-and-Cut-Verfahren* [104]. Es kombiniert die beiden exakten Lösungsalgorithmen *Schnittebenenverfahren* und *Branch-and-Bound-Verfahren* [98]. Ist die Lösung der LP-Relaxierung durch das Simplex-Verfahren nicht ganzzahlig und gehört somit nicht zur zulässigen Lösungsmenge ($M_1$), fügt das Schnittebenenverfahren weitere Ungleichungen hinzu, um die Lösungsmenge weiter einzugrenzen. Ziel der Ungleichungen ist es, Teile der relaxierten Lösungsmenge ($M_2$) abzuschneiden, ohne dabei die zulässige Lösungsmenge ($M_1$) zu betreffen. Folglich sollte die Neuberechnung mit der hinzugefügten Ungleichung näher am gesuchten Optimum liegen. Solch eine Schnittebene ist exemplarisch in Abbildung 4.8 dargestellt. Ist diese Lösung weiterhin nicht ganzzahlig, werden weitere Schnittebenen hinzugefügt, bis die Lösung gefunden wird oder keine weiteren Schnittebenen mehr zur Verfügung stehen. Wurde die Lösung nicht gefunden, startet das Branch-and-Bound-Verfahren, wodurch das Problem in mehrere Teilprobleme zerteilt wird. Hierbei wird der Wertebereich einer oder mehrerer Variablen der Teilprobleme eingeschränkt, was zu einer Vereinfachung des ursprünglichen Problems führt. Durch iterative Anwendung der Aufteilung in Teilprobleme entsteht eine Baumstruktur (Suchbaum). Dieser erste Schritt entspricht dem *Branch* und ist ebenfalls in Abbildung 4.8 verdeutlicht. Der zweite Schritt *Bound* hat zum Ziel, Teile des Suchbaumes, die nicht die optimale Lösung enthalten können, von der weiteren Lösungsfindung (Berechnung) auszuschließen. Dies verfolgt den Zweck, den gesamten Rechenaufwand und somit die Berechnungsdauer zu reduzie-

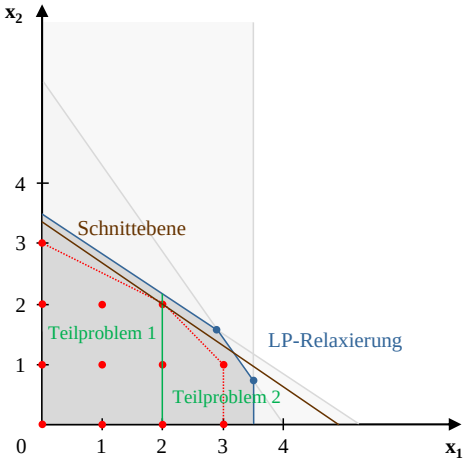



Abbildung 4.8: Durch das Hinzufügen von Schnittebenen wird die Lösungsmenge eingegrenzt. Führt dies zu keiner zulässigen Lösung, wird die Lösungsmenge durch das Branch-and-Bound-Verfahren sukzessiv in kleinere Teilprobleme zerlegt.

ren. Eine konkrete Implementierung des Branch-and-Cut-Algorithmus stellt der CPLEX-Solver der Firma *International Business Machines Corporation* (IBM) dar [60]. Weitere Informationen zum Branch-and-Cut-Algorithmus sind in [104] zu finden.

Die Idee, das Parallelisierungsproblem als MILP zu modellieren, ist in Abbildung 4.9 dargestellt. Zunächst wird die Zeit in diskrete Zeitschlitze aufgeteilt. Für jeden AUS und jeden Zeitschlitz werden drei binäre Variablen eingeführt, die beschreiben, wann ein AUS ausgeführt wird. $bs_{i,j}$ und $be_{i,j}$ sind gleich eins, wenn ein AUS $i$ im Zeitschlitz $j$ beginnt beziehungsweise endet. Der Start und das Ende eines AUS liegen in der Mitte eines Zeitschlitzes. Im Beispiel in Abbildung 4.9 sind $bs_{i,1} = 1$ und $be_{i,5} = 1$. $b_{i,j}$ beschreibt, ob ein AUS im Zeitschlitz $j$ durchgeführt wird und ist vom Start- bis zum Endzeitpunkt gleich eins. Diese Variable wird verwendet, um die Anforde-

rungen i) bis iv) aus Abschnitt 4.1 als lineare Bedingungen zu modellieren.

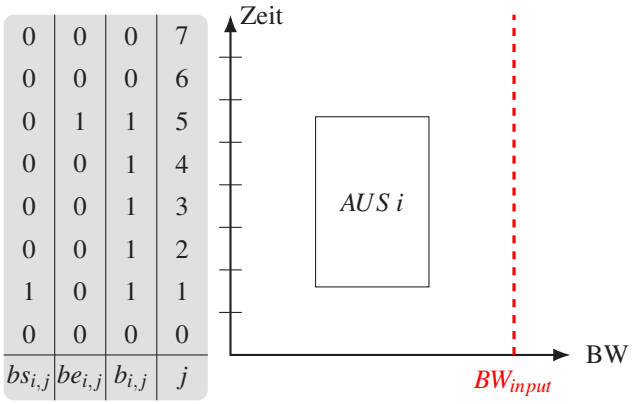



Abbildung 4.9: Darstellung von binären Variablen, welche die Start- und Endzeitpunkte eines AUS anzeigen. Die Zeit ist in diskrete Zeitschlitze unterteilt (nach [56]).

Es sei $I_{SB_k}$ die Menge aller AUS, deren Zielsteuergerät zum Subbus $k$ für $k = 1, \ldots, n$ gehört. Des weiteren sei

$$I = I_{SB_1} \cup \ldots \cup I_{SB_n}$$

die Menge aller AUS und

$$J = \{0, 1, \ldots, T\}$$

die Menge aller Zeitschlitze, wobei $T \in \mathbb{N}$ hinreichend groß ist. Die Dauer eines Zeitschlitzes kann beliebig gewählt werden, sei im Folgenden aber zur einfacheren Darstellung des Verfahrens eine Sekunde. Für jedes $i \in I$ bezeichne $a_i, d_i \in \mathbb{N}$ die ABB beziehungsweise die Dauer des AUS $i$. Die Dauer eines AUS entspricht dabei der Anzahl der benötigten Zeitschlitze und muss in Abhängigkeit der gewählten Dauer der Zeitschlitze erfolgen.

Der Parameter $p_i$ beschreibt, ob AUS $i$ auf dem Eingangsbus parallelisiert werden kann. Zu diesem Zweck sei

$$p_i = \begin{cases} 1, & i \text{ kann auf Eingangsbus parallelisiert werden} \\ P_{input}, & \text{sonst} \end{cases} \quad \forall i \in I,$$

wobei $P_{input} \in \mathbb{N}$ die Anzahl der AUS bezeichnet, die gleichzeitig auf dem Eingangsbus eingeplant werden können. Der Parameter $q_i^k$ beschreibt, ob ein AUS $i \in SB_k$ auf seinem entsprechenden Subbus $k$ für $k = 1, \ldots, n$ parallelisiert werden kann. Hierzu sei

$$q_i^k = \begin{cases} 1, & i \text{ kann auf Subbus } k \text{ parallelisiert werden} \\ P_{SB_k}, & \text{sonst} \end{cases} \quad \forall i \in I_{SB_k},$$

wobei $P_{SB_k} \in \mathbb{N}$ die maximale Anzahl von AUS bezeichnet, die auf dem Subbus $k$ parallelisiert werden können. Die Reihenfolge bestimmter AUS wird durch den Parameter $r_{i,\hat{i}}$ modelliert,

$$r_{i,\hat{i}} = \begin{cases} 1, & i \text{ muss eingeplant werden vor } \hat{i} \\ 0, & \text{sonst} \end{cases} \quad \forall i, \hat{i} \in I.$$

Weiter sei $BW_{input} \in \mathbb{N}$ die maximale Bandbreite des Eingangsbusses und $BW_{SB_k} \in \mathbb{N}$ die maximale Bandbreite des Subbusses $k$.
Ziel dieses MILP ist es, die gesamte Updatedauer $t_{ins}$ so zu minimieren, dass die Anforderungen i) bis iv) aus Abschnitt 4.1 erfüllt sind. Die binären Variablen $bs_{i,j}$ und $be_{i,j}$ werden dafür als SOS-Variablen[3] vom Typ 1 definiert und müssen zunächst den richtigen Werten zugeordnet werden. Die Variable $bs_{i,j}$ wird auf eins gesetzt, wenn AUS $i$ im Zeitschlitz $j$ beginnt (Gleichung

[3] SOS1: Eine Menge von Variablen, von denen höchstens eine einen Wert ungleich Null annehmen darf.

(4.3)). Ebenso wird $be_{i,j}$ auf eins gesetzt, wenn AUS $i$ im Zeitschlitz $j$ endet (Gleichung (4.4)).

$$\sum_{j \in J} j \cdot bs_{i,j} = start_i, \quad \forall i \in I, \tag{4.3}$$

$$\sum_{j \in J} j \cdot be_{i,j} = end_i, \quad \forall i \in I, \tag{4.4}$$

wobei die Variablen $start_i$, $end_i \in \mathbb{R}$ die Start- und Endzeitpunkte von AUS $i$ bezeichnen. Mit diesen Variablen garantiert Ungleichung (4.5), dass $t_{ins}$ größer ist als der Endzeitpunkt jedes AUS.

$$end_i \leq t_{ins}, \quad \forall i \in I \tag{4.5}$$

Außerdem muss der Startzeitpunkt von AUS $i$ genau $d_i$ Zeitschlitze vor seinem Endzeitpunkt sein (4.6).

$$start_i = end_i - d_i, \quad \forall i \in I \tag{4.6}$$

Um die Anforderungen bezüglich eines Softwareupdates bei einem Steuergerät als lineare Nebenbedingungen zu formulieren, wird eine weitere binäre Variable $b_{i,j}$ mit der Gleichung (4.7) eingeführt.

$$b_{i,j} = b_{i,j-1} + bs_{i,j} - be_{i,j-1}, \quad \forall i \in I \, \forall j \in J, \tag{4.7}$$

wobei

$$b_{i,j-1} = be_{i,j-1} = 0, \quad \text{für } j = 0 \, \forall i \in I.$$

Dadurch wird garantiert, dass $b_{i,j}$ tatsächlich entweder 0 oder 1 ist. $b_{i,j}$ ist wie in Abbildung 4.9 dargestellt für alle Zeitschlitze, in denen AUS $i$ ausgeführt wird 1 und ansonsten 0. $b_{i,j}$ wird zur Umsetzung der vier Anforderungen (siehe Kapitel 4.1) als lineare Nebenbedingungen verwendet. Die

erste Anforderung besagt, dass die gemeinsame Bandbreite aller simultanen AUS die Bandbreite des Eingangsbusses (Ungleichung (4.8)) sowie die Bandbreite jedes Subbusses (Ungleichung (4.9)) nicht überschreitet.

$$\sum_{i \in I} a_i \cdot b_{i,j} \leq BW_{input}, \quad \forall j \in J \tag{4.8}$$

$$\sum_{i \in I_{SB_k}} a_i \cdot b_{i,j} \leq BW_{SB_k}, \quad \forall j \in J,\ k = 1, \ldots, n \tag{4.9}$$

Die nächste Nebenbedingung erfüllt die beiden Anforderungen iii) und iv). Diese besagen, dass eine maximale Anzahl gleichzeitiger AUS $P_{input}$ nicht überschreiten darf und dass es einige AUS geben kann, die exklusiv behandelt werden müssen (Ungleichung (4.10)). Diese Anforderungen müssen ebenfalls für alle Subbusse gelten (Ungleichung (4.11)).

$$\sum_{i \in I} p_i \cdot b_{i,j} \leq P_{input}, \quad \forall j \in J \tag{4.10}$$

$$\sum_{i \in I_{SB_k}} q_i^k \cdot b_{i,j} \leq P_{SB_k}, \quad \forall j \in J,\ k = 1, \ldots, n \tag{4.11}$$

Die letzte Anforderung wird durch die lineare Ungleichung (4.12) erfüllt.

$$r_{i,\hat{i}} \cdot end_i \leq start_{\hat{i}}, \quad \forall i, \hat{i} \in I \tag{4.12}$$

Mit diesen Formulierungen reduziert sich die MILP auf

$$\min t_{ins}, \text{ so dass die Bedingungen (4.3) - (4.12) eingehalten werden.}$$

Alle Mengen, Parameter und Variablen, die in dem MILP verwendet werden, sind in den Tabellen 4.1 und 4.2 zusammengefasst.
Der CPLEX-Solver [60] ist nun in der Lage, mit Hilfe eines Branch-and-Cut-Algorithmus, die optimale Lösung des Parallelisierungsproblems zu finden. Da es jedoch viele Lösungen mit der gleichen Updatedauer geben

kann, benötigt der Branch-and-Cut-Algorithmus unter Umständen eine lange Rechenzeit. Der Grund dafür ist, dass der Algorithmus nicht in der Lage ist, den Suchbaum effektiv zu reduzieren, wenn es eine große Anzahl von Lösungen gibt [96]. Außerdem kann die große Anzahl von Binärvariablen die Rechenzeit negativ beeinflussen.

### 4.2.3 Scheduling-Algorithmus II: Hybrid-Algorithmus

Aufgrund der möglichen langen Rechenzeit des im Abschnitt 4.2.2 definierten Optimierungsproblems, die sich aus einer großen Anzahl von binären Variablen und möglichen Lösungen ergeben kann, wird in diesem Abschnitt ein alternativer, heuristischer Algorithmus vorgestellt. Dieser Ansatz kombiniert eine *deterministische Platzierungsroutine* mit einer *nicht deterministischen Meta-Heuristik*. Ziel dieses *Hybrid-Algorithmus* (HA) ist es, ein hinreichend gutes Scheduling in einer angemessenen Rechenzeit zu finden. Das Grundgerüst des Hybrid-Algorithmus ist ein *Evolutionärer Algorithmus* (EA), welcher zur Klasse der Meta-Heuristischen Optimierungsverfahren gehört. Meta-Heuristiken sind allgemeine Lösungsstrategien, die durch manuelle Definition und Anpassung der problemspezifischen Parameter auf beliebige Probleme angewendet werden können. Dabei sind EA an den biologischen Evolutionsprozess angelehnt. Die vier Grundschritte sind: Initia-

Tabelle 4.1: Im MILP verwendete Mengen (nach [56]).

| **Mengen** | |
|---|---|
| $I_{SB_k}$ | Menge aller AUS, die zu Subbus $k$ gehören, $k = 1, \ldots, n$ |
| $I = I_{SB_1} \cup \ldots \cup I_{SB_n}$ | Menge aller AUS |
| $J = \{0, 1, \ldots, T\}$ | Menge aller Zeitschlitze, wobei $T \in \mathbb{N}$ hinreichend groß ist |

Tabelle 4.2: Im MILP verwendete Parameter und Variablen (nach [56]).

| **Parameter** | |
|---|---|
| $a_i \in \mathbb{N}$ | ABB von AUS $i \in I$ |
| $d_i \in \mathbb{N}$ | Dauer von AUS $i \in I$ |
| $p_i \in \mathbb{N}$ | Beschreibt, ob AUS $i \in I$ auf dem Eingangsbus parallelisiert werden kann |
| $q_i^k \in \mathbb{N}$ | Beschreibt, ob AUS $i \in I$ auf seinem Subbus $k$ parallelisiert werden kann |
| $r_{i,\hat{i}} \in \{0,1\}$ | Beschreibt, ob AUS $i \in I$ vor AUS $\hat{i} \in I$ eingeplant werden muss |
| $BW_{input} \in \mathbb{N}$ | Maximale Bandbreite des Eingangsbus |
| $BW_{SB_k} \in \mathbb{N}$ | Maximale Bandbreite des Subbus $k$ |
| $P_{input} \in \mathbb{N}$ | Maximale Anzahl von parallelen AUS auf dem Eingangsbus |
| $P_{SB_k} \in \mathbb{N}$ | Maximale Anzahl von parallelen AUS auf dem Subbus $k$ |

| **Variablen** | |
|---|---|
| $t_{ins} \in \mathbb{R}$ | Gesamtdauer des Updates |
| $start_i \in \mathbb{R}$ | Startzeitpunkt von AUS $i \in I$ |
| $end_i \in \mathbb{R}$ | Endzeitpunkt von AUS $i \in I$ |
| $bs_{i,j} \in \{0,1\}$ | SOS Typ 1 Variable gibt an ob AUS $i \in I$ in Zeitschlitz $j \in J$ beginnt |
| $be_{i,j} \in \{0,1\}$ | SOS Typ 1 Variable gibt an ob AUS $i \in I$ in Zeitschlitz $j \in J$ endet |
| $b_{i,j} \in \{0,1\}$ | Binäre Variable gibt an ob AUS $i \in I$ in Zeitschlitz $j \in J$ aktiv ist |

lisierung (Erzeugen einer neuen Population bestehend aus Individuen), Evaluation (Bestimmung der Lösungsgüte für alle Individuen mittels Gütefunktion), Selektion (Bestimmung der Individuen für die nächste Generation anhand der Güte) und Variation (Kombination und zufällige Mutation der ausgewählten Individuen). Die letzten drei Schritte werden so lange wiederholt, bis ein definiertes Abbruchkriterium erreicht ist [121].

Das übergeordnete Ziel des hier betrachteten EA ist es, eine Reihenfolge von $m \in \mathbb{N}$ AUS zu finden, die eine möglichst geringe Gesamtdauer ($t_{ins}$) des Updates ergibt. Das Scheduling einer Reihenfolge wird dabei durch die Platzierungsroutine berechnet und ergibt die gesamte Updatedauer. Die Grundstruktur des Hybrid-Algorithmus in Form eines Flussdiagramms ist in Abbildung 4.10 verdeutlicht.

Zur Initialisierung wird eine zufällige Population von Individuen durch den *Generator* erzeugt. Jedes Individuum ist dabei eine Folge von $m$ AUS.

Im nächsten Schritt wird mit der Platzierungsroutine ein valides Scheduling für jedes dieser Individuen generiert, das auch die erforderliche Updatedauer liefert. Die Updatedauer wird als Güte (Fitness) eines Lösungskandidaten herangezogen, um eine Vergleichbarkeit herzustellen. Die Platzierungsroutine ist somit als Unterfunktion in die Meta-Heuristik integriert und dient als *Evaluator*.

Für die Implementierung der Platzierungsroutine werden die gleichen Definitionen wie in den Tabellen 4.1 und 4.2 aus Abschnitt 4.2.2 verwendet. Zusätzlich sei $I_0 = (i_1, \ldots, i_m)$ eine vorgegebene Reihenfolge von $m \in \mathbb{N}$ AUS, $BW_{input}(j)$ die verfügbare Bandbreite am Zeitschlitz $j$ des Eingangsbusses und $BW_{SB_k}(j)$ die verfügbare Bandbreite des Subbusses $k$ am Zeitschlitz $j$. Die Platzierungsroutine plant jeden AUS von unten nach oben ein (bottom-up). Dazu sucht der Algorithmus nach dem ersten Zeitschlitz, in dem die Einplanung eines AUS zu einem gültigen Scheduling führt. Das bedeutet, dass sowohl die Bandbreite des Eingangsbusses als auch die Bandbreite des entsprechenden Subbusses nicht überschritten werden und die weiteren Anforderungen ii) bis iv) aus Abschnitt 4.1 gelten. Schließlich gibt die Plat-

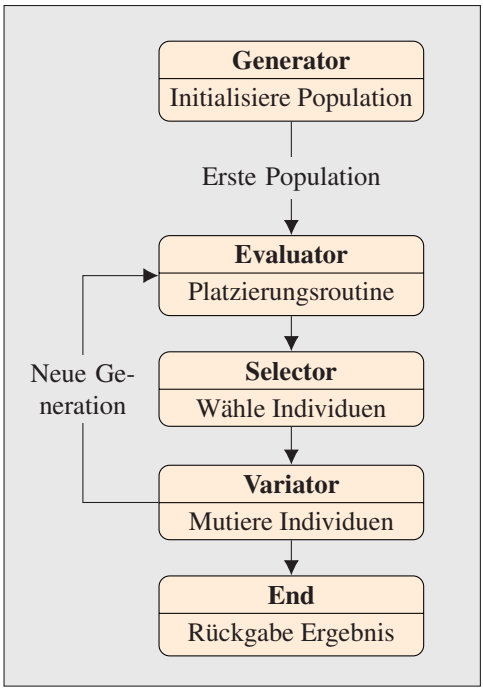



Abbildung 4.10: Flussdiagramm des Hybrid-Algorithmus. Die Platzierungsroutine ist als Evaluator in die Meta-Heuristik eingebettet (nach [56]).

zierungsroutine die Start- und Endzeitpunkte jedes AUS sowie die gesamte Updatedauer zurück. Die Funktionsweise ist in Algorithmus 1 zusammengefasst.

Im dritten Schritt des HA wählt der *Selektor* eine geeignete Teilmenge der aktuellen Population aus, die wiederum als Eltern der nächsten Generation dient. Basierend auf der erzielten Updatedauer (Ergebnis vom Evaluator) werden Individuen mit geringerer Updatedauer bevorzugt selektiert.

Im letzten Schritt modifiziert der *Variator* die zuvor ausgewählte Teilmenge, um eine neue Generation zu bilden. Bei Folgen (wie es hier der Fall ist) existieren diverse Permutationsoperatoren wie Einfügen, Umkehren, Ver-

**Algorithm 1** Platzierungsroutine

**Input:** $I_0 = (i_1, \ldots, i_m)$ Folge von $m$ AUS
$J = (0, 1, 2, \ldots)$ Menge an Zeitschlitze
$a_{i_1}, \ldots, a_{i_n}$ ABB von jedem AUS
$d_{i_1}, \ldots, d_{i_n}$ Dauer von jedem AUS
**Output:** Installationsdauer $t_{ins}$

setze $BW_{input}(j) = BW_{input}\ \forall j \in J$
setze $BW_{SB_k}(j) = BW_{SB_k}\ \forall j \in J,\ k = 1, \ldots, n$
**for each** $i_l \in I_0$ **do**
    finde kleinstes $j \in J$
    mit $\min\{BW_{input}(j), \ldots, BW_{input}(j + d_{i_l})\} \geq a_{i_l}$
    und $\min\{BW_{SB_k}(j), \ldots, BW_{SB_k}(j + d_{i_l})\} \geq a_{i_l}$
    und Anforderungen ii) bis iv) eingehalten werden
    $start_{i_l} = j$
    $end_{i_l} = j + d_{i_l}$
    $BW_{input}(\hat{j}) = BW_{input}(\hat{j}) - a_{i_l},\ \hat{j} = j, \ldots, j + d_{i_l}$
    $BW_{SB_k}(\hat{j}) = BW_{SB_k}(\hat{j}) - a_{i_l},\ \hat{j} = j, \ldots, j + d_{i_l}$
**end for**
setze $t_{ins} = \max\{end_{i_1}, \ldots, end_{i_m}\}$
**return** $t_{ins}$

schieben oder Austauschen von Positionen [124]. Dieses iterative Verfahren stoppt nach einer festen Anzahl von Generationen oder bei Erreichen einer bestimmten Lösung und gibt die Folge mit der kürzesten Updatedauer zurück.

Für die Implementierung des HA wird die *Inspyred* Bibliothek [49] verwendet, die ein Open-Source Framework für evolutionäre Berechnungen ist. Zu beachten ist, dass dieser HA im Allgemeinen nicht die optimale Lösung findet. Selbst wenn dies der Fall ist, gibt es keine Möglichkeit zu wissen, dass die Lösung optimal ist. Ein weiterer Nachteil ist, dass das Ergebnis dieses Algorithmus variieren kann, da der evolutionäre Teil nicht deterministisch ist.

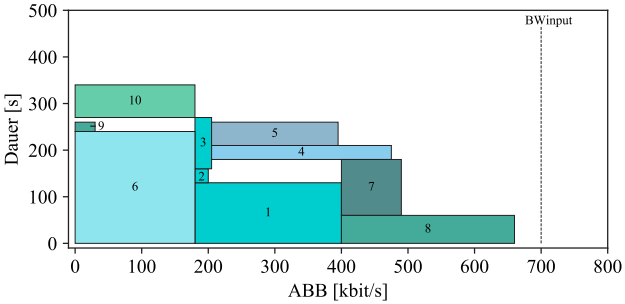
### 4.2.4 Exemplarisches Scheduling

Um die Funktionalität des ausgearbeiteten Modells zu verdeutlichen, zeigt dieses Kapitel ein exemplarisches Scheduling von beiden Scheduling-Algorithmen mit den gleichen Eingangsdaten. Die Hauptmotivation besteht darin, die Umsetzung der Anforderungen i) bis iv) aus Abschnitt 4.1 zu veranschaulichen und die Besonderheiten eines solchen Schedulings hervorzuheben.
Zu diesem Zweck werden alle Arten von Updateschritten wie Installation, Kodierung und Inbetriebnahme sowie ein exklusiver Aktualisierungsschritt berücksichtigt. Der verwendete Datensatz ist in Tabelle 4.3 aufgeführt. Aus der Tabelle ist zu entnehmen, dass die ABB und die Dauer für verschiedene AUS je nach Art des Aktualisierungsschritts sogar für das gleiche Steuergerät variieren können. Da die Reihenfolge von Installation, Kodierung und Inbetriebnahme eingehalten werden muss, ist die Reihenfolge bestimmter AUS zu berücksichtigen. Die Spalte *Reihenfolge* gibt dabei an, welche AUS abgeschlossen sein müssen, bevor der jeweilige AUS ausgeführt werden darf. Ebenso sind AUS, die exklusiv ausgeführt werden müssen in der Spalte *Sequentiell* aufgeführt.
$P_{input}$ und $P_{SB_k}$ sind auf drei begrenzt, was bedeutet, dass maximal drei parallele AUS gleichzeitig auf jedem Bus betrieben werden können. Betrachtet werden insgesamt vier Subbusse, bei denen die maximale Bandbreite jedes Subbusses auf die maximale Bandbreite des Eingangsbusses eingestellt ist. Auf diese Weise muss nur $BW_{Input}$ berücksichtigt werden, um ein gültiges Scheduling zu gewährleisten. Die Vereinfachung kann getroffen werden, da alle Datenmengen immer zuerst über den Eingangsbus übertragen werden.

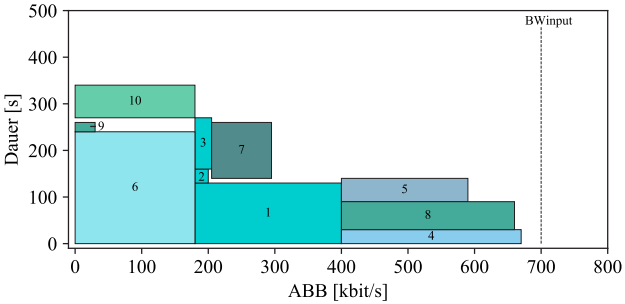
**Beispiel 4.3.** *Abbildung 4.11 zeigt die resultierenden Schedulings beider Algorithmen. Jede Farbe in Abbildung 4.11 visualisiert ein individuelles Steuergerät, während die Flash-, Kodier- und Inbetriebnahme-Schritte (AUS) durch ihre jeweilige ID aus Tabelle 4.3 identifiziert werden können. Um die Beziehungen zwischen den einzelnen AUS besser zu veranschaulichen, wird*

Tabelle 4.3: Auszug aus dem Steuergeräte Datenpool (nach [56]).

| ID | Name | ABB [$KBit/s$] | Dauer [$s$] | Reihenfolge | Sequentiell | Bussystem | Typ |
|---|---|---|---|---|---|---|---|
| 1 | LL_ThermoManag. | 220 | 130 | 0 | 0 | CAN_Conf | Installation |
| 2 | LL_ThermoManag. | 20 | 30 | 1 | 0 | CAN_Conf | Kodierung |
| 3 | LL_ThermoManag. | 25 | 110 | 1,2 | 0 | CAN_Conf | Inbetriebnahme |
| 4 | LL_SeatAdjust_SL | 270 | 30 | 0 | 0 | CAN_Tran | Installation |
| 5 | LL_SeatAdjust_MA | 190 | 50 | 4 | 0 | CAN_Tran | Installation |
| 6 | LL_RearViewCamera | 90 | 120 | 0 | 0 | FlexRay_Driv | Installation |
| 7 | LL_TirePressure | 180 | 240 | 0 | 0 | CAN_Body | Installation |
| 8 | LL_GearboxControl | 260 | 60 | 0 | 0 | FlexRay_Driv | Installation |
| 9 | LL_GearboxControl | 30 | 20 | 8 | 0 | FlexRay_Driv | Kodierung |
| 10 | LL_DoorDriveSide | 180 | 70 | 0 | 1 | CAN_Body | Installation |




(a)




(b)

Abbildung 4.11: Beispielhaftes Scheduling des verwendeten Datensatzes aus Tabelle 4.3. (a) zeigt das durch die MILP erzeugte Ergebnis an, während (b) das durch den HA erzeugte Ergebnis zeigt. Beide weisen eine berechnete Updatedauer von 340 Sekunden auf (nach [56]).

*in der Darstellung kein AUS entlang der horizontalen Achse geschnitten. Beide Schedulings erfüllen die Anforderungen dieses Datensatzes. So sind beispielsweise die AUS mit ID 1, 2 und 3 in beiden Schedulings in der richtigen Reihenfolge.*

*Es zeigt sich, dass sich die beiden Schedulings leicht voneinander unterscheiden, da für die in Tabelle 4.3 dargestellten Daten mehrere Lösungen zu existieren scheinen. Die berechnete Updatedauer ist jedoch für beide Schedulings mit 340 Sekunden gleich, was gleichzeitig der optimalen Lösung für*

*den gegebenen Datensatz entspricht. So können viele andere optimale Lösungen durch minimal vertikale Verschiebungen bestimmter AUS geschaffen werden, solange die Anforderungen noch erfüllt sind.* □

Es ist offensichtlich, dass die Schedulings aus Abbildung 4.11 nicht das volle Potential der Buskapazität ausschöpfen. Dies ist auf die Anforderungen des Datensatzes, wie zum Beispiel eine exklusive Ausführung des AUS mit ID 10 oder die Begrenzung auf maximal drei parallele AUS zurückzuführen. Um die maximal mögliche Zeitersparnis zu bewerten, werden daher in den folgenden Kapiteln Abhängigkeiten und Exklusivitäten zwischen den AUS nicht weiter berücksichtigt.

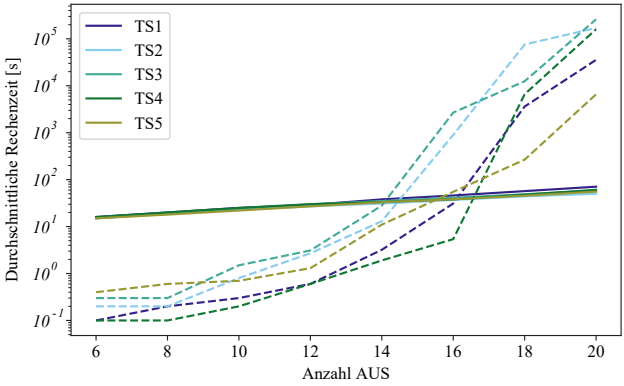
## 4.3 Evaluation: Optimales Scheduling paralleler Steuergeräte-Softwareupdates

Die im Abschnitt 4.2.2 formulierte MILP und der im Abschnitt 4.2.3 eingeführte HA werden nachfolgend ausgewertet und mit Hilfe realer Daten miteinander verglichen. In Abschnitt 4.3.1 erfolgt die Analyse der theoretisch berechneten Ergebnisse, während die real erzielten Zeiteinsparungen in Abschnitt 4.3.2 verifiziert werden.
Dazu werden fünf *Testreihen* (TR) aus einem Datenpool von 50 Steuergeräten zufällig erstellt. Für jeden Flash-, Kodier- und Inbetriebnahme-Container werden die ABB in $KBit/s$ und die Dauer in $s$ ermittelt. Die Dauer eines AUS wird bei der exklusiven Ausführung des AUS gemessen, während die ABB eines AUS wie in Formel 4.1 berechnet wird. Die Parameter für die Tests beider Algorithmen sind wie folgt gewählt: $BW_{input}$ und $BW_{SB_k}$ werden auf 700 $KBit/s$ gesetzt. Die Berechnung des HA wird mit einer *Populationsgröße* von 100 über 20 *Generationen* durchgeführt.
Um eine maximale Zeitersparnis zu erreichen und eine bessere Vergleichbarkeit mit bestehenden Konzepten aus der aktuellen Literatur zu ermöglichen, gibt es in allen Testszenarien keine Abhängigkeiten (wie Reihenfol-

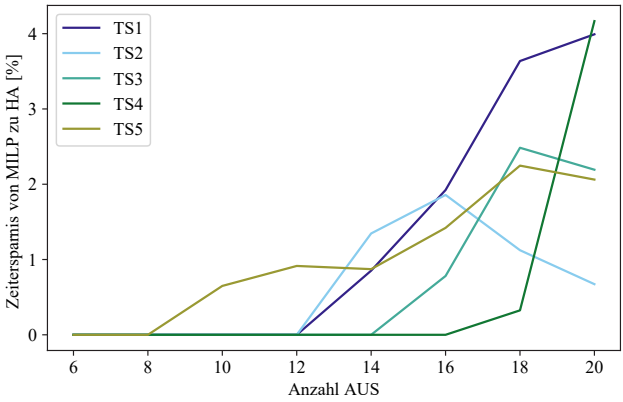
gen) zwischen den AUS oder exklusiv eingeplante AUS. Für jede der fünf Testreihen erfolgt zunächst das Scheduling von sechs AUS. Anschließend werden schrittweise zwei AUS hinzugefügt, bis insgesamt 20 AUS erreicht sind. Folglich berechnen beide Algorithmen ein Scheduling für $6, 8, \ldots, 20$ AUS für jede Testreihe. Dieses Verfahren wurde gewählt, um sicherzustellen, dass die gesamte Updatedauer innerhalb einer Testreihe monoton ansteigt. Die Leistung beider Algorithmen bezüglich der *Rechenzeit* und der erzielten *Updatedauer* wird nachfolgend verglichen. Alle Berechnungen wurden auf einem PC mit Intel Core™ i7, 16GB RAM durchgeführt.

### 4.3.1 Berechnete Ergebnisse der Algorithmen

In diesem Abschnitt werden die MILP und der HA hinsichtlich Rechenzeit und berechneter Updatedauer verglichen. Dies ermöglicht die Überprüfung der Ergebnisse, ohne ein echtes Softwareupdate am Steuergerät durchführen zu müssen. $P_{input}$ und $P_{SB_k}$ werden ohne Einschränkungen gewählt, um das volle Potential des beschriebenen Ansatzes zu ermöglichen. Abbildung 4.12(a) zeigt die durchschnittliche Rechenzeit des MILP (gestrichelte Linien) und des HA (durchgezogene Linien) auf einer logarithmischen Skala. Für jede Testreihe haben beide Algorithmen ein Scheduling berechnet, wobei gleiche Testreihen in der gleichen Farbe dargestellt werden. Die Abbildung 4.12(a) zeigt, dass für eine kleine Anzahl von AUS die MILP den HA bezüglich der benötigten Rechenzeit übertrifft. Da die MILP die optimale Lösung findet, ist dies die Methode der Wahl für eine geringe Anzahl von Steuergeräten. In den Testreihen zeigen beide Algorithmen eine ähnliche Rechenzeit für 14 bis 16 AUS. Für mehr als 16 AUS zeigt der HA eine bessere Rechenzeit. Darüber hinaus liefert die MILP eine größere Varianz in der benötigten Rechenzeit, während der HA für jede Testreihe etwa die gleiche Zeit benötigt.
Die Vorteile des HA in der Rechenzeit für eine große Anzahl von AUS gehen auf Kosten einer insgesamt schlechteren Updatedauer. Abbildung




(a)

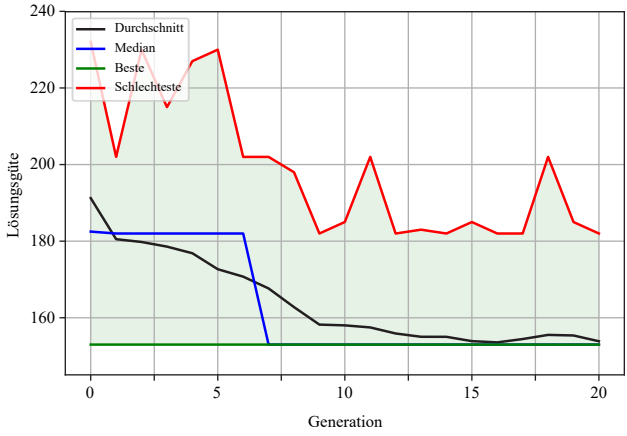



(b)

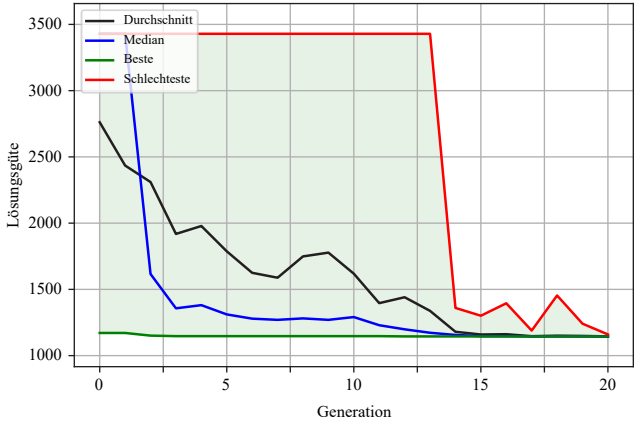
Abbildung 4.12: Vergleich von MILP und HA in Bezug auf Rechenzeit der Algorithmen und durchschnittliche Zeitersparnis der Updatedauer. (a) zeigt die erforderliche Rechenzeit des MILP (gestrichelte Linie) im Vergleich zum HA (durchgezogene Linie). Die entsprechende durchschnittliche Differenz der Updatedauer zwischen MILP und HA ist in (b) dargestellt (nach [56]).

4.12(b) zeigt die durch den HA berechnete durchschnittliche Differenz der Updatedauer gegenüber der optimalen Lösung in Prozent. Insgesamt wächst dieser Unterschied mit der Anzahl der eingeplanten AUS und damit

der Komplexität des Optimierungsproblems. Die Differenz der berechneten Updatedauer beträgt bei den definierten Testreihen bis zu vier Prozent. Bei mehr als 20 AUS wird ein noch größerer Unterschied erwartet. Insgesamt ist der Vorteil des MILP in der berechneten Updatedauer jedoch eher gering und kann für eine große Anzahl von AUS nur dann gerechtfertigt sein, wenn eine Flotte von Fahrzeugen ein Softwareupdate erhalten soll. Dadurch ist es möglich, die längere Rechenzeit des MILP zu kompensieren. Die dazugehörige Detailansicht des Fitnessverlaufs des HA über die Generationen zeigt Abbildung 4.13 exemplarisch für 6 und 20 AUS.
Zusätzlich soll die MILP mit einem Stopp-Kriterium von 600 Sekunden mit dem HA verglichen werden, da die Berechnungszeit des MILP für eine große Anzahl von Steuergeräten sehr lang ist. 600 Sekunden werden gewählt, da dies aus praktischer Sicht eine akzeptable Rechenzeit darstellt. Abbildung 4.12(a) und 4.12(b) zeigen bereits, dass die MILP den HA sowohl in der Rechenzeit als auch in der erforderlichen Updatedauer für bis zu 14 - 16 AUS übertrifft. Untersucht werden soll nun, wie sich die MILP mit Stopp-Kriterium im Vergleich zum HA für mehr als 16 AUS verhält. Abbildung 4.14(a) zeigt die Rechenzeit des MILP (gestrichelte Linien) und des HA (durchgezogene Linien) für alle fünf Testreihen für bis zu 30 AUS. Es ist zu erkennen, dass die Rechenzeit des CPLEX-Solvers auf 600 Sekunden begrenzt ist. Wenn der CPLEX-Solver nach 600 Sekunden aufgrund des Stopp-Kriteriums endet, ist die gefundene Lösung in der Regel nicht optimal. Wenn die MILP früher beendet wird, ist die optimale Lösung gefunden. Der Fokus der Betrachtung liegt dabei auf Ergebnissen, bei denen die genaue Lösung nicht gefunden wurde. Dies ist bei etwa 16 oder mehr AUS der Fall. Abbildung 4.14(b) vergleicht die Ergebnisse des MILP inklusive Stopp-Kriterium mit dem HA. Eine positive Zeitersparnis auf der y-Achse bedeutet, dass die berechnete Updatedauer des MILP das Ergebnis des HA übertrifft. Dies ist bei bis zu ca. 22 AUS der Fall. Für mehr AUS ergeben die Ergebnisse des HA eine kürzere Updatedauer. Dies zeigt die bessere praktische Anwendbarkeit des HA für mehr als 22 parallele AUS. Zu beachten ist,




(a)



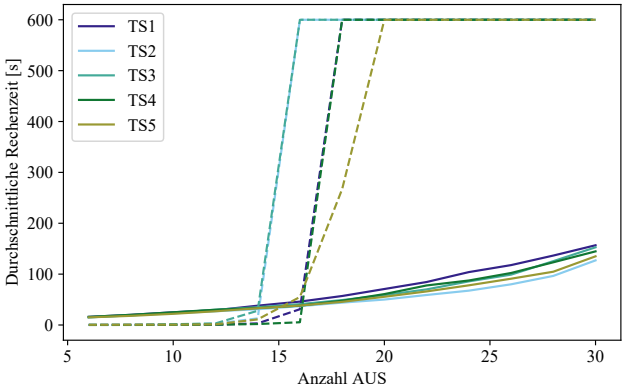

(b)

Abbildung 4.13: Verlauf der Lösungsgüte (Fitness) der Meta-Heuristik über 20 Generationen. (a) zeigt den Verlauf für 6 AUS während (b) den Verlauf von 20 AUS verdeutlicht.

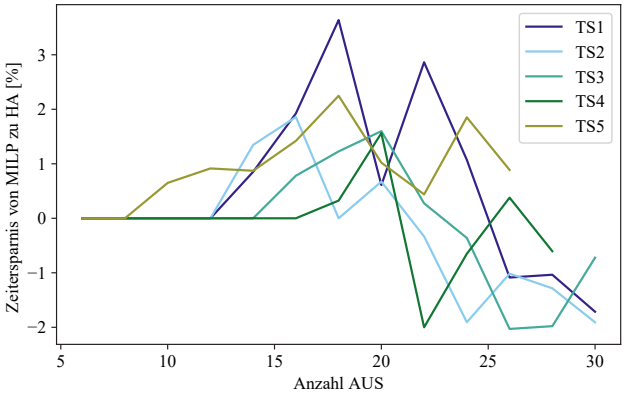
dass der CPLEX-Solver für einige Testreihen innerhalb von 600 Sekunden keine Lösung finden konnte.
Abbildung 4.15(a) zeigt die sequentielle und optimale parallele Updatedauer über eine steigende Anzahl von AUS für alle fünf Testreihen. Sie zeigt das gesamte Potential der Zeiteinsparungen durch die Parallelisierung auf. Der Durchschnitt beider Updateprozesse wird durch eine gestrichelte Linie dargestellt. Es zeigt sich, dass die Zeitersparnis durch die Parallelisierung gegenüber der steigenden Anzahl von AUS linear zunimmt. Im Durchschnitt liegt die Zeitersparnis bei 20 AUS bei 69 %. Zusätzlich ist die Zeitersparnis aller fünf Testreihen als Box-Plot in Abbildung 4.15(b) dargestellt. Abhängig von der jeweiligen Testreihe variiert die durchschnittliche Zeitersparnis zwischen 61 % und 73 %. Eine potentielle Zeitersparnis von bis zu 77 % kann erreicht werden. Damit liegt das hier vorgestellte Konzept im Durchschnitt über den Zeitersparnissen aus der aktuellen Literatur von ca. 39 % bis 61 % [68, 77, 78, 115, 122]. Es gibt auch einige Ausreißer (Testreihen 1 bis 3), die daraus resultieren, dass einzelne AUS mit langer Laufzeit die gesamte Updatedauer dominieren können. Dies gilt insbesondere für das Scheduling einer kleinen Anzahl von AUS.

### 4.3.2 Reale Ergebnisse am Kraftfahrzeug

In diesem Abschnitt werden die durch das Parallelisierungsmodell gezeigten theoretischen Zeiteinsparungen verifiziert. Der generelle Fokus aller Tests liegt darauf, die Machbarkeit des vorgestellten Ansatzes in einer realen Testumgebung zu demonstrieren. Das Hauptziel ist daher, die Abweichung zwischen den theoretischen Ergebnissen aus Abschnitt 4.3.1 und dem realen parallelen Updateprozess zu bestimmen. Alle Tests werden auf einem Prüfstand durchgeführt, der eine exakte Kopie einer aktuellen E/E-Architektur von einem Porsche Panamera (2018) ohne Fahrzeugkarosserie ist. Somit gibt es keine Abweichungen zu einem realen Fahrzeug hinsichtlich der elek-




(a)




(b)

Abbildung 4.14: Aufgrund der langen Berechnungszeit des MILP in Abbildung 4.12 wird die Leistung des MILP in (a) und (b) bewertet, wenn die Berechnungszeit auf 600 Sekunden begrenzt ist. (a) zeigt die Begrenzung von 600s für die Berechnungszeit des MILP (gestrichelte Linie) und eine gleichzeitig erhöhte Anzahl von AUS auf 30. Die zugehörige Differenz zwischen der Updatedauer beider Algorithmen wird in (b) für bis zu 30 AUS dargestellt. Die Begrenzung der Rechenzeit kann zu suboptimalen Lösungen des MILP führen, die insbesondere zwischen 20 und 30 AUS zu sehen sind (nach [56]).

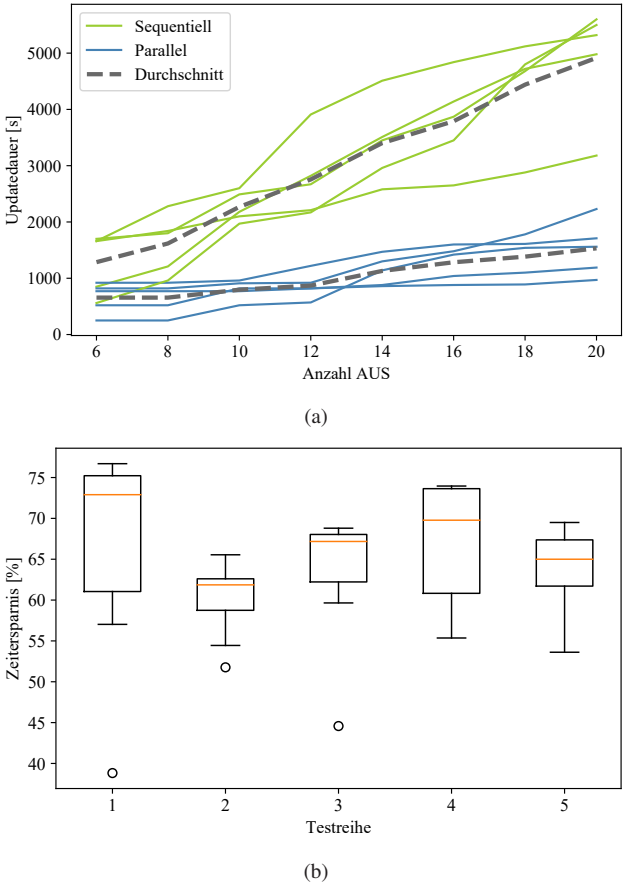



Abbildung 4.15: Zeitersparnis durch optimale Parallelisierung im Vergleich zur sequentiellen Updatedauer. a) zeigt das Potential für eine Reduzierung der Updatedauer für alle fünf Testreihen von 6 bis 20 AUS. (b) zeigt die prozentuale Zeitersparnis aller Testreihen in einem Box-Plot an. Die durchschnittliche Zeitersparnis durch die Parallelisierung liegt zwischen 61 % und 73 % (nach [56]).

tronischen Komponenten. Die E/E-Architektur des Panamera basiert auf einem zentralen Gateway, wie in Abbildung 2.1 exemplarisch dargestellt.

Ein OTX-Skript wird zur Verarbeitung des berechneten parallelen Schedulings genutzt. Dieses Skript startet und steuert mehrere Instanzen von parallelen Flashjobs, die im MCD-Kernel laufen. Alle für das Update benötigten Softwareversionen und Datenobjekte werden in den Diagnosedaten (ODX) gespeichert. Die aufgeführten standardisierten Softwarekomponenten (OTX, MVCI und ODX) definieren den Diagnosetester, der über die DoIP-Schnittstelle mit dem Fahrzeug verbunden ist (vgl. Kapitel 2.2.1). Abbildung 4.16 zeigt den Testaufbau mit den verwendeten Softwarekomponenten. In einem alternativen Testszenario, wie zum Beispiel bei einem OTA-Update, können sich diese standardisierten Diagnose-Softwarekomponenten auch im Fahrzeug selbst befinden (vgl. asynchrones Konzept *Onboard-Toolchain* in Kapitel 2.3.2).
Ist die Dauer eines AUS zu niedrig oder zu hoch angesetzt, wird dies durch eine robuste Implementierung des OTX-Skripts kompensiert, so dass das

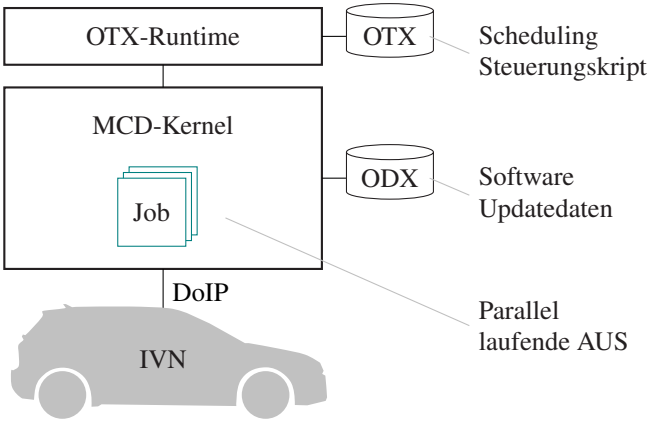


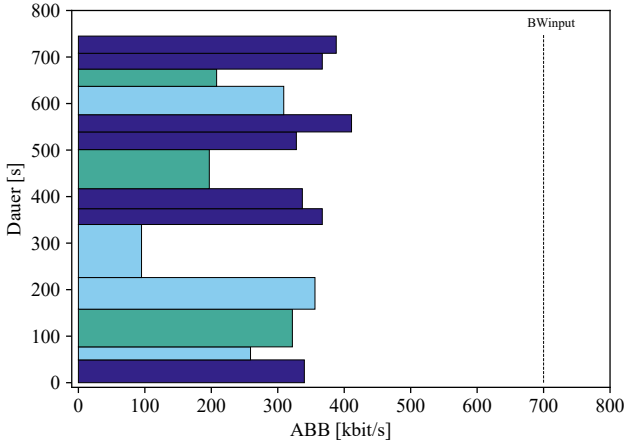

Abbildung 4.16: Detailansicht des Testaufbaus mit den standardisierten Diagnose-Softwarekomponenten. Die Einhaltung des berechneten parallelen Schedulings wird in einem OTX-Skript gesteuert, während parallel ausgeführte AUS im MCD-Kernel instanziiert sind (nach [56]).

Softwareupdate nicht fehlschlägt. Zu diesem Zweck werden sogenannte Warteschritte definiert, um die Abhängigkeiten der AUS im Scheduling einzuhalten. Der Startzeitpunkt eines AUS basiert somit nicht auf festen Zeitstempeln. Beispielsweise muss im Falle einer Abhängigkeit zwischen zwei AUS zuerst ein AUS erfolgreich abgeschlossen werden, bevor das nächste AUS beginnen kann. Eine Unterschätzung der ABB wird jedoch nicht erkannt und kann im schlimmsten Fall zu einem Abbruch des Updates führen. Dieser Fall soll vermieden werden. Wird hingegen die ABB als zu hoch angenommen, geht lediglich Potential bei der Zeiteinsparung verloren. Wenn ein AUS tatsächlich fehlerhaft ausgeführt wird, werden alle parallel laufenden AUS abgeschlossen, bevor das Gesamtupdate vorzeitig beendet wird. Der hierbei entstandene Prozess sowie die Schemas der einzelnen XML-Dateien sind in Anhang A zu finden.

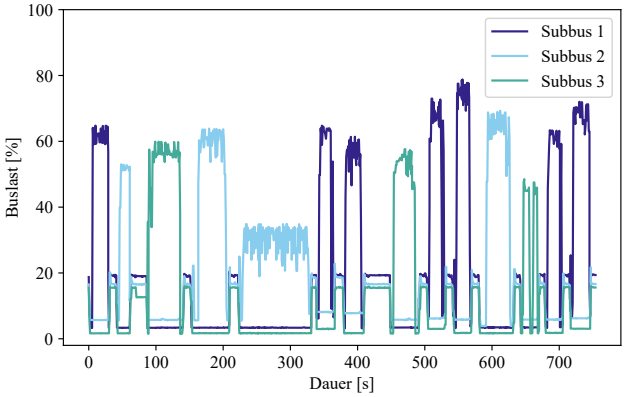
Für die Tests wird ein Datensatz von 20 AUS auf verschiedenen Subbussen gewählt, wobei der HA das parallele Scheduling berechnet. Anschließend werden die Updates am Testaufbau ausgeführt. Diese Ergebnisse werden mit einem sequentiellen Updateprozess (dem heutigen technischen Standard) verglichen. $P_{input}$ und $P_{SB_k}$ werden für alle realen Tests auf drei gesetzt, um einen stabilen Aktualisierungsprozess zu gewährleisten.

Abbildung 4.17 gibt einen detaillierten Einblick in den sequentiellen Updateprozess für 14 beispielhafte AUS. Die Farben symbolisieren den Subbus, zu dem die AUS gehören. Abbildung 4.17(a) zeigt das berechnete Scheduling, wobei jedes Rechteck ein AUS darstellt. In Abbildung 4.17(b) werden die zugehörigen Buslasten des sequentiellen Updates, die für jeden Subbus gemessen wurden, dargestellt. Es zeigt sich, dass die einzelnen Buslasten aus 4.17(b) die berechneten ABBs aus 4.17(a) widerspiegeln.

Die gemessenen Buslasten belegen, dass der Ansatz, die durchschnittlich benötigte Bandbreite wie in Formel (4.1) zu berechnen, einen validen Wert ergibt. Außerdem zeigt sich, dass die Bandbreite über die Dauer des AUS annähernd konstant bleibt. Abbildung 4.18(a) zeigt den berechneten parallelen Schedule der gleichen 14 AUS wie in Abbildung 4.17(a). Da ein AUS




(a)




(b)

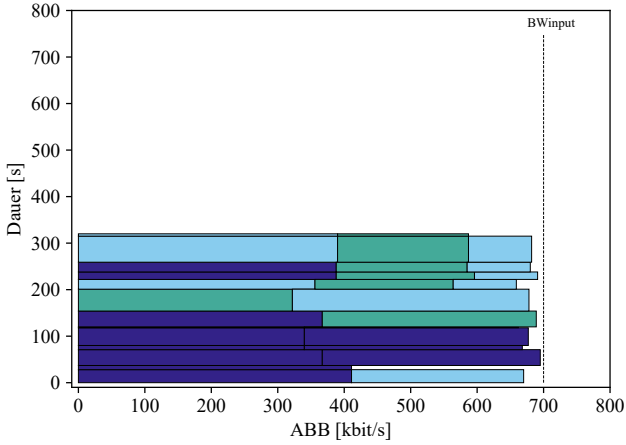
Abbildung 4.17: Reale Testergebnisse mit 14 exemplarischen AUS, welche sequentiell prozessiert werden. (a) zeigt den sequentiellen Update-Schedule, während (b) die dazugehörigen Buslasten der Subbusse veranschaulicht (nach [56]).

horizontal geschnitten werden kann, können mehrere Rechtecke einem AUS zugeordnet werden, wie es bereits in Abbildung 4.5 zu sehen ist. Die zu 4.18(a) zugehörige Buslast ist in Abbildung 4.18(b) dargestellt. Auch hier

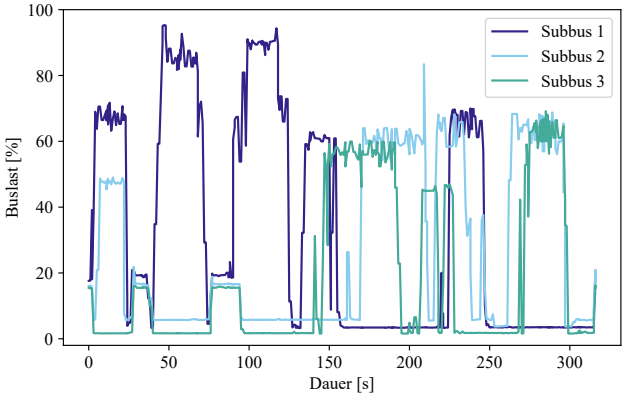
kann ein eindeutiger Zusammenhang zwischen ABB und Buslast hergestellt werden. Außerdem ist in Abbildung 4.18(b) zu sehen, dass noch Potential für höhere Buslasten einzelner Subbusse vorhanden ist. Das ist genau dann der Fall, wenn verschiedene Subbusse gleichzeitig parallelisiert werden und $BW_{input}$ kleiner oder gleich $BW_{SB_k}$ gewählt ist. Durch Setzen von $BW_{input} > BW_{SB_k}$ kann die maximale Auslastung auch auf mehreren verschiedenen Subbussen gleichzeitig erreicht werden. Dies muss jedoch von der Bandbreite des Eingangsbusses und der Routing-Leistung des Gateways unterstützt werden.

Alle experimentellen Testergebnisse sind in Abbildung 4.19 zusammengefasst. Insgesamt spiegelt die berechnete parallele Updatedauer die reale parallele Updatedauer sehr gut wider. Bei einem realen parallelen Update ergibt sich jedoch eine minimal höhere Updatedauer, da im Modell keine Übergangs- oder Umschaltzeiten zwischen den AUS berücksichtigt werden. Darüber hinaus kann sich das Aufwachen oder Neustarten von Steuergeräten während des Updateprozesses ebenfalls negativ auf die Gesamtdauer auswirken. Im Durchschnitt ist die reale parallele Updatedauer nur 4 % höher als die berechnete parallele Updatedauer. Das gleiche Verhalten kann für die berechnete und reale sequentielle Updatedauer beobachtet werden, wobei die durchschnittliche Abweichung 5 % beträgt.

Abbildung 4.19 zeigt auch, dass die effektive Zeitersparnis mit der Anzahl der parallelisierten Steuergeräte zunimmt. Die Zeiteinsparung reicht von 39 % bis 67 %, wobei 67 % für 20 AUS erreicht werden. Dieser Effekt stimmt mit den theoretischen Ergebnissen aus Abbildung 4.15(a) überein. Die experimentellen Gesamtergebnisse bestätigen somit das Zeiteinsparungspotential, das in der Realität durch das vorgestellte Schedulingmodell erreicht werden kann. Zu beachten ist, dass alle erzielten Zeiteinsparungen nur beispielhaft für den hier verwendeten Versuchsaufbau sind. Für andere Systemspezifikationen, wie zum Beispiel unterschiedliche maximale Routing-Leistungen des Gateways oder Datenübertragungsraten der Subbusse, können diese abweichen.




(a)




(b)

Abbildung 4.18: Reale Testergebnisse mit 14 exemplarischen AUS, welche parallel prozessiert werden. (a) zeigt den parallelen Update-Schedule, während (b) die dazugehörigen Buslasten der Subbusse veranschaulicht (nach [56]).

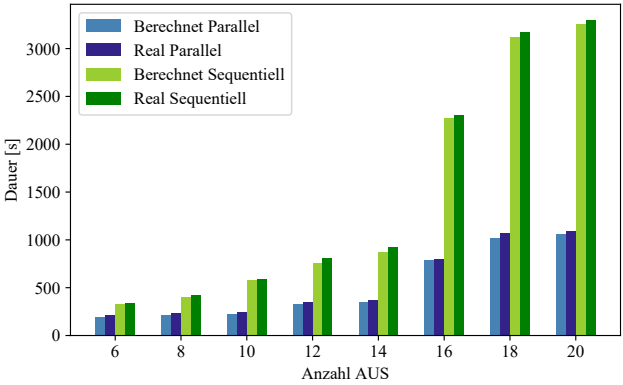



Abbildung 4.19: Vergleich zwischen realer sequentieller und realer paralleler Updatedauer. Zudem wird jeweils die Abweichung zwischen berechneter und tatsächlicher Dauer für den sequentiellen und parallelen Updateprozess dargestellt (nach [56]).

## 4.4 Alternative Ausrichtungen des Schedulings

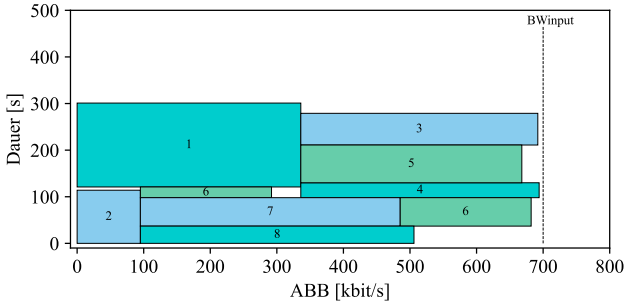
Die bisherige Ausrichtung des parallelen Schedulings zielt ausschließlich auf eine maximale Zeiteinsparung beim Softwareupdate über mehrere Steuergeräte ab. Die Zeitersparnis durch eine Parallelisierung kann jedoch auch hinsichtlich einer erhöhten Prozesssicherheit oder Energieeffizienz erfolgen. Daher sollen im Folgenden zwei alternative Ausrichtungen des parallelen Schedulings betrachtet werden, welche sich nicht ausschließlich auf eine maximal mögliche Reduzierung der Updatedauer fokussieren. Beide Ausrichtungen werden erstmals in [126] behandelt.

Während eines Softwareupdates im Kraftfahrzeug müssen entweder die gesamte Fahrzeugelektronik oder nur aktuell benötigte Subbusse wachgehalten werden. Um Energie während des Gesamtfahrzeugupdates zu sparen, sollten die Subbusse nur so lang wie nötig aktiv gehalten werden. Aus diesem Gedanken resultiert eine Strategie mit möglichst *energieeffizienter Ausrichtung*. Sie lässt sich durch eine ausschließlich sequentielle Bearbei-

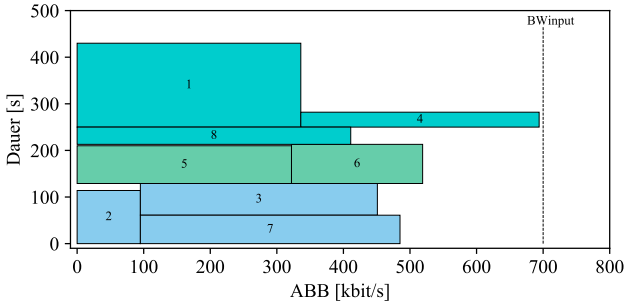
tung der *unterschiedlichen* Subbusse realisieren, wobei eine Parallelisierung auf dem jeweiligen Subbus nach wie vor erfolgen darf. Umgesetzt werden kann dies, indem die zu aktualisierenden AUS ihrer Gruppe des jeweiligen Subbusses zugeordnet werden. Der Scheduling-Algorithmus wird danach für die jeweilige Gruppierung (Subbus) ausgeführt, um sie anschließend zu einem gesamtheitlichen Scheduling zusammenzuführen. Der Aufbau des Schedulings ähnelt dem Prinzip einer Schichttorte, wobei die einzelnen Schichten den Subbussen (auf denen jeweils parallelisiert werden darf) entsprechen.

Alternativ dazu sieht eine möglichst *prozesssichere Ausrichtung* vor, keine Parallelisierung auf den Subbussen zuzulassen. Die Parallelisierung findet dadurch ausschließlich Subbus-übergreifend statt. Diese Ausrichtung kann insbesondere dem Verlust von Datenpaketen auf Subbussen entgegenwirken oder dabei helfen, systembedingte Einschränkungen bei der Parallelisierung auf bestimmten Subbussen zu umgehen. Umgesetzt werden kann es, indem für alle Subbusse $P_{SB_k} = 1$ gesetzt wird. Zur Verdeutlichung der beiden vorgestellten Varianten dient Abbildung 4.20.

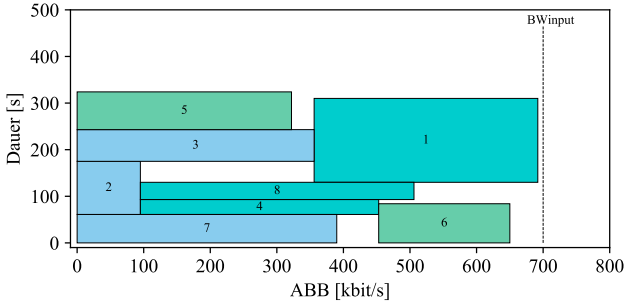
Es werden acht AUS an drei Subbussen betrachtet, wobei die Farbe den jeweiligen Subbus des AUS symbolisiert. Abbildung 4.20(a) dient als Referenz und entspricht einem Scheduling mit maximaler Zeiteinsparung, wie es im vorherigen Kapitel 4.3.1 vorgestellt wurde. Die Variante mit möglichst kurzer Wachhaltedauer der Subbusse zeigt Abbildung 4.20(b). Die einzelnen *Schichten* sind durch die farblichen Markierung der Subbusse deutlich zu erkennen. Diese energieeffiziente Ausrichtung hat den Vorteil, dass die durchschnittliche Wachhaltedauer der Subbusse sinkt, jedoch auf Kosten der benötigten Updatedauer im Vergleich zur reinen Parallelisierung ohne Ausrichtung (vgl. Abbildung 4.20(a)). Die prozesssichere Ausrichtung der gleichen acht AUS ist in Abbildung 4.20(c) veranschaulicht. Es ist zu erkennen, dass keine AUS vom selben Subbus (gleiche Farbe) parallelisiert werden.




(a)




(b)




(c)

Abbildung 4.20: Vergleich unterschiedlicher Ausrichtungen des parallelen Schedulings für acht AUS. (a) zeigt die Parallelisierung mit maximaler Zeitersparnis. (b) zeigt eine energieeffiziente Ausrichtung, wobei die Wachhaltedauer der einzelnen Subbusse auf Kosten der gesamten Updatedauer reduziert wird. (c) zeigt eine prozesssichere Ausrichtung wobei dies zu Kosten der Updatedauer und Energieeffizienz geht. Die Farben repräsentieren den jeweiligen Subbus (in Anlehnung an [126]).

Diese Variante kann trotz Parallelisierung zu einem stabileren Updatevorgang beitragen. Nachteile sind hierbei sowohl eine insgesamt längere Updatedauer als auch Wachhaltedauer der Subbusse und somit höherem Energieverbrauch.
Aufgrund der gezeigten Praxistauglichkeit des HA sind die unterschiedlichen Ausrichtungen lediglich im HA implementiert und getestet. Ziel hierbei ist es, das Potential der unterschiedlichen Ausrichtungen abzuschätzen und aufzuzeigen. Die erzielten Vor- und Nachteile (Einsparungen) der jeweiligen Ausrichtung sind abschließend in Tabelle 4.4 für die Ergebnisse aus Abbildung 4.20 gegenübergestellt.

Tabelle 4.4: Vor- und Nachteile der Scheduling Ausrichtungen.

| | Maximale Zeitersparnis | Energie-effizienz | Prozess-sicherheit |
|---|---|---|---|
| Dauer Sequentiell | 657 s | 657 s | 657 s |
| Dauer Parallel | 301 s | 430 s | 324 s |
| Zeitersparnis | 54 % | 34 % | 50 % |
| Wachhaltedauer Subbusse | 587 s | 430 s | 657 s |

## 4.5 Diskussion und Fazit

In diesem Kapitel wird eine Methode beziehungsweise ein Modell vorgestellt, das eine optimale und automatisierte Parallelisierung von Steuergeräte-Softwareupdates ermöglicht. Dazu ist ein Teilschritt eines Steuergeräte-Softwareupdates als abstraker Updateschritt mit den Dimensionen Dauer und allokierter Busbandbreite definiert. Um das Optimierungsproblem auf den Bussystemen im Fahrzeug optimal zu lösen, werden zwei Scheduling-

Algorithmen vorgestellt und hinsichtlich Rechenzeit und maximaler Zeitersparnis miteinander verglichen.
Die MILP liefert ein optimales Ergebnis. Die Rechenzeit steigt jedoch bei zunehmenden Updateschritten überproportional an. Bei identischen Updates für einen großen Fuhrpark oder bei Updates mit geringer Anzahl an AUS könnte sich die einmalige Berechnung dennoch lohnen.
Der HA liefert ein ausreichend gutes Ergebnis. Das Scheduling ist höchstens 4 % schlechter als das Optimum der fünf Testreihen. Damit ist der HA eine geeignete Methode, um ein Update für ein einzelnes Fahrzeug oder eine kleine Flotte zu parallelisieren.
Gesamtergebnisse mit bis zu 20 AUS haben gezeigt, dass die gesamte Updatedauer um durchschnittlich 69 % und bis zu 77 % im Vergleich zu sequentiellen Updates reduziert werden kann. Dies hängt jedoch von der Art der eingeplanten AUS ab. AUS mit relativ kleiner ABB und etwa gleicher Dauer können am effizientesten parallelisiert werden. Bezogen auf OTA-Updates, im Kontext der OTA-Dienste (vgl. Tabelle 2.1), hat eine Parallelisierung somit den größten Effekt im Falle eines Gesamtfahrzeugupdates (Software Maintenance).
Der vorgestellte Ansatz wurde auf einer E/E-Architektur mit zentralem Gateway getestet, ist aber aufgrund der hohen Abstraktion eines AUS leicht an zukünftige Änderungen der E/E-Architektur anpassbar (siehe Abschnitt 2.1). Zukünftige dateibasierte Steuergeräte-Softwareupdates anstelle von flashbasierten Steuergeräte-Softwareupdates können ebenfalls berücksichtigt werden, indem die Ermittlung beziehungsweise Berechnung der ABB entsprechend adaptiert wird (siehe Abschnitt 2.2.2). Des Weiteren lässt sich der Scheduling-Algorithmus, durch die in Kapitel 4.1 vorgestellten Anforderungen, frei auf die Systemspezifikationen eines Fahrzeugs zuschneiden.
Durch die in Abschnitt 4.2.1 erfolgte Abstraktion, einen Aktualisierungsschritt als AUS zu modellieren, ist es grundsätzlich möglich, die *Parallelen Updates* auch mit den Konzepten *Delta Updates* und *Code Compression* aus Abschnitt 3.3 zu kombinieren. Grundsätzlich reduzieren beide Konzepte

(DU und CC) lediglich die zu installierende Datenmenge. Dies betrifft somit ausschließlich die beiden Modellparameter ABB und Dauer eines AUS. Es ist zu erwarten, dass sich die Dauer aufgrund der geringeren zu installierenden Datenmenge verkürzt. Die ABB kann auch durch andere Verarbeitungsschritte im Steuergerät, wie zum Beispiel dem Dekomprimieren der Daten, variieren. Zusammengefasst konnten die folgenden Merkmale durch das hier vorgestellten Scheduling-Verfahren erzielt werden:

- *Optimales paralleles Scheduling:* Das Scheduling kann so konfiguriert werden, dass es das volle Potential der E/E-Architektur nutzt.
- *Automatisierte Scheduling Strategie:* Das Scheduling berücksichtigt updatespezifische Anforderungen und Abhängigkeiten aller Steuergeräte und Updateschritte.
- *Hohe Abstraktion:* Eine einfache Adaption an bestehende und zukünftige Bussysteme, E/E-Architekturen oder dateibasierte Softwareupdates ist möglich.
- *Kompatibilität zu anderen Lösungsansätzen:* Die Kombination mit anderen Konzepten zur Reduktion der Updatedauer, beispielsweise *Delta Updates* und *Code Compression*, ist denkbar.

Eine zukünftige Möglichkeit die Rechenzeit des MILP zu reduzieren, könnte die sequentielle Kombination beider Algorithmen sein. Hierbei berechnet der HA eine mögliche, aber nicht zwangsläufig optimale Lösung. Diese Lösung wird dann als Eingangsgröße, beziehungsweise Startpunkt für die MILP verwendet. Somit beginnt die Lösungsfindung des MILP idealerweise nahe am Optimum, was auch als Warmstart bezeichnet wird. Auch die Betrachtung einer flexiblen ABB ermöglicht eine weitere Verbesserung des Modells, wie es bereits in Abschnitt 4.2.2 erwähnt wurde. Konkret wäre eine Reduktion der ABB bei gleichzeitiger Erhöhung der Dauer denkbar, da die Verarbeitungsgeschwindigkeit der Steuergeräte begrenzt ist. Die hier vorgestellte Ermittlung der ABB würde daher die maximale ABB für diesen zukünftigen Ansatz darstellen. Diese Idee kann zu einer weiteren Verbes-

serung der gesamten Aktualisierungsdauer führen, ist jedoch nur für den Installationsschritt anwendbar und erfordert eine anspruchsvollere Implementierung des eigentlichen Flashprozesses im Steuergerät.

# 5 Ermittlung des Update-Installationszeitpunktes

Ziel dieses Kapitels ist es, die identifizierten Defizite bezüglich Mobilitätsvorhersagen aus Abschnitt 3.4 aufzugreifen und ein Prognoseverfahren zur zukünftigen Fahrzeugnutzung für die automatisierte Ermittlung des Update-Installationszeitpunktes auszuarbeiten. In Anbetracht der drei Teilschritte zur automatisierten Ermittlung des Installationszeitpunktes (siehe Abbildung 3.4), besteht die eigentliche Herausforderung weniger in der Einplanung selbst, sondern vielmehr in einer möglichst zuverlässigen und genauen Prognose des zukünftigen Mobilitätsverhaltens. Ist die Prognosegüte zu ungenau, kann auch die darauf aufbauende Einplanung selbst keine hohe Güte erzielen. Daher soll der Schwerpunkt dieses Kapitels auf einer möglichst präzisen Prognose der zukünftigen Fahrzeugnutzung liegen. Die bereits in Abschnitt 3.4 erfolgte Kategorisierung der hier benötigten Mobilitätsvorhersage in *unimodal*, *nutzerspezifisch* und *mittelfristige Mobilitätsvorhersage* stellt dabei konkrete Anforderungen an das zu erarbeitende Prognoseframework. Dies ist entsprechend bei der Modellbildung zur berücksichtigen.
Als Ausgangslage der Modellbildung erläutert Abschnitt 5.1 die Erhebung und Aufbereitung der Mobilitätsdaten. Das eigentliche Prognoseframework, welches aus den beiden Teilschritten *Inter-* und *Intra-Tages-Prädiktion* besteht, wird in Abschnitt 5.2 behandelt. Mittels eines ausführlichen Beispiels zweier exemplarischer Nutzerprofile wird die konkrete Funktionsweise des Prognoseframeworks veranschaulicht. Die Analyse der erzielten Prognosegüte des Prognoseframeworks erfolgt in Abschnitt 5.3. Durch einen direkten Vergleich zweier unterschiedlicher Datensätze von realen Personen, wird ei-

ne bessere Einordnung der erzielten Prognosegüte gewährleistet. Abschnitt 5.4 erläutert abschließend die automatisierte Einplanung des OTA-Updates mit dem daraus abgeleiteten Installationszeitpunkt, bevor Abschnitt 5.5 die gewonnenen Erkenntnisse zusammenfasst. Einige Inhalte dieses Kapitels wurden bereits in verkürzter Form in [57, 93] veröffentlicht.

## 5.1 Definition und Vorverarbeitung von Mobilitätsprofilen

In diesem Abschnitt erfolgt zunächst die Definition von Fahrzeugnutzungsprofilen, bevor auf die konkrete Erhebung und Aufbereitung der zugehörigen Daten eingegangen wird.

### 5.1.1 Definition von Fahrzeugnutzungsprofilen

Da sich ein OTA-Softwareupdate auf das Kraftfahrzeug als System bezieht und nicht auf die verschiedenen Fahrzeugnutzer selbst, wird auch nicht zwischen mehreren potentiellen Fahrzeugführern bei der Mobilitätsvorhersage unterschieden. Dies würde die Prognose ausschließlich erschweren, hätte aber keinen weiteren Vorteil bezüglich der vorgesehenen Einplanung von OTA-Updates. Als Grundlage für die Prognose der zukünftigen Fahrzeugnutzung muss folglich pro Fahrzeug ein eigenes *Fahrzeugnutzungsprofil* (FNP) erhoben und angelegt werden. Dabei soll sich ein FNP wiederum aus der eigentlichen *Fahrzeugbewegungsaufzeichnung* und den *Fahrzeugumgebungsbedingungen* zusammensetzen. Beide sind nachfolgend näher erläutert:

- *Fahrzeugbewegungsaufzeichnung*: Die Literatur behandelt viele Möglichkeiten zur Aufzeichnung von zeitlichen und örtlichen Mobilitätsprofilen von Personen [136]. In Abhängigkeit des Anwendungsfalls können diese stark variieren, wodurch gleichzeitig diverse Vor- und Nachteile entstehen. Als Datenquellen dienen hierbei meist das

*Global Positioning System* (GPS), Mobilfunkzellen, Wireless-LAN Access Points oder Ticketsysteme von öffentlichen Nahverkehrssystemen (zum Beispiel U-Bahn). Ortungsverfahren wie GPS können einen bis auf wenige Meter genauen Standort am Boden ermitteln [95]. Aufzeichnungen auf Basis von Mobilfunkzellen oder Ticketsystemen wiederum erschließen lediglich einen potentiellen Bereich (nicht zwangsläufig am Boden), durch den bereits systembedingt die Genauigkeit zur Aussage des konkreten Aufenthalts leidet.

- *Fahrzeugumgebungsbedingungen*: Dabei handelt es sich um zusätzliche Informationen, wie beispielsweise die Verfügbarkeit einer Ladestation oder Wireless-LAN an einem bestimmten Standort, mit denen das Fahrzeugnutzungsprofil angereichert werden kann. Diese Zusatzinformationen sind für die eigentliche Einplanung das OTA-Updates von Nutzen.

### 5.1.2 Erhebung und -aufbereitung von Fahrzeugnutzungsprofilen

Aufgrund der in Abschnitt 5.1.1 genannten Vorteile und der hohen Verfügbarkeit im Fahrzeug fällt die Auswahl zur Fahrzeugbewegungsaufzeichnung auf das Ortungsverfahren GPS. Tabelle 5.1 zeigt, wie die Daten bei der Erhebung in Rohform vorliegen. Diese bestehen aus einer Abfolge von GPS-Koordinaten (Breiten- und Längengrad), die mit einem Zeitstempel versehen sind. Diese aufgezeichneten Abfolgen entsprechen somit den zurückgelegten Fahrten eines Fahrzeugs, also dem Wechsel zwischen zwei Standorten.

Zur Weiterverarbeitung müssen die GPS-Rohdaten aufbereitet und in ein einheitliches Datenformat überführt werden. Dabei lässt sich grundsätzlich das Mobilitätsverhalten eines Fahrzeugs durch die Standorte und Fahrten dazwischen charakterisieren. Folglich werden wiederkehrende Standorte und identische Fahrten aus der Fahrzeugbewegungsaufzeichnung extrahiert.

Tabelle 5.1: GPS-Rohdaten einer Fahrt mit Zeitstempel.

| Zeit | Breitengrad | Längengrad |
|---|---|---|
| $t_1$ | $lat_1$ | $lon_1$ |
| $t_2$ | $lat_2$ | $lon_2$ |
| $t_3$ | $lat_3$ | $lon_3$ |
| ⋮ | ⋮ | ⋮ |

Aufenthaltsorte entsprechen dabei sogenannten *Points of Interest* (POIs) und werden anhand einer eindeutigen *POI-ID* unterschieden, während wiederkehrende Fahrten über eine *Trip-ID* identifiziert werden.

#### Bildung Trip-ID

Trip-IDs werden direkt aus der Folge von GPS-Koordinaten, wie die Fahrzeugbewegungsaufzeichnung aus Tabelle 5.1, abgeleitet. Kanten verbinden die einzelnen GPS-Punkte miteinander. Dabei ist es grundsätzlich möglich, einen Standort $x$ von einem Standort $y$ über verschiedene Routen zu erreichen. Über die Trip-ID sollen ähnliche und häufig gefahrene Routen zusammengefasst und abgespeichert werden. Zwei Routen A und B gelten als gleich, wenn der Durchschnitt der kleinsten Entfernungen zwischen allen Kanten der Route A und ihren korrespondierenden Koordinaten der Route B unter einem bestimmten Schwellenwert liegt. Die Berechnung basiert auf der *Hausdorff-Metrik* und ist abgeleitet aus den Methoden in [14] und [20]. Die Bildung von Trip-IDs ermöglicht somit nicht nur die Vorhersage wohin jemand gefahren ist, sondern auch welche Route voraussichtlich genommen wird.

### Bildung POI-ID

Vergleichbar zu wiederkehrenden Fahrten (Trips) werden häufig besuchte Standorte als POIs identifiziert. Einzelnen Standorten wird dabei eine eindeutige POI-ID zugeordnet. Die Bildung von POIs auf Grundlage einer GPS-Trajektorie ist ein weitreichend untersuchtes Forschungsgebiet [12, 42, 87, 138]. In den meisten Fällen findet die Erstellung von POIs mittels zwei Größen statt. Der *Radius* definiert den Umkreis eines POI, während die *Standzeit* entscheidet, ob ein neuer POI gebildet wird. Die Parametrierung des Radius als auch der Standzeit variiert dabei in den unterschiedlichen Studien [12, 42, 138].
Da es sich bei der Fahrzeugbewegungsaufzeichnung lediglich um Fahrten handelt, ergeben sich die POIs aus den GPS-Abfahrts- und Ankunftskoordinaten eines Trips. Dies entspricht der ersten und letzten Zeile aus Tabelle 5.1. Die Standzeit spielt somit keine Rolle. Im hier betrachteten Fall sollen die Abfahrts- und Ankunftskoordinaten im Radius von empirisch ermittelten 800 Metern zu einem POI zusammengefasst werden. Hierzu wird das *Mean-Shift Clustering* verwendet [29, 43]. Die GPS-Abfahrts- und Ankunftskoordinaten des ersten Trips definieren dabei das jeweilige Zentrum der Cluster. Kommen neue GPS-Abfahrts- beziehungsweise GPS-Ankunftskoordinaten innerhalb eines Radius von 800 Metern hinzu, entspricht das neue Zentrum dem Mittelpunkt beider Koordinaten. Dieser Prozess setzt sich kontinuierlich fort, sobald neue Fahrten aufgezeichnet werden. Anzumerken ist, dass POIs zusätzliche Informationen, wie Verfügbarkeit von Ladesäulen, hinterlegt haben können. Dies entspricht den in Abschnitt 5.1 erläuterten Fahrzeugumgebungsbedingungen.
Jedes FNP kann nun mittels der aggregierten Information aus POIs und Trips beschrieben werden. Eine Visualisierung beider Größen für ein exemplarisches FNP liefert Abbildung 5.1. Unterschiedliche Trip-IDs sind farblich hervorgehoben. Zudem zeigt der Durchmesser eines POIs jeweils die Häufigkeit der Besuche an.

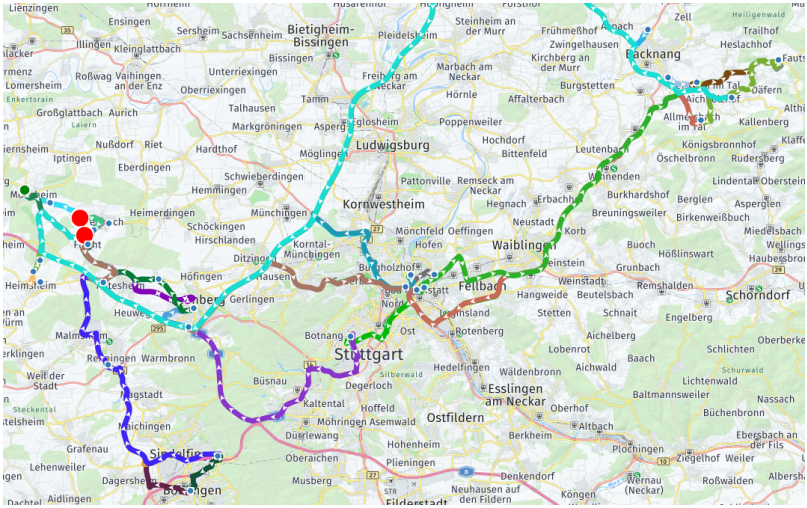



Abbildung 5.1: Darstellung aufbereiteter POIs und Trips anhand einer beispielhaften Fahrzeugbewegungsaufzeichnung. Unterschiedliche Trip-IDs werden über verschiedene Farben visualisiert. Kreise symbolisieren POIs. Je öfter ein POI besucht wurde, desto größer ist der Kreis. Gleich große Kreise haben dieselbe Farbe.

Tabelle 5.2 zeigt entsprechend, wie die Profildaten nach der Vorverarbeitung vorliegen. Dabei entspricht jede Zeile einer Fahrt. Mit Hilfe dieses Datenformats lässt sich ein kompletter Mobilitätsverlauf aus gefahrenen Routen und Aufenthaltsorten mit den jeweiligen Aufenthalts- und Fahrtdauern beschreiben.

Eine direkt daraus resultierende Visualisierung liefert Abbildung 5.2. Dabei handelt es sich um das gleiche FNP wie in Abbildung 5.1. Konkret ist das Mobilitätsverhalten über mehrere aufeinanderfolgende Tage veranschaulicht. Jede Zeile zeigt den Verlauf eines Tages, wobei jede Farbe eine POI-ID symbolisiert. Zur vereinfachten Darstellung sind trotz unterschiedlicher Trip-IDs alle Trips in der gleichen Farbe (dunkelblau) dargestellt. Zudem ist ein Tag in diskrete Zeitslots (hier Minuten) eingeteilt. Die Pfeile

Tabelle 5.2: Darstellung des Datenformats der Fahrzeugbewegungsaufzeichnung anhand von POI-IDs und Trip-IDs. Jede Zeile entspricht einer Fahrt zwischen zwei POIs einschließlich Abfahrts- und Ankunftszeit (nach [57]).

| $\text{POI}_{Abfahrt}$ | $\text{Zeit}_{Abfahrt}$ | Trip | $\text{POI}_{Ankunft}$ | $\text{Zeit}_{Ankunft}$ |
|---|---|---|---|---|
| $\text{POI-ID}_1$ | $t_1$ | $\text{Trip-ID}_1$ | $\text{POI-ID}_2$ | $t_2$ |
| $\text{POI-ID}_2$ | $t_3$ | $\text{Trip-ID}_2$ | $\text{POI-ID}_3$ | $t_4$ |
| $\text{POI-ID}_3$ | $t_5$ | $\text{Trip-ID}_3$ | $\text{POI-ID}_1$ | $t_6$ |
| $\text{POI-ID}_1$ | $t_7$ | $\text{Trip-ID}_1$ | $\text{POI-ID}_2$ | $t_8$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

symbolisieren erkennbare tagesspezifische beziehungsweise tagesübergreifende Mobilitätsmuster.

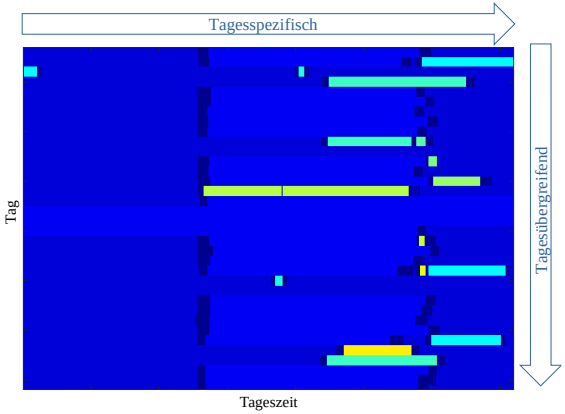



Abbildung 5.2: Visualisierung der Fahrzeugbewegungsaufzeichnung am Beispiel von 30 Tagen über die Reihenfolge der POIs des jeweiligen Tages. Jede Farbe stellt einen POI dar, wobei alle Fahrten einheitlich dunkelblau dargestellt sind.

## 5.2 Modellbildung: Örtliche- und zeitliche Mobilitätsvorhersage

Das hier vorgestellte Prognoseframework ist ein zweistufiges Verfahren, welches aus einer *Inter-Tages-Prädiktion* und einer nachgelagerten *Intra-Tages-Prädiktion* besteht. Diese Zweistufigkeit ergibt sich aus der Anforderung tagesspezifische und tagesübergreifende Mobilitätsmuster ermitteln und prognostizieren zu können (vgl. Abbildung 5.2).
Tagesübergreifende Muster behandelt dabei die Inter-Tages-Prädiktion, wobei zwei alternative Varianten betrachtet werden. Die erste Variante (*Wochentagabhängig*) verfolgt einen klassischen Ansatz, bei dem die Prognose des zukünftigen Tages über den jeweiligen Wochentag erfolgt. Hierfür werden sogenannte *Weekday Cluster* (WC) bestimmt. So wird beispielsweise ein zukünftiger Dienstag auf Basis aller in der Fahrzeugbewegungsaufzeichnung historisch aufgezeichneten Dienstage prognostiziert. Für Fahrzeugführer, die einem klassischen wöchentlichen Rhythmus folgen, kann dadurch bereits eine einfache und sinnvolle Mobilitätsvorhersage erstellt werden. Dieses Vorgehen schlägt jedoch bei Personen fehl, die sich außerhalb eines wöchentlichen Rhythmus bewegen und somit ein atypisches Mobilitätsmuster aufweisen, wie beispielsweise Schichtarbeiter. Deshalb wird eine weitere Variante (*Wochentagunabhängig*) vorgestellt, bei der ähnliche 24-Stunden-Perioden unabhängig vom Wochentag erfasst und bei Ähnlichkeit geclustert werden. Sie werden als *Feature Cluster* (FC) bezeichnet. Diese Clusterbildung entkoppelt die Abhängigkeit der natürlichen Sequenz der Wochentage im Vergleich zur wochentagabhängigen Variante. Die Prognose des nächsten Tages erfolgt nun über die Folge der zuvor geclusterten Tage, die durch ein *Künstliches Neuronales Netz* (KNN) realisiert wird.
Insgesamt betrachtet ermittelt die Inter-Tages-Prädiktion aus dem jeweiligen FNP alle *relevanten Tage*, die für die nachgelagerte Intra-Tages-Prädiktion zu verwenden sind. Basierend auf dem Ergebnis der Inter-Tages-Prädiktion prognostiziert die Intra-Tages-Prädiktion somit das tagesspezifische

Mobilitätsverhalten während eines zukünftigen Tages. Die Gesamtstruktur des Prognoseframeworks, inklusive der Teilschritte Inter- und Intra-Tages-Prädiktion, ist in Abbildung 5.3 dargestellt.

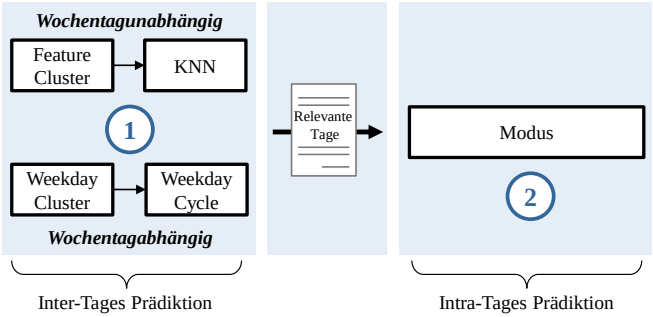



Abbildung 5.3: Übersicht des Prognoseframeworks. Der erste Schritt entspricht der Inter-Tages-Prädiktion, die aus zwei alternativen Varianten besteht. Der erste Variante *Wochentagunabhängig* erzeugt *Feature Cluster* und verwendet ein KNN für die Vorhersage des nächsten Clusters. Die zweite Variante ist *Wochentagabhängig* und erstellt *Weekday Clusters*. Im zweiten Schritt wird die Intra-Tages-Prädiktion auf Grundlage der in Schritt eins ausgewählten Daten (relevante Tage) erstellt. Um mehr als einen Tag vorherzusagen, werden die Teilschritte entsprechend oft wiederholt (nach [57]).

Durch die abstrakte Herangehensweise des Prognoseframeworks ist eine universelle Verwendung jeglicher GPS-Rohdaten (Fahrzeug, Smartphone, etc.) möglich. Einzige Voraussetzung ist die Überführung der GPS-Rohdaten in POI-IDs und Trip-IDs, wie in Abschnitt 5.1.2 betrachtet. Dies schafft eine Vergleichbarkeit zu anderen Forschungsprojekten wie [117] oder [123]. Im Fokus steht dabei die mittelfristige Mobilitätsvorhersage von mehreren Tagen bis zu einer Woche. Sowohl die Inter- als auch die Intra-Tages-Prädiktion wird in den nachfolgenden Abschnitten näher erläutert.

### 5.2.1 Inter-Tages-Prädiktion

Ziel der Inter-Tages-Prädiktion ist es, zu bestimmen, welche historischen Tage aus der Fahrzeugbewegungsaufzeichnung dem zukünftigen zu pro-

gnostizierenden Tag ähneln. Hierfür sei $\mathscr{X} = \{X^{(1)}, \ldots, X^{(N)}\}$ die Menge aller aufgezeichneten Tage des Fahrzeugnutzungsprofils für $N \in \mathbb{N}$ und $\mathscr{P}$ die Menge aller POIs. Ein aufgezeichneter Tag $X^{(i)} = \left(X_1^{(i)}, \ldots, X_T^{(i)}\right) \in \mathscr{P}^T$ ist eine Folge aus $T \in \mathbb{N}$ POIs. $T$ gibt hierbei die Anzahl der Zeitslots eines Tages an, zum Beispiel $T = 24$ für eine stündliche oder $T = 1440$ für eine minütliche Auflösung. Bei der Inter-Tages-Prädiktion des kommenden Tages wird ein Cluster $\widehat{Cl}_{N+1} \subseteq \mathscr{X}$ für die Vorhersage bestimmt, das aus historischen Tagen besteht, die zum kommenden Tag ähnlich sind. Für die Vorhersage des $m$-ten zukünftigen Tages wird analog ein Cluster $\widehat{Cl}_{N+m} \subseteq \mathscr{X}$ ähnlicher Tage bestimmt.
Zur Vereinfachung wird nachfolgend nur die Vorhersage eines Tages betrachtet. Die Menge relevanter Tage $\widehat{Cl}_{N+1}$ wird anschließend für die nachgelagerte Intra-Tages-Prädiktion benötigt. Nachfolgend soll auf die beiden alternativen Varianten wochentagabhängig und wochentagunabhängig der Inter-Tages-Prädiktion eingegangen werden.

#### Wochentagabhängig

Der einfachste und in der Literatur häufig angewendete Ansatz ist es, Mobilitätsmuster tageweise aufgrund des Wochentags zu clustern [17, 35, 108]. Diese Art der Clusterbildung wird im Folgenden als *Weekday Cluster* (WC) bezeichnet. Die Idee dahinter ist, dass viele Menschen an gleichen Wochentagen ähnliche Mobilitätsmuster aufweisen. Somit wird bei dieser Variante implizit von einer wöchentlichen Regelmäßigkeit des Verhaltens des Fahrzeugführers ausgegangen. Dies ist jedoch eine verallgemeinerte Unterstellung und muss nicht für alle FNP beziehungsweise Tage korrekt sein. Als weitere Einschränkung dieses Verfahrens ist eine zusätzliche Unterscheidung in Arbeitstage und Wochenenden möglich [17, 35, 118, 131]. Die Reihenfolge der Tage und damit die relevanten Daten für die Vorhersage des nächsten Tages sind bei dieser Variante a priori bekannt, da sie durch die Reihenfolge der Wochentage gegeben sind: Zum Beispiel, dass auf einen

Donnerstag ein Freitag folgt. Soll ein Dienstag vorhergesagt werden, basieren die relevanten Tage $\widehat{Cl}_{N+1}$ für die Prognose auf allen historischen Dienstagen der erfassten Fahrzeugbewegungsaufzeichnung.
Bei dieser Art von Clustering ist jedoch eine schlechte Prognosegüte für Personen mit Regelmäßigkeiten außerhalb des Wochenzyklus zu beobachten. Goulet et al. [52] veranschaulichen mit Hilfe von Smartcard-Daten für den öffentlichen Verkehr in London, dass für einige Personen ein Teil des Mobilitätsverhaltens außerhalb der wöchentlichen Periodizität auftritt. Daher kann es in einigen Fällen sinnvoll sein, Tage nach den individuellen Bewegungsmerkmalen der Person statt nach dem Namen des Wochentags zu clustern. Diese Alternative hilft dabei, Faktoren wie tagesübergreifende oder atypische Mobilitätsmuster zu identifizieren und wird im nächsten Abschnitt vorgestellt.

#### Wochentagunabhängig

Bei dieser Variante der Inter-Tages-Prädiktion werden Tage nicht automatisch anhand des Wochentags einem Cluster zugeordnet, sondern aufgrund ihrer tatsächlichen Ähnlichkeit der Mobilitätsmuster. Diese werden als *Feature Cluster* (FC) bezeichnet [9, 112]. Die Clusterbildung der FC besteht dabei aus den drei Grundschritten *Bestimmung Ähnlichkeitsmaß*, *Auswahl Clusterverfahren* und *Bestimmung Clusteranzahl*, die nachfolgend erläutert werden.
*1. Bestimmung Ähnlichkeitsmaß:* Um ähnliche Objekte zu identifizieren wird eine Ähnlichkeitsfunktion benötigt. Ein Ähnlichkeitsmaß lässt sich allgemein wie folgt definieren [112].

**Definition 5.1.** *Es sei $I$ eine beliebige Menge, die Funktion $S : I \times I \to \mathbb{R}$ heißt Ähnlichkeitsmaß falls für alle $i, j \in I$ gilt:*

1. $S(i,j) = S(j,i)$,
2. $S(i,i) \geq S(i,j)$,

3. $S(i,j) \geq 0$

*Ein Ähnlichkeitsmaß heißt normalisiert, falls $S(i,i) = 1$ für alle $i \in I$ gilt.*

Die Wahl des Ähnlichkeitsmaßes muss in Abhängigkeit der vorliegenden Daten, insbesondere des verwendeten Skalenniveaus, erfolgen [134]. Für reellwertige Daten existieren eine Vielzahl metrischer Maße, wie beispielsweise die Euklidische, Manhattan oder Minkowski Norm [41].
In der hier vorliegenden Modellierung werden jedoch kategoriale Daten verwendet (konkret POIs). Boriah et al. [24] lieferten eine gute Übersicht möglicher kategorialer Ähnlichkeitsmaße. Zur Bestimmung ähnlicher Mobilitätsmuster zweier Tage wird daher das Ähnlichkeitsmaß *Goodall1* [24] adaptiert.
Dazu sei $\mathscr{P}_t \subseteq \mathscr{P}$ die Menge aller POIs, die jemals zum Zeitpunkt $t$ auftraten für $t = 1,\ldots,T$. Die Fahrzeugbewegungsaufzeichnung zweier Tage sei gegeben durch die Folgen $X = (X_1,\ldots,X_T)$ und $X' = (X'_1,\ldots,X'_T)$, wobei $X_t, X'_t \in \mathscr{P}_t$ den Standort zum $t$-ten Zeitpunkt für $t = 1,\ldots,T$ angebe. Das Goodall Ähnlichkeitsmaß $S_G$ ergibt sich dann durch

$$\mathrm{S_G}(X,X') = \frac{1}{T}\sum_{t=1}^{T} S(X_t,X'_t), \tag{5.1}$$

wobei

$$S(X_t,X'_t) = \begin{cases} 1 - \sum\limits_{q\in Q} p_t^2(q) & \text{falls } X_t = X'_t \\ 0 & \text{sonst} \end{cases}, \tag{5.2}$$

mit

$$Q = \{\mathrm{POI} \in \mathscr{P}_t \mid f_t(\mathrm{POI}) \leq f_t(X_t)\}$$

und

$$p_t^2(X_t) = \frac{f_t(X_t)(f_t(X_t)-1)}{N(N-1)}. \tag{5.3}$$

$N \in \mathbb{N}$ sei dabei wieder die Anzahl der aufgezeichneten Tage des Fahrzeugnutzungsprofils und $f_t(x)$ die Häufigkeit von POI $x \in \mathscr{P}_t$ zum Zeitpunkt $t$. Das Ähnlichkeitsmaß ist normalisiert, dass heißt es gilt

$$0 \leq \mathrm{S_G}(X, X') \leq 1. \tag{5.4}$$

Dieses Ähnlichkeitsmaß vergleicht Standorte aller Zeitpunkte zweier Tage. Selten besuchte Orte werden höher gewichtet, um den Einfluss häufig vorkommender POIs zu minimieren, wie zum Beispiel das Verweilen zu Hause.

**Beispiel 5.2.** *Es sei $T = 24$, also stündliche Auflösung eines Tages und $\mathscr{P} = \{A, B, C\}$. Betrachtet werden $N = 5$ Tage $X^{(1)}$, $X^{(2)}$, $X^{(3)}$, $X^{(4)}$ und $X^{(5)}$:*

$$\begin{aligned}
X^{(1)} &= (A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A)\\
X^{(2)} &= (A,A,A,A,A,A,A,A,B,B,B,B,B,B,B,B,B,A,A,A,A,A,A,A)\\
X^{(3)} &= (C,C,C,C,C,C,C,C,B,B,B,B,B,B,B,B,B,C,C,C,C,C,C,C)\\
X^{(4)} &= (A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A)\\
X^{(5)} &= (A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A)
\end{aligned}$$

*Die Goodall Ähnlichkeit der ersten drei Tage zueinander ist somit*

$$\mathrm{S_G}\left(X^{(1)}, X^{(2)}\right) = 0{,}25,\ \mathrm{S_G}\left(X^{(1)}, X^{(3)}\right) = 0,\ \mathrm{S_G}\left(X^{(2)}, X^{(3)}\right) = 0{,}3375.$$

*Durch das hohe Aufkommen von POI A wird sein Einfluss geringer gewichtet. Dadurch hat $X^{(1)}$ zu $X^{(2)}$ nach Goodall eine geringere Ähnlichkeit als $X^{(2)}$ zu $X^{(3)}$.* □

*2. Auswahl Clusterverfahren:* Die zwei wichtigsten Typen von Clusterverfahren sind *partitionierende* und *hierarchische Verfahren* [134]. *Partitionierende Verfahren* zerlegen den Suchraum in eine a priori bestimmte Anzahl von Clustern. Dabei wird oft der Abstand aller Objekte zu ihren jeweiligen Clustermittelpunkten minimiert. Damit Mittelpunkte bestimmt werden kön-

nen, ist aber ein räumlicher Zusammenhang der Objekte erforderlich. Zu den bekanntesten partitionierenden Verfahren gehört der *k-Means-Algorithmus* [41, 133].

*Hierarchische Verfahren* hingegen clustern Objekte aufgrund ihrer Ähnlichkeit zueinander. Unterschieden wird zwischen agglomerativen und divisiven Verfahren. Bei agglomerativen Verfahren ist jedes Objekt zu Beginn ein eigenes Cluster. Anschließend werden iterativ die jeweils ähnlichsten Cluster fusioniert. Die Ähnlichkeitsbestimmung von zwei ein-elementigen Clustern ist trivial. Für die Bestimmung der Ähnlichkeit von mehr-elementigen Clustern existieren verschiedene Vorgehensweisen, wie beispielsweise *Single-Linkage, Complete-Linkage, Average-Linkage* oder *Ward* [41]. Das Verfahren endet, wenn nur noch ein Cluster existiert. Divisive Verfahren funktionieren analog, beginnen jedoch mit einem großen Cluster, das alle Objekte enthält. Dieses Cluster wird sukzessive in kleinere Cluster zerlegt. Das Verfahren endet, wenn jedes Objekt ein eigenes Cluster darstellt. Die finale Anzahl der Cluster muss extrinsisch bestimmt werden. Dies kann graphisch mit Hilfe eines sogenannten *Dendrogrammes* erfolgen. Ein Dendrogramm zeigt, in welcher Reihenfolge und Ähnlichkeit die Cluster gebildet wurden. Eine weitere Möglichkeit ist die Bestimmung der Clusteranzahl durch eine quantitative Bewertung, wie dem Silhouettenkoeffizient. Da in der Modellierung eines Fahrzeugnutzungsprofils mit kategorialen Daten gearbeitet wird, bietet sich ein hierarchisches Clusterverfahren an.

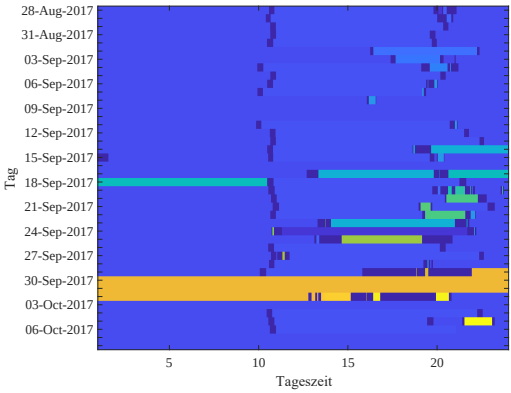
*3. Bestimmung Clusteranzahl:* Die Bestimmung der Clusteranzahl erfolgt unter dem Kompromiss der Ähnlichkeit innerhalb eines Clusters und der Gesamtanzahl erstellter Cluster. Dabei entsteht ein Zielkonflikt zwischen Handhabbarkeit (geringe Clusteranzahl) und Lösungsgüte (hohe Clusteranzahl). Wie bereits erwähnt, kann die Anzahl der Cluster eines hierarchischen Verfahrens grafisch durch das Dendrogramm bestimmt werden, was jedoch bei einer automatisierten Berechnung nicht praxistauglich anwendbar ist. Zusätzlich existieren Verfahren, die die Clusteranzahl quantitativ bewerten,

wie beispielsweise der Silhouettenkoeffizient [110]. Der Silhouettenkoeffizient eines einzelnen Objektes berechnet sich durch

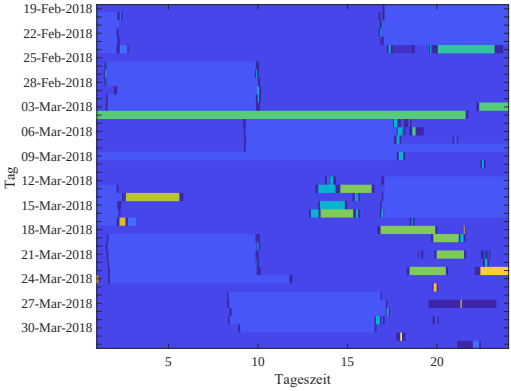
$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}, \tag{5.5}$$

wobei $a_i$ dem durchschnittlichen Abstand des $i$-ten Objekts zu den anderen Objekten im selben Cluster entspricht und $b_i$ der Abstand zum nächsten Cluster ist [11]. Für den Silhouettenkoeffizienten des gesamten Clusterverfahrens wird das arithmetische Mittel aller Objekte gebildet. Die Wahl der Clusteranzahl fällt auf das Clustering mit dem höchstem Silhouettenkoeffizienten. Beispiel 5.3 soll nachfolgend das gesamte Prinzip der Clusterbildung für zwei exemplarische Fahrzeugnutzungsprofile verdeutlichen.

**Beispiel 5.3.** *Das Vorgehen der Bildung von Feature Cluster soll an zwei echten Fahrzeugnutzungsprofilen A und B demonstriert werden. Abbildung 5.4 zeigt die Fahrzeugbewegungsaufzeichnungen beider Fahrzeugnutzungsprofile über sechs Wochen. Die Art der Darstellung der Fahrzeugbewegungsaufzeichnung ist dabei identisch zu Abbildung 5.2 und basiert auf dem in Tabelle 5.2 definierten Datenformat. Jede Zeile repräsentiert den Verlauf eines Tages, wobei die POIs mit unterschiedlichen Farben dargestellt sind. Für die vereinfachte Darstellung sind auch hier alle Trips in dunkelblau dargestellt.*
*FNP A (Abbildung 5.4(a)) zeigt einen typischen Verlauf eines Arbeitnehmers. Es lässt sich deutlich ein wöchentlicher Rhythmus erkennen. Die einzelnen Tage lassen sich außerdem in Arbeitstag, nicht-Arbeitstag und vereinzelte Ausreißer unterteilen. Abbildung 5.4(b) zeigt hingegen ein FNP, dessen Mobilitätsmuster keinem wöchentlichen Rhythmus folgt. Es lässt sich deutlich eine wöchentliche Verschiebung des größten Fahrtenblocks (hellblau) beobachten. FNP B scheint also ein Schichtarbeiter zu sein.*
*Für die eigentliche Bildung der Feature Cluster müssen zunächst die Goodall-Ähnlichkeiten (Gleichung 5.1) aller Tage zueinander berechnet werden.*

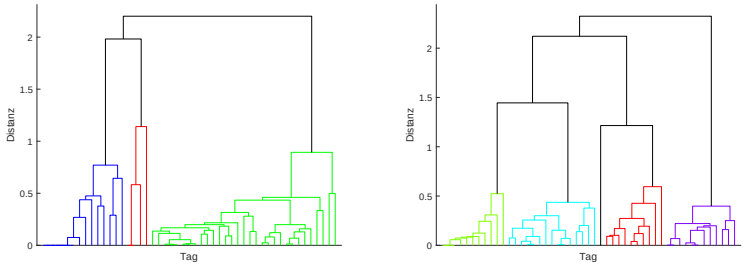



(a)




(b)

Abbildung 5.4: Fahrzeugbewegungsaufzeichnungen zweier realer Fahrzeugnutzungsprofile. Während FNP *A* einen einwöchigen Rhythmus aufweist (a), zeigt FNP *B* einen Rhythmus außerhalb des typischen Wochenzyklus (b). Jede Farbe entspricht dabei einer POI-ID. Zur vereinfachten Darstellung sind alle Trip-IDs in dunkelblau dargestellt (nach [57]).




(a) (b)

Abbildung 5.5: Dendrogramme der Fahrzeugnutzungsprofile *A* und *B*.

*Die Gruppierung ähnlicher Tage erfolgt daraufhin durch das hierarchische Clusterverfahren. Abbildung 5.5 zeigt die entstandenen Dendrogramme des Fahrzeugnutzungsprofils A (a) und Fahrzeugnutzungsprofils B (b).*
*Für die Bestimmung der optimalen Anzahl der Cluster wird anschließend der Silhouettenkoeffizient wie in Gleichung 5.5 berechnet. Um die optimale Anzahl der Cluster zu bestimmen, wird die Anzahl der zu bildenden Cluster in mehreren Durchgängen schrittweise erhöht und der durchschnittliche Silhouettenkoeffizient berechnet. Die optimale Anzahl wird dann durch den höchsten durchschnittlichen Silhouettenkoeffizienten bestimmt. Abbildung 5.6 zeigt den Silhouettenkoeffizient für beide Profile für 1 bis 25 Cluster. Daraus ist zu erkennen, dass für FNP A drei und für FNP B fünf Cluster als ideal bestimmt wurden.*
*Abbildung 5.7 zeigt das Ergebnis des Feature Clusterings der Daten aus Abbildung 5.4. Zum direkten Vergleich sind auch die Weekday Cluster (Clusterbildung nach Wochentag) der beiden Fahrzeugnutzungsprofile in Abbildung 5.7 enthalten. Für FNP A (Abbildung 5.7(a)) besitzen die WC eine große Ähnlichkeit zwischen den Clustern 1 bis 5. Diese Cluster sind typischerweise Arbeitstage (Montag bis Freitag). Beim FC wählt das Verfahren drei Cluster. In Cluster 1 befinden sich hauptsächlich Tage, die sich beim WC*

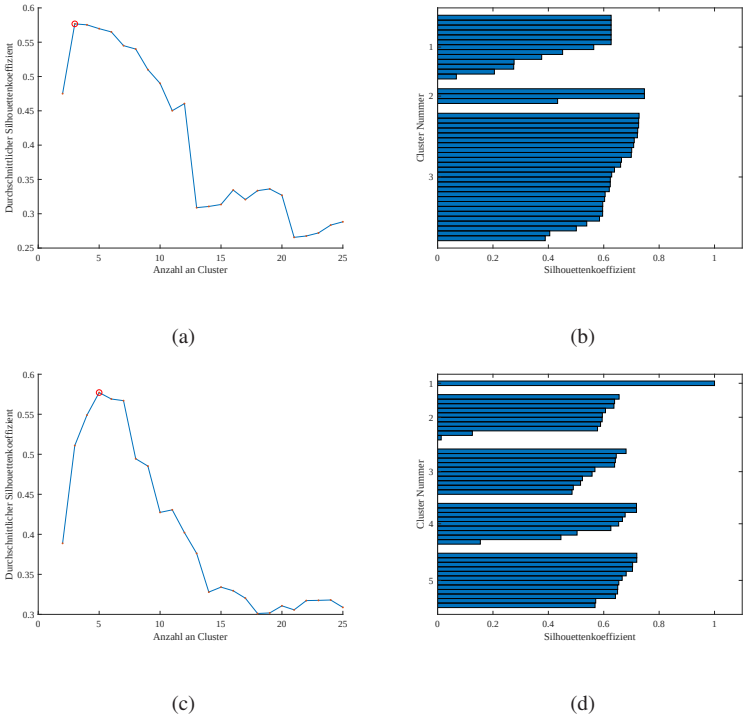
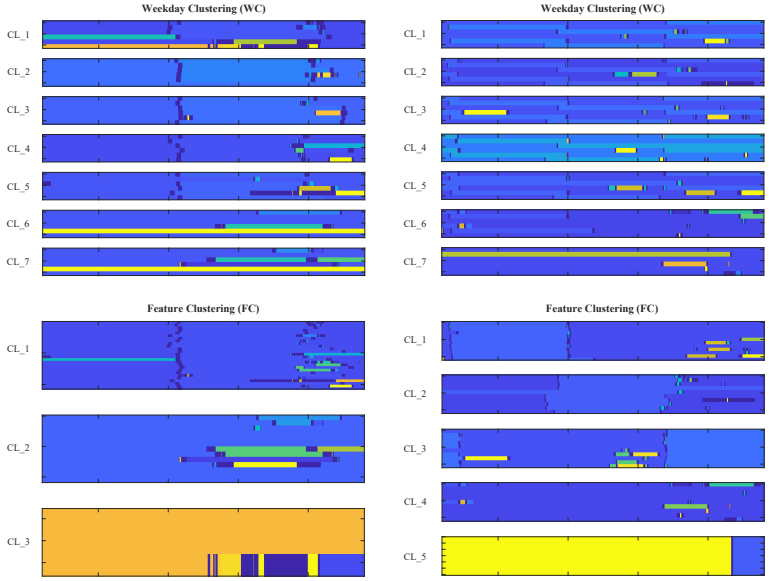



Abbildung 5.6: Silhouettenkoeffizienten beider Fahrzeugnutzungsprofile *A* und *B*. Es ergibt sich eine optimale Anzahl an Feature Cluster von drei für FNP *A* und fünf für FNP *B*.

*in den Clustern 1 bis 5 befinden. Im zweiten Cluster sind im Wesentlichen die Tage aus den Wochenendclustern 6 und 7 enthalten. Das dritte Cluster fasst Tage zusammen, die einzelne Ausreißer darstellen. Es zeigt sich insgesamt, dass für FNP A beide Varianten (WC und FC) Cluster mit homogenen Objekten liefern.*

*Für FNP B (Abbildung 5.7(b)) zeigt sich hingegen ein anderes Bild. Beim WC sind die einzelnen Tage in einem Cluster teilweise sehr unterschiedlich. Es ist insbesondere die Verschiebung des täglichen Nutzungsverhalten er-*



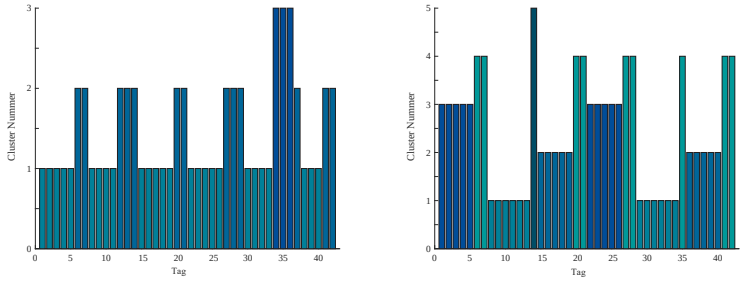

(a) (b)

Abbildung 5.7: Darstellung Clustering-Methoden (WC und FC) zweier realer Fahrzeugnutzungsprofile. (a) FNP *A* stellt ein typisches wöchentliches Mobilitätsverhalten dar, (b) während FNP *B* ein atypisches wöchentliches Mobilitätsverhalten aufweist. Bei (a) bilden sowohl WC als auch FC annehmbare Cluster, wohingegen bei (b) nur FC sinnvolle Cluster bildet (nach [57]).

*kennbar. Das FC liefert deutlich homogenere Gruppen. In den Clustern 1 bis 3 sind die jeweils verschobenen Tagesprofile korrekt zusammengefasst. Cluster 5 besteht aus einem einzelnen Tag, der sich zu sehr von allen anderen unterscheidet und somit als Ausreißer bewertet wird. Dies verdeutlicht nochmals, dass eine Clusterbildung auf Basis der Wochentags (Weekday Cluster) nicht zwangsläufig zu einem guten Resultat führt.* □

Für beide Fahrzeugnutzungsprofile aus Beispiel 5.3 lässt sich nach der Clusterbildung eine Folge erstellen, die angibt, an welchen Tagen welches Clus-

ter vorliegt. Für WC ist dies trivial und ergibt eine alternierende Folge von 1 bis 7. Die Clusterfolge beider Fahrzeugnutzungsprofile der FC ist in Abbildung 5.8 dargestellt. Für FNP *A* sind drei Cluster vorhanden, die eine wöchentliche Regelmäßigkeit zeigen. Die Clusterfolge von FNP *B* zeigt den Rhythmus außerhalb einer Woche. Dennoch ist auch für *B* ein Muster zu erkennen.




(a) (b)

Abbildung 5.8: (a) zeigt die Clusterfolge von FNP *A*, während (b) die Clusterfolge von FNP *B* darstellt. In beiden Fällen wird ein Zeitraum von sechs Wochen gezeigt. Die Cluster wurden durch Feature Clustering gebildet, wie in Abbildung 5.7 dargestellt. Typische wöchentliche Regelmäßigkeiten sind in (a) zu erkennen, während in (b) ein dreiwöchiger Rhythmus zu erkennen ist. Jedes Cluster wird zusätzlich mit einer individuellen Farbe hervorgehoben (nach [57]).

Das Ergebnis des Feature Clusterings ist somit eine Folge von Clustern (vgl. Abbildung 5.8), die im nächsten Schritt auf wiederkehrende Muster hin untersucht und prognostiziert wird. Die Idee ist es, den nächsten Tag basierend auf den Feature Clustern der Vortage vorherzusagen. Die Vorhersage soll dabei bestimmen, welches Feature Cluster im Folgetag vorkommt. Abbildung 5.9 illustriert den Ansatz. In diesem Beispiel werden die Feature Cluster der zwei Vortage $Cl_{N-1}$ und $Cl_N$ und die Information, um welchen Wochentag es sich handelt, als Input verwendet, um das Feature Cluster $\widehat{Cl}_{N+1}$ vorherzusagen.

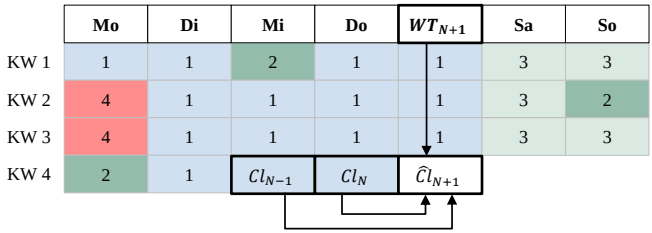



Abbildung 5.9: Diese Darstellung veranschaulicht beispielhaft die Idee der Feature Cluster Vorhersage (wochentagunabhängig). Es zeigt die Reihenfolge der Feature Cluster aller erfassten Tage eines Fahrzeugnutzungsprofils. Die Zahl und Farbe repräsentieren das berechnete Feature Cluster für den jeweiligen Tag. In diesem Beispiel fließen zwei Vortage und der Wochentag in die Prognose ein. Eine sinnvolle Vorhersage wäre in diesem Beispiel $\widehat{Cl}_{N+1} = 1$ (nach [57, 72]).

Zur Umsetzung des in Abbildung 5.9 vorgestellten Ansatzes wird ein *Künstliches Neuronales Netz* (KNN) verwendet. KNNs sind von der Verarbeitungsweise des Gehirns inspiriert und sind auf eine große Zahl von Problemstellungen wie Klassifikation, Mustererkennung und Prognose von Zeitreihen anwendbar [38, 73]. Ein KNN leitet dabei den Input durch eine Reihe von Neuronen in der versteckten Schicht auf den Output. Durch verschiedene arithmetische Operationen in den Neuronen können nichtlineare Zusammenhänge zwischen Input und Output gefunden werden. Ein KNN lernt dabei mit Hilfe von Daten, bei denen der Output zum jeweiligen Input bekannt ist. Beim Trainingsvorgang werden Gewichte im Inneren des KNN solange angepasst, bis der Input auf den gewünschten Output abgebildet wird. Ein Vorteil des KNN ist, dass externe Informationen einfach durch eigene Inputneuronen in die Prognose einfließen können. Um die Vorhersagegenauigkeit insbesondere bezüglich atypischem Mobilitätsverhaltens zusätzlich zu verbessern, sollen daher externe Informationen wie bundeslandspezifische Feiertage in die Prognose mit einfließen. Abbildung 5.10 zeigt den Aufbau eines auf die Problemstellung zugeschnittenen KNN mit einer versteckten Schicht (Hidden-Layer).

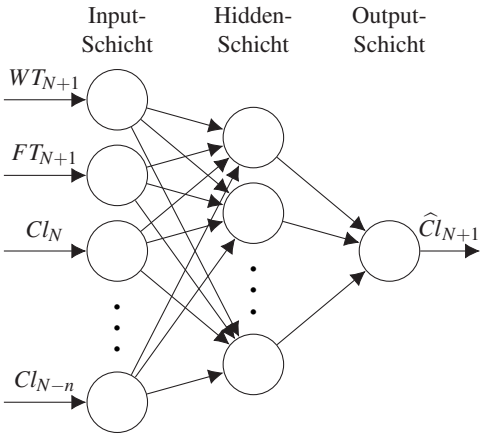



Abbildung 5.10: Struktur des KNN für die Vorhersage des nächsten Feature Clusters $\widehat{Cl}_{N+1}$. Ein Neuron entspricht dem Wochentag des gesuchten Tages ($WT_{N+1}$). Ein zweites Neuron zeigt, ob der gesuchte Wochentag ein Feiertag ist ($FT_{N+1}$). Die restlichen Neuronen entsprechen den Feature Clustern der möglichen einzubeziehenden Vortage ($Cl_N, \ldots, Cl_{N-n}$). Die Ausgabeschicht gibt das gesuchte Feature Cluster des folgenden Tages zurück (nach [57, 72]).

Ausgangspunkt der hier vorgestellten Prognose ist eine Folge von Feature Clustern wie in Abbildung 5.8. Durch das KNN sollen die Feature Cluster der nächsten $m \in \mathbb{N}$ Tage auf Basis der Feature Cluster der letzten $n \in \mathbb{N}$ Tage und weiteren externen Informationen vorhergesagt werden. Da es sich dabei um kategoriale Daten handelt, müssen diese auf besondere Weise codiert werden. Dies erfolgt mit einer *One-Hot-Kodierung* [18], bei der für jeden Inputtag mehrere Inputneuronen verwendet werden. Die Anzahl hängt dabei von der Anzahl möglicher Cluster ab. Alle bis auf eines dieser Neuronen eines Inputtages sind null. Das Neuron, welches das Cluster repräsentiert, ist hingegen eins.

Die Kodierung des Outputs erfolgt analog, wobei zusätzlich die *Softmax-Funktion* [22] angewendet wird. Dadurch liegen alle Outputwerte zwischen null und eins und können als Wahrscheinlichkeit interpretiert werden. Das

Feature Cluster mit der höchsten Wahrscheinlichkeit wird schließlich für die Mobilitätsvorhersage des nächsten Tages herangezogen.
Die kodierten Daten können nun zum Trainieren des KNN verwendet werden, um eine funktionale Beziehung zwischen den Input- und Output-Clustern zu finden. Die Struktur ist in Abbildung 5.10 dargestellt. Um die Muster im jeweiligen Profil zu erkennen, werden mehrere KNNs mit einer unterschiedlichen Anzahl von Input-Clustern $Cl_{N-n}$ für jedes FNP trainiert ($n = 1, \ldots, 6$). Es wird die Konfiguration mit der besten Vorhersagegenauigkeit ausgewählt. Das trainierte KNN wird dann verwendet, um eine Prognose des nächsten Feature Clusters für den kommenden Tag zu erhalten. Alle Tage des Fahrzeugnutzungsprofils, die diesem vorhergesagten Feature Cluster ($\widehat{Cl}_{N+1}$) zugeordnet sind, liefern die Basis für die nachfolgende Intra-Tages-Prädiktion (siehe Abbildung 5.3). Das hier beschriebene Prinzip wird abschließend in Beispiel 5.4 verdeutlicht. Es knüpft direkt an Beispiel 5.3 an.

**Beispiel 5.4.** *In diesem Beispiel wird für das FNP B aus Beispiel 5.3 eine Mobilitätsvorhersage mit Hilfe der Feature Cluster aus Abbildung 5.7(b) und der Clusterfolge aus Abbildung 5.8(b) erstellt. Ziel dieses Beispiels ist es, den 43. Tag ($N = 42$) der Clusterfolge aus Abbildung 5.8(b) mit dem KNN vorherzusagen. Dazu wird das KNN mit der Clusterfolge der vorherigen sechs Wochen aus Abbildung 5.8(b) trainiert. Die dazugehörige Konfiguration des KNN ist in Anhang B.1 zu finden. Mit Hilfe des trainierten KNN kann nun das nächste Feature Cluster $\widehat{Cl}_{N+1}$, welches dem 43. Tag der Clusterfolge entspricht, bestimmt werden. Die dazugehörigen Eingangsparameter sind in Tabelle 5.3 dargestellt.*
*In diesem Beispiel werden die letzten drei Tage (Feature Cluster) aus Abbildung 5.8(b) als Input verwendet ($Cl_{40}$ - $Cl_{42}$). Bei dem zu prognostizierenden Tag handelt es sich um einen Montag ($WT_{43} = Montag$) und keinen Feiertag ($FT_{43} = Nein$). Als Ergebnis wird Feature Cluster 3 mit 97,3 % als wahrscheinlichstes Cluster bestimmt, wodurch $\widehat{Cl}_{43} = 3$. Somit wird die*

Tabelle 5.3: Exemplarische Input Daten (Input-Schicht) für das trainierte KNN zur Prognose des 43. Tages ($\widehat{Cl}_{N+1}$) für FNP *B*. Aus der Output-Schicht lässt sich das wahrscheinlichste Cluster $\widehat{Cl}_{43} = 3$ ableiten.

| Input-Schicht | | Output-Schicht | |
|---|---|---|---|
| $WT_{43}$ | *Montag* | 1 | 1,2 % |
| $FT_{43}$ | *Nein* | 2 | 1,5 % |
| $Cl_{42}$ | 4 | 3 | 97,3 % |
| $Cl_{41}$ | 4 | 4 | 0 % |
| $Cl_{40}$ | 2 | 5 | 0 % |

*Clusterfolge aus Abbildung 5.8(b) nach dem bisherigen Muster korrekt fortgesetzt. Möchte man auch den 44. Tag vorhersagen, würde wiederum der hier vorhergesagte 43. Tag als Input für das trainierte KNN dienen. Das hier prognostizierte Feature Cluster ($\widehat{Cl}_{43} = 3$) beinhaltet nun alle relevanten Tage für die nachgelagerte Intra-Tages-Prädiktion.* □

### 5.2.2 Intra-Tages-Prädiktion

Aufbauend auf das durch die Inter-Tages-Prädiktion prognostizierte Cluster, bestimmt die Intra-Tages-Prädiktion nun den genauen Tagesverlauf (Fahrten und Aufenthaltsorte) des zu prognostizierenden Tages. Das Cluster $\widehat{Cl}$ kann dabei auf Grundlage der Weekday oder Feature Cluster prognostiziert worden sein. Bei WC basiert die Intra-Tages-Prädiktion auf allen historischen Tagen des gleichen Wochentags. Ein Dienstag wird also durch alle vergangenen Dienstage des Fahrzeugnutzungsprofils vorhergesagt. Wenn die Intra-Tages-Prädiktion jedoch auf FC basiert, verwendet sie alle Tage aus dem vom KNN vorhergesagten Feature Cluster (siehe Abbildung 5.10). In beiden Fällen ist das Ergebnis somit eine Menge an *relevanten Tagen* aus dem FNP (vgl. Abbildung 5.3).

Eine verhältnismäßig einfache Umsetzung der Intra-Tages-Prädiktion ist die Anwendung des *Modus* auf die Menge der relevanten Tage [52]. Diese mathematische Operation wählt zu jedem Zeitslot den häufigsten POI, der in der Menge der relevanten Tage in diesen Zeitslot vorkommt. Die relative Häufigkeit des häufigsten POI in jedem Zeitslot kann darüber hinaus als Gütemaß der Vorhersage herangezogen werden. Dieses Verfahren, das eine Variante der Kompressionsmethode *Prediction-by-Partial-Match* nullter Ordnung ist, wurde in einer ähnlichen Form schon in [26] zur Mobilitätsvorhersage verwendet. Das Format der zu vergleichenden relevanten Tage entspricht dabei einer Folge von POIs, wobei ein Tag $X = (X_1, \ldots, X_T)$ entspricht (vgl. Abschnitt 5.2.1). Nachteil dieser Methode ist, dass durch dieses Format der Daten und den Modus selbst keine Standortwechsel (Fahrten) zwischen den POIs berücksichtigt werden. Um eine vollständige Mobilitätsvorhersage (vgl. Tabelle 5.2) inklusive Fahrten zu erhalten, müssen diese somit nach Anwendung des Modus hinzugefügt werden. Das Vorgehen hierfür ist in Algorithmus 2 beschrieben.

**Algorithm 2** Intra-Tages-Prädiktion

**Input:** Relevante Tage $\widehat{Cl} = \{X_t^{(1)}, \ldots, X_t^{(k)}\}$ aus Inter-Tages-Prädiktion
**Output:** POI- und Trip-ID Sequenz

1: **for each** $t = 1, \ldots, T$ **do**
2: $\quad \widehat{POI}_t = modus\left(\{X_t^{(1)}, \ldots, X_t^{(k)}\}\right)$
3: **end for**
4: **for each** Standortwechsel **do**
5: $\quad$ Suche häufigste Trip-ID für Standortwechsel
6: $\quad$ Füge Dauer des Trips der Trip-ID mittig ein
7: **end for**
8: **return** POI- und Trip-ID Sequenz

Nach Anwendung des Algorithmus 2 ist das vorliegende Ergebnis der Mobilitätsvorhersage identisch zu dem in Tabelle 5.2 vorgestellten Datenformat.

Es handelt sich somit um eine vollständige zeitliche und örtliche Mobilitätsvorhersage im Format von POIs und Trips.
Ergänzend ist zu erwähnen, dass es sich bei der hier vorgestellten Intra-Tages-Prädiktion (Modus) um ein vom aktuellen Standort *unabhängiges* Verfahren handelt. Das bedeutet, dass der aktuelle POI keinen Einfluss auf die Vorhersage des nächsten POI hat. Ein vom Standort *abhängiges* Verfahren hingegen ist beispielsweise ein Markov-Modell. Ein konkretes Konzept hierzu wird in den Abhandlungen [56] und [93] vorgestellt. Es beruht auf einer Kombination aus Markov-Modell und Kerndichteschätzung. Dabei wird das Markov-Modell zur Vorhersage des nächsten POI verwendet, während die Kerndichteschätzung den Abfahrtszeitpunkt ermittelt. Grundgedanke ist es, mit Hilfe eines Markov-Modells die zeitliche und örtliche Mobilitätsvorhersage zu ermöglichen, ohne gleichzeitig die Nachteile eines Time-Based Markov (vgl. Abschnitt 3.4) zu erhalten. Dieses Konzept entspricht einer alternativen Variante der hier vorgestellten Intra-Tages-Prädiktion, die im Gegensatz zum Modus vom aktuellen Standort abhängig ist. Da jedoch die Gesamtergebnisse der Prognosegüte in [56, 93] schlechter ausfallen als der hier angewendete Operator Modus, wird das Konzept nicht näher betrachtet. Detaillierte Informationen sind unter [56, 93] zu finden.

## 5.3 Evaluation: Örtliche- und zeitliche Mobilitätsvorhersage

Das in Abschnitt 5.2 vorgestellte Prognoseframework soll anhand von zwei erhobenen und aufbereiteten Datensätzen evaluiert werden. Der erste Datensatz *DS*1 umfasst 30 Fahrzeugnutzungsprofile. Die Fahrzeugbewegungsaufzeichnungen wurden über einen GPS-fähigen OBD-Adapter erhoben und drahtlos an ein Backend-System übertragen. Die Übertragung startet beim Betätigen der Zündung und sendet im circa minütlichen Takt die GPS-Position des Fahrzeugs. Dies ergibt eine Folge von GPS-Koordinaten mit Zeitstempel, wie in Tabelle 5.1 dargestellt. Bei Deaktivierung der Zündung wer-

den keine weiteren Koordinaten gesendet. Folglich werden nur die Fahrten der Fahrzeuge aufgezeichnet. Die Aufbereitung der rohen GPS-Koordinaten zu POI-IDs und Trip-IDs erfolgt dabei automatisiert im Backend wie in Abschnitt 5.1.2 beschrieben.
Zum direkten Vergleich wurde ein zweiter Datensatz *DS*2 bestehend aus GPS-Koordinaten von 96 individuellen Smartphone-Benutzern erhoben. Jeder einzelne Smartphone-Benutzer des Datensatzes *DS*2 besteht dabei in seiner Rohform aus einem diskreten Strom an GPS-Koordinaten mit Zeitstempel. Jedoch wurden die Daten ohne Angabe des verwendeten Verkehrsmittels und ohne zeitliche Konsistenz ermittelt. Um *DS*2 in ein zu *DS*1 vergleichbares Format zu überführen, sind zusätzliche Verarbeitungsschritte des *DS*2 Datensatzes erforderlich. Zu diesem Zweck wird der Datensatz zunächst neu abgetastet, um Profile mit einer GPS-Koordinate pro Minute zu erstellen. Anschließend werden die GPS-Aufzeichnungen in *Bewegungsdaten* und *Nicht-Bewegungsdaten* unterteilt. Auch hierfür werden, wie bereits in Abschnitt 5.1.2 beschrieben, der Radius und die Standzeit herangezogen. Falls die Daten eine Durchschnittsgeschwindigkeit von $\leq$10 km/h für mindestens 180 Sekunden aufweisen, wird daraus ein POI gebildet. Liegen weitere Koordinaten innerhalb eines 500 Meter-Radius, werden sie ebenfalls diesem POI zugeordnet. Dadurch werden sämtliche POI-IDs und Trip-IDs abgeleitet. Die Unterscheidung von Fahrt- und Bewegungsdaten ist dabei anfällig für falsche Klassifizierungen und die Identifizierung des jeweiligen Verkehrsmittels nicht trivial [62, 132, 137]. Anzumerken ist, dass *DS*2 ausschließlich als Referenzdatensatz dient, um das Potential der Prognose von fahrzeugspezifischen Mobilitätsprofilen (*DS*1) besser einordnen zu können. Er wird nicht für die nachgelagerte Einplanung von OTA-Updates genutzt.
Im nächsten Abschnitt werden zunächst beide Datensätze deskriptiv analysiert, bevor im Anschluss das hier vorgestellte Prognoseframework angewendet und evaluiert wird. Für die Bewertung der Prognose des Mobilitätsverhaltens werden dazu drei verschiedene Gütekriterien vorgestellt. Insbe-

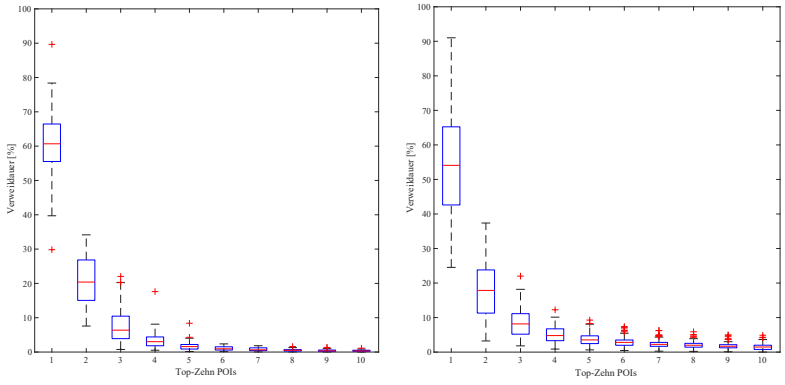
sondere sollen die Unterschiede beider Datensätze und die Implikationen auf die Prognose verdeutlicht werden.

### 5.3.1 Deskriptive Datensatzanalyse

In diesem Abschnitt werden die Datensätze *DS*1 und *DS*2 deskriptiv analysiert, um die jeweiligen Charakteristiken zu identifizieren.

#### Verweildauer an Standorten

Fahrzeugdaten (*DS*1) sind im Vergleich zu Smartphone-Daten (*DS*2) weniger komplex, da sie einen *unimodalen* Bewegungsablauf aufweisen, sich also nur auf ein Transportmittel beziehen. Die Daten zeigen weniger Standorte (POIs), die für den Benutzer als relevant erachtet werden und die Trajektorien sind im Allgemeinen auf den für Fahrzeuge zugänglichen Raum (Straßenverkehrsnetz) beschränkt. Smartphone-Daten sind im Vergleich zu Fahrzeugdaten komplexer, da sie aus *multimodalen* Mobilitätsmustern bestehen und somit eine größeren Freiheitsgrad bezüglich der Bewegungsmöglichkeiten abbilden.
Miteinander verglichen zeigen beide Datensätze unterschiedliche Mobilitätsverhalten. Abbildung 5.11 zeigt die durchschnittliche Verweildauer an den zehn meistbesuchten Standorten für Fahrzeuge (*DS*1) und Smartphone-Benutzer (*DS*2). Abbildung 5.11(a) zeigt, dass Fahrzeuge durchschnittlich zu 61 % an ihrem meistbesuchten Standort geparkt werden. Smartphone-Benutzer hingegen verweilen durchschnittlich 54 % der Zeit an ihrem meistbesuchten POI. Außerdem ist die Streuung innerhalb der Stichprobe größer. Bei Smartphone-Daten werden mehr als 80 % der gesamten aufgezeichneten Zeit an einem der drei meistbesuchten Standorte verbracht. Die aufgezeichneten Fahrzeugdaten zeigen eine um 9 % höhere Wahrscheinlichkeit (89 %), dass ein Fahrzeug an einem der drei meistbesuchten POIs zu finden ist.




(a) (b)

Abbildung 5.11: (a) Verweilzeit in Prozent an den Top-Zehn POIs für *DS*1. Etwa 89 % der gesamten Verweildauer wird an einem der drei meistbesuchten Standorte verbracht. (b) Verweilzeit in Prozent an den Top-Zehn Standorten für *DS*2. Etwa 80 % der gesamten Verweildauer wird an einem der drei meistbesuchten Standorte verbracht (nach [57]).

### Ermittlung der Profilregularität und maximaler Prognostizierbarkeit

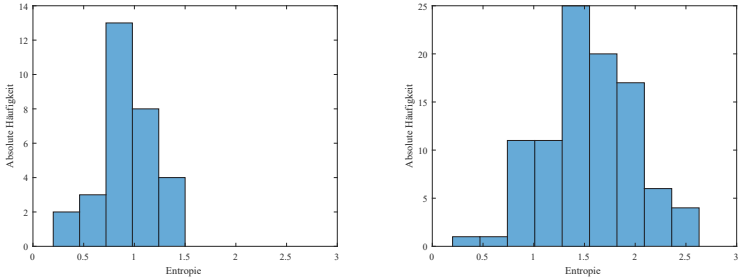
Die maximal erreichbare Vorhersagegenauigkeit des zukünftigen Mobilitätsverhaltens wird durch die Zufälligkeit der Mobilitätsmuster innerhalb der aufgezeichneten Fahrzeugnutzungsprofile begrenzt. Die Messung der Entropie pro FNP ist eine Möglichkeit, diese Zufälligkeit zu quantifizieren und gibt einen Hinweis auf die maximal erreichbare Vorhersagegenauigkeit [52]. Eine niedrige Entropie deutet dabei auf das Vorhandensein gleicher Muster und somit einer guten Prognostizierbarkeit hin. Im Umkehrschluss bedeutet eine hohe Entropie einen Mangel ähnlicher Muster, was wiederum mit einer schlechteren Prognostizierbarkeit einhergeht.

Die Literatur schlägt verschiedene Ansätze zur Schätzung der Entropie in Mobilitätsdaten vor. Song et al. [123] nutzten die Entropie als eine der ersten, um die Regelmäßigkeit in Mobilitätsprofilen zu bestimmen. Sie entdeckten, dass die Entropie als Obergrenze für die Vorhersagbarkeit des

menschlichen Mobilitätsverhaltens verwendet werden kann. Ihr Ansatz ist jedoch nicht unabhängig von der vorliegenden örtlichen und zeitlichen Auflösung der aufgezeichneten Daten. Wie Burbey in [27] schreibt, hat die Form der Darstellung von Daten einen Einfluss auf das Vorhersageergebnis. Dies geht mit den Ergebnissen von [52] und [61] einher und zeigt, dass die örtliche und zeitliche Auflösung von Daten die maximale Vorhersagbarkeit von Mobilitätsdaten beeinflusst. Ikanovic et al. veranschaulichen in [61] explizit, dass die maximal erreichbare Vorhersagbarkeit mit einer geringeren örtlichen (Clusterradius der POIs) und zeitlichen ($T$) Auflösung derselben Daten zunimmt.
Deshalb wird hier eine zu [123] alternative, zeitlich und örtlich unabhängige Formel verwendet, um die Entropie wie in [105] zu berechnen. Die Entropie $E$ berechnet sich durch

$$E = -\frac{\sum_{t=1}^{T} \sum_{i \in \mathscr{P}_t} \frac{l_{it}}{N} \cdot \log\left(\frac{l_{it}}{N}\right)}{T}, \tag{5.6}$$

wobei $T \in \mathbb{N}$ wie zuvor die Anzahl der Zeitslots innerhalb eines Tages, $N \in \mathbb{N}$ die Gesamtanzahl an Tagen des beobachteten FNP, $\mathscr{P}_t$ die Menge aller jemals in Zeitslot $t$ beobachteten POIs und $l_{it}$ die Anzahl ist, wie oft POI $i \in \mathscr{P}_t$ Zeitslot $t$ für $t = 1, \ldots, T$ dominiert hat.
Abbildung 5.12 veranschaulicht die Entropieverteilung für beide Datensätze in einem Histogramm. Es zeigt, dass die durchschnittliche Entropie von $DS1$ im Vergleich zu $DS2$ niedriger ist. Dies ist unter anderem auf den höheren örtlichen Freiheitsgrad der Smartphone-Daten ($DS2$) als der Fahrzeugdaten ($DS1$) zurückzuführen. Die höhere Entropie wird auch durch die höhere Anzahl relevanter POIs für $DS2$ begünstigt, wie in Abbildung 5.11 dargestellt.




(a) (b)

Abbildung 5.12: Histogramm der Entropie für die beiden Datensätze: (a) *DS*1, (b) *DS*2 (nach [57]).

### 5.3.2 Ermittelte Prognosegüte

Grundsätzlich ist sowohl eine angemessene Bestimmung der Prognosegüte als auch ein direkter Vergleich verschiedener Modelle zur Mobilitätsvorhersage zu anderen Forschungsprojekten schwierig. Dies liegt unter anderen an den vielfältigen Arten und Anwendungsgebieten der Mobilitätsvorhersage (vgl. Abschnitt 3.4), als auch an der Verwendung und Handhabung unterschiedlichster Datensätze. Des Weiteren existieren viele verschiedene Methoden und Metriken, um die Genauigkeit eines Vorhersagemodells zu bewerten [17, 26, 136]. Aus diesem Grund werden eigens drei Gütekriterien definiert, um die Prognosegüte des in Abschnitt 5.2 vorgestellten Prognoseframeworks zu bewerten. Zusätzlich soll die in Kapitel 5.3.1 beschriebene Entropieberechnung dabei helfen, die erzielten Ergebnisse mit der maximal möglichen Vorhersagegenauigkeit zu vergleichen. Diese Handhabung ist auch in anderen Forschungsprojekten üblich [17, 61, 86, 105, 123].
Basierend auf der in Abschnitt 3.4 gegebenen Definition der mittelfristigen Mobilitätsvorhersage (gekennzeichnet durch einen Prognosehorizont von bis zu einer Woche), wird der Trainingsdatensatz so bestimmt, dass ein Verhältnis von Trainings- und Testdaten von 6:1 entsteht. Dies ist mit ande-

ren Forschungsprojekten zur Mobilitätsvorhersage vergleichbar [40, 137]. Dazu werden die verfügbaren Fahrzeugnutzungsprofile auf eine betrachtete Aufzeichnungsdauer von sieben Wochen beschränkt. Die siebte und somit letzte Woche jedes Fahrzeugnutzungsprofils wird gesondert betrachtet. Sie dient ausschließlich zur Validierung und wird nicht verwendet, um das Prognoseframework zu trainieren. Somit ergibt sich ein Trainingsdatensatz von sechs Wochen und ein Testdatensatz von einer Woche pro FNP, was einem *Out-of-Sample* Test entspricht (Verhältnis 6:1).

Für das erste Gütekriterium ('1 Tag') wird der erste Tag nach den sechswöchigen Trainingsdaten prognostiziert. Dieser prognostizierte Tag wird mit dem ersten Tag der nicht zum Training herangezogenen siebten Woche (Testdatensatz) verglichen. Dazu wird zu jedem diskreten Zeitslot die Übereinstimmung der POI-IDs, beziehungsweise Trip-IDs, beider Tage geprüft, was 1440 Zeitslots bei einer minütlichen Auflösung entspricht. Stimmen sie im jeweiligen Zeitslot überein, gilt dies als korrekte Vorhersage, ansonsten nicht. Über alle Zeitslots eines Tages ergibt sich somit die erzielte Prognosegüte in Prozent.

Die zweite Genauigkeitsmetrik ('1 Woche') verfährt vom Prinzip identisch, abgesehen davon, dass sich der zu prognostizierende und vergleichende Zeitraum erhöht. Statt einem Tag werden insgesamt sieben Tage prognostiziert. Ein Vergleich findet dabei entsprechend mit der gesamten siebten Validierungswoche (Testdatensatz) statt.

Bei der dritten Genauigkeitsmetrik ('7 Wochen') sollen ebenfalls sieben Tage, wie bei dem zweiten Gütekriterium, prognostiziert werden. Allerdings wird diese prognostizierte Woche nicht nur mit dem Testdatensatz (siebte Woche), sondern auch mit dem eigentlichen Trainingsdatensatz (sechs Wochen) wochenweise verglichen. Der Anteil der korrekten Überlappung von POI-IDs und Trip-IDs trägt auch hier zur Leistungsbeurteilung bei und liefert ein Indiz, wie gut die Mobilitätsmuster des Trainingsdatensatzes erkannt werden. Dieser Ansatz entspricht einem *In-Sample* Test.

Die Tabellen 5.4 und 5.5 zeigen das erzielte Ergebnis der Mobilitätsvorhersagen für die Datensätze *DS*1 und *DS*2. In den Spalten sind die drei zuvor beschriebenen Gütekriterien aufgeführt, während die Zeilen zwischen den *Wochentagabhängigen* (WTA) und *Wochentagunabhängigen* (WTU) Varianten der Inter-Tages-Prädiktion aus Kapitel 5.2 unterscheiden. In allen Fällen ist sowohl der Mittelwert als auch Median der erzielten Prognosegüte aufgeführt.

Tabelle 5.4: Erzielte Prognosegüte für *DS*1 (nach [57]).

| Verfahren | | '1 Tag' | '1 Woche' | '7 Wochen' |
|---|---|---|---|---|
| WTA | ∅ | 76 % | 68 % | 71 % |
| | Median | 81 % | 72 % | 72 % |
| WTU | ∅ | 77 % | 66 % | 66 % |
| | Median | 86 % | 70 % | 68 % |
| Auto-Select | ∅ | 83 % | 70 % | 69 % |
| | Median | 86 % | 75 % | 70 % |

Tabelle 5.5: Erzielte Prognosegüte für *DS*2 (nach [57]).

| Verfahren | | '1 Tag' | '1 Woche' | '7 Wochen' |
|---|---|---|---|---|
| WTA | ∅ | 53 % | 55 % | 57 % |
| | Median | 57 % | 56 % | 57 % |
| WTU | ∅ | 49 % | 49 % | 49 % |
| | Median | 52 % | 49 % | 48 % |
| Auto-Select | ∅ | 58 % | 56 % | 55 % |
| | Median | 61 % | 58 % | 57 % |

Da die Mobilitätsvorhersage im ersten Schritt der Inter-Tages-Prädiktion über zwei alternative Varianten (WTA und WTU) durchgeführt werden kann, ist in den Tabellen 5.4 und 5.5 noch eine dritte Zeile *Auto-Select* hinzugefügt. Auto-Select wählt dabei automatisch das bessere erzielte Ergebnis zwischen der WTA- und WTU-Variante pro Profil. Zu beachten ist, dass die Auswahl nicht pro Genauigkeitsmetrik erfolgt, sondern über den Mittelwert aller drei Genauigkeitsmetriken.
Im Detail fällt die Auswahl bei Auto-Select aller Profile (*DS*1 und *DS*2) auf 22 % der WTU-Variante und somit 78 % auf die WTA-Variante. Das heißt, dass 22 % aller Profile mit der WTU-Variante der Inter-Tages-Prädiktion und 78 % aller Profile mit der WTA-Variante der Inter-Tages-Prädiktion eine bessere Prognosegüte erzielen. Dies lässt sich auf den erwartet hohen Anteil an Profilen mit typischen einwöchigen Mobilitätsmustern zurückführen. Hierfür ist die Methode der wochentags basierten Clusterbildung (Weekday Cluster) und Vorhersage (WTA-Variante) bereits ausreichend genau. Dies ist auch darauf zurückzuführen, dass bei Profilen mit einem typischen einwöchigen Mobilitätsmuster die Arbeitstage bei Weekday Cluster in fünf Cluster (Montag bis Freitag) und bei Feature Cluster meist in nur ein Cluster zusammengefasst werden. Darunter leidet wiederum die Genauigkeit der WTU-Variante der Intra-Tages-Prädiktion bei Profilen mit typischen einwöchigen Mobilitätsmustern. Bei Profilen mit scheinbar atypischen Mobilitätsmustern konnte dennoch durch den 22 % Anteil an wochentagunabhängigen Mobilitätsvorhersagen bei Auto-Select eine durchschnittlich um 5,8 % erhöhte Prognosegüte erzielt werden.
Beim direkten Vergleich der beiden Tabellen 5.4 und 5.5, zeigt *DS*1 eine insgesamt bessere Prognostizierbarkeit als *DS*2 auf. Dies war aufgrund der geringeren Entropie von *DS*1 (vgl. Abbildung 5.12) bereits zu erwarten. Außerdem wird für *DS*1 deutlich, dass die Vorhersage eines einzelnen Tages einfacher ist als die Vorhersage der kompletten Woche. Die Prognosegüten von WTA und WTU bewegen sich auf ähnlichem Niveau. Datensatz *DS*2 (Tabelle 5.5) hingegen zeigt eine insgesamt geringere Prognostizierbarkeit.

WTA und WTU bewegen sich auch hier in einem vergleichbaren Wertebereich, wobei WTA eine leicht bessere Prognosegüte besitzt. Dies kann darauf zurückgeführt werden, dass ein Großteil der Profile über einen wöchentlichen Rhythmus verfügt.
Ein Einblick in die Prognosegüte des KNN selbst (betrifft lediglich die WTU-Variante) ist in Abbildung 5.13 gegeben. Es zeigt die durchschnittliche Prognosegüte jeden Tages, beziehungsweise Feature Cluster, bei einem Prognosehorizont von einer Woche. Die dargestellte Prognose ist somit ein Auszug des Gütekriteriums '1 Woche'. Um die Prognosegüte des KNN zu bestimmen, werden die sieben Tage des Testdatensatzes nach Feature Cluster geclustert und mit dem vorhergesagten Feature Cluster des KNN verglichen. Dies führt zu einem binären Vergleich, ob das richtige Feature Cluster vorhergesagt wurde oder nicht. Aus Abbildung 5.13 lässt sich erkennen, dass die Prognosegüte über mehrere Tage abnimmt. Dies lässt sich dadurch erklären, dass vorhergesagte Feature Cluster wieder als Eingabewert für das KNN für die Prognose des nächsten Feature Clusters dient. Falsch vorhergesagte Feature Cluster führen so zu einer Fehlerfortpflanzung. Diese Fehlerfortpflanzung lässt sich auch aus den beiden Tabellen 5.4 und 5.5 entnehmen. Sowohl der Mittelwert, als auch der Median der Prognosegüte ist für das Genauigkeitskriterium '1 Tag' am höchsten, was sich dadurch erklären lässt, dass die Mobilitätsvorhersage eines Tages weniger Folgefehler aufweist als die Vorhersage einer ganzen Woche.
Abschließend veranschaulicht Abbildung 5.14 den Zusammenhang zwischen erzielter Prognosegüte und berechneter Entropie. Dabei wird die detaillierte Verteilung der erreichten Prognosegüte pro Profil aus $DS1$ und $DS2$ über die jeweilige Entropie visualisiert. Die Prognosegüte entspricht den erzielten Ergebnissen aus den Tabellen 5.4 und 5.5, wobei nur die Zeile Auto-Select betrachtet wird. Abbildung 5.14(a) entspricht dem Gütekriterium '1 Tag', (b) '1 Woche' und (c) '7 Wochen'. Die farbige Kennzeichnung der Daten dient der Unterscheidung zwischen den Datensätzen $DS1$ (blau) und $DS2$ (rot).

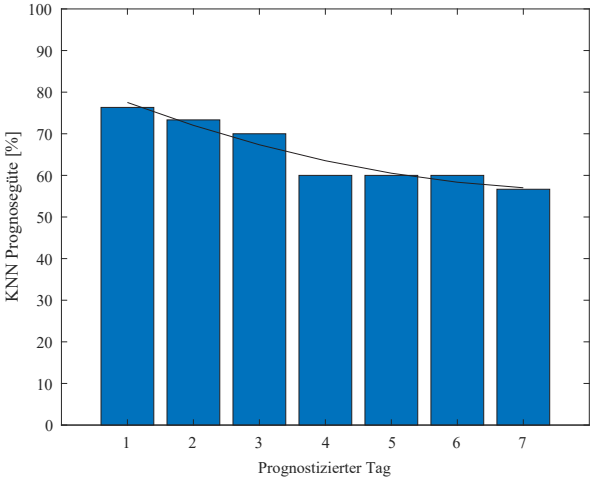
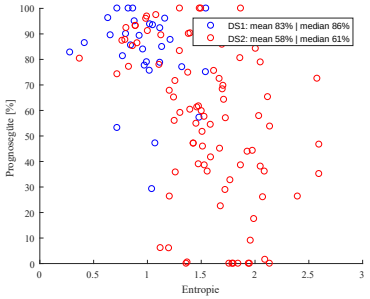
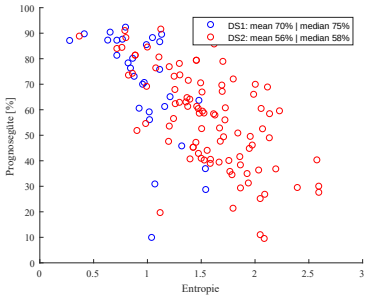



Abbildung 5.13: Durchschnittliche Prognosegüte des KNN selbst, für eine siebentägige Mobilitätsvorhersage der Feature Cluster ('1 Woche'). Da falsch vorhergesagte Feature Cluster wiederum als Input für die Prognose des nächsten Feature Clusters dienen, kann sich die Vorhersagegenauigkeit bei einer Prognose über mehrere Tage verringern (Fortpflanzungsfehler) (nach [57]).

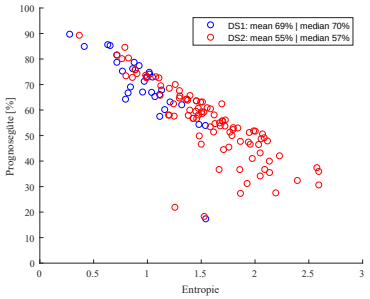
Die drei verschiedenen Abbildungen verdeutlichen die Herausforderung, einen einheitlichen Indikator für die Prognosegüte der Mobilitätsvorhersage zu finden. Während der Vergleich der Entropie und Vorhersagegenauigkeit für einen Tag nur eine schwache Korrelation mit verhältnismäßig vielen Ausreißern aufweist (Abbildung 5.14(a)), zeigt die Vorhersage einer ganzen Woche bereits eine stärkere Korrelation zwischen Entropie und Vorhersagegenauigkeit (Abbildung 5.14(b)). Die Grenzen der Vorhersagbarkeit werden bei dem '7 Wochen' Gütekriterium in Abbildung 5.14(c) anhand der starken Korrelation am deutlichsten. Die Beurteilung der Streudiagramme von oben nach unten verdeutlicht, dass die Korrelation zwischen Prognosegüte und Entropie geringer wird, je kürzer der Prognosehorizont ist. Dies ist nachvollziehbar, da der für die Ermittlung der Prognosegüte zufällig prognostizierte Tag einem Ausreißer entsprechen kann, welcher wiederum von




(a)




(b)




(c)

Abbildung 5.14: Darstellung der erzielten Prognosegüte über die Entropie für alle Profile aus *DS*1 (blau) und *DS*2 (rot). (a) entspricht der Darstellung des Gütekriteriums ’1 Tag’, während (b) und (c) ’1 Woche’ beziehungsweise ’7 Wochen entspricht. Grundsätzlich ist die Prognosegüte von *DS*1 bei allen Gütekriterien höher als für *DS*2 (nach [57]).

der durchschnittlichen Entropie des gesamten Profils in Abbildung 5.14(a) abweicht.
Insgesamt zeigen die Fahrzeugdaten aus *DS*1 (blau) eine systematisch geringere Entropie und damit eine bessere Vorhersagegenauigkeit als die Smartphone-Daten aus *DS*2 (rot). Somit zeigt sich, dass eine bessere Vorhersagbarkeit für Fahrzeugdaten gegeben ist, da Fahrzeuge im Vergleich zu Menschen eine geringere Bewegungsfreiheit besitzen. Außerdem wird von weniger Messfehlern in *DS*1 ausgegangen, was unter anderem an der guten GPS-Abdeckung von Fahrzeugen im Straßenverkehr liegt (Ausnahmen: Tunnel oder Tiefgarage). Im Vergleich dazu weisen die Smartphone-Daten aus *DS*2 größere Lücken in der GPS-Abdeckung auf, welche beispielsweise innerhalb eines Gebäudes oder U-Bahn Station auftreten können. Des Weiteren kann die in Kapitel 5.3 erwähnte Aufbereitung des Datensatzes, zur Unterscheidung von Fahrt- und Bewegungsdaten, für falsche Klassifizierungen im Datensatz sorgen.

## 5.4 Einplanung von OTA-Updates auf Basis der Mobilitätsvorhersage

Der letzte Schritt eines optimierten OTA-Updateprozesses ist die Einplanung des Softwareupdates auf Basis der nutzerindividuellen Mobilitätsvorhersage. Dabei wurde bei der in Kapitel 3.4 geschilderten Idee nur die Einplanung des Installationszeitpunktes vorgeschlagen. Wie bereits in Kapitel 2.3.3 beschrieben, kann ein OTA-Update für den Fahrzeugführer jedoch in zwei relevante Teilschritte gegliedert werden: Softwaredownload und Softwareinstallation. Grundsätzlich ist die automatisierte Einplanung des OTA-Updates unter Betrachtung unterschiedlicher Prämissen für beide Prozessschritte denkbar. Die Ermittlung des nutzerspezifisch optimalen *Downloadzeitfensters* $t_{free\_dow}$ verfolgt die Prämisse, Kosteneinsparungen durch Reduktion beziehungsweise Verhinderung von Mobilfunkdaten zu erzielen (*save money*). Je nach Kostenmodell des Mobilfunkvertrags können dadurch

Mobilfunkkosten für den Fahrzeugführer oder Fahrzeughersteller eingespart werden. Die Ermittlung des nutzerspezifisch optimalen *Installationszeitfensters* $t_{free\_ins}$, hat hingegen die Erhöhung der Fahrzeugverfügbarkeit für den jeweiligen Fahrzeugführer als Primärziel (*save time*).
Dabei sind in beiden Fällen neben der Mobilitätsvorhersage unterschiedliche örtliche Aspekte für die Einplanung von Bedeutung. Während für das Herunterladen der Software eine stabile und für den Fahrzeugführer kostenneutrale Übertragungstechnik im Vordergrund steht, sind für die eigentliche Installation Gegebenheiten, wie das Vorhandensein einer Ladesäule, entscheidend. Durch Letzteres kann beispielsweise ein nicht einkalkulierter Energiebedarf eines OTA-Updates abgesichert werden. Somit kann die Einplanung auch von örtlichen Gegebenheiten abhängig gemacht werden. Dies ermöglicht es zu bestimmen, unter welchen Randbedingungen das OTA-Update durchgeführt werden soll (zum Beispiel nur Zuhause mit Ladesäule).
Für jeden POI lässt sich aus den in Abschnitt 5.1.1 beschriebenen Fahrzeugumgebungsbedingungen bestimmen, welche dieser Gegebenheiten erfüllt sind. Daraus erschließt sich, welche POIs sowohl für das Herunterladen als auch die Installation von Softwareupdates geeignet sind (Menge geeigneter POIs). Das Ziel der Einplanung des Updates ist es, dass diese Updatephasen beim Verweilen des Fahrzeugs an einem dieser geeigneten POIs stattfinden soll. Dazu wird eine möglichst lange Verweildauer an einem dieser POIs mit hoher Wahrscheinlichkeit auf Basis der Mobilitätsvorhersage gefunden.
Dazu sei $\mathscr{P}$ die Menge der besuchten POIs eines Fahrzeugnutzungsprofils und

$$\mathscr{P}_G = \{POI \in \mathscr{P} | \text{Gegebenheiten sind an } POI \text{ erfüllt}\}$$

die Menge aller für den Updateprozess geeigneter POIs. Weiter sei $\widehat{Cl} = \{X^{(1)}, \ldots, X^{(k)}\}$ die Menge der relevanten Tage, die von der Inter-Tages-Prädiktion bestimmt wurden, wobei $X^{(i)} = \left(X_1^{(i)}, \ldots, X_T^{(i)}\right) \in \mathscr{P}^T$ eine Fol-

ge von POIs der Länge $T \in \mathbb{N}$ ist für $i = 1, \ldots, k$. Da das Ergebnis der Inter-Tages-Prädiktion (sowohl WTA als auch WTU) jeweils eine Menge von relevanten Tagen ist, können ebenfalls beide Verfahren analog betrachtet werden. Bei der WTU-Variante entspricht die Menge relevanter Tage dem Feature Cluster ähnlicher Tage, die durch das KNN prognostiziert werden. Bei der WTA-Variante sind dies alle historischen Tage (Weekday Cluster) des jeweiligen Wochentags.
Für die Einplanung wird die Information der relativen Häufigkeit jedes POIs wieder relevant, weshalb die hier beschriebene Einplanung auf Basis der Inter- und nicht der Intra-Tages-Prädiktion aufbaut. Um eine vollständige örtliche und zeitliche Mobilitätsvorhersage inklusive Fahrten zu erhalten, wurde bei der Intra-Tages-Prädiktion (vgl. Abschnitt 5.2.2) der Operator Modus angewendet. Durch diesen Einsatz wurde der am häufigsten vorkommende POI bereits ausgewählt. Allerdings geht dadurch die Information der exakten relativen Häufigkeit jedes einzelnen POI verloren, was für den Vergleich der unterschiedlichen POIs bei der Einplanung wiederum benötigt wird.
Für jeden $POI_i \in \mathscr{P}_G$ und Zeitpunkt $t = 1, \ldots, T$ sei $r_{i,t}$ die relative Häufigkeit des Auftretens von $POI_i$ zum Zeitpunkt $t$ innerhalb aller relevanten Tage $\widehat{Cl}$. In Tabelle 5.6 sind die relativen Häufigkeiten schematisch aufgezeigt.

Tabelle 5.6: Relative Häufigkeiten jedes geeigneten POIs zu den Zeitpunkten $1, \ldots, T$.

| | 1 | … | $T$ |
|---|---|---|---|
| $POI_1$ | $r_{1,1}$ | … | $r_{1,T}$ |
| $POI_2$ | $r_{2,1}$ | … | $r_{2,T}$ |
| ⋮ | ⋮ | ⋮ | ⋮ |

Weiter sei $t_{ins} \in \mathbb{N}$ die Installationsdauer eines Updatevorganges gemessen in Zeitslots. Damit lässt sich

$$p_{i,t}(t_{ins}) = \min\{r_{i,t}, \ldots, r_{i,t+t_{ins}-1}\},$$
$$\text{für } i = 1, \ldots, k \text{ und } t = 1, \ldots, T - t_{ins} + 1$$

als Wahrscheinlichkeit interpretieren, mit der die Installation zum Zeitpunkt $t$ an $POI_i$ durchgeführt werden kann. Gesucht wird der POI $\hat{i}$ und der Zeitpunkt $\hat{t}$, an dem die Wahrscheinlichkeit $p_{\hat{i},\hat{t}}(t_{ins})$ maximal ist. Gibt es mehr als einen Zeitpunkt mit maximaler Wahrscheinlichkeit, wird der längste zusammenhänge Block gewählt. Die Installation wird schließlich mittig in diesem Zeitfenster platziert. Das Vorgehen wird schematisch in Algorithmus 3 dargestellt.

**Algorithm 3** Ermittlung Installationszeitfenster

**Input:** Menge geeigneter POIs $\mathscr{P}_G$,
Menge relevanter Tage $\widehat{Cl}$,
Installationsdauer $t_{ins}$
**Output:** Installationszeitfenster $t_{free\_ins}$
1: Berechne relative Häufigkeiten $r_{i,t}$
2: Berechne $p_{i,t}(t_{ins})$
3: **if** $|\arg\max p_{i,t}(t_{ins})| = 1$ **then**
4: $t_{free\_ins} = \arg\max p_{i,t}(t_{ins})$
5: **else**
6: Wähle einen der längsten zusammenhängenden Blöcke
7: $t_{free\_ins}$ wird mittig im Block eingeplant
8: **end if**
9: **return** $t_{free\_ins}$

Abbildung 5.15 zeigt, wie die Einplanung des Installationszeitfenster eines zweistündigen Updates aussehen könnte. Gezeigt wird die relative Häufigkeit eines geeigneten POI über den Tagesverlauf. In diesem Beispiel ist der POI ebenfalls der meistbesuchte Standort. Der Algorithmus identifiziert das Zeitfenster mit höchster relativer Häufigkeit und plant das Installationszeit-

fenster mittig darin ein. Das hier beschriebene Vorgehen beim Downloadzeitfenster ist analog, lediglich die Menge geeigneter POIs $\mathscr{P}_G$ ändert sich entsprechend.

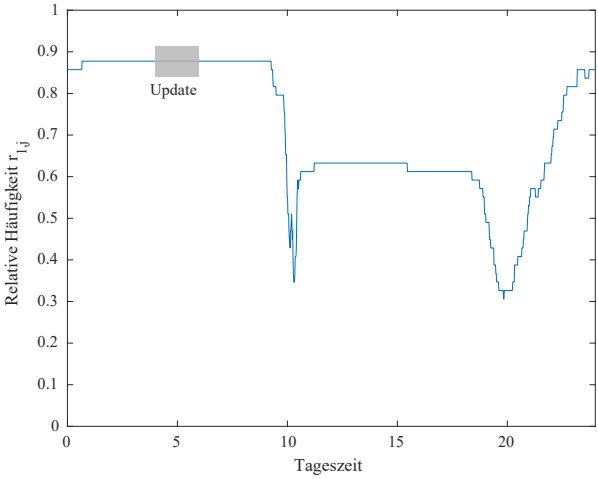



Abbildung 5.15: Die zweistündige Installation wird mittig im Zeitfenster mit höchster relativer Häufigkeit eingeplant.

Aus Gründen der Übersichtlichkeit wird in diesem Kapitel nur die Einplanung für einen Tag betrachtet. Die Einplanung für einen längeren Zeitraum (zum Beispiel eine Woche) erfolgt äquivalent durch Anpassung der Parameter. Es muss lediglich die relative Häufigkeit für den gewünschten Zeitraum berechnet werden. Dazu wird die Folge der prognostizierten relevanten Tage verwendet. Die Ermittlung des Zeitfensters mit maximaler relativer Häufigkeit in Abhängigkeit der Download- beziehungsweise Installationsdauer erfolgt anschließend ebenfalls identisch.

## 5.5 Diskussion und Fazit

In diesem Kapitel wird ein Verfahren vorgestellt, das eine nutzerindividuelle und automatisierte Einplanung von Download- und Installationszeitpunkten eines OTA-Updates in Kraftfahrzeugen ermöglicht. Die zentrale Herausforderung der Einplanung liegt dabei in einer möglichst genauen Prognose des zukünftigen Fahrzeugnutzungsverhaltens, weshalb ein Prognoseframework zur örtlichen und zeitlichen Mobilitätsvorhersage ausgearbeitet und evaluiert wird. Als geeigneter Prognosehorizont steht der Zeitraum von einem Tag bis zu einer Woche (mittelfristige Mobilitätsvorhersage) im Fokus. Dabei werden zur tagesübergreifenden Prognose (Inter-Tages-Prädiktion) zwei alternative Varianten vorgeschlagen. Eine Variante ist wochentagabhängig, die andere wochentagunabhängig. Die darauf aufbauende tagesspezifische Prognose (Intra-Tages-Prädiktion) liefert letztlich den genauen Tagesverlauf mit minütlichem Standortaufenthalt. Durch die Aufbereitung der rohen GPS-Koordinaten zu POI-IDs und Trip-IDs wird dabei nicht nur eine Standortvorhersage getroffen, sondern auch die exakte Route zwischen den Standortwechseln mit jeweiligem Abfahrtszeitpunkt prognostiziert.
Durch Validierung des Prognoseframeworks an zwei verschiedenen Mobilitätsdatensätzen, bestehend aus Fahrzeugdaten ($DS1$) und Smartphone-Daten ($DS2$), findet ein direkter Vergleich von personenbezogener und fahrzeugbezogener Mobilität statt. Insgesamt wird im Median eine Prognosegüte von 75 % für fahrzeugbezogene Mobilitätsprofile und 58 % für personen- beziehungsweise smartphonebezogene Mobilitätsprofile für eine einwöchige Mobilitätsvorhersage erreicht. Durch den direkten Vergleich beider Datensätze lässt sich die Erkenntnis ableiten, dass sich fahrzeugbezogene Mobilität deutlich besser als individuelle personenbezogene Mobilität prognostizieren lässt. Dies liegt nicht zuletzt am geringeren Freiheitsgrad an Bewegungsmöglichkeiten, bedingt durch das Straßenverkehrsnetz, an das Kraftfahrzeuge im Regelfall gebunden sind. Dadurch kann eine automatisier-

te Einplanung von OTA-Updates für Kraftfahrzeuge mit dem vorgestellten Prognoseframework als durchaus praxistauglich erachtet werden.
Zusätzlich hat sich in der Auswertung gezeigt, dass nicht in allen Fällen eine wochentagabhängige Mobilitätsvorhersage, wie sie von einer Reihe modernster Vorhersagemethoden verwendet wird, zur besten benutzerspezifischen Mobilitätsvorhersage führt. Stattdessen übertrifft bei Individuen mit atypischen Mobilitätsmustern eine wochentagunabhängige Clustering-Methode die wochentagabhängige Vorhersage. Bezogen auf beide Datensätze führt diese wochentagunabhängige Variante der Inter-Tages-Prädiktion für 22 % der Mobilitätsprofile zu einem besseren Ergebnis. Über die Korrelation der Prognosegüte mit der profilbezogenen Entropie wird außerdem gezeigt, dass das vorgeschlagene Prognoseframework dazu tendiert, die maximale Vorhersagbarkeit zu erreichen (vgl. Abbildung 5.14). Die Entropie liefert dabei einen vom Prognoseverfahren unabhängigen Leistungsindex, wodurch die Einordnung zu anderen Forschungsprojekten ermöglicht wird. Ein direkter Vergleich gestaltet sich dennoch schwierig, da Faktoren, die einen wesentlichen Einfluss auf die Prognosegüte haben (zum Beispiel Prognosehorizont bezüglich Fehlerfortpflanzung), zu berücksichtigen sind. Zusammengefasst konnten die folgenden Merkmale durch das hier vorgestellte Prognoseframework erzielt werden:

- *Robustheit bezüglich typischen und atypischen Mobilitätsmustern*: Durch Auswahl der alternativen Inter-Tages-Prädiktionen können sowohl wöchentliche als auch außer-wöchentliche Mobilitätsmuster erkannt und prognostiziert werden.
- *Hohe Anpassungsfähigkeit an andere Anwendungsfälle*: Eine einfache Adaption ist auch für andere Datenquellen als GPS möglich. Grundvoraussetzung ist lediglich die Identifikation von POIs und Trips. Faktoren, wie der Prognosehorizont, sind ebenfalls parametrierbar.

- *Kombinierte örtliche und zeitliche Mobilitätsvorhersage*: Eine minutengenaue Standortvorhersage, unter Angabe der spezifisch gefahrenen Routen bei Standortwechseln, wird ermöglicht.
- *Erweiterbarkeit durch alternative Informationsquellen*: Feiertage werden bereits durch das KNN in die Prognose mit einbezogen. Gleichzeitig schafft das KNN den Vorhalt, weitere Informationsquellen wie den persönlichen Kalender mit einzubeziehen.

Die betrachtete Trainingsphase des Prognoseframeworks umfasst in der aktuellen Evaluation sechs Wochen. Die Auswirkungen der dynamischen Änderung dieser Zeitspanne des Trainingsdatensatzes sollte in weiterführenden Untersuchungen analysiert werden. Insbesondere eine länger gewählte Zeitspanne ist dabei gesondert in Betracht zu ziehen. Die aktuelle Mobilitätsvorhersage findet zudem im Wesentlichen auf Basis der Fahrzeugbewegungsaufzeichnungen (GPS-Trajektorien) statt. Auch wenn bereits beim KNN Feiertage mit einbezogen werden, könnten weitere Verbesserungen der Prognosegüte durch eine umfassendere Ergänzung unterschiedlicher Datenquellen erfolgen. Dies hilft die maximale Abdeckung der Mobilitätsdaten zu gewährleisten. So können fehlende oder lückenhafte Informationen durch andere Quellen, wie beispielsweise benutzerspezifische Kalendereinträge, ausgeglichen werden.

# 6 Gesamtheitlich optimierter OTA-Updateprozess

In diesem Kapitel erfolgt die Zusammenführung der jeweils in Kapitel 4 und Kapitel 5 vorgestellten Konzepte. Daraus erschließt sich ein gesamtheitliches Verfahren entlang des OTA-Updateprozesses, wie er in Abbildung 2.8 dargestellt ist. Zunächst wird eine Gesamtübersicht veranschaulicht, bevor anhand eines durchgängigen Beispiels das Prinzip der Einzelschritte verdeutlicht wird.

Die Gesamtübersicht des Prozesses liefert Abbildung 6.1. Insgesamt sind daraus sechs Einzelschritte ersichtlich sowie deren Abhängigkeiten untereinander. Durch die Verortung ist erkennbar, ob der jeweilige Schritt im Backend oder Fahrzeug prozessiert wird. Ebenfalls angegeben ist, welcher Schritt in welchem Kapitel dieser Arbeit behandelt wird. Die Einzelschritte werden nachfolgend näher erläutert und anhand eines durchgängig durchgeführten Beispiels verdeutlicht. Dabei wird das Beispiel mit Hilfe der beiden Fahrzeugnutzungsprofile *A* und *B* aus Abbildung 5.4 durchgeführt.

**OTA-Update initiieren:** Der Bedarf, beziehungsweise die Initiierung eines Softwareupdates seitens des Fahrzeugherstellers, definiert den Startpunkt eines OTA-Updates. Durch Anlegen einer neuen Update-Kampagne erfolgt die Konsolidierung der benötigten Information und Daten in der Backend-Domäne. Aus einer Datenbank aller potentiell im Fahrzeug verfügbaren Steuergeräte werden nur die vom aktuellen OTA-Update betroffenen Steuergeräte ausgewählt. Die Datenbank muss hierbei sowohl Informationen wie Dauer des Updates und ABB, als auch Abhängigkeiten der ausgewählten Steuergeräte untereinander zur Verfügung stellen. Die in der Da-

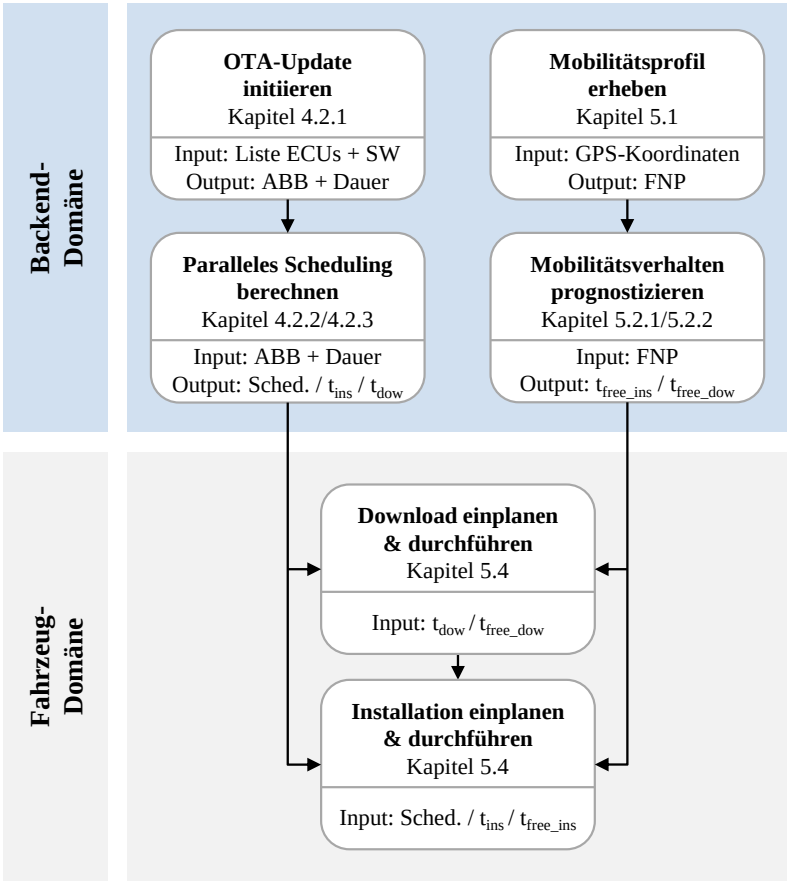



Abbildung 6.1: Gesamtübersicht des entstandenen Prozesses, mit Darstellung der jeweiligen Verortung (Backend oder Fahrzeug). Ebenfalls referenziert sind die jeweils zugehörigen Kapitel in denen das Thema behandelt wird.

tenbank hinterlegten Informationen entsprechen der Darstellung aus Tabelle 4.3. Ebenfalls müssen alle für das OTA-Update relevanten Daten (Flash- und Kodiercontainer) inklusive deren Datenmenge durch die Backend-Domäne bereitgestellt werden.

**Paralleles Scheduling berechnen:** Auf Grundlage der vorliegenden Informationen (Dauer, ABB und Abhängigkeiten) zu den betroffenen Steuergeräten erfolgt die Berechnung des Schedulings zur Anweisung der parallelen Steuergeräte-Softwareupdates. Grundsätzlich können hierfür beide in Kapitel 4.2 vorgestellten Scheduling-Algorithmen in Frage kommen. Das Scheduling kann individuell für ein Fahrzeug oder eine ganze Fahrzeugflotte gelten. In beiden Fällen ergibt sich daraus ein konkretes Scheduling mit einer erwarteten Installationsdauer $t_{ins}$. Werden darüber hinaus noch alternative Ausrichtungen des Schedulings (vgl. Abschnitt 4.4) berechnet, muss eine entsprechende Auswahl bezüglich einer konkreten Ausrichtung getroffen werden. Die erwartete Downloaddauer $t_{dow}$ hingegen kann nur in Abhängigkeit der Kenntnis eines POI mit Wireless-LAN Verbindung (pro Fahrzeugnutzungsprofil) bestimmt werden. Dazu dient folgende Gleichung

$$t_{dow} = \frac{d_{dow}}{WLAN_{rate}} \tag{6.1}$$

wobei $d_{dow}$ der Datenmenge für den Download und $WLAN_{rate}$ der Datenübertragungsrate des Wireless-LAN am jeweiligen POI entspricht.

**Mobilitätsprofil erheben:** Die Erhebung des Fahrzeugnutzungsprofils beginnt unmittelbar nach Überstellung des Fahrzeugs an den Fahrzeughalter und ist grundsätzlich komplett entkoppelt von der Initiierung eines OTA-Updates. Grundvoraussetzung ist lediglich die Zustimmung zur Nutzung ortsbezogener Dienste und somit die Einwilligung der Aufzeichnung von GPS-Koordinaten seitens des Fahrzeughalters. In der Backend-Domäne erfolgt dabei die Ablage und Aufbereitung der GPS-Koordinaten zu POI- und Trip-IDs. Daraus entsteht kontinuierlich das wie in Abschnitt 5.1 beschriebene Fahrzeugnutzungsprofil (FNP).

**Mobilitätsverhalten prognostizieren:** Die eigentliche Mobilitätsvorhersage basiert auf dem aufgezeichneten Fahrzeugnutzungsprofil und wird nur bei Bedarf (zum Beispiel Einplanung des Installationszeitpunktes) abgeru-

fen. Sie erfolgt wie in Kapitel 5.2 beschrieben. Dabei ist der für die Einplanung gewünschte Zeitraum als Prognosehorizont zu parametrieren.

**Download einplanen & durchführen:** Die Einplanung des Softwaredownloads ist eine Maßnahme, um mögliche Mobilfunkkosten für den Fahrzeughersteller oder -halter zu vermeiden. Der hierbei ermittelte Zeitpunkt (Zeitfenster) dient lediglich als Information für den Fahrzeughersteller, wann es sinnvoll sein kann, den Download einzuplanen. Eine Einplanung, beziehungsweise Zustimmung vom Fahrzeughalter, ist für diesen Schritt in der Regel nicht notwendig. Für die automatisierte Einplanung des Downloadzeitfensters wird die erwartete Downloaddauer $t_{dow}$ am jeweiligen POI und die durchgeführte Mobilitätsvorhersage benötigt. Aus beiden Informationen kann die Einplanung, wie in Algorithmus 3 beschrieben, bestimmt werden. Aus dem Algorithmus resultiert der entsprechende Downloadzeitpunkt $t_{free\_dow}$. Gibt es keinen POI im Fahrzeugnutzungsprofil mit Wireless-LAN entfällt der Schritt zur Einplanung des Downloads. Der Download muss somit über die Mobilfunkverbindung erfolgen, wobei kein spezifischer Downloadzeitpunkt benötigt wird.

**Installation einplanen & durchführen:** Erst nachdem die Daten für das OTA-Update vollständig heruntergeladen wurden, wird das Update dem Fahrzeughalter zur Installation angezeigt. Im Gegensatz zur Einplanung des Downloads soll der dabei automatisiert ermittelte Installationszeitpunkt als konkreter Vorschlag per HMI kommuniziert werden. Ausschlaggebend für die Einplanung der Installation sind die erwartete Installationsdauer $t_{ins}$ und eine durchgeführte Mobilitätsvorhersage. Die Installationsdauer $t_{ins}$ ergibt sich dabei direkt aus dem Scheduling-Algorithmus. Wie bereits beim Download wird der Installationszeitpunkt mit Hilfe des Algorithmus 3 ermittelt. Durch das stochastische menschliche Verhalten muss die getroffene Mobilitätsvorhersage nicht zwangsläufig zutreffen, wodurch der vorgeschlagene Installationszeitpunkt ungünstig liegen kann. Die Option zur selbstbestimmten Einplanung eines OTA-Updates muss daher weiterhin bestehen (HMI-Eingabe). Dem Fahrzeughalter wird bei der Entscheidung zur Einplanung

eines Softwareupdates somit lediglich ein individueller Vorschlag gemacht, gänzlich ohne Mehraufwand.
Abschließend sollen die beschriebenen Einzelschritte des Gesamtprozesses anhand eines durchgängigen Beispiels im Detail veranschaulicht werden. Hierfür wird das Szenario betrachtet, das berechnete Scheduling eines initiierten OTA-Updates über mehrere Steuergeräte für zwei beispielhafte Fahrzeugnutzungsprofile einzuplanen.

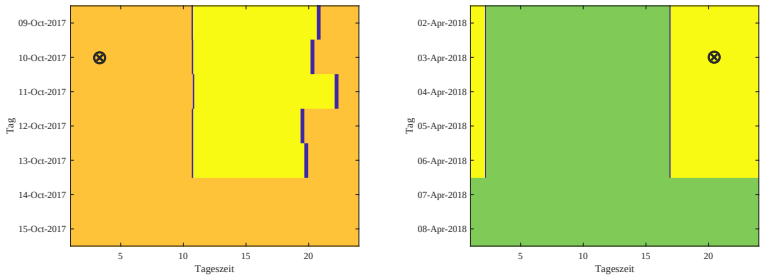
**Beispiel 6.1.** *Mobilitätsprofil erheben: Für das hier durchgeführte Beispiel werden wieder die beiden Fahrzeugnutzungsprofile A und B aus Beispiel 5.3 genutzt. Die jeweilige aufbereitete Fahrzeugbewegungsaufzeichnung (POI- und Trip-IDs) ist in Abbildung 5.4 gegeben. Als Besonderheit der Fahrzeugumgebungsbedingungen verfügen die beiden Fahrzeugnutzungsprofile A und B jeweils über einen POI mit Wireless-LAN Verbindung. Bei Fahrzeugnutzungsprofil A verfügt POI 3 über eine maximale Datenübertragungsrate von* $WLAN_{rate} = 54\,MBit/s$*, während bei Fahrzeugnutzungsprofil B am POI 5 lediglich* $11\,MBit/s$ *verfügbar sind.*
*OTA-Update initiieren: Alle Einträge aus Tabelle 4.3 sollen für dieses Beispiel die für das OTA-Update relevanten Steuergeräte beziehungsweise durchzuführende AUS entsprechen. Somit sind insgesamt sieben Steuergeräte mit zehn AUS vom hier exemplarisch initiierten OTA-Update betroffen. Als summierte Datenmenge* $d_{dow}$ *aller 10 AUS ergeben sich 7,8 Gigabyte für den Softwaredownload. Dabei sollen beide Fahrzeuge (Fahrzeugnutzungsprofile A und B) dasselbe OTA-Update erhalten.*
*Paralleles Scheduling berechnen: Die Berechnung des Schedulings wird mit Hilfe des Hybrid-Algorithmus (vgl. Abschnitt 4.2.3) auf Basis der in Tabelle 4.3 hinterlegten Informationen durchgeführt. Daraus resultiert das Scheduling aus Abbildung 4.11(b), das einer Ausrichtung mit maximaler Zeitersparnis entspricht (vgl. Kapitel 4.4). Damit ergibt sich eine zu erwartende Installationsdauer von* $t_{ins} = 340\,sec.$ *Mit einem Puffer von 20 % wird bei diesem OTA-Update somit eine Installationsdauer von rund* $t_{ins} = 7\,min$

*erwartet. Dies entspricht einer verhältnismäßig kurzen Installationsdauer, wobei durch die Parallelisierung knapp 60 % der Installationsdauer gegenüber dem sequentiellen Updatevorgang eingespart werden. Die Berechnung der jeweiligen Downloaddauer erfolgt nach Gleichung 6.1. Für die ermittelte Datenmenge von 7,8 Gigabyte (entspricht 62400 Megabit) ergibt sich für eine Datenübertragungsrate von* $WLAN_{rate} = 54\,MBit/s$ *eine Downloaddauer von* $t_{dow} = 1155\,sec.$ *Mit der Beaufschlagung eines Puffers von 20 %, resultiert daraus eine geschätzte Downloaddauer von* $t_{dow} = 23\,min$ *am POI 3 für Fahrzeugnutzungsprofil A. Für Fahrzeugnutzungsprofil B ergibt sich aus der Datenübertragungsrate von* $WLAN_{rate} = 11\,MBit/s$ *eine Downloaddauer von* $t_{dow} = 5672\,sec.$ *Durch den 20 % Puffer resultiert daraus eine geschätzte Downloaddauer von* $t_{dow} = 113\,min$ *am POI 5.*

*Mobilitätsverhalten prognostizieren: Für die Mobilitätsvorhersage der beiden Fahrzeugnutzungsprofile A und B wird davon ausgegangen, dass der heutige Tag dem jeweilig letzten Tag in Abbildung 5.4 entspricht (A: 08-Oct-2017, B: 01-Apr-2018). Ziel der Prognose ist eine konsekutive siebentägige Mobilitätsvorhersage der in Abbildung 5.4 dargestellten Profile. Das Ergebnis zeigt Abbildung 6.2, wobei (a) die Mobilitätsvorhersage für FNP A und (b) die Mobilitätsvorhersage für FNP B repräsentiert.*

*Ergänzend ist zu erwähnen, dass die Mobilitätsvorhersage für Fahrzeugnutzungsprofil A auf Basis der WTA- und für Fahrzeugnutzungsprofil B auf Basis der WTU-Variante durchgeführt wurde. Beim Vergleich der Abbildung 6.2 mit Abbildung 5.4 ist in beiden Fällen gut zu erkennen, dass das bisherige Mobilitätsmuster erkannt und prognostiziert wurde.*

*Download plus Installation einplanen und durchführen: Zur Vereinfachung des Beispiels wird die Einplanung von Softwaredownload und -installation nicht separat betrachtet. Die summierte Dauer wird weiterhin zur einfacheren Handhabung als* $t_{ins}$ *bezeichnet. Für Fahrzeugnutzungsprofil A ergibt sich dadurch eine summierte Download- und Installationsdauer von* $t_{ins} = 30\,min$ *und für Fahrzeugnutzungsprofil B* $t_{ins} = 120\,min.$ *Aus Algorithmus 3 ergibt sich bei einer Updatedauer* $t_{ins} = 30\,min$ *und der Mobilitäts-*




(a) (b)

Abbildung 6.2: Siebentägige Mobilitätsvorhersage der beiden exemplarischen Fahrzeugnutzungsprofile *A* in (a) und *B* in (b). Jede Farbe stellt einen POI dar, wobei alle Fahrten einheitlich dunkelblau dargestellt sind. Die ermittelten Updatezeitpunkte sind als Kreuze markiert.

*vorhersage aus Abbildung 6.2(a) ein vorgeschlagener Updatezeitpunkt von* $t_{free\_ins}$ *= 3:15 Uhr | 10.10.2017 für Fahrzeugnutzungsprofil A. Aus Algorithmus 3 ergibt sich bei einer Updatedauer* $t_{ins} = 120\,min$ *und der Mobilitätsvorhersage aus Abbildung 6.2(b) ein vorgeschlagener Updatezeitpunkt von* $t_{free\_ins}$ *= 20:50 Uhr | 03.04.2018 für Fahrzeugnutzungsprofil B. Beide Zeitpunkte sind in Abbildung 6.2 entsprechend markiert.* □

Das abschließende Beispiel verdeutlicht die Funktionsweise und Wirksamkeit des entstandenen Prozesses. Für beide exemplarische FNP konnte sowohl eine Reduzierung der Updatedauer als auch automatisiert ein Updatezeitpunkt ermittelt werden.

# 7 Zusammenfassung und Ausblick

Häufigkeit und Umfang von Softwareupdates in Fahrzeugen werden in den nächsten Jahren aufgrund der aktuellen Megatrends rasant zunehmen. Durch OTA-Updates erfolgt dabei eine Verlagerung des Updateprozesses von der Werkstatt hin zum Fahrzeughalter. Die zentrale Aufgabe bestand darin, den OTA-Updateprozess unter der Prämisse *Erhöhung der Nutzerzufriedenheit* zu optimieren. Dazu wurden zunächst in Kapitel 2 alle notwendigen Grundlagen zur Fahrzeugelektronik und -diagnose vermittelt, während anschließend in Kapitel 3 fünf mögliche Optimierungspotentiale vorgestellt wurden. Als Schwerpunkte der vorliegenden Arbeit sind die beiden Potentiale *Reduzierung der Installationsdauer* und *Ermittlung des Installationszeitpunktes* vertieft betrachtet worden. Entsprechend der identifizierten Defizite aus dem Stand der Forschung und Technik zu beiden Optimierungspotentialen wurden dazu in Kapitel 4 beziehungsweise Kapitel 5 geeignete Lösungsansätze und Methoden vorgestellt.

Die wesentliche Herausforderung bezüglich der *Reduzierung der Installationsdauer* bestand darin, einen Scheduling-Algorithmus herzuleiten, der die Anordnung eines optimalen parallelen Updatevorgangs mehrerer Steuergeräte ermittelt. Diese optimale Anordnung hat zum Ziel die Busbandbreiten der einzelnen Bussysteme vollständig auszunutzen und somit alle Steuergeräte in einer möglichst kurzen Zeit zu aktualisieren. Um das Optimierungsproblem zu lösen, wurden zwei Scheduling-Algorithmen, namentlich *MILP* und *HA*, vorgestellt. Zur Evaluation wurden beide hinsichtlich Rechenzeit und maximaler Zeitersparnis miteinander verglichen. Das MILP lieferte ein optimales Ergebnis, wobei die Rechenzeit bei zunehmenden Ak-

tualisierungsschritten überproportional ansteigt. Der HA lieferte ein ausreichend gutes Ergebnis, wobei das Scheduling höchstens 4% schlechter als das Optimum bei den vorgestellten fünf Testreihen ist. Die Evaluation hat zudem gezeigt, dass die gesamte Updatedauer um durchschnittlich 69% sowie bis zu 77% im Vergleich zu sequentiellen Updates reduziert werden kann. Dabei hat im Falle eines Gesamtfahrzeugupdates die Parallelisierung den größten Effekt. Der vorgestellte Ansatz wurde auf einer E/E-Architektur mit zentralem Gateway getestet und ist aufgrund der hohen Abstraktion leicht an zukünftige Änderungen der E/E-Architektur anpassbar. Dies betrifft sowohl neue Bussysteme als auch dateibasierte Steuergeräte. Des Weiteren lässt sich der Scheduling-Algorithmus, durch die in Kapitel 4.1 vorgestellten Anforderungen, frei bezüglich der Systemspezifikationen eines Fahrzeuges konfigurieren. Dadurch ist es möglich, das volle Potential einer E/E-Architektur auszunutzen.

Zukünftige Untersuchungen sollten sich auf die Reduzierung der benötigten Rechenzeit konzentrieren. Dabei könnte eine Definition als stetiges Problem hilfreich sein. Alternativ kann der HA auch mit dem MILP kombiniert werden. Hierbei berechnet der HA eine mögliche Lösung, die dann als Startpunkt (Warmstart) für das MILP verwendet wird. Zur weiteren Reduzierung der benötigten Installationsdauer kann das vorgestellte Konzept der *Parallelen Updates* auch mit anderen Konzepten wie *Delta Updates* und *Code Compression* kombiniert werden. Ein Vorhalt ist durch die in Abschnitt 4.2.1 erfolgte Abstraktion bereits geschaffen.

Für die *Ermittlung des Installationszeitpunktes* bestand die zentrale Herausforderung in der Herleitung eines Prognoseverfahrens für die zukünftige Fahrzeugnutzung. Als geeigneter Prognosehorizont lag der Zeitraum von einigen Tagen bis zu einer Woche (mittelfristige-Mobilitätsvorhersage) im Fokus. Dabei setzt sich das vorgestellte Prognoseframework aus den beiden Teilschritten Inter-Tages und Intra-Tages Prädiktion zusammen. Die Aufteilung dient der Erkennung von tagesspezifischen und tagesübergreifenden Mobilitätsmustern. Durch die Aufbereitung von rohen GPS-Koordi-

naten zu POI-IDs und Trip-IDs wurde nicht nur eine Standortvorhersage getroffen, sondern auch die exakte Route zwischen den Standortwechseln mit jeweiligen Abfahrtszeitpunkt prognostiziert. Die Validierung des Prognoseframeworks an zwei verschiedenen Mobilitätsdatensätzen, bestehend aus Fahrzeugdaten (*DS*1) und Smartphone-Daten (*DS*2), ermöglichte einen direkten Vergleich personenbezogener und fahrzeugbezogener Mobilität. Für eine einwöchige Mobilitätsvorhersage wurde im Median eine Prognosegüte von 75% für fahrzeugbezogene Mobilitätsprofile und 58% für personenbezogene (smartphonebezogene) Mobilitätsprofile erreicht. Durch den direkten Vergleich beider Datensätze ließ sich die Erkenntnis gewinnen, dass sich fahrzeugbezogene Mobilität deutlich besser als die individuelle personenbezogene Mobilität prognostizieren lässt. Über die Korrelation der Prognosegüte mit der profilbezogenen Entropie wurde außerdem veranschaulicht, dass das vorgeschlagene Prognoseframework dazu tendiert, eine hinreichend gute Vorhersagbarkeit zu erreichen (vgl. Abbildung 5.14). Die Entropie liefert dabei einen vom Prognoseverfahren unabhängigen Leistungsindex. Auf Grundlage dieser Mobilitätsvorhersage erfolgte anschließend die Einplanung des OTA-Updates. Damit wurde eine nutzerindividuelle und automatisierte Einplanung von Download- und Installationszeitpunkten eines OTA-Updates in Kraftfahrzeugen ermöglicht.

Die aktuelle Mobilitätsvorhersage findet im Wesentlichen auf Basis der Fahrzeugbewegungsaufzeichnungen (GPS-Trajektorien) statt. Auch wenn beim KNN bereits Feiertage mit einbezogen werden, könnten weitere Ergänzungen unterschiedlicher Datenquellen zur Steigerung der Prognosegüte beitragen. So würden fehlende oder lückenhafte Informationen durch andere Quellen, wie beispielsweise benutzerspezifische Kalendereinträge, ausgeglichen werden. Des Weiteren kann die hier getroffene Mobilitätsvorhersage auch für andere Anwendungsfälle, die eine Einplanung im Fahrzeug voraussetzen, genutzt werden. Dabei entspricht die Mobilitätsvorhersage einem *prognostizierten Terminkalender*. Reservierungen, wie OTA-Updates oder Ladevorgänge von Elektrofahrzeugen, könnten entsprechend

in den *prognostizierten Terminkalender* eingeplant und dadurch synchronisiert werden.
Die beiden in dieser Arbeit ausgearbeiteten Optimierungspotentiale wurden abschließend in Kapitel 6 zu einem Gesamtprozess zusammengeführt. Ein Beispiel veranschaulicht dabei den entstandenen Prozess und zeigt die Wirksamkeit. Für zukünftige Untersuchungen können die weiteren Optimierungspotentiale aus Kapitel 3 ausgearbeitet werden. Potentiale wie *Reduzierung des Ressourcenbedarfs* lassen sich gut in den bisherigen Gesamtprozess (vgl. Abbildung 6.1) einbinden, während Potentiale wie *Sicherstellung der Prozesssicherheit* in allen Teilschritten gewährleistet sein muss.

# A Anhang

Nachfolgend werden Implementierungs- und Umsetzungsdetails des entstandenen Prozesses bezüglich paralleler Steuergeräte-Softwareupdates geliefert. Abbildung A.1 liefert eine Übersicht des Prozesses, während Anhang A.1 und A.2 die XML-Schemas der Topologiedatei beziehungsweise Parallelisierungsanweisung veranschaulichen.

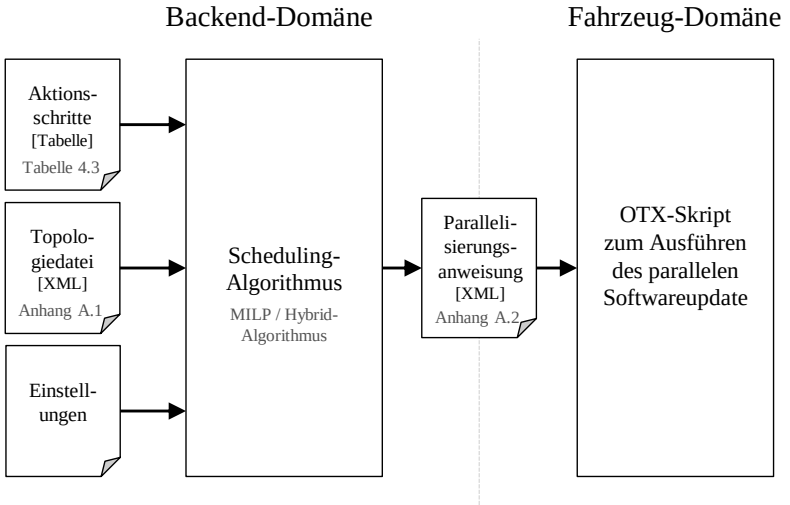



Abbildung A.1: Übersicht des Scheduling Prozesses.

## A.1 Schema Topologiedatei

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Baureihe NameBaureihe="Fahrzeug">
 <topologie name="Basis">
  <ecu Addr="0x19" LL="Gateway" Dauer="54" ABB="700">
   <segment SegmentABB="700" name="CAN_A">
    <ecu Addr="0x81" LL="OverTheAir" Dauer="1260" ABB="300"/>
    <ecu Addr="0x75" LL="ConBox" Dauer="420" ABB="200"/>
    <ecu Addr="0x6D" LL="DeckLid" Dauer="310" ABB="140"/>
    <ecu Addr="0x67" LL="SeatAdjust" Dauer="140" ABB="410"/>
   </segment>
   <segment SegmentABB="700" name="CAN_B">
    <ecu Addr="0xC6" LL="BatteCharg" Dauer="800" ABB="800"/>
    <ecu Addr="0x8C" LL="BatteEnerg" Dauer="420" ABB="320"/>
    <ecu Addr="0x81" LL="DCDCConvert" Dauer="720" ABB="70"/>
    <ecu Addr="0x51" LL="DriveMotor" Dauer="1620" ABB="110"/>
    <ecu Addr="0x2A" LL="WirelCharg" Dauer="820" ABB="420"/>
    <ecu Addr="0xC5" LL="ThermManag" Dauer="120" ABB="560"/>
   </segment>
   <segment SegmentABB="700" name="CAN_C">
    <ecu Addr="0x6C" LL="CamerSystem" Dauer="1020" ABB="240"/>
    <ecu Addr="0x84" LL="NightVisio" Dauer="810" ABB="720"/>
    <ecu Addr="0x3C" LL="LaneChang" Dauer="320" ABB="440"/>
    <ecu Addr="0xC0" LL="ActuaNoise" Dauer="470" ABB="230"/>
    <ecu Addr="0x65" LL="TirePress" Dauer="620" ABB="290"/>
    <ecu Addr="0xCF" LL="LaneChangAss" Dauer="20" ABB="500"/>
   </segment>
   <segment SegmentABB="1000" name="FlexRay_A">
    <ecu Addr="0x47" LL="SoundSystem" Dauer="920" ABB="1000"/>
    ...
   </segment>
   ...
```

```
  </ecu>
 </topologie>
</Baureihe>
```

## A.2 Schema Parallelisierungsanweisung

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Aktionsschrittfolge>
 <Lanes>
  <Lane>
   <Schritt Inhalt="Flashen">
    <LogicalLink>DoorElect</LogicalLink>
    <Nutzdaten>Flashcontainer_DE</Nutzdaten>
    <SWVersionNachFlashen>DE94</SWVersionNachFlashen>
   </Schritt>
   <Schritt Inhalt="Flashen">
    <LogicalLink>TransModul</LogicalLink>
    <Nutzdaten>Flashcontainer_TM</Nutzdaten>
    <SWVersionNachFlashen>TM03</SWVersionNachFlashen>
   </Schritt>
   <Schritt Inhalt="Warten">
    <Warteschritte>
     <Warteschritt>
      <LaneID>3</LaneID>
      <SchrittID>2</SchrittID>
     </Warteschritt>
     <Warteschritt>
      <LaneID>2</LaneID>
      <SchrittID>3</SchrittID>
     </Warteschritt>
    </Warteschritte>
```

```
  </Schritt>
  <Schritt Inhalt="Flashen">
   <LogicalLink>SeatAdjust</LogicalLink>
   <Nutzdaten>Flashcontainer_SA</Nutzdaten>
   <SWVersionNachFlashen>SA94</SWVersionNachFlashen>
  </Schritt>
 </Lane>
 <Lane>
  <Schritt Inhalt="Flashen">
   <LogicalLink>DriveMotor</LogicalLink>
   <Nutzdaten>Flashcontainer_DM</Nutzdaten>
   <SWVersionNachFlashen>DM22</SWVersionNachFlashen>
  </Schritt>
  <Schritt Inhalt="Warten">
   <Warteschritte>
    <Warteschritt>
     <LaneID>1</LaneID>
     <SchrittID>1</SchrittID>
    </Warteschritt>
   </Warteschritte>
  </Schritt>
  <Schritt Inhalt="Flashen">
   <LogicalLink>LaneChang</LogicalLink>
   <Nutzdaten>Flashcontainer_LC</Nutzdaten>
   <SWVersionNachFlashen>LC23</SWVersionNachFlashen>
  </Schritt>
  <Schritt Inhalt="Flashen">
   <LogicalLink>OperatUnit</LogicalLink>
   <Nutzdaten>Flashcontainer_OU</Nutzdaten>
   <SWVersionNachFlashen>OU03</SWVersionNachFlashen>
  </Schritt>
```

```
  </Lane>
  <Lane>
   <Schritt Inhalt="Warten">
    <Warteschritte>
     <Warteschritt>
      <LaneID>2</LaneID>
      <SchrittID>3</SchrittID>
     </Warteschritt>
     <Warteschritt>
      <LaneID>1</LaneID>
      <SchrittID>1</SchrittID>
     </Warteschritt>
    </Warteschritte>
   </Schritt>
   <Schritt Inhalt="Flashen">
    <LogicalLink>ContrModul</LogicalLink>
    <Nutzdaten>Flashcontainer_CM</Nutzdaten>
    <SWVersionNachFlashen>CM94</SWVersionNachFlashen>
   </Schritt>
  </Lane>
 </Lanes>
</Aktionsschrittfolge>
```

# B Anhang

## B.1 KNN-Konfiguration

Anzahl Input-Neuronen : 23
Anzahl Hidden-Neuronen : 14
Anzahl Output-Neuronen : 5
Performance Funktion : 'crossentropy'
Trainings Funktion : ' scaled conjugate gradient'

```
epochs  150  Maximum number of epochs to train
show  25  Epochs between displays
goal  0  Performance goal
time  inf  Maximum time to train in seconds
min_grad 1e-6 Minimum performance gradient
max_fail 6  Maximum validation failures
```

# C Anhang

## C.1 Fahrzeuginterne Bussysteme

Tabelle C.1: Übersicht der Eigenschaft fahrzeuginterner Bussysteme.

| | Datenübertragungsrate | Nutzdaten | Anwendung |
|---|---|---|---|
| LIN | 20KBit/s | 8Byte | Karosserie |
| CAN | 500KBit/s - 1MBit/s | 0 - 8Byte | Karosserie, Fahrwerk |
| FlexRay | 10MBit/s | 254Byte | Antriebsstrang |
| Ethernet | 100MBit/s | 46 - 1500Byte | Infotainment, Backbone |

# Abkürzungs- und Symbolverzeichnis

**Formelzeichen**

| | | |
|---|---|---|
| $\mathscr{P}$ | Menge aller POIs | - |
| $\mathscr{X}$ | Menge aller aufgezeichneten Tage des FNP | - |
| $\widehat{Cl}$ | Relevante Tage | - |
| $a_i$ | ABB von AUS $i$ | KBit/s |
| $ABB$ | Allokierte Busbandbreite | KBit/s |
| $b_{i,j}$ | Binäre Variable | - |
| $be_{i,j}$ | SOS Typ 1 Variable | - |
| $bs_{i,j}$ | SOS Typ 1 Variable | - |
| $BW_{input}$ | Maximale Bandbreite des Eingangsbusses | KBit/s |
| $BW_{SB_k}$ | Maximale Bandbreite des Subbusses $k$ | KBit/s |
| $d_i$ | Dauer von AUS $i$ | s |
| $d_{dow}$ | Downloaddatenmenge | MBit |
| $E$ | Entropie | - |
| $end_i$ | Endzeitpunkt von AUS $i$ | s |
| $I$ | Menge aller AUS | - |
| $I_{SB_k}$ | Menge aller AUS, die zu Subbus $k$ gehören | - |
| $J$ | Menge aller Zeitschlitze | - |
| $N$ | Gesamtanzahl an Tagen des beobachteten FNP | - |
| $p_i$ | Parallelisierung AUS $i \in I$ auf Eingangsbus | - |
| $P_{input}$ | Maximale Anzahl paralleler AUS auf Eingangsbus | - |
| $P_{SB_k}$ | Maximale Anzahl paralleler AUS auf Subbus $k$ | - |
| $q_i^k$ | Parallelisierung AUS $i \in I$ auf seinem Subbus $k$ | - |
| $S$ | Ähnlichkeitsmaß | - |

| | | |
|---|---|---|
| $s_i$ | Silhouettenkoeffizient | - |
| $start_i$ | Startzeitpunkt von AUS $i$ | s |
| $T$ | Anzahl der Zeitslots eines Tages | - |
| $t_{dow}$ | Downloaddauer | s |
| $t_{free\_dow}$ | Downloadzeitfenster | - |
| $t_{free\_ins}$ | Installationszeitfenster | - |
| $t_{ins}$ | Gesamtdauer des Updates | s |
| $WLAN_{rate}$ | WLAN Übertragungsgeschwindigkeit | MBit/s |

**Abkürzungen**

| | |
|---|---|
| ABB | Allokierte Busbandbreite |
| AUS | Abstrakte Updateschritte |
| BEV | Battery Electric Vehicle |
| C2C | Car-to-Car |
| C2I | Car-to-Infrastructure |
| CAN | Controller Area Network |
| CC | Code Compression |
| DCU | Domain Control Units |
| DoCAN | Diagnostics over Controller Area Network |
| DoIP | Diagnostic over Internet Protocol |
| DTC | Diagnostic Trouble Code |
| DU | Delta Updates |
| EA | Evolutionärer Algorithmus |
| ECU | Electronic Control Unit |
| EVA | Eingabe-Verarbeitung-Ausgabe |
| FC | Feature Cluster |
| FNP | Fahrzeugnutzungsprofil |
| FoD | Function on Demand |
| GPS | Global Positioning System |
| GW | Gateway |
| HA | Hybrid-Algorithmus |

| | |
|---|---|
| HMI | Human Machine Interface |
| HV | Hochvolt |
| IoT | Internet of Things |
| IVN | In-Vehicle Network |
| KI | Künstliche Intelligenz |
| KNN | Künstliches Neuronales Netz |
| LIN | Local Interconnect Network |
| MCD | Measurement, Calibration und Diagnostic |
| MILP | Mixed-Integer Linear Programming |
| ML | Maschinelles Lernen |
| MM | Markov Modell |
| MOST | Media Oriented Systems Transport |
| OBD | On-Board Diagnostics |
| ODX | Open Diagnostic Data Exchange |
| OEM | Original Equipment Manufacturer |
| OTA | Over-the-Air |
| OTX | Open Test Sequence Exchange |
| PDU | Protocol Data Units |
| PHEV | Plug-in Hybrid Electric Vehicle |
| PLU | Power Line Communication |
| POI | Point of Interest |
| PU | Parallele Updates |
| RW | Random Walk |
| TBM | Time-Based Markov |
| TR | Testreihe |
| UDS | Unified Diagnostic Service |
| VANET | Vehicular Ad Hoc Network |
| VCI | Vehicle Communication Interface |
| WC | Weekday Cluster |
| WTA | Wochentagabhängig |
| WTU | Wochentagunabhängig |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Literaturverzeichnis


[1] *Road Vehicles–Modular vehicle communication interface (MVCI).* : *Road Vehicles–Modular vehicle communication interface (MVCI).* ISO Std. 22900, 2008

[2] *Road Vehicles–Open diagnostic data exchange (ODX).* : *Road Vehicles–Open diagnostic data exchange (ODX).* ISO Std. 22901, 2008

[3] *Road vehicles–Diagnostic communication over Controller Area Network (DoCAN).* : *Road vehicles–Diagnostic communication over Controller Area Network (DoCAN).* ISO Std. 15765, 2011

[4] *Road Vehicles–Diagnostic Communication Over Internet Protocol (DoIP).* : *Road Vehicles–Diagnostic Communication Over Internet Protocol (DoIP).* ISO Std. 13400, 2011

[5] *Road Vehicles–Open Test sequence eXchange format (OTX).* : *Road Vehicles–Open Test sequence eXchange format (OTX).* ISO Std. 13209, 2011

[6] *Road vehicles–Implementation of World-Wide Harmonized On-Board Diagnostics (WWH-OBD).* : *Road vehicles–Implementation of World-Wide Harmonized On-Board Diagnostics (WWH-OBD).* ISO Std. 27145, 2012

[7] *Road vehicles–Unified diagnostic services (UDS).* : *Road vehicles–Unified diagnostic services (UDS).* ISO Std. 14229, 2013

[8] ADAC-DEUTSCHLAND: *ADAC Rückruf-Jahresbericht 2017*. https://www.adac.de/infotestrat/reparatur-pflege-und-wartung/rueckrufe/rueckruf_jahresbericht_2017.aspx, 2017. – [Online; accessed 16.08.2019]

[9] AGGARWAL, C. C.: *Data Mining - The Textbook*. 1. Aufl. Heidelberg New York : Springer-Verlag, 2015. – ISBN 978–3–319–14141–1

[10] ALBERIO, M. ; PARLADORI, G.: Innovation in automotive: A challenge for 5G and beyond network. In: *International Conference of Electrical and Electronic Technologies for Automotive*, IEEE, 2017. – ISBN 978–1–5090–6006–1, S. 1–6

[11] ANDERSON, R. K.: *Visual Data Mining*. New York : John Wiley and Sons, Ltd, 2012. – 155–173 S. – ISBN 978–1–1184–4481–8

[12] ANDRADE, T. ; GAMA, J.: Identifying Points of Interest and Similar Individuals from Raw GPS Data. In: *Mobility IoT 2018 - 5th EAI International Conference on Smart Cities* (2018), S. 1–13

[13] ASHBROOK, D. ; STARNER, T.: Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users. In: *Personal and Ubiquitous Computing* 7 (2003), Nr. 5, S. 275–286. – ISSN 1617–4917

[14] ATEV, S. ; MILLER, G. ; PAPANIKOLOPOULOS, N. P.: Clustering of vehicle trajectories. In: *IEEE Transactions on Intelligent Transportation Systems* 11 (2010), Nr. 3, S. 647–657. – ISSN 1558–0016

[15] BANERJEE, S. ; BOUZEFRANE, S. ; MÜHLETHALER, P.: Mobility Prediction in Vehicular Networks: An Approach Through Hybrid Neural Networks Under Uncertainty. In: *Mobile, Secure, and Programmable Networking*, Springer International Publishing, 2017. – ISBN 978–3–319–67806–1, S. 178–194

[16] BARBOSA, H. ; BARTHELEMY, M. ; GHOSHAL, G. ; JAMES, C. R. ; LENORMAND, M. ; LOUAIL, T. ; MENEZES, R. ; RAMASCO, J. ; SIMINI, F. ; TOMASINI, M.: Human mobility: Models and applications. In: *Physics Reports* 734 (2018), S. 1–74. – ISSN 0370–1573

[17] BAUMANN, P.: *Human Mobility And Aplication Usage Prediction Algortihms For Mobile Devices*. Dresden, TU Dresden, Dissertation, 2016

[18] BECK, J. E. ; WOOLF, B. P.: High-Level Student Modeling with Machine Learning. In: *Intelligent Tutoring Systems*, Springer Berlin Heidelberg, 2000. – ISBN 978–3–540–45108–2, S. 584–593

[19] BELLO, L. L.: ACM The Case for Ethernet in Automotive Communications. In: *SIGBED Rev.* 8 (2011), Nr. 4, S. 7–15. – ISSN 1551–3688

[20] BESSE, P. C. ; GUILLOUET, B. ; LOUBES, J. ; ROYER, F.: Review and Perspective for Distance-Based Clustering of Vehicle Trajectories. In: *IEEE Transactions on Intelligent Transportation Systems* 17 (2016), Nr. 11, S. 3306–3317. – ISSN 1558–0016

[21] BHATTACHARYA, A. ; DAS, S. K.: LeZi-Update: An Information-Theoretic Approach to Track Mobile Users in PCS Networks. In: *Proceedings of the 5th Annual International Conference on Mobile Computing and Networking*, ACM, 1999. – ISBN 1–58113–142–9, S. 1–12

[22] BISHOP, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg : Springer-Verlag, 2006. – ISBN 038–7–31073–8

[23] BOGDAN, D. ; BOGDAN, R. ; POPA, M.: Delta flashing of an ECU in the automotive industry. In: *11th International Symposium on Applied Computational Intelligence and Informatics*, IEEE, 2016. – ISBN 978–1–5090–2381–3, S. 503–508

[24] BORIAH, S. ; CHANDOLA, V. ; KUMAR, V.: Similarity Measures for Categorical Data: A Comparative Evaluation. In: *Society for Industrial and Applied Mathematics - 8th SIAM International Conference on Data Mining 2008, Proceedings in Applied Mathematics 130*, 2008. – ISBN 978–1–6056–0317–9, S. 243–254

[25] BRATZEL, S.: *Die Rückruf - Trends der globalen Automobilhersteller im Jahr 2017*. `http://auto-institut.de/index_htm_files/Rueckrufe_2017_Pressemeldung_Referenzmarkt_USA.pdf`, 2017. – [Online; accessed 16.08.2019]

[26] BURBEY, I.: *Predicting Future Locations and Arrival Times of Individuals*. Blacksburg, Virginia Polytechnic Institute and State University, Dissertation, 2011

[27] BURBEY, I. ; MARTIN, T. L.: Predicting Future Locations Using Prediction-by-partial-match. In: *Proceedings of the First ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments*, ACM, 2008. – ISBN 978–1–60558–189–7, S. 1–6

[28] CHARETTE, R. N.: Why software fails. In: *IEEE Spectrum* 42 (2005), Nr. 9, S. 42–49. – ISSN 0018–9235

[29] CHENG, Y.: Mean shift, mode seeking, and clustering. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17 (1995), Nr. 8, S. 790–799. – ISSN 0162–8828

[30] CHENG, Y. ; QIAO, Y. ; YANG, J.: An Improved Markov Method for Prediction of User Mobility. In: *12th International Conference on Network and Service Management*, IEEE, 2016. – ISBN 978–1–5090–3236–5, S. 394–399

[31] CHOI, M. ; RO, Y. ; LEE, H.: Human Mobility Pattern Prediction Algorithm using Mobile Device Location and Time Data. In: *Stanford University* (2013), S. 1–9

[32] CHONG, E. K. P. ; ZAK, S. H.: *An Introduction to Optimization*. 3. Aufl. New York : John Wiley and Sons, Ltd, 2011. – ISBN 978–1–118–03155–1

[33] COMITO, C.: Human Mobility Prediction Through Twitter. In: *Procedia Computer Science* 134 (2018), S. 129–136. – ISSN 1877–0509

[34] CUTTONE, A. ; LEHMANN, S. ; GONZALEZ, M. C.: Understanding Predictability and Exploration in Human Mobility. In: *EPJ Data Science* 7 (2018), Nr. 2, S. 1–17. – ISSN 2193–1127

[35] DO, T. M. T. ; DOUSSE, O. ; MIETTINEN, M. ; GATICA-PEREZ, D.: A Probabilistic Kernel Method for Human Mobility Prediction with Smartphones. In: *Pervasive and Mobile Computing* 20 (2015), S. 13–28. – ISSN 1574–1192

[36] DO, T. M. T. ; GATICA-PEREZ, D.: Contextual Conditional Models for Smartphone-based Human Mobility Prediction. In: *Conference on Ubiquitous Computing*, ACM, 2012. – ISBN 978–1–4503–1224–0, S. 163–172

[37] EBERSPÄCHER, M. ; GRIMM, M. ; REUSS, H.-C.: Erweiterte server- und datengestützte Diagnosekonzepte. In: *Diagnose in mechatronischen Fahrzeugsystemen IX*. Dresden : TUDpress, 2015. – ISBN 978–3–95908–006–4, S. 191–203

[38] ERTEL, W.: *Grundkurs Künstliche Intelligenz - Eine praxisorientierte Einführung*. 4. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2016. – ISBN 978–3–658–13549–2

[39] ESCH, F.-R.: *Strategie und Technik des Automobilmarketing*. 1. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2013. – 346 S. – ISBN 978–3–8349–3831–2

[40] ETTER, V. ; KAFSI, M. ; KAZEMI, E. ; GROSSGLAUSER, M. ; THIRAN, P.: Where to Go from Here? Mobility Prediction from Instanta-

neous Information. In: *Pervasive and Mobile Computing* 9 (2013), Nr. 6, S. 784–797. – ISSN 1574–1192

[41] EVERITT, B. S. ; LANDAU, S. ; LEESE, M. ; STAHL, D.: *Cluster Analysis*. 5. Aufl. New York : John Wiley and Sons, Ltd, 2011. – 43–69 S. – ISBN 978–0–4709–7781–1

[42] FU, Z. ; TIAN, Z. ; XU, Y. ; QIAO, C.: A Two-Step Clustering Approach to Extract Locations from Individual GPS Trajectory Data. In: *ISPRS International Journal of Geo-Information* 5 (2016), Nr. 10, S. 166–183. – ISSN 2220–9964

[43] FUKUNAGA, K. ; HOSTETLER, L. D.: The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. In: *IEEE Transactions on Information Theory* 21 (1975), Nr. 1, S. 32–40. – ISSN 1557–9654

[44] FÜRST, S.: Challenges in the design of automotive software. In: *Design, Automation and Test in Europe Conference and Exhibition*, IEEE, 2010. – ISBN 978–1–4244–7054–9, S. 256–258

[45] GAMBS, S. ; KILLIJIAN, M.-O. ; PRADO CORTEZ, M. N.: Show Me How You Move and I Will Tell You Who You Are. In: *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, ACM, 2010. – ISBN 978–1–4503–0435–1, S. 103–126

[46] GAMBS, S. ; KILLIJIAN, M.-O. ; PRADO CORTEZ, M. N.: Towards Temporal Mobility Markov Chains. In: *1st International Workshop on Dynamicity Collocated with OPODIS* (2011)

[47] GAMBS, S. ; KILLIJIAN, M.-O. ; PRADO CORTEZ, M. N.: Next Place Prediction using Mobility Markov Chains. In: *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, ACM, 2012. – ISBN 978–1–4503–1163–2, S. 1–6

[48] GAO, H. ; TANG, J. ; LIU, H.: Mobile Location Prediction in Spatio-Temporal Context. In: *Nokia Mobile Data Challenge Workshop* 41 (2012), Nr. 2, S. 1–4

[49] GARRETT, A.: *Inspyred: Bio-inspired Algorithms in Python*. `https://pythonhosted.org/inspyred/`, 2015. – [Online; accessed 13.08.2019]

[50] GOOGLE: *Android 9*. `https://www.android.com/versions/pie-9-0/`, 2019. – [Online; accessed 13.08.2019]

[51] GOPALRATNAM, K. ; COOK, D. J.: Active LeZi: An Incremental Parsing Algorithm for Sequential Prediction. In: *International Journal on Artificial Intelligence Tools* 13 (2004), Nr. 04, S. 917–929. – ISSN 0218–2130

[52] GOULET-LANGLOIS, G. ; KOUTSOPOULOS, H. N. ; ZHAO, Z. ; ZHAO, J.: Measuring regularity of individual travel patterns. In: *IEEE Transactions on Intelligent Transportation Systems* 19 (2018), Nr. 5, S. 1583–1592. – ISSN 1558–0016

[53] HANK, P. ; MÜLLER, S. ; VERMESAN, O. ; KEYBUS, J. Van D.: Automotive Ethernet: In-vehicle networking and smart mobility. In: *Design, Automation and Test in Europe Conference and Exhibition*, IEEE, 2013. – ISBN 978–1–4673–5071–6, S. 1735–1739

[54] HE, H. ; QIAO, Y. ; GAO, S. ; YANG, J. ; GUO, J.: Prediction of User Mobility Pattern on a Network Traffic Analysis Platform. In: *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, ACM, 2015. – ISBN 978–1–4503–3695–6, S. 39–44

[55] HERBERTH, R. ; KOCH, M. ; GAUTERIN, F.: Kundenorientierte Selbstoptimierung des OTA-Updateprozesses vernetzter Kraftfahrzeuge. In: *Diagnose in mechatronischen Fahrzeugsystemen XI*. Dresden : TUDpress, 2017. – ISBN 978–3–95908–099–6, S. 13–30

[56] HERBERTH, R. ; KÖRPER, S. ; STIESCH, T. ; GAUTERIN, F. ; BRINGMANN, O.: Automated Scheduling for Optimal Parallelization to Reduce the Duration of Vehicle Software Updates. In: *IEEE Transactions on Vehicular Technology* 68 (2019), Nr. 3, S. 2921–2933. – ISSN 0018–9545

[57] HERBERTH, R. ; MENZ, L. ; ; KÖRPER, S. ; LUO, C. ; GAUTERIN, F. ; GERLICHER, A. ; WANG, Q.: Identifying Atypical Travel Patterns for Improved Medium-Term Mobility Prediction. In: *IEEE Transactions on Intelligent Transportation Systems* 21 (2020), Nr. 12, S. 5010–5021. – ISSN 1558–0016

[58] HERPEL, T. ; KLOIBER, B. ; GERMAN, R. ; FEY, S.: Routing of Safety-Relevant Messages in Automotive ECU Networks. In: *70th Vehicular Technology Conference Fall*, IEEE, 2009. – ISBN 978–1–4244–2514–3, S. 1–5

[59] HOPPER, E. ; TURTON, B. C. H.: An empirical investigation of metaheuristic and heuristic algorithms for a 2D packing problem. In: *European Journal of Operational Research* 128 (2001), Nr. 1, S. 34–57

[60] IBM: *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual*. V12.6, 2016

[61] IKANOVIC, E. L. ; MOLLGAARD, A.: An alternative approach to the limits of predictability in human mobility. In: *EPJ Data Science* 6 (2017), Nr. 1, S. 1–10. – ISSN 2193–1127

[62] JAHANGIRI, A. ; RAKHA, H. A.: Applying Machine Learning Techniques to Transportation Mode Recognition Using Mobile Phone Sensor Data. In: *IEEE Transactions on Intelligent Transportation Systems* 16 (2015), Nr. 5, S. 2406–2417. – ISSN 1558–0016

[63] JENELIUS, E. ; KOUTSOPOULOS, H. N.: Urban network travel time prediction based on a probabilistic principal component analysis mo-

del of probe data. In: *IEEE Transactions on Intelligent Transportation Systems* 19 (2018), Nr. 2, S. 436–445. – ISSN 1558–0016

[64] JOHANSON, M. ; DAHLE, P. ; SÖDERBERG, A.: Remote Vehicle Diagnostics over the Internet using the DoIP Protocol. In: *The Sixth International Conference on Systems and Networks Communications*, 2011, S. 1–6

[65] KARMARKAR, N.: A new polynomial-time algorithm for linear programming. In: *Combinatorica* 4 (1984), Nr. 4, S. 373–395. – ISSN 1439–6912

[66] KATSAROS, D. ; MANOLOPOULOS, Y.: Prediction in Wireless Networks by Markov Chains. In: *IEEE Wireless Communications* 16 (2009), Nr. 2, S. 56–64. – ISSN 1558–2612

[67] KERN, A.: *Ethernet and IP for Automotive E-E-Architectures: Technology Analysis, Migration Concepts and Infrastructure*. Erlangen, University of Erlangen-Nuremberg, Dissertation, 2012

[68] KIM, J. H. ; SEO, S. H. ; HAI, N. T. ; CHEON, B. M. ; LEE, Y. S. ; JEON, J. W.: Gateway Framework for In-Vehicle Networks Based on CAN, FlexRay, and Ethernet. In: *IEEE Transactions on Vehicular Technology* 64 (2015), Nr. 10, S. 4472–4486. – ISSN 0018–9545

[69] KIM, T.-H. ; KIM, J.-W.: Handover Optimization with User Mobility Prediction for Femtocell-based Wireless Networks. In: *International Journal of Engineering and Technology* 5 (2013), Nr. 2, S. 1829–1837. – ISSN 0975–4024

[70] KIYOHARA, R. ; MII, S. ; MATSUMOTO, M. ; NUMAO, M. ; KURIHARA, S.: A new method of fast compression of program code for ota updates in consumer devices. In: *IEEE Transactions on Consumer Electronics* 55 (2009), Nr. 2, S. 812–817. – ISSN 0098–3063

[71] KIYOHARA, R. ; MII, S. ; TANAKA, K. ; TERASHIMA, Y. ; KAMBE, H.: Study on binary code synchronization in consumer devices. In: *IEEE Transactions on Consumer Electronics* 56 (2010), Nr. 1, S. 254–260. – ISSN 0098–3063

[72] KÖRPER, S.: *Verfahren zur Ermittlung einer situationsoptimierten Durchführung von OTA-Updates*. Karlsruhe, Karlsruher Institut für Technologie, Masterarbeit, 2017

[73] KRIESEL, D.: *Ein kleiner Überblick über Neuronale Netze*. http://www.dkriesel.com/_media/science/neuronalen etze-de-zeta2-2col-dkrieselcom.pdf, 2007. – [Online; accessed 16.08.2019]

[74] KUPPUSAMY, T. K. ; DELONG, L. A. ; CAPPOS, J.: Uptane: Security and Customizability of Software Updates for Vehicles. In: *IEEE Vehicular Technology Magazine* 13 (2018), Nr. 1, S. 66–73. – ISSN 1556–6072

[75] LAURILA, J. K. ; GATICA-PEREZ, D. ; AAD, I. ; BORNET, O. ; DO, T. M. T. ; DOUSSE, O. ; EBERLE, J. ; MIETTINEN, M.: The mobile data challenge: Big data for mobile computing research. In: *Nokia Mobile Data Challenge Workshop*, 2012, S. 1–8

[76] LEE, S. ; LIM, J. ; PARK, J. ; KIM, K.: Next Place Prediction Based on Spatiotemporal Pattern Mining of Mobile Device Logs. In: *Sensors* 16 (2016), Nr. 2, S. 145–164. – ISSN 1424–8220

[77] LEE, Y. S. ; KIM, J. H. ; HUNG, H. V. ; JEON, J. W.: A parallel reprogramming method for in-vehicle gateway to save software update time. In: *International Conference on Information and Automation*, IEEE, 2015. – ISBN 1–4673–9104–2, S. 1497–1502

[78] LEE, Y. S. ; KIM, J. H. ; JANG, S. J. ; JEON, J. W.: Automotive ECU Software Reprogramming Method Based on Ethernet Backbone Net-

work to Save Time. In: *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication*, ACM, 2016. – ISBN 978–1–4503–4142–4, S. 1–8

[79] Li, A. ; Yuan, W. ; Huang, X. ; Qiu, X. ; Ban, X. ; Zhang, Y.: Traffic Control System Based on 5G Communication Network. In: *28th International Symposium on Industrial Electronics*, IEEE, 2019. – ISBN 978–1–7281–3667–7, S. 1950–1955

[80] Lim, H.-T. ; Völker, L. ; Herrscher, D.: Challenges in a Future IP/Ethernet-based In-car Network for Real-time Applications. In: *Proceedings of the 48th Design Automation Conference*, ACM, 2011. – ISBN 978–1–4503–0636–2, S. 7–12

[81] Lin, J. ; Chen, S.-C. ; Shih, Y.-T.: A study on remote on-line diagnostic system for vehicles by integrating the technology of OBD, GPS, and 3G. In: *World Academy of Science, Engineering and Technology* 56 (2009), Nr. 8, S. 435–441. – ISSN 1307–6892

[82] Liu, Y.-J. ; Yao, Y. ; Liu, C.-X. ; Chu, L.-T. ; Liu, X.: A Remote On-Line Diagnostic System for Vehicles by Integrating OBD, GPS and 3G Techniques. In: *Practical Applications of Intelligent Systems*, Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–25658–5, S. 607–614

[83] Liu, Z. ; Hu, L. ; Wu, C. ; Ding, Y. ; Zhao, J.: A novel trajectory Similarity-Based approach for location prediction. In: *International Journal of Distributed Sensor Networks* 12 (2016), Nr. 11, S. 1–13. – ISSN 1550–1329

[84] Liu, Z. ; Li, Z. ; Wu, K. ; Li, M.: Urban Traffic Prediction from Mobility Data Using Deep Learning. In: *IEEE Network* 32 (2018), Nr. 4, S. 40–46. – ISSN 0890–8044

[85] Liyuan, T. ; Guihe, Q. ; Jindong, Z.: Remote diagnosis system of vehicle based on telematics. In: *International Conference on Compu-*

*ter, Mechatronics, Control and Electronic Engineering* Bd. 4, IEEE, 2010. – ISBN 978–1–4244–7957–3, S. 217–220

[86] Lu, X. ; Wetter, E. ; Bharti, N. ; Tatem, A. J. ; Bengtsson, L.: Approaching the Limit of Predictability in Human Mobility. In: *Scientific Reports* 3 (2013), S. 2923–2932

[87] Luong, C. ; Do, S. ; Hoang, T. ; Deokjai: A Method for Detecting Significant Places from GPS Trajectory Data. In: *Journal of Advances in Information Technology* 6 (2015), Nr. 1, S. 44–49. – ISSN 1798–2340

[88] Lv, Q. ; Qiao, Y. ; Ansari, N. ; Liu, J. ; Yang, J.: Big Data Driven Hidden Markov Model Based Individual Mobility Prediction at Points of Interest. In: *IEEE Transactions on Vehicular Technology* 66 (2017), Nr. 6, S. 5204–5216. – ISSN 1939–9359

[89] Ma, Z. ; Xing, J. ; Mesbah, M. ; Ferreira, L.: Predicting short-term bus passenger demand using a pattern hybrid approach. In: *Transportation research part C: Emerging technologies* 39 (2014), S. 148–163. – ISSN 1879–2359

[90] Mandour, M. ; Gebali, F. ; Elbayoumy, A. D. ; Abdel Hamid, G. M. ; Abdelaziz, A.: Handover Optimization and User Mobility Prediction in LTE Femtocells Network. In: *International Conference on Consumer Electronics*, IEEE, 2019. – ISBN 978–1–5386–7910–4, S. 1–6

[91] Mansour, K. ; Farag, W. ; ElHelw, M.: AiroDiag: A sophisticated tool that diagnoses and updates vehicles software over air. In: *International Electric Vehicle Conference*, IEEE, 2012. – ISBN 978–1–4673–1562–3, S. 1–7

[92] Marscholik, C. ; Subke, P.: *Datenkommunikation im Automobil: Grundlagen, Bussysteme, Protokolle und Anwendungen*. Berlin, Offenbach : VDE-Verlag, 2011. – ISBN 978–3–8007–3275–3

[93] MENZ, L. ; HERBERTH, R. ; LUO, C. ; GAUTERIN, F. ; GERLICHER, A. ; WANG, Q.: An improved method for mobility prediction using a Markov model and density estimation. In: *Wireless Communications and Networking Conference*, IEEE, 2018. – ISBN 978–1–5090–4184–8, S. 1–6

[94] MICROSOFT: *Windows 10*. `https://blogs.windows.com/windowsexperience/2018/07/25/announcing-windows-10-insider-preview-build-17723-and-build-18204/`, 2018. – [Online; accessed 13.08.2019]

[95] MISRA, P. ; ENGE, P.: *Global Positioning System: Signals, Measurements, and Performance*. 2. Aufl. Lincoln MA : Ganga-Jamuna Press, 2010. – ISBN 978–0–9709–5442–8

[96] MITCHELL, J. E.: Branch and Cut. In: *Wiley Encyclopedia of Operations Research and Management Science*, American Cancer Society, 2011. – ISBN 978–0–4704–0053–1

[97] MOHAMED, A. ; ONIRETI, O. ; HOSEINITABATABAEI, S. A. ; IMRAN, M. ; IMRAN, A. ; TAFAZOLLI, R.: Mobility prediction for handover management in cellular networks with control/data separation. In: *International Conference on Communications*, IEEE, 2015. – ISBN 978–1–4673–6432–4, S. 3939–3944

[98] NEMHAUSER, G. L. ; WOLSEY, L. A.: *Integer and Combinatorial Optimization*. 1. Aufl. New York : John Wiley and Sons, Ltd, 1988. – ISBN 978–0–4718–2819–8

[99] NILSSON, D. K. ; SUN, L. ; NAKAJIMA, T.: A Framework for Self-Verification of Firmware Updates over the Air in Vehicle ECUs. In: *Globecom Workshops*, IEEE, 2008. – ISBN 978–1–4244–3062–8, S. 1–5

[100] OLIVEIRA, J. F. ; NEUENFELDT, A. ; SILVA, E. ; CARRAVILLA, M. A.: A Survey on Heuristics for the two-dimensional Rectangu-

lar Strip Packing Problem. In: *Pesquisa Operacional* 36 (2016), Nr. 2, S. 197–226. – ISSN 0101–7438

[101] ONUMA, Y. ; NOZAWA, M. ; TERASHIMA, Y. ; KIYOHARA, R.: Improved Software Updating for Automotive ECUs: Code Compression. In: *40th Annual Computer Software and Applications Conference* Bd. 2, IEEE, 2016. – ISBN 978–1–4673–8845–0, S. 319–324

[102] ONUMA, Y. ; TERASHIMA, Y. ; SUMIKA ; NAKAMURA ; KIYOHARA, R.: Compression method for ECU software updates. In: *Tenth International Conference on Mobile Computing and Ubiquitous Network*, IEEE, 2017. – ISBN 978–1–5386–0594–3, S. 1–6

[103] PASCHOS, V. T.: *Concepts of Combinatorial Optimization*. 2. Aufl. New York : John Wiley and Sons, Ltd, 2012. – ISBN 978–1–1186–0023–8

[104] POCHET, Y. ; WOLSEY, L. A.: *Production Planning by Mixed Integer Programming*. 1. Aufl. New York : Springer Publishing Company, Incorporated, 2010. – ISBN 978–1–4419–2132–1

[105] QIN, S.-M. ; VERKASALO, H. ; MOHTASCHEMI, M. ; HARTONEN, T. ; ALAVA, M.: Patterns, entropy, and predictability of human mobility and life. In: *PLoS ONE* 7 (2012), Nr. 12, S. 1–8. – ISSN 1932–6203

[106] REVFI, M. ; EBERSPÄCHER, M. ; ZOPPELT, G. ; REUSS, H.-C.: Ganzheitliche Modularchitektur zur remote Diagnostizierung vernetzter Kraftfahrzeuge. In: *Diagnose in mechatronischen Fahrzeugsystemen X*. Dresden : TUDpress, 2016. – ISBN 978–3–95908–050–7, S. 64–80

[107] REVFI, M. ; KOCH, M.: Die Diagnose(R)evolution: Remote Diagnose als Basis für Big Automotive Data und Remote Update – Chancen

und Herausforderungen. In: *Diagnose in mechatronischen Fahrzeugsystemen IX*. Dresden : TUDpress, 2015. – ISBN 978–3–95908–006–4, S. 22–39

[108] RICHTER, F. ; DI MARTINO, S. ; MATTFELD, D. C.: Temporal and Spatial Clustering for a Parking Prediction Service. In: *26th International Conference on Tools with Artificial Intelligence*, IEEE, 2014. – ISBN 978–1–4799–6572–4, S. 278–282

[109] RODRIGUEZ-CARRION, A. ; GARCIA-RUBIO, C. ; CAMPO, C. ; CORTES-MARTIN, A. ; GARCIA-LOZANO, E. ; NORIEGA-VIVAS, P.: Study of LZ-Based Location Prediction and Its Application to Transportation Recommender Systems. In: *Sensors* 12 (2012), Nr. 6, S. 7496–7517. – ISSN 1424–8220

[110] ROUSSEEUW, P. J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. In: *Journal of Computational and Applied Mathematics* 20 (1987), Nr. 1, S. 53–65. – ISSN 0377–0427

[111] RUI, Z. ; GUI-HE, Q. ; JIA-QIAO, L.: Gateway system for CAN and FlexRay in automotive ECU networks. In: *International Conference on Information, Networking and Automation* Bd. 2, 2010. – ISBN 978–1–4244–8104–0, S. 49–53

[112] RUNKLER, T. A.: *Data Mining - Modelle und Algorithmen intelligenter Datenanalyse*. 2. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2015. – ISBN 978–3–834–82171–3

[113] SADILEK, A. ; KRUMM, J.: Far Out: Predicting Long-Term Human Mobility. In: *Twenty-sixth AAAI Conference on Artificial Intelligence*, AAAI Press, 2012. – ISBN 978–1–5773–5568–7, S. 1–7

[114] SCELLATO, S. ; MUSOLESI, M. ; MASCOLO, C. ; LATORA, V. ; CAMPBELL, A. T.: NextPlace: A Spatio-temporal Prediction Framework for Pervasive Systems. In: *Proceedings of the 9th International*

*Conference on Pervasive Computing*, Springer-Verlag, 2011. – ISBN 978–3–642–21725–8, S. 152–169

[115] SCHMIDGALL, R.: *Automotive Embedded Systems Software Reprogramming*. London, Brunel University, Dissertation, 2012

[116] SCHNABEL, P.: *Kommunikationstechnik-Fibel*. Elektronik-Kompendium, 2015. – ISBN 978–3–9815–3022–3

[117] SCHNEIDER, C. M. ; BELIK, V. ; COURONNÉ, T. ; SMOREDA, Z. ; GONZÁLEZ, M. C.: Unravelling daily human mobility motifs. In: *Journal of The Royal Society Interface* 10 (2013), Nr. 84, S. 1–8. – ISSN 1742–5689

[118] SCOTT, J. ; BRUSH, A. B. ; KRUMM, J. ; MEYERS, B. ; HAZAS, M. ; HODGES, S. ; VILLAR, N.: PreHeat: controlling home heating using occupancy prediction. In: *Proceedings of the 13th international conference on Ubiquitous computing*, ACM, 2011. – ISBN 978–1–4503–0630–0, S. 281–290

[119] SHI, G. ; KE, Z. ; YAN, F. ; HU, J. ; YIN, W. ; JIN, Y.: A vehicle electric control unit over-the-air reprogramming system. In: *International Conference on Connected Vehicles and Expo*, IEEE, 2015. – ISBN 978–1–5090–0265–8, S. 48–51

[120] SHREEJITH, S. ; MUNDHENK, P. ; ETTNER, A. ; FAHMY, S. A. ; STEINHORST, S. ; LUKASIEWYCZ, M. ; CHAKRABORTY, S.: VEGa: A High Performance Vehicular Ethernet Gateway on Hybrid FPGA. In: *IEEE Transactions on Computers* 66 (2017), Nr. 10, S. 1790–1803. – ISSN 0018–9340

[121] SOLGI, M. ; BOZORG-HADDAD, O. ; LOAICIGA, H.: *Meta-heuristic and Evolutionary Algorithms for Engineering Optimization*. 1. Aufl. New York : John Wiley and Sons, Ltd, 2017. – 53–67 S. – ISBN 978–1–1193–8705–3

[122] SOMMER, J. ; FEIL, V. ; SANZ, E. A.: Scheduling Algorithms for Simultaneous Software Updates of Electronic Devices in Vehicles. In: *International Conference on Communications Workshops*, IEEE, 2008. – ISBN 978–1–4244–2052–0, S. 393–399

[123] SONG, C. ; QU, Z. ; BLUMM, N. ; BARABÁSI, A.-L.: Limits of predictability in human mobility. In: *Science* 327 (2010), Nr. 5968, S. 1018–1021

[124] SONI, N. ; KUMAR, T.: Study of Various Mutation Operators in Genetic Algorithms. In: *International Journal of Computer Science and Information Technologies* 5 (2014), Nr. 3, S. 4519–4521. – ISSN 0975–9646

[125] STEFFELBAUER, M. ; ERBER, J.: *Diagnose 4.0 - die Antwort auf zunehmende Vernetzung*. `https://automotive.softing.com/fileadmin/sof-files/img/ae/fachartikel/2017/Diagnose_4.0_%E2%80%93_die_Antwort_auf_zunehmende_Vernetzung_1710.pdf`, 2017. – [Online; accessed 22.08.2019]

[126] STIESCH, T.: *Prozess- und Performanceoptimierung von Over-the-Air Updates durch Parallelisierung der Steuergeräte-Programmierung*. Pforzheim, Hochschule Pforzheim, Bachelorarbeit, 2018

[127] TALBOT, S. C. ; REN, S.: Comparision of FieldBus Systems CAN, TTCAN, FlexRay and LIN in Passenger Vehicles. In: *29th International Conference on Distributed Computing Systems Workshops*, IEEE, 2009. – ISBN 978–0–7695–3660–6, S. 26–31

[128] TERAOKA, H. ; NAKAHARA, F. ; KUROSAWA, K.: Incremental update method for vehicle microcontrollers. In: *6th Global Conference on Consumer Electronics*, IEEE, 2017. – ISBN 978–1–5090–4046–9, S. 1–2

[129] THOMAS, J. ; CHAUDHARI, N. S.: A new metaheuristic genetic-based placement algorithm for 2D strip packing. In: *Journal of In-*

*dustrial Engineering International* 10 (2014), Nr. 1, S. 1–16. – ISSN 1735–5702

[130] VALVO, E. L.: Meta-heuristic Algorithms for Nesting Problem of Rectangular Pieces. In: *Procedia Engineering* 183 (2017), S. 291–296. – ISSN 1877–7058

[131] VU, L. ; DO, Q. ; NAHRSTEDT, K.: Jyotish: A novel framework for constructing predictive model of people movement from joint Wi-fi/Bluetooth trace. In: *International Conference on Pervasive Computing and Communications*, IEEE, 2011. – ISBN 978–1–4244–9528–3, S. 54–62

[132] WANG, B. ; GAO, L. ; JUAN, Z.: Travel Mode Detection Using GPS Data and Socioeconomic Attributes Based on a Random Forest Classifier. In: *IEEE Transactions on Intelligent Transportation Systems* 19 (2018), Nr. 5, S. 1547–1558. – ISSN 1558–0016

[133] WU, J.: *Cluster Analysis and K-means Clustering: An Introduction*. 1. Aufl. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. – 1–16 S. – ISBN 978–3–642–29807–3

[134] XU, R. ; II, D. C. W.: *Clustering*. 1. Aufl. New York : John Wiley and Sons, Ltd, 2008. – ISBN 978–0–4703–8277–6

[135] YANG, X.-S.: *Optimization Techniques and Applications with Examples*. 1. Aufl. New York : John Wiley and Sons, Ltd, 2018. – ISBN 978–1–1194–9054–8

[136] ZHANG, H. ; DAI, L.: Mobility Prediction: A Survey on State-of-the-Art Schemes and Future Applications. In: *IEEE Access* 7 (2019), S. 802–822. – ISSN 2169–3536

[137] ZHAO, Z. ; KOUTSOPOULOS, H. N. ; ZHAO, J.: Individual mobility prediction using transit smart card data. In: *Transportation research part C: Emerging technologies* 89 (2018), S. 19–34. – ISSN 1524–9050

[138] ZHENG, V. W. ; ZHENG, Y. ; YANG, X. Xieand Q.: Collaborative Location and Activity Recommendations with GPS History Data. In: *Proceedings of the 19th International Conference on World Wide Web*, ACM, 2010. – ISBN 978–1–60558–799–8, S. 1029–1038

[139] ZHENG, Y. ; LI, F. ; LUO, F.: Vehicle Remote Diagnostic System Implementation Based on 3G Communication and Browser/Server Structure. In: *Lecture Notes in Information Technology* 25 (2012), S. 160–166. – ISSN 2070–1918

[140] ZOPPELT, G.: *Optimierung des Software-Downloads in Steuergeräte über den Diagnosezugang am Fahrzeug*. Ulm, Hochschule Ulm, Bachlorarbeit, 2014

[141] ZOPPELT, G.: *Verfahren zur parallelen Programmierung einer Mehrzahl von Fahrzeugsteuergeräten*. `http://www.freepatentsonline.com/DE102014107072.html`. Version: DE102014107072, 2015. – [Online; accessed 13.08.2019]