universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Smart Approaches to the Parking Spot Search Problem

verfasst von / submitted by

Moritz Bastian Reinhardt, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Diplom-Ingenieur (Dipl.-Ing.)

Wien, 2018 / Vienna 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 940
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Scientific Computing UG2002
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Wilfried Gansterer, M.Sc.

Abstract

The parking spot search problem (PSSP) is a problem millions of drivers face each day. It poses an economic cost as well as an environmental burden. Smart cities offer new ways of assisting drivers to reduce the time spent looking for a free parking spot and the distance between the final parking spot and the actual destination. We compare three fundamentally different approaches to the PSSP in extensive simulations on a grid network with randomly generated routes and varying vehicle densities: First, and as a reference, a naïve random approach that does not use any smart hardware. Second, a global approach where communication of atomic parking spot availability data flows through a central server that can be reached by all vehicles in the network. Third, a completely distributed approach where vehicles gather such information themselves and only share it with their geographical neighbors. Our results show that such smart approaches do reduce search times and remaining distances significantly. The centralized approach performs best in all scenarios. However, it poses the strongest assumptions, from a theoretical perspective as well as on actual infrastructure, and its deployment would be much more expensive than a decentralized solution. Such a distributed approach achieves nearly as good results with a local memory that only stores information about 5 parking spots. It is therefore an important option to consider when trying to improve driver satisfaction in urban areas.

Zusammenfassung

Das Parking Spot Search Problem (PSSP) ist ein wohlbekanntes Problem, mit dem Millionen von Autofahrern täglich konfrontiert werden. Es bedingt wirtschaftliche Kosten und Umweltverschmutzung. Smart Cities bieten neue Möglichkeiten um Fahrern dabei zu helfen, die Zeit, die sie zur Parkplatzsuche benötigen, und die verbleibende Distanz zwischen dem Parkplatz und dem eigentlichen Ziel zu reduzieren. Wir vergleichen drei fundamental verschiedene Ansätze das PSSP zu lösen mittels umfangreichen Simulationen in einem Gitter-Netzwerk mit zufälligen Routen und unterschiedlichen Verkehrsdichten: Erstens, zum Vergleich, einen naiven zufälligen Ansatz, bei dem Autos keinerlei intelligente Hardware nutzen. Zweitens, einen globalen Ansatz, bei dem jegliche Kommunikation von atomaren Parkplatzinformationen über einen zentralen Server läuft. Drittens, einen komplett verteilten dezentralen Ansatz, wobei Autos diese Informationen selber sammeln und nur mit ihren unmittelbaren Nachbarn teilen. Unsere Resultate zeigen, dass solch intelligente Methoden die Dauer der Parkplatzsuche und die verbleibende Distanz signifikant verringern können. Dabei erzielt der zentrale Ansatz die besten Werte. Er stellt allerdings auch die stärksten Bedingungen, aus theoretischer Sicht wie auch an die Hardware in der Praxis, und seine Umsetzung wäre teurer als die eines verteilten Ansatzes. So ein verteilter Ansatz erreicht beinahe die gleiche Leistung mit einem lokalen Speicher der nur Informationen über 5 Parkplätze speichern kann. Daher ist eine dezentrale Lösung für das PSSP eine wichtige Option, wenn man die Zufriedenheit von Autofahrern in städtischen Räumen erhöhen möchte.

Contents

Abstract i

Zusammenfassung iii

Contents v

List of Figures vii

List of Tables viii

List of Algorithms ix

1 Introduction 1
- 1.1 Problem Statement . . . 1
- 1.2 Motivation . . . 2
- 1.3 Synopsis . . . 4

2 Related Work 9
- 2.1 Traffic Flow Optimization . . . 10
- 2.2 Vehicular Communication . . . 11
- 2.3 Parking Spot Search Problem . . . 12
  - 2.3.1 Disseminating Parking Spot Locations . . . 12
  - 2.3.2 Predicting Parking Spot Locations . . . 16
  - 2.3.3 Predicting Parking Spot Occupancy . . . 17
  - 2.3.4 Efficient Candidate Traversal . . . 18
  - 2.3.5 Simulation Setups . . . 19
  - 2.3.6 Similar Studies . . . 21

3 Three Approaches to Parking Spot Search 25
- 3.1 The Naïve Approach . . . 26
- 3.2 The Global Approach . . . 28
- 3.3 The Fully Distributed Approach . . . 30
  - 3.3.1 Data Merging . . . 31
  - 3.3.2 Candidate Ranking . . . 33
  - 3.3.3 The Advanced Fully Distributed Approach . . . 35

4 Simulation Environment 37
- 4.1 Basic Model . . . 37

4.2 Demand Generation 39
4.3 Simulation Generator Pipeline 42
4.4 Simulation Pipeline 44
4.5 Algorithm Parameter Definitions 45
4.6 System 46

5 Simulation Results **49**
5.1 Time Spent During Parking Spot Search 49
5.2 Distance Driven During Parking Spot Search 52
5.3 Distance Between Parking Spot and Destination 55
5.4 Metrics of the Distributed Approach 55
5.5 Message Cost 61
5.6 Discussion 63

6 Conclusion and Future Work **67**

Bibliography **69**

List of Figures

3.1 UML state machine—naïve approach 27
3.2 UML state machine—global approach 29
3.3 Information tuple 31
3.4 UML state machine—fully distributed approach 33

4.1 Screenshot of the SUMO-GUI 38
4.2 Route generation for the hot spot scenario 41
4.3 Visualization of the simulation generator pipeline 43
4.4 Visualization of the simulation pipeline 44

5.1 Average times spent looking for a parking spot (uniformly distributed) 50
5.2 Average times spent looking for a parking spot (hot spot) 51
5.3 Distribution of time spent looking for a parking spot (uniformly distributed) 52
5.4 Average distances driven looking for a parking spot (uniformly distributed) 53
5.5 Average distances driven looking for a parking spot (hot spot) 54
5.6 Average distances between final parking spot and destination (uniformly distributed) 56
5.7 Average distances between final parking spot and destination (hot spot) 57
5.8 Average number of vehicles with knowledge of a free candidate 58
5.9 Target area coverage 59
5.10 Accuracy of data on all relevant parking spots 60
5.11 Accuracy of data on free relevant parking spots 61
5.12 Accuracy of coverage of free relevant parking spots 62
5.13 Average number of messages per vehicle 63

List of Tables

4.1 Demand definition parameters for both scenarios 42
4.2 Algorithm parameters used in our simulations 47
4.3 System specifications of the platform running the simulation pipeline 47

List of Algorithms

1 Naïve parking spot search 27
2 Parking spot search with a global instance 29
3 Fully distributed cooperative parking spot search 32

1 Introduction

Millions of drivers around the world drive into a city each day. All of them need to park their car upon reaching their destination. Finding a place to do so can be an enormous struggle and take a large amount of time due to the limited availability of parking spots especially in city centers. In recent years urban planners and automobile manufacturers realized the importance of this parking spot search problem which lead to the development and installation of assisting hardware and software.

1.1 Problem Statement

To the best of our knowledge the problem stated in this section has only been investigated in various forms but never been defined in one universally accepted way. Therefore we define the *parking spot search problem (PSSP)* as the following problem. A set V of active vehicles competes for a set P of free parking spots. Usually $|V| > |P|$, at least in certain areas of the underlying road network. While we present different traffic density control mechanisms in this thesis, in general it can be said that the number of active vehicles and the number of free parking spots (i.e. $|V|$ and $|P|$) are dependent. When a parking vehicle becomes active, it leaves a parking spot, generating a new free spot. On the contrary, a vehicle that parks at a free spot, occupies it. Equation (1.1) shows the cost function of the parking spot search for vehicle v_i

$$C(v_i) = t_{lfp_{v_i}} + d_{pd_{v_i}} \quad \rightarrow min, \tag{1.1}$$

where t_{lfp} denotes the time spent looking for a parking spot, which means the time from search initiation until the final parking maneuver and d_{pd} is the distance between the final parking spot and the original destination. In fact, since the remaining distance d_{pd} has to be overcome walking—which in turn is slower than driving—it can be converted to time as well. Hence, the actual constituent of cost is *time*.

The PSSP can be tackled from two perspectives, with two similar but effectively different objectives. Using the *individual perspective* every driver wants to find a parking spot as fast as possible as close to his destination as possible, ignoring the needs of other drivers. This means every vehicle $v_i \in V$ tries to minimize its cost function $C(v_i) \rightarrow min$. Such an egoistic approach can still rely on teamwork of

different drivers to fulfill their goals. However, the competition and greedy behavior of the actors may lead to very bad results (long time spent for parking spot search, long distance between final parking spot and real destination) for some, and therefore to worse results on average than taking a cooperative approach. The second perspective is the *general perspective* in which we consider all actors that compete for the limited amount of parking spots at the same time. This means we try to minimize the cost function given in Equation (1.2) which is the sum of the costs of all vehicles in V.

$$C(V) = \sum_{v \in V} C(v) = \sum_{v \in V} t_{lfp_v} + d_{pd_v} \quad \rightarrow min \tag{1.2}$$

Note that due to the nature of the problem (i.e. the interdependence between the solutions of the individual vehicles) it is likely that

$$min(C(V)) = min(\sum_{v \in V} C(v)) < \sum_{v \in V^*} min(C(v)) \tag{1.3}$$

where V^* denotes the same set of vehicles as V, but with a different assignment to parking spots (i.e. a different solution to the problem). This means that if a driver keeps the objective function for the global minimum (Equation (1.2)) in mind, in some cases he might experience a worse result than he would if he was being greedy, but on average he experiences a better one.

This thesis will present different algorithmic approaches to the PSSP. These approaches try to solve the objective of the general perspective (i.e. achieve good performance in the average case, Equation (1.2)) and are evaluated by these means while taking the individual perspective (each driver acts egoistically according to Equation (1.1)). Therefore they should be insusceptible to individual actors that are not part of the system and one could consider to be *cheating*.

While this theoretical problem definition does not distinguish between single parking spots and aggregations in form of designated parking lots the experiments conducted in this thesis only consider curb parking spots. In addition, one has to consider that drivers who are on the lookout for a free parking spot are usually going at a lower speed than other traffic. This reduces the flow within the underlying road network even further than just the increased vehicle density itself.

1.2 Motivation

The time spent looking for a parking spot can essentially be considered *wasted*. In addition, vehicles, whose drivers are looking for roadside parking spots, are usually going at a slower speed than normal traffic flow which leads to road congestion and,

hence, more wasted time. According to [3] the authors of [4] state that "traversing vehicles looking for car park spaces occupy about 30% of the existing vehicles on the road in the downtown area." This excess traffic does not only result in an economic loss but also poses an enormous environmental burden due to emissions and noise exposure. According to [57] vehicles cruising for parking in the district Westwood Village, Los Angeles, drive almost 4000 miles every day. Shoup sums up the consequences very descriptively:

> Over a year, cruising in Westwood Village creates 950,000 excess VMT [vehicle miles of travel]—equivalent to 38 trips around the earth, or four trips to the moon. The obvious waste of time and fuel is even more appalling when we consider the low speed and fuel efficiency of cruising cars. Because drivers average about ten miles an hour in the Village, cruising 950,000 miles a year wastes about 95,000 hours (eleven years) of drivers' time every year. And here's another inconvenient truth about underpriced curb parking: cruising 950,000 miles wastes 47,000 gallons of gasoline and produces 730 tons of CO_2 emissions in a small business district. ([57])

With the emergence of integrated hardware and increased computing capabilities in *Smart Cities* [14] it is an obvious idea to use this infrastructure to solve the PSSP. Urban planners realized the importance of the problem and some municipal authorities have already deployed parking guidance systems (PGS) which are similar to the well-known systems of modern parking garages [48]. These systems use stationary sensors, some interconnecting hardware, and dynamic road signs to guide motorists to vacant parking spots. Such sensors are usually either based on ultrasound [26, 47], infrared light [50], changes in the magnetic field of the earth [77], visual imaging [16], or a combination of those. However, in all cases there needs to be at least one more advanced sensor per group of parking spots, or one simpler sensor per single parking spot. Usually it is the municipal authority that provides them and bears the costs. Given the fact that modern vehicles are already equipped with a wide range of sensors to gather information about their environment it is worth considering shifting the costs of sensor deployment to vehicle owners. Current vehicle-based sensors gather information about other vehicles or traffic in general, road or weather conditions, or the status of the vehicle itself [67]. However, in recent years there have been efforts to develop vehicle-based sensors that can detect (free) parking spots along the road while driving past them [42].

To tackle the PSSP the gathered information has to be shared and made use of in some way. Therefore modern vehicles are not only equipped with sensors but also with communication infrastructure. They can communicate with each other creating Vehicular Ad-Hoc Networks (VANETs) which are a special type of Mobile Ad-Hoc Networks (MANETs) [73]. Due to the mobility of the nodes (i.e. the vehicles) of the network, its topology changes constantly. Neighborhoods split and unite quickly, depending on the transmission ranges and driving speeds of the vehicles.

These aspects have to be taken into consideration when thinking about the different technologies and protocols available to realize the communication as well as the specific use case. Another aspect that stands out when comparing VANETs to classical sensor networks is that while energy and computational power is still limited in vehicles, it is much larger than in a small sensor device. However, this thesis focuses on the high-level algorithmic characteristics of a distributed approach neglecting the low-level technical details of the implementation of the communication. For a comprehensive overview about VANETs—and more generally—Wireless Ad-Hoc and Sensor Networks, refer to [27] and [73].

1.3 Synopsis

This section gives a quick overview of the structure of this thesis and the contents of the respective chapters. As already noted Chapter 1 gives an introduction to the topic of Smart Cities, sensor networks, and vehicular communication within VANETs. Emphasis is placed on the parking spot search problem (PSSP) and a formal problem statement given. In short, it is an often experienced problem of a motorist to find a suitable parking spot upon reaching the destination. Since the number of vehicles is often larger than that of vacant parking spots competition between drivers is high. Everyone of them tries to minimize both the time spent on the search for a free spot and the distance between the final parking spot and the actual destination. The main motivation to solve this problem is the reduction of excess traffic due to motorists that are solely trying to park their vehicle. This increased vehicle density leads to unnecessary pollution and economic loss.

In Chapter 2 we give an overview about related work done in the field. We start off by presenting several exemplary applications of smart systems in the context of vehicles such as the electronic brake warning system (EBW) or the vehicle stability warning (VSW). The literature analysis is divided into three sections: First, traffic flow optimization. Second, vehicular communication; and third, the parking spot search problem. Traffic flow optimization is a typical use case for smart systems in an urban environment. It is already common to use adaptive traffic lights to keep traffic fluid and such have been widely investigated. There are two ways traffic is modelled in theory: (i) as cellular automata (ii) with a continuous model that only discretizes time. Traffic flow optimization is a prominent example that suits the application of an automaton-based model. In this thesis we use the continuous model. We distinguish between vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. There have been efforts by authorities in the United States as well as Europe to standardize such communication and they revolve around the frequency band around 5.9 GHz. Many scientists have investigated the parking spot search problem or certain aspects of it. A focal point of interest—and also at the heart of this thesis—is the question of how to disseminate parking spot locations to vehicles on the search. They can be aggregated or atomic, like in our case, and the

distribution can happen through a centralized server or completely decentralized. We compare these two dissemination methods. Multiple methods have been published that showcase the retrieval of parking spot locations in the first place (e.g. through GPS traces). Since information about free parking spots is of spatio-temporal nature, it is important to predict its reliability upon a possible arrival. Among other ways, this has been done using Markov chains. Once a driver knows multiple candidate parking spots his task is to traverse them efficiently. This can be seen as a Time-Varying Travelling Salesman problem (TVTSP) and solved approximately using heuristics. In the last subsection of Chapter 2 we present a few studies that have very similar problem settings and methods applied to solve them. Namely, three approaches to the parking spot search problem that resemble the core ideas of our three approaches explained in Chapter 3 have been published: (i) Non-assisted parking search (NAPS) (ii) opportunistically-assisted parking search (OAPS) (iii) and centrally-assisted parking search (CAPS). However, in the details those analyses differ from ours crucially. We point out these differences in Section 2.3.6.

We present three fundamentally different approaches to the parking spot search problem in Chapter 3. They mainly differ in the underlying assumptions and illustrate three common concepts of solving a problem with distributed data: (i) The simplest solution is a random approach without any smart logic. Such behaviour—randomly driving through the vicinity of the destination while looking for a free parking spot—can be seen nowadays by drivers that do not use any assisting hardware and software. We call this the *naïve* approach. (ii) A typical idea is using a centralized approach. Such a solution, however, poses the algorithmically very strong assumption that at least one node in the network can be reached from every other node, independent of its actual position. We call this the *global* approach. It simulates drivers that request available parking spot locations from a central server through long-range communication like mobile internet. Another important assumption for this solution is that this central server always knows of all parking spot states in the network at any point in time. In reality this could be implemented by a multitude of sensors throughout a city. (iii) The last, and most complex approach in terms of implementation, is a fully decentralized distributed approach. In this case all vehicles form a VANET and communicate only with their geographical neighbors using short-range communication. The vehicles also are the ones that detect free parking spots using on-board sensors. We call this the *distributed* approach. Vehicles following this approach are equipped with a local memory that stores information about recorded free parking spots and share this information with vehicles in their vicinity. When initiating the search for a parking spot close to their destination they query their own local memory for candidates. If the system does not know of any free parking spot within the search radius around the destination, the driver behaves as if he was following the naïve approach—he randomly traverses his search area until he finds a free parking spot or receives new information. All three approaches only consider curb parking and we therefore assume that vehicles that are actively searching for a free parking spot reduce their maximum speed from 50 km/h to

30 km/h. In any case the on-board system only supplies the driver with information. The decision where to park is left to the driver, which means he stops his search and parks whenever he encounters a free parking spot. In the naïve and the distributed case the search radius increases the longer the search takes—this shall simulate natural driver behavior and guarantee termination. As an additional model we present an advanced variant of the distributed approach where vehicles also store information about occupied parking spots they encounter.

In Chapter 4 we outline the simulation environment we built and used in the course of this thesis. At the core of our simulation framework we use the microscopic traffic simulator SUMO (Simulation of Urban MObility) with a 10×10 grid network. That means we simulate every single vehicle as an entity within a road network with continuous space and only discretize time to seconds. Each vehicle has a predefined route and the sum of all vehicle and route definitions is called *vehicular demand*. We specify two scenarios that differ in the way this demand is initialized: (i) One with *uniformly distributed* random routes and (ii) one with a simulated *hot spot* in the center area of the grid where vehicles are more likely to target this area. To compare the performance of the different algorithms we fix the number of free parking spots within the network by always starting a vehicle when another parks. However, we simulate varying active vehicle densities to analyze the effect of the density on the different approaches. In the course of this thesis we developed a pipeline of python scripts that generates the complete simulation setup with different parameter initializations, multiple random runs for each of them, and result collection and visualization. The individual runs are executed in parallel and consist of one call to SUMO which is then controlled by a python script via the TraCI (Traffic Control Interface) API that is based on socket communication. Chapter 4 also lists and discusses the parameters used for the simulation environment as well as those used for the initialization of the different algorithms.

The results of these simulations are presented in Chapter 5. We define four metrics to compare the performance of the different approaches: (i) the time spent during parking spot search (ii) the distance driven during parking spot search (iii) the 2D-distance between the final parking spot and the actual destination (iv) message cost; and, additionally, analyze the local memory accuracy for the distributed approach and its advanced variant with different memory sizes. Since the time spent and the distance driven during the search are related, the approaches behave similarly considering both metrics. In all cases the global approach achieves the best performance, closely followed by the distributed approach. The naïve random approach performs worst in all regards. The main findings are the following: (i) An increasing number of active vehicles degrades the performance of the global approach more than that of the others (ii) the advanced variant of the distributed approach, that tries to accelerate information decay by also storing information about occupied parking spots, always performs worse than the standard distributed approach (iii) as expected, this advanced variant needs a larger local memory to perform, and otherwise behaves like the naïve approach (iv) as few as five memory slots are sufficient for

the distributed approach; its performance does not improve with larger memories. For the 2D-distance between the final parking spot and the actual destination the relative performances are similar, however, we experience an inverse relationship to the number of active vehicles. The more vehicles are active at the same time, the more fluctuation in the positions of available parking spots, and therefore the shorter these distances for both smart algorithms. Again, the global approach is affected more than the distributed variants, which in this case means its performance improves more. We compare the mean values of these times and distances measured. However, looking at the actual distributions of the original values we notice that they all have positive skew. This means most drivers experience a pleasant result while only a few experience extremely bad results. The analysis of the local memory accuracy for the different distributed variants shows that the additional storage of information about occupied parking spots does not actually lead to more accurate data on free parking spots within the relevant area. It just rather clogs the memory with irrelevant data. In terms of message cost a decentralized approach obviously needs more messages per vehicle than a global approach. However, this load is distributed over all vehicles within the VANET, whereas in a centralized approach the server could be the bottleneck. As expected, for both algorithms the number of messages within the network increases as the number of active vehicles increases. In all cases the results of the hot spot scenario are very similar to those of the uniformly distributed scenario. This could either mean that the algorithms are not affected differently by such a different setting, or—more likely—that the demand definition of the hot spot scenario is not ideal to simulate such a setting.

Finally, in Chapter 6 we conclude the thesis and give an outlook on possible future work in the field. While a centralized approach to the PSSP does lead to the best results for drivers, it is more expensive for authorities than a decentralized approach where the cost of sensor deployment and communication infrastructure could be delegated to the drivers. Such a distributed approach still heavily outperforms a naïve random approach and is therefore a reasonable solution to improve driver satisfaction within city centers. The presented algorithms use a multitude of parameters that are initialized with reasonable defaults, whose effects, however, are not analyzed in detail. This is an immediate task for future work. Also, it is desirable to repeat and prove this thesis's findings using a real world road network in combination with real traffic demand data. An idea for future research is a combination of the smart approaches, for example using a central server for data aggregation but using vehicle-based sensors for opportunistic parking spot discovery.

2 Related Work

As mentioned before, modern vehicles are already equipped with a wide range of ICT hardware. This includes sensors to gather data about their environment, as well as computational devices to turn this data into information, and communication hardware to interact with other actors [67]. Given the introduction of ICT to urban areas of life modern media coined the term *Smart City* (e.g. [49]). In the general Smart City ICT hardware is not only vehicle-based but also stationary and controlled by complex software. The idea is to create certain benefits for users of the new infrastructure. Users of urban infrastructure can be residents, pedestrians, cyclists, drivers, ... These benefits can be of different nature. The most important ones are safety, performance optimization, and entertainment. Since the PSSP does not pose an immediate safety hazard, it falls into the category of performance optimization.

Mousannif et al. present two exemplary applications of how *smart* hardware can assist drivers in traffic to increase safety [44]. One is an electronic brake warning (EBW) system. Consider an obstacle that requires one vehicle to brake sharply, which leads to a sharp braking of a second vehicle. A third vehicle that has its view blocked by the second vehicle can then only induce an emergency braking with a delay due to the visual information being delayed. This delay can be enough to result in a collision and can be avoided if the vehicles are communicating. Vehicle one could broadcast an Electronic Brake Warning (EBW) message which would induce an emergency braking in the following vehicles immediately [44]. The other example is a vehicle that suddenly experiences hazardous road conditions. It can then broadcast a Vehicle Stability Warning (VSW) message which alerts following vehicles in advance [44]. Both applications need an existing network connection between the vehicles beforehand but can improve traffic safety using very simple event-based logic.

Caveney gives a deeper insight into possible safety applications that require communication between vehicles [15]. In addition to the aforementioned applications—and among others—he also presents the on-coming traffic warning (OTW) and the lane change warning (LCW) applications. The former emerges when a vehicle has to cross lanes with the oncoming traffic to overtake another vehicle. In such situations it is very difficult for humans to estimate the distance needed to avoid any head-on collisions or even just deceleration of oncoming vehicles. Using complex path estimation and long-range communication between the vehicle trying to overtake and an oncoming vehicle the system could warn the driver if the distance is insufficient [15].

The lane change warning (LCW) application considers a vehicle about to initiate a lane change while doing so would result in a crash with a vehicle going the same direction. A faster vehicle could approach the initial vehicle from its driver's blind spot. In the possibility of a crash the system alerts the driver to the danger and prevents the lane change. This application requires vehicle-to-vehicle communication between the faster vehicle coming from behind and the vehicle about to change lane. It also requires path prediction to estimate possible intersections of vehicles going the same direction [15].

2.1 Traffic Flow Optimization

A typical goal of using ICT in urban environments is to improve traffic flow which can be considered performance optimization. A very common approach is using adaptive traffic lights [8, 22–25, 32, 41]. While the scientific community is still studying new possibilities and their performance many municipalities in the U.S. are already using a system called *InSync* [51]. The system uses stationary hardware in the form of cameras at intersections to monitor traffic density. It measures the waiting times of the vehicles and adjusts the green phase timings dynamically. Many cities all over the world are already optimizing their traffic flows using similar systems. For a more detailed overview refer to [53] or [78].

In this thesis we are simulating urban traffic at a microscopic level. That is, each vehicle is considered an entity. In comparison a macroscopic approach focuses on general traffic flow definitions and road densities at its lowest level. There are two different models for simulating urban traffic on a microscopic level. One is the discrete model which basically uses cellular automata, and the other is the continuous model. The idea of using cellular automata was first introduced by Nagel and Schreckenberg [45]. The model is discrete in space, time, and state variables. Every road is considered to consist of a certain limited amount of cells. Every one of these cells has two possible states: (i) there is a vehicle inside or (ii) there is no vehicle inside (i.e. it is empty road). The only parameters describing a vehicle are its position (=which cell) and its velocity, which has to be an integer value. At every time step every vehicle's velocity is adjusted depending on the distance to the leading vehicle. Only then, all vehicles are moved. This procedure makes the Nagel-Schreckenberg model prominent for traffic jam simulation and research [45].

In this work we use the continuous model. We consider the space on a road to be continuous and only discretize time for computational reasons. Every vehicle has a certain velocity as well as acceleration at every given point in time. Using this information its position along the road at the next time step is calculated. The model will be described in more detail in Section 4.1.

Gier et al. investigated traffic flow performance gains depending on adaptive traffic lights using a cellular automaton model that extends the original Nagel-Schreckenberg

model [24]. As input they used the road network of the Australian city of Melbourne with empirical real world traffic data as well as a Manhattan grid with generated traffic data. Then they compared the traffic flow using non-adaptive and adaptive traffic lights. Their conclusion is that adaptive traffic lights lead to shorter travel times on average. In particular they state that the fluctuation of the travel time is lowest if the system is fed with traffic data of both sides—upstream and downstream—of the intersection [24].

In [32] Jiang et al. introduced the concept of *Network Operation Reliability (NOR)* emphasizing on global gridlocks in a road network. They also base their analysis on an extended cellular automaton model. In contrast to the aforementioned publication, however, they use adaptive traffic lights to avoid and resolve network gridlocks [32]. Their results show that adaptive traffic lights improve traffic flow significantly if all vehicles use the geometric shortest path in a grid network. In such a setting there are multiple shortest paths with the same length if one ignores intersection and turn details treating them all the same. However, when traffic reaches a certain overall density eventually global gridlock is inevitable. Adaptive traffic lights that adjust green phase timings depending on the density of incoming lanes help in reducing global density hot spots and hence, keeping the network operational at higher global density levels. Further analysis of the authors shows that up-to-date traffic information for drivers to base their route choices on is even more effective in reaching a high NOR. In the case of drivers choosing the time shortest path adaptive traffic lights do not improve the NOR further [32].

2.2 Vehicular Communication

We distinguish between vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. While this thesis neglects the in-depth analysis of the underlying hardware realization it is worth noting that this is of course part of the scientific discourse [27, 55]. The requirements for the communication technology depend closely on the application. The most important parameters are the communication range, bandwidth, speed, and whether the connection is constant or not. Examples of some existing technologies are “direct short range communication (DSRC), 2.5/3G cellular systems, Bluetooth, radar systems or ultra wideband (UWB) communication” ([55]).

The large number of applicable technologies poses the danger of incompatibility. Since it is not of much use if only the vehicles of the same car manufacturer can communicate with each other, authorities are making efforts to standardize communication hardware and protocols. In 2002 the *Vehicle Safety Communications (VSC) Project* was initiated by the *VSC Consortium* which consisted of seven major car manufacturers. The project was supported by the United States Department of Transportation (USDOT) and its final report was released in April 2006 [55, 68].

The goals of the VSC project were to "evaluate the potential safety benefits of communication-based vehicle safety applications" ([68]), "define and evaluate the communications requirements" ([68]) of said applications, and ensure that proposed standards fulfill these requirements [68].

From 2006 to 2009 the follow-up *Vehicle Safety Communications - Applications (VSC-A) Project* was conducted by the *Vehicle Safety Communications 2 Consortium (VSC2)* which had its final report published in 2011 [2]. The project focused on developing and testing vehicle-based communication systems to solve safety applications and evaluate if they outperform autonomous single-vehicle based systems that do not use any communication. The main technology that was investigated was Dedicated Short Range Communications (DSRC) at 5.9 GHz in combination with vehicle positioning [2].

This frequency band around 5.9 GHz has been designated exclusively to Intelligent Transportation Systems (ITS) or more generally speaking vehicle-to-everything (V2X) communication by the European Union in Europe where it is referred to as *ITS-G5* [29, 30]. In the United States the radio technology is standardized by the USDOT as the WLAN standard *IEEE 802.11p* referred to as *Wireless Access in Vehicular Environments (WAVE)* [28]. Jiang et al. were one of the first to design protocols using the WAVE standard [31]. They proposed a set of protocols to address three identified key challenges: channel congestion control, broadcast performance enhancement, and concurrent multichannel operation [31].

2.3 Parking Spot Search Problem

The parking spot search problem is a typical use case every driver frequently experiences. Although it has been investigated in the scientific literature to our knowledge there is no initial work that describes the general problem to be cited. Scientists refer to it as the parking problem [11], the parking space problem [69], the parking search problem [33], the parking space search problem [13], the parking assignment problem [1], or the parking space allocation problem [19]. Therefore we defined the problem we are addressing in this thesis ourselves in Section 1.1. In this section we give a short overview of other work that has been done in this field and compare its problem settings to ours. For a recent "Survey of Smart Parking Solutions" refer to [38].

2.3.1 Disseminating Parking Spot Locations

There are many ways of how information about parking spots can be disseminated to individual drivers. They can be categorized based on *what* information is communicated and *how*. As for the *what*, we can distinguish between aggregates and

individual data points that are sent over the network. In our analysis presented in this thesis we consider atomic information about single parking spots (cf. Section 3.3 and Figure 3.3). The main differing principles of *how* information is shared are (i) centralized over some server that can be reached from all vehicles (ii) distributed directly across the vehicles themselves as mobile nodes (iii) distributed semi-directly across multiple stationary nodes (iv) or a combination of those. Below we present various systems and scientific publications that propose and analyze systems that implement the dissemination of parking spot locations to vehicles on the search.

Since 2010 the San Francisco Municipal Transportation Agency uses a system called *SFpark* [54] in the downtown area of San Francisco. The system consists of stationary sensors at a almost every parking spot and a central server that distributes parking spot locations to drivers. The aim of the project is not only blindly advertising free parking spots to reduce search time, but also to enforce a dynamic pricing policy to keep a certain amount of parking spots free at all times. This is done by increasing and lowering parking cost based on current demand. As a result vehicles on the search for a parking spot do not have to do so for long which in turn reduces the number of vehicles on the streets [54].

In 2004 Basu and Little used short-range V2I communication and the GPS location service to guide vehicles to available parking spots [5]. They created a multi-hop wireless parking meter network which they call PMNET. A PMNET is a special class of ad hoc networks that consists of some static immobile nodes (parking meters) and mobile nodes (vehicles). The parking meters are equipped with radio frequency (RF) transceivers and auxiliary hardware and software to communicate with vehicle clients. The vehicles use wireless ad hoc networking to request the location of a free parking spot. The authors stress that the special benefit of their system is that it does not need any *fixed* wired or wireless infrastructure like cellular or satellite networks in addition to its own hardware [5]. However, for the initial installation of the system a location service like GPS is still needed.

Caliskan et al. developed a topology independent, scalabale information dissemination algorithm for spatio-temporal traffic information within a VANET [13]. Those VANETs are based on the WLAN IEEE 802.11 standard and as an example of spatio-temporal traffic information the authors use parking spot availability data. Special emphasis is put on the fact that the presented algorithm uses periodic broadcasts of aggregated data which actually leads to information dissemination. Stationary parking automats monitor parking space availability and broadcast this information. Vehicles create aggregates of the received data based on a grid partitioning of the area and broadcast these aggregates within the VANET again. Hence, the approach presented in [13] does not only consider V2V but also V2I communication. In their simulations the authors assume that each vehicle has a local resource cache that is unlimited in size, however parking spot availability data is considered invalid if it reaches a certain age. The idea is that atomic information, which in this case corresponds to parking spot availability data of one parking automat—which are very

likely multiple actual parking spots—is less time stable than aggregated information. Also, information aggregated over a certain area should give a driver a good hint of the parking situation upon entering a dense area or possibly even beforehand [13]. In our fully distributed algorithm we use the concept of valuing older information less as well (cf. Section 3.3).

In [3] Aliedani et al. present the *CoPark* approach—a simple negotiation model based on which vehicles decide which areas of a parking lot to target for parking spot search. The approach is largely decentralized, as vehicles negotiate with each other using short-range V2V communication. However, the actual detection of free parking spots is done by stationary sensors and the information is passed on to the vehicles entering the parking lot by a stationary roadside unit using V2I communication. The authors focus on solving the problem of competition for a free parking spot between multiple vehicles rather than the process of initially finding it [3]. This is similar to the work presented in [19] but different to our work which leaves the problem of competition to be discussed for further research (cf. Section 5.6). In [19] Delot et al. present a cooperative reservation protocol for parking spaces in a fully distributed VANET that is purely based on V2V communication. The authors emphasize that solely informing neighboring vehicles of free parking spots does not solve the problem but in fact increase the competition and, hence, reduce the satisfaction of the drivers. For their analysis they use their own system called *VESPA (Vehicular Event Sharing with a mobile P2P Architecture)* which has a built-in information dissemination mechanism [18]. VESPA can be used for different types of events; in [19] the authors define the event "available parking space" when a vehicle leaves a parking position. That vehicle communicates this event. This is similar to our fully distributed approach where the leaving vehicle communicates the time and position of the vacated parking spot (cf. Section 3.3). While the authors explicitly state that the event could also be triggered by a (stationary) sensor [19], in its original form this is a different approach to previously discussed works where stationary infrastructure needs to discover free parking spots (or the event of a vehicle vacating it). Furthermore, every vehicle calculates an *Encounter Probability (EP)* for every event to assess its relevance based on the positions, directions, and speeds, of the vehicle and the event itself [19]. For every free parking spot one vehicle acts as the *coordinator* who advertises it to neighboring vehicles and decides which vehicle the parking spot gets reserved for. Other vehicles that are not chosen do receive the information about the free parking spot; the system, however, does not show this information to the driver [19]. In [20] Delot et al. present their distributed reservation protocol in more detail.

In 2010 Mathur et al. presented a system called *ParkNet* [42]. Selected vehicles are equipped with ultrasonic sensors that detect free parking spots and feed the information into a central server that aggregates the data. Client vehicles can then query the server to receive the locations of free parking spots [42]. The paper especially addresses the GPS location accuracy limits by using an environmental fingerprinting approach. The authors performed experiments in the real world by

equipping vehicles with said sensors and achieved over 90% accuracy when creating occupancy maps for parking spots [42]. In addition they show that equipping as few as 500 taxi cabs of San Francisco with these sensors would be enough to cover the whole city. That approach would be around 10-15 times cheaper than stationary sensors at every parking spot [42]. Coric and Gruteser follow up on this work focusing on the differentiation between legal and illegal parking spots gathered from this crowdsourced data [17]. They stress that multiple recordings of the same parking spot are necessary to achieve satisfying results, but as few as 8 lead to around 90 % classification accuracy.

Already in 2008 Boehle et al. had developed *CBPRS*, a City Based Parking and Routing System [10]. CBPRS monitors and reserves parking spots for participating vehicles using stationary sensors at every parking spot in addition to intelligent lamp posts at intersections. Vehicles in fact only communicate with these lamp posts (V2I) to receive parking spot and/or routing information [10]. Vice versa the vehicles send their travel times for specific roads to the lamp posts at the intersections. Using the vehicle density data the lamp posts exchange so-called *ants* to calculate routing tables. The presented ant based algorithm then uses these routing tables based on current traffic density to guide the vehicles to their reserved parking spot [10]. It is of hierarchical nature because the authors distinguish between different types of streets (residential, highway, ...). Simulations show that the proposed algorithm leads to a significant reduction of travel times in comparison to static routing information generated by Dijkstra's algorithm [10]. This in turn leads to an increase of city wide traffic flows, and with an increasing number of participants the number and duration of traffic jams decreases [10].

Lu et al. focused on large parking lots when creating the smart parking scheme *SPARK* [40]. The system uses stationary roadside units (RSUs) to monitor the parking lot. Similar to the previously discussed approach vehicles communicate with these RSUs and are then guided towards their reserved parking spot. The authors emphasize two things: First, while the RSUs provide real-time parking navigation, they also offer intelligent anti-theft protection, as well as the whole system provides high security. Communication is encrypted and the privacy of the drivers is protected by using pseudo IDs for identification. Second, as GPS has the aforementioned accuracy limitations SPARK does the localization of parking spots and vehicles itself using triangulation between three RSUs. Simulations show an efficient reduction of searching time delay for an available parking space [40]. In [46] Panayappan et al. proposed a similar approach that also uses RSUs but to inform drivers of the current occupancy of relevant parking lots. They specifically addressed the problem of malicious drivers that could try to trick the system into believing they target a different parking lot, hence, distorting the occupancy data reported by the system [46].

2.3.2 Predicting Parking Spot Locations

A common assumption in related literature—and in this thesis as well—is that maps containing the locations of parking spots are given. However, with the exception of systems that install stationary sensors at every parking spot (like [54] for example), these maps usually do not exist. There has been research into generating them by predicting parking spot locations from GPS traces [75] and additional data from mobile phones [52, 58].

In [58] Stenneth et al. propose a method to detect parking spot locations by monitoring GPS traces of mobile phones in combination with monitoring the respective device's bluetooth status. The method is based on their previous work [59] which shows how to detect the current transportation mode of a mobile phone user from his GPS traces. The concept presented in [58] states that the transition of the transport modes *car* → *stationary* → *walking* very likely suggests that the user parked his vehicle at his location during the stationary phase. The same holds for a transition the other way round; which means the user left a parking spot in his vehicle. Therefore it can be concluded, that there must be a parking spot at the position of the stationary phase [58]. To reassure this conclusion, the authors propose combining the GPS data with the timestamps when the mobile phones disconnect from or connect to the vehicle's on-board bluetooth system. If the user approaches a parking meter or uses his phone to pay the parking fee immediately after leaving the car, the probability of him having parked his car increases even further [58]. The authors tested the proposed system with five drivers in the area of Chicago and achieved an average parking detection of over 80 % and an average un-parking (i.e. a vehicle leaving a parking spot) detection of over 85 %.

Later, Salpietro et al. present *Park Here!* [52], a mobile application that tries to identify parking and un-parking events using the data from the accelerometer and gyroscope of mobile phones to detect the transition between the modes *driving* and *walking*. If requested by the user, the status of the bluetooth connection is used as an additional feature in the same way as in the previously mentioned publication. This bluetooth monitoring basically reduces false positives to zero [52]. The actual classification of the feature vectors acquired from the sensor readings is done using a random forest algorithm. To avoid individual outliers in the measurements, the app uses an averaging mechanism and considers the last h readings. The value of h is a trade-off between the accuracy and the responsiveness of the system (i.e. how fast is a mode change detected). Experiments conducted by the authors show that the system achieves a parking detection accuracy of over 90 % [52]. Given a sensor sampling rate of 5 Hz the authors suggest setting h to ≥ 5 which results in a response time of about 1 second.

2.3.3 Predicting Parking Spot Occupancy

A pivotal problem of parking spot location is the reliability of the data. A parking spot becomes available when a previously parked vehicle departs. Depending on the system used this information might reach interested vehicles instantly or with a possibly random delay. Hence, when a vehicle receives it, it might already be outdated, i.e. the parking spot is occupied again. Although some approaches try to avoid this problem by reserving certain parking spots for certain vehicles, greedy drivers that do not act according to the system could still block these spots. There has been effort to predict parking lot occupancy (e.g. [11]), also based on information exchanged among vehicles (e.g. [12]), and related studies occasionally implement these prediction algorithms.

A typical idea to increase the reliability of the data is using aggregates of parking spot data for a certain area of interest. The Park Here! mobile app (cf. Section 2.3.2), for example, only shows probabilities of an empty parking spot in a grid cell to the user [52]. It polishes this probability with an additional confidence value that describes how much relevant information the system actually has. [52] and [58] also present one possibility of predicting parking spot occupancy: The mode changes of mobile phones always have a position and a location attached which could then be disseminated to other vehicles.

In 2007 Caliskan et al. developed a model that predicts the occupancy of parking lots based on queuing theory and continuous-time homogeneous Markov chains [12]. The argument for using a model that is continuous in time is that a parking spot can become available or occupied at any point in time [12]. One parking automat per parking lot monitors its capacity, the number of free parking spots, the arrival rate, and the parking rate, which is based on the time vehicles spend parking. This information in combination with a timestamp then gets distributed through the VANET and disseminated to the vehicles by V2I and V2V communication [12]. The vehicle's on-board computers then compute the predicted occupancy (precisely the probability that at least one parking spot is free) upon the vehicle's possible arrival time for each parking lot and present the results to the driver to make a choice. The authors show efficient ways of performing the numerical computations using the limited resources of on-board hardware of vehicles [12]. It is important to note that the information on which the prediction relies consists of aggregated information of multiple actual parking spots instead of individual atomic parking spots. The described algorithm is therefore not directly applicable for the approach taken in this thesis. In their previously discussed publication [13] the authors even used aggregates of these aggregates (cf. Section 2.3.1).

More recently, in 2015, Bogoslavskyi et al. published another method for predicting the occupancy of single parking spots in combination of efficient routing towards the chosen parking spot within a parking lot [11]. Their approach is also based on a Markov decision process (MDP), however, it does not include any communication

and can be applied by an individual vehicle. The layout of the parking lot (or area that contains the parking spots) and the positions of the parking spots have to be known previously as they are modelled as a graph using different types of vertices for the parking positions and the road positions from which the parking spots are reachable [11]. Vehicles can detect if a parking spot is occupied by another vehicle by using cameras and image recognition. Based on a prior occupancy probability estimate and a series of observations of a parking spot its current occupancy probability can be calculated using a static state binary Bayes filter [11]. Initially the prior probability of all parking spots is 0.5. Upon every new observation or if an estimate hasn't been recalculated in a predefined amount of time, it gets adjusted using the Bayes filter. Just like in [12] the events of cars leaving or parking since the previous observation of a parking spot are modelled as a Poisson distributed random variable [11]. The resulting MDP is solved using policy iteration and the solution guarantees to minimize the time to find a free parking spot and the time it takes to walk from that parking spot to the actual destination. The authors conducted experiments based on real world data gathered from a parking lot to compute the prior probabilities. In simulations they compared their MDP-based approach to three naïve heuristics (searching near the goal, targeting parking spots with the lowest prior occupation probability, searching right at the start) and it achieved significantly better results [11].

2.3.4 Efficient Candidate Traversal

Closely related to the prediction if a parking spot is occupied or not is the choice in which order possible candidates are traversed. Verroios et al. formulate this as a Time-Varying Travelling Salesman problem (TVTSP) and present a method to calculate a *good* solution [69]. They define a cost-function to rate the paths between the different candidate parking spots that depends on the distance between them, the distance between the target parking spot and the real destination, and the probability that the parking spot is still free at the time of arrival.However, the calculation of this probability relies on the average time a parking spot is free and the average number of occupied parking spots visited before having success [69]. The authors assume that while the vehicles keep track of these statistics they are originally "collected" from others [69]. Then they solve this TVTSP using an exact dynamic-programming algorithm. However, this is only possible for small instances (about 9 candidates max.) as the time complexity of the proposed algorithm is $O(n^3T2^n)$ where n is the number of candidate parking spots and T the time of the longest trip. In addition to the inherent problem of complexity explosion the computational on-board hardware of vehicles is also likely limited. Therefore, the authors show ways of reducing the problem size by performing a clustering of the initial set of candidates to apply the exact algorithm only to calculate the optimal route between the different clusters. Within each cluster a simple best-first traversal is done. Tests show that the best results are achieved by an initial clustering with a small radius and then reducing the problem size even further by selecting a good subset of clusters using the k-medoids

method [69]. An important feature of the approach presented in [69] is the so-called *live-mode*. As it is possible that a vehicle receives new information about free parking spots while already being on a search trip, the initially best route might have to get updated. The authors show an algorithm that reacts to these updates by calculating the difference between the original set of candidates and the new set. If the difference is small, the clusters just get updated and a new route calculated. If it is big, the whole clustering has to be redone. In general, the calculation overhead caused by these updates is quite small. Finally, the authors ran extensive simulations to show the impact of clustering and cutting off and to compare their new live method to a simple best-first search. Their results show that it significantly reduces the time spent to find a parking spot while the remaining walking time from the parking spot to the destination stays about the same [69].

In 2015 Abidi et al. published "A New Heuristic for Solving the Parking Assignment Problem" [1]. They explain the parking slot assignment problem for groups (PSAPG). This means they consider the problem of assigning a number of parking spots to a number of vehicles based on individual driver preference (location, parking duration, cost for parking). In fact, they assign vehicles to parking zones, which are aggregates of multiple single parking spots that have the same parameters (general location and cost). A novel hybrid heuristic which they call "hybrid genetic assignment search procedure (HGASP)" ([1]) is presented and compared to three other algorithms. The authors ran simulations based on real world map data from the city of Tunis, including road network, parking zone capacities, and parking rates. The vehicle routes and parking requests, however, were randomly generated with a bias towards the city center. The proposed HGASP algorithm shows significant improvement over simpler algorithms in the distance driven while looking for a parking spot [1]. However, the way this assignment problem is presented it can only be solved by a *global* entity which could be a municipal parking guiding system server. This server knows the state of all parking spots and supplies drivers that request a free parking spot with a route to the one that fits this driver's requirements best while also considering other requests. The study is of basic algorithmic research nature and does not consider any underlying vehicular communication.

2.3.5 Simulation Setups

Another important problem of investigating smart city applications, especially ones that are based on vehicular communication, is that real world tests are often not possible (without huge investments into hardware) or want to be avoided in the first place when doing basic research. Just like for this thesis, the aim is often to get a basic understanding and evaluation of how algorithms and problem approaches perform before deciding which one to implement. In life sciences one would call this *in-vitro* testing as opposed to *in-vivo* testing. Therefore, as presented, computer scientists often rely on simulations for evaluation and comparison of the different methods.

Although there have been many publications on the topic of smart city applications and the parking spot search problem, there is no clear standard simulation software that fits the needs of all applications.

An approach many teams take is implementing their own simulator. As did the authors of [19], [69], and [33] in the programming languages Java and C. The latter is to be discussed in more detail in Section 2.3.6. In the presentation of CBPRS [10] the authors used their own environment which was previously developed by one of them and is described in more detail in his Master thesis [9]. It is written in C# and based on cellular automata (cf. Section 2.1) using an advanced version of the Nagel-Schreckenberg model [45]. For testing the CoPark approach [3] the authors used a combination of tools. To simulate the vehicles within the road network they used the traffic simulator SUMO (Simulation of Urban MObility) [37] which we also use in this work (cf. Chapter 4). On top of SUMO they used JADE (Java Agent Development framework) [6] to simulate the application itself. Both tools were connected through TraSMAPI [66], a generic API for microscopic traffic simulators. While some use generic grid road networks, others import the networks of real cities. However, vehicle routes are usually generated randomly as well as the parking requests. Usually this is done with a bias towards the center to increase competition (like in [56] for example, where the authors use MATLAB [43] in combination with the Google Maps service). We analyze two settings, one with uniformly distributed route origins and destinations, and one with such a bias towards a hot spot (cf. Section 4.2). The remainder of this section discusses two exemplary publications of an interlinking framework and a simulator itself that aim at simulating vehicular communication applications.

In [39] Lochert et al. present a "Multiple Simulator Interlinking Environment for IVC" which the contributing authors use in subsequent publications (e.g. [12, 13]). As the title suggests the proposed interlinking environment is specifically designed to simulate applications involving vehicular communication. Originally such applications are split into three levels, all needing different simulation software: First, the road network and vehicle movements within. Second, the network traffic within the VANET. Third, an approximation of the application data according to some stochastic or deterministic process. The problem with such a setup is that the previously existing tools did not offer any option for the vehicle movements to react to application specific inputs [39]. The proposed architecture consists of the open-source network simulator ns-2 [64] as the central module, simulating the VANET with its nodes and messages; the VISSIM traffic simulator [65] performing the movements of the vehicles; and the Matlab/Simulink [62] environment as an application level simulator [39]. Since VISSIM runs on the MS Windows operating system and the other parts on Unix systems, the authors developed a cross-platform communication tool. They discuss the cost of the message based control of the traffic simulation which is also used within the framework used in this thesis (cf. Chapter 4). Finally, the proposed architecture is tested on the example application of broadcasting emergency warning messages within a VANET [39].

In 2008 Wang and Lin published version 5.0 of NCTUns [63], an open-source network simulator, which introduced the functionality of simulating vehicular communication according to the IEEE 802.11p standard [70]. NCTUns runs on Linux and makes use of the real-life TCP/IP protocol stack of the kernel to simulate the messages. It ships with a GUI and, while the previous release could already simulate road networks, realistic vehicle movement, and V2I communication by defining RSUs, the 5.0 release adds the possibility of adding on-board units (OBUs) to vehicles. This allows to add VANETs directly into the simulation and have the vehicles react to events from within the VANET [70]. In 2011 NCTUns was renamed EstiNet [61] which is commercially distributed and updated to date and has much higher functionality.

2.3.6 Similar Studies

In addition to the aforementioned literature there are publications that are even closer related to our analysis. This mainly means that the problem setting or the simulation setup are very similar to ours. We will present them in more detail in this section and emphasize differences to our work.

In 2011 and 2012 Kokolaki et al. published a very similar study regarding the topic of this thesis in two papers which are titled "Value of information exposed" [33] and "Opportunistically assisted parking service discovery" [36]. Therein they investigate three simple dissemination methods for parking spot availability data in VANETs and conduct exhaustive simulations to compare their performance.

- As a reference method they use the same method we call *naïve* and name it ***non-assisted parking search (NAPS)***. Upon reaching an initial parking search radius around the real destination the vehicle starts targeting random destinations within that radius. When it encounters a free parking spot it parks. The search radius is increased with time.
- What we call the *distributed* approach Kokolaki et al. call ***opportunistically-assisted parking search (OAPS)***. They argue that vehicles communicate with stationary sensors at each parking spot to gather availability information whereas we consider the sensors vehicle-based. However, this differentiation is mainly of theoretical nature and only possibly impacts the method's performance if we consider the detection of parking spots on the opposite side of the road. Vehicles have a local cache, store this data of parking spots they encounter, and share it with other vehicles that are within communication range. When looking for a parking spot, the driver targets the parking spot within his vehicle's cache, that is the closest to the original destination. In doing so, information that is older than a threshold is discarded. This is different to our distributed approach where the ranking (decision making) process is more sophisticated (cf. Section 3.3.2). The authors give no extensive description of the information sharing process used (cf. Section 3.3.1).

- The third approach that includes a central server is called ***centrally-assisted parking search (CAPS)*** and similar to our *global* approach. This omniscient instance knows the status of all parking spots and vehicles can query it. An important difference to our work, however, is that vehicles do not only receive information about free parking spots, but rather only the location of one free parking spot that is closest to their destination. The global authority reserves this parking spot for the assigned vehicle. This is a fundamental difference to our global approach that leaves the decision making to the drivers by only serving as a live database. In [33] the central server handles the requests in a First-Come-First-Served (FCFS) manner. This reservation approach is susceptible to cheaters since it might not be enforceable physically, but only by legislative means. For as long as there is no parking spot available within a vehicle's parking search radius it traverses random points within that area. This is also different to our model of a global approach which does not consider an area of interest but always uses the free parking spot that is closest to the driver's destination.

The authors developed their own simulation environment in C and ran multiple simulations to compare the three paradigms of information dissemination between vehicles [33]. They investigate two scenarios—one where the destinations are uniformly distributed, and one where they are biased towards a hot spot. For the uniformly distributed scenario it can be said that with an increasing number of vehicles and a limited number of parking spots the general competition increases and leads to worse results in terms of time spent looking for a parking spot as well as distance between destination and parking spot for all three approaches. This is intuitive and expected. The distributed OAPS outperforms the naïve NAPS in all regards. The global CAPS manages to achieve shorter parking search times for a higher amount of vehicles (increasing it even further CAPS gets worse than OAPS and NAPS) but at the expense of immensely increased distances between parking spot and destination. The central server obviously reserves parking spots that are sub-optimal and due to vehicles taking a long time reaching them, the requests of other vehicles cannot be satisfied [33].

In the hot spot scenario results are different. As expected, the performance of all three approaches deteriorates. However, CAPS outperforms NAPS and OAPS for all vehicle densities tested significantly in terms of parking search time. The distance between parking spot and destination is worse for CAPS than for NAPS and OAPS, albeit not as much worse as for the uniform scenario. Also, NAPS and OAPS perform more or less the same, with OAPS achieving minimal better parking search time [33].

We will compare the results presented in [33] to ours in more detail in the discussion (Section 5.6).

In two subsequent publications Kokolaki et al. investigate the effect of vehicular node selfishness in the opportunistically-assisted parking search [34, 35]. Therein they

specifically consider two types of misbehavior: information denial and information forgery. Vehicles that belong to the first group do not share information themselves, but only consume information received from other nodes. Vehicles of the second group deliberately share wrong information. This means that they communicate all parking spots within their own area of interest as occupied, thereby driving competitors away from it. The main findings presented in [34] are, that rather counter-intuitively such selfish behavior does not necessarily lead to better results for those showing it. Also, in the first case of vehicles not sharing information with others at all, if the overall vehicle density is not too low, a negative effect on other vehicles cannot be seen. We will analyze the results from [34] in more detail in Section 5.6 as well.

Bessghaier et al. present an "agent-based approach to management of urban parking" [7]. This approach is similar to our distributed approach but not the same. Instead of one, in their work vehicles store two lists about parking spots: one only containing free spots (FS) and one only containing occupied spots (OS) [7]. They then compare the parking spot search time of vehicles that are part of the system to those that are not. The second performance metric the authors present is the number of messages per participant of the network. This is compared to a central approach where one server handles all requests for parking information. The authors argue that in terms of message cost it is therefore beneficial to distribute the work load across multiple nodes. No information about the simulation setup is given [7].

A very fundamental analysis about "Opportunistic resource exchange in inter-vehicle ad-hoc networks" is performed in [74]. In contrast to our work therein the authors prescind from the level of vehicles on roads and rather investigate the phenomenon of information decay within a simple network of nodes that follow straight paths on a two-dimensional pane. These nodes broadcast their location upon leaving their initial position to other nodes within close vicinity and exchange their local stores of such data points when coming in communication range with one another. The authors give a theoretical proof of a *boundary radius* beyond which data points do not leave their origin based on simple parameters and confirm it with simulations. With parameters similar to ours (cf. Section 4.5) in a 50×50 mile grid this radius turns out to be around 0.8 miles [74].

3 Three Approaches to Parking Spot Search

This chapter will explain three fundamentally different approaches to the PSSP that we investigated in more detail. The differences lie within the assumptions of the underlying model and its requirements. This mainly regards the availability of *smart* hardware, including sensors and different communication technologies, and *smart* software for computations.

Nonetheless, all three approaches have the following things in common:

- They take the *individual perspective*. This means the decision where to park is always left to the driver. Any system only assists the driver during the search by suggesting possible candidate parking spots. This avoids the risk of cheaters that do not act according to some implicit set of rules. One could say that the ultimate decision where to park is always made independently and in a decentralized manner.
- They only consider curb parking. In all simulations run and explained in Chapters 4 and 5 the possibility of designated parking lots is neglected. However, the addition of such would not break any approaches, but could rather produce different results in performance. The investigation of such scenarios in combination with the proposed approaches is therefore left open for further research.
- In all cases drivers have a specific destination within a road network. While going towards this destination the vehicles are in the state *driving*. Upon reaching a certain threshold distance $d_{lfpInit}$ from the destination, a vehicle switches state to *lookingForParking (lfp)*. From this moment on it will park at any free parking spot it encounters on its side of the road. While in the state *lookingForParking* a vehicle's maximum speed is lowered to 30 km/h to simulate a driver actively looking for a free spot while being able to abruptly break when he encounters one.

The following point only holds for the distributed and naïve approaches:

- Vehicles start with a personal initial search radius r_{init}. Only parking spots within this radius around the destination are considered relevant. With increasing time spent looking for a free parking spot this search radius increases. Given that the number of available parking spots is larger than the number

of vehicles looking for such spots, this procedure assures that all approaches terminate, i.e. every vehicle finds a free parking spot. It is also an intuitive thing a driver would do. The exact value of the initial search radius as well as the increment function can be defined by driver's preference. The values we used in our simulations are listed in Section 4.5 and the search radius increment function can be seen in Equation (3.1)

$$r = r_{init} + t_{lfp}/60 \cdot r_{init} \tag{3.1}$$

where t_{lfp} is the time spent looking for a parking spot in seconds. Obviously r increases linearly, continuously, and increases by r_{init} every minute. The idea is to simulate realistic driver behavior and the values produced by this function seem reasonable. Naturally, a driver first tries to find a free parking spot close to his destination and only with increasing search time trades off the remaining distance against a higher chance of actually finding such a spot sooner.

3.1 The Naïve Approach

The *naïve* approach shall simulate what most people do today. It involves no smart hardware for parking spot detection or (inter-vehicle) communication and serves as a reference base for the following more sophisticated approaches. Figure 3.1 visualizes a vehicle taking the naive approach as a UML state machine. As soon as the driving distance to the destination is less than the *search initiation distance* $d_{lfpInit}$ the driver starts looking for a free parking spot. Henceforth, the vehicle is considered to be in the state *looking for parking spot (lfp)*. From this state on the following conditions apply:

- If the vehicle encounters a free parking spot on its side of the road, it parks.
- If the driver sees one on the opposite side of the road, he targets it by turning at the next intersection.

Upon reaching such a previously targeted parking spot or the original destination he randomly targets an address within his search radius r around the original destination. This process repeats until the driver manages to park his car. The search radius r, which is used for the random secondary destination generation, is increased constantly using Equation (3.1). As stated above, given that the number of available parking spots is higher than the number of vehicles looking for such spots, this search radius increase assures that the naïve approach terminates at some point, i.e. every vehicle finds a free parking spot.

Algorithm 1 shows the naïve approach in pseudo code.

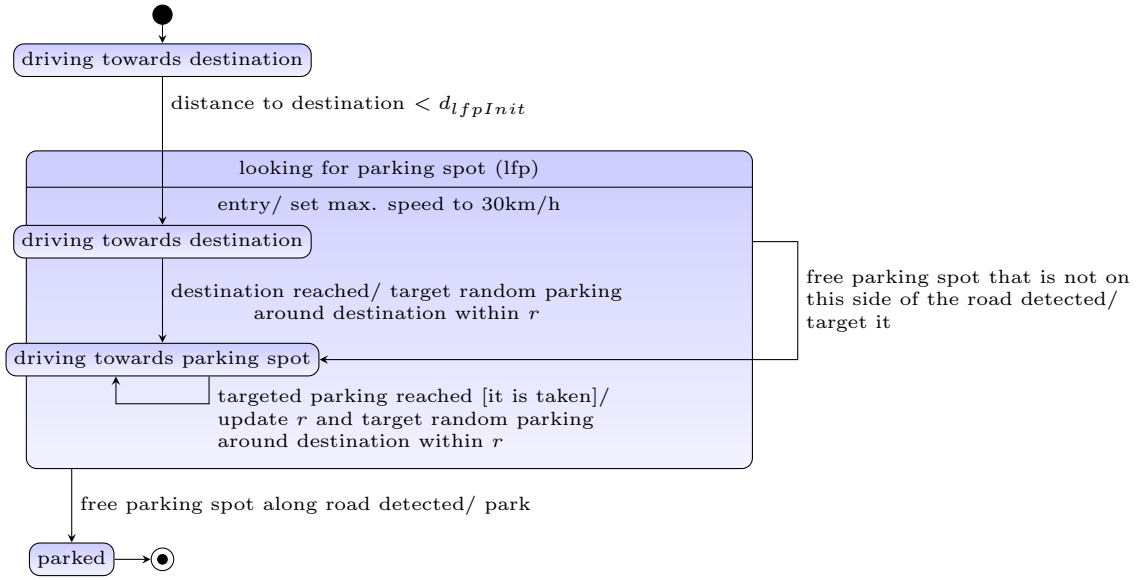


Figure 3.1: UML state machine of a vehicle taking the naïve approach to the PSSP

Algorithm 1: Naïve parking spot search

CurrentDestination : OriginalDestination
r **:** r_{init}
while DistanceToDestination $> d_{lfpInit}$ **do**
  `DriveTowards(CurrentDestination);`
end
`StartLfpTimer();`
MaxSpeed $\leftarrow$ 30 km/h;
while *not* Parked **do**
  if *detected free parking spot on the same side of the road* **then**
    `Park();`
  end
  if *detected free parking spot* p *on the opposite side of the road* **then**
    CurrentDestination $\leftarrow$ p;
  end
  if *reached* CurrentDestination **then**
    $r \leftarrow$ `UpdateSearchRadius(LfpTimer);`
    CurrentDestination $\leftarrow$ `GenerateRandomPointAround(OriginalDestination,` r`);`
  end
  `DriveTowards(CurrentDestination);`
end

3.2 The Global Approach

The scenario in which all vehicles within the road network can communicate with some global instance shall henceforth be called the *global* approach. This global instance which offers the possibility to exchange information over large distances throughout the whole network could be considered a server on the internet. Therefore, taking the global approach requires the availability of such a server and long-range communication hardware. Cellular networks which are already highly available in urban areas could well be used for the purpose of communication of parking spot availability data. However, the strongest requirement of this approach is the presence of stationary sensors that detect if a parking spot is available or not. There are two main possibilities of how such sensors could be implemented: (i) as (underground) sensors at every single parking spot [76, 79], or (ii) as overground sensors that could be mounted on lamp posts and possibly monitor a group of parking spots [16, 26]. Such sensors are often based on magnetic field changes, radar, or visual detection (cf. Section 1.2).

Figure 3.2 visualizes a vehicle taking the global approach as a UML state machine. The event that initiates the parking spot search is the same as for the naïve approach. Upon reaching the destination by a driving distance of $d_{lfpInit}$ the driver starts looking for a parking spot; he enters the state *lfp*. From now on the following same conditions as in the naïve approach apply:

- If the vehicle encounters a free parking spot on its side of the road, it parks.
- If the driver sees one on the opposite side of the road, he targets it by turning at the next intersection.

In addition to the naïve approach, upon initiating the search the vehicle automatically requests the *best* candidate parking spot from the server using long-range communication. In doing so it communicates its destination to the server. The *best* available parking spot is the one that is closest to the driver's destination. Its location is reported back to the driver who targets it and is now registered as a prospect for this parking spot. Should its availability change, the server notifies all vehicles currently targeting it and supplies them with new *best* candidates. Note that in comparison to [33] the server in our model does not reserve any parking spots for specific vehicles and neither performs any optimization as to schedule all vehicles based on their interests. It merely serves as a live database for free parking spots. Thereby all participating vehicles have the same chances and the possibility of aforementioned *cheaters* who may not even use the service is accounted for. However, the server also does not alert vehicles if a better candidate has become available.

Algorithm 2 shows the global approach in pseudo code.

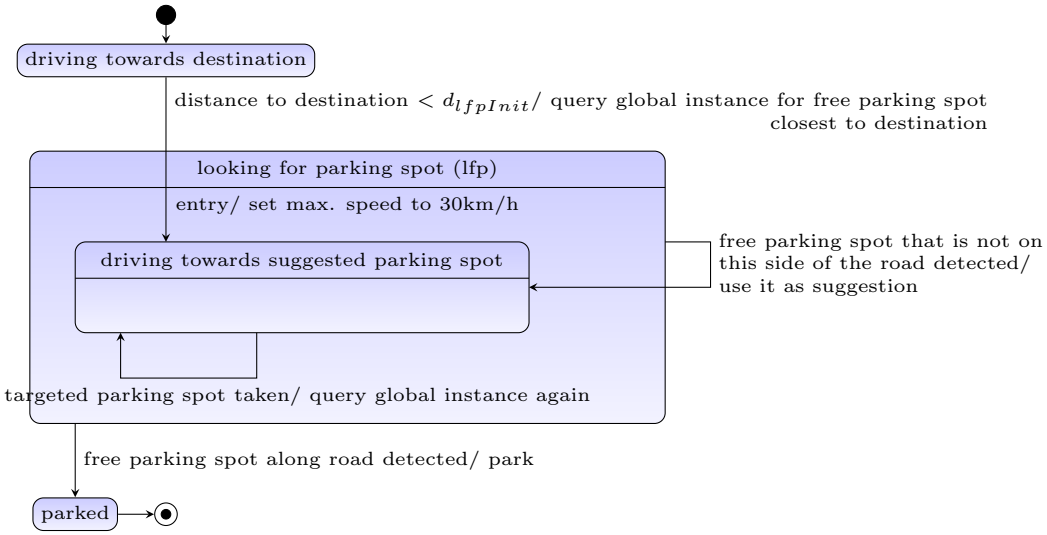


Figure 3.2: UML state machine of a vehicle taking the global approach to the PSSP

Algorithm 2: Parking spot search with a global instance

CurrentDestination : OriginalDestination

while DistanceToDestination > $d_{lfpInit}$ **do**
  `DriveTowards(CurrentDestination);`
end
MaxSpeed ← 30 km/h;
CurrentDestination ← `QueryServer(OriginalDestination);`
while *not* Parked **do**
  if *detected free parking spot on the same side of the road* **then**
    `Park();`
  end
  if *detected free parking spot* p *on the opposite side of the road* **then**
    CurrentDestination ← p;
  end
  if *reached* CurrentDestination **or** *received notification that* CurrentDestination *is taken* **then**
    CurrentDestination ← `QueryServer(OriginalDestination);`
  end
  `DriveTowards(CurrentDestination);`
end

3.3 The Fully Distributed Approach

The third approach to parking spot search we investigate is a fully distributed one which means that there is no long-range communication between vehicles—or more precisely within the VANET—but only between vehicles within a certain neighborhood. The size of this neighborhood depends on the communication radius of the on-board short-range radio communication hardware and its protocols. We assume that this communication radius is around 100 m at most, which is a reasonable value [31]. The exact values used in our simulations are stated in Section 4.5.

The second most important aspect of this scenario after its limited communication range is that sensors for the detection of free parking spots are purely vehicle-based. As mentioned in Section 2.3.1 such sensors are being developed and have already been tested [42]. In [42] these sensors are based on ultrasonic sound. Unfortunately, the detection of free parking spots by passing vehicles has the drawback that it is not faultless. So far field tests have achieved an accuracy of 95 % [42]. However, in this thesis we ignore this fact knowingly as we focus on the performance of the fully distributed approach in an ideal scenario. We assume a detection accuracy of 100 %. The other inherent disadvantage of communicating such availability data from vehicle to vehicle after detection is that any information is always outdated. A parking spot that was detected to be free is not necessarily still available by the time a vehicle receives the information. This is not to be confused with the problem that it might not be available anymore by the time the vehicle would reach it. Therefore the probability for every information to be valid is closely related to its age. Section 3.3.2 describes in more detail how vehicles handle this problem in our fully distributed scenario.

Every vehicle is equipped with sensors that constantly scan its side of the road for free parking spots. Therefore it needs a map containing the positions of legal parking spots beforehand (cf. Section 2.3.2 and [17]). For simplicity these parking spots are unanimously encoded by an ID. This should not be considered a requirement since ultimately they can be differentiated by their position. Additional hardware requirements are access to a location service like GPS, a local memory for storing parking spot availability data, and short-range communication hardware.

Figure 3.4 visualizes a vehicle taking the fully distributed approach as a UML state machine. During the whole time the vehicle is active its sensors scan the environment for the availability of legal parking spots (according to the official map). If it detects a free parking spot, an information tuple containing its ID, its position, and the timestamp of its detection (cf. Figure 3.3) is stored in local memory. If it detects a parking spot to be occupied, it removes any possible entry from the local memory.

This local memory is defined as a sorted set which can be used like a queue while adding new entries. It has a maximum size q. The exact values used in our simulations are stated in Section 4.5. Therefore any new detection is always simply added to

ID (string)	time of detection (timestamp)	position (double[2])

(a)

1/1to1/2_6	1510667613	(13.524, 24.213)

(b)

Figure 3.3: (a) Schema of an information tuple describing the detection of a free parking spot including the data types. (b) An example tuple.

one end while the oldest entry is dropped if the queue would overflow. Additionally, a vehicle constantly communicates with its neighbors. That is, it creates wireless links to other vehicles within its communication radius and exchanges parking spot information. Section 3.3.1 explains this process in detail. Like in both previously presented approaches, as soon as the remaining distance towards the destination becomes less than $d_{lfpInit}$, the vehicle enters the state *lfp*, which means the driver starts looking for a parking spot. Thus, again he parks at any free parking spot he encounters and targets free parking spots he sees on the opposite side of the road.

When starting the search, the on-board system checks the local memory for information about a free parking spot within the initial search radius r_{init} around the driver's destination. If there are multiple candidates their ranking is based on a certain logic that uses the age of the information tuples and the parking spot positions. This ranking process is described in more detail in Section 3.3.2. The system only supplies the driver with the best option which he targets. If there is no entry of a relevant parking spot in the local memory, the driver targets random points within his search radius (he behaves like a driver that follows the naïve approach) while continuously checking the vehicle's memory for new information. The targeting of the suggested parking spot only stops if its entry is dropped from the local memory due to a merge. In such a case the local system is queried again for the best candidate parking spot. The detection of a free parking spot would result in an immediate stop occupying it. If the targeted spot is detected to be in use when reaching it, its information tuple is deleted and the system queried again. Like in the naïve approach a vehicle constantly increases its search radius according to Equation (3.1).

Algorithm 3 shows the distributed approach in pseudo code.

3.3.1 Data Merging

When a vehicle enters the neighborhood (i.e. communication radius) of another vehicle these vehicles merge their local memories. This happens once when they establish a connection and does not repeat while this wireless connection is established.

Algorithm 3: Fully distributed cooperative parking spot search

```
CurrentDestination : OriginalDestination
r                  : r_init
while DistanceToDestination > d_lfpInit do
    if vehicle enters communication radius then
        MergeMemories()
    end
    if parking spot p detected then
        if p is free then
            Save(p_ID, CurrentTime, CurrentPosition);
        else
            remove possible entry p from memory;
        end
    end
    DriveTowards(CurrentDestination);
end
MaxSpeed ← 30 km/h;
CurrentDestination ← GetBestLocalCandidateOrRandom(OriginalDestination, r);
while not Parked do
    if detected free parking spot on the same side of the road then
        Park();
    end
    if parking spot p detected then
        if p is free then
            CurrentDestination ← Save(p_ID, CurrentTime, CurrentPosition);
        else
            remove possible entry p from memory;
        end
    end
    if vehicle enters communication radius then
        temporarily remove CurrentDestination from local memory;
        MergeMemories();
    end
    if reached CurrentDestination or not CurrentDestination ∈ LocalMemory then
        r ← UpdateSearchRadius(LfpTimer);
        CurrentDestination ← GetBestLocalCandidateOrRandom(OriginalDestination,
         r);
    end
    DriveTowards(CurrentDestination);
end
```

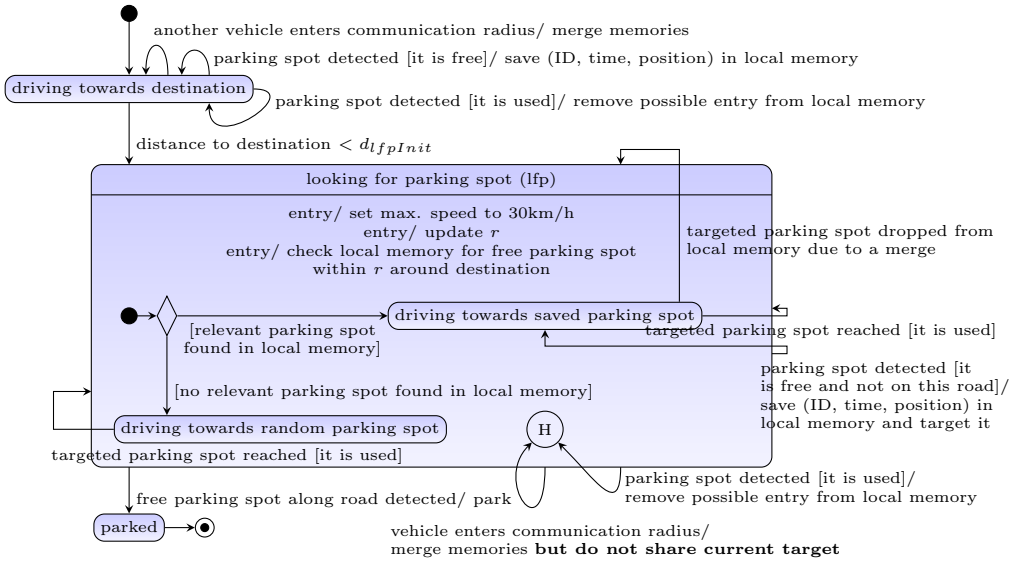


Figure 3.4: UML state machine of a vehicle taking the fully distributed approach to the PSSP

The merging process keeps the q newest information tuples of both original queues. In doing so the *set* property has to be kept intact meaning every parking spot is only stored once. Its newest detection timestamp is kept. Given that both vehicles have the same q the resulting local memories are the same.

It is important to note, that a vehicle in the state *lfp* does not share the information about its current target. However, the effect of this behavior is likely not very high if there is at least one other vehicle around sharing the specific information tuple [34].

3.3.2 Candidate Ranking

Let Q be the set of all parking spots stored in a vehicle's local memory. Then the set R of relevant parking spots is defined as

$$R \coloneqq \{p \in Q \mid d_{pd} \leq r\}, \tag{3.2}$$

where d_{pd} is the distance from p to the original destination and r is the current parking search radius. It is possible that R contains (tuples for) multiple relevant parking spots. In such a case the on-board system has to choose the *best* candidate to report to the driver. In Section 2.3.3 we presented and discussed multiple approaches how this problem has been tackled before. An intuitive idea is to value older information less [13]. However, one cannot only consider the probability of a parking spot still being free at the current moment but at the time the vehicle would arrive at its

location. Therefore, as already mentioned in [19], the authors of [74] present the relevance function

$$F(p) = -\alpha \times t - \beta \times d, \qquad \alpha, \beta \geq 0, \tag{3.3}$$

where p is the parking spot candidate, t is the age of p, d is the distance from the location of p, and α and β are non-negative constants that represent the relative importance of time and distance [74].

In this thesis we use the following slightly different relevance function

$$F(p) = -t - T, \qquad p \in R \tag{3.4}$$

for the parking spot candidate p, where t is the age of p (i.e. the time since its detection stored in memory), T is the expected travel time from the current position to p and R is the set of relevant parking spot candidates defined in Equation (3.2). With respect to the age t this is very similar to the relevance function given in [74] (Equation (3.3)) and just setting α to 1. However, in contrast to [74], as the constraint states, we use a hard cut to determine whether a parking spot within the memory is relevant at all which depends on the vehicle's current parking search radius. The function itself is then only used to rank relevant parking spot tuples. It does so by estimating the travel time from the current vehicle position to the respective candidate and adding it to the age of the information. This is slightly more complex than just using the distance to the parking spot itself. The benefit of this transformation of the distance to time is that it is easily possible to extend the model by incorporating traffic density or traffic light information into the travel time calculation. It is also more practical to use travel time as a factor instead of distance to properly handle cases where the shortest route is not the quickest to drive. However, we do not implement factors like α and β to put any individual bias on the distance to p or the age of the information t. After calculating the relevance for all parking spot candidates we impose another hard cut as we discard all candidates whose information would be *too old* upon reaching them. The best, and hence proposed, parking spot candidate is the one that has the highest relevance value $F(p)$ while still fulfilling

$$|F(p)| \leq maxAge \tag{3.5}$$

with $maxAge$ being the user-defined threshold which could be individual according to each driver's preference.

3.3.3 The Advanced Fully Distributed Approach

In addition to the previously described fully distributed approach we analyze a variant that we shall call the *advanced distributed approach*. The only difference to the standard version is that vehicles do not only store information about free parking spots but instead also record information about parking spots that are detected to be occupied. The idea behind this modification is that a possible downside of the standard version is that outdated information is only slowly removed from the local memories (only when new information about free parking spots is acquired) and the only mechanism to avoid its usage is the $maxAge$ parameter. This means the standard approach has no active mechanism for *information decay* [74]. Hence, by storing information about used parking spots any outdated information tuples listing them as free is dropped during merges. However, since the amount of occupied parking spots is usually inherently larger than that of free parking spots, this means that most of the on-board storage is used for information about occupied spots. Therefore larger local memories are needed which in turn leads to higher message cost during merges (cf. Sections 5.4 and 5.5).

4 Simulation Environment

The three approaches to parking spot search presented in Chapter 3 are analyzed through extensive simulations. This chapter will explain the nature of these simulations, underlying assumptions, used programs, and the way they are initialized, while Chapter 5 will present and discuss the results.

4.1 Basic Model

In various publications discussed in Chapter 2 the authors simulated their problem settings to calculate results for their approaches. The wide range of simulation setups used shows the difficulty of finding the *perfect* tool for the specific task one is interested in. Section 2.3.5 gave a short overview of simulation setups that have been used in similar studies. While many authors developed their own simulators [10, 19, 33, 69], we take the same road as the authors of [3] and use the microscopic traffic simulator SUMO (Simulation of Urban MObility) [37] at the core of our simulations. However, we use basic python scripts that communicate with SUMO through the TraCI (Traffic Control Interface) [71] API to implement the logic. Figure 4.1 shows a screenshot of SUMO's GUI version running one of our simulations. The possibility to run it with a graphical user interface along with our python script in the background makes it a very convenient tool for such an application as this makes it easier to follow the actions and decisions of actors within the simulation.

SUMO is open source and freely available online[1]. It is a completely microscopic simulator meaning that it simulates every single vehicle by itself with individual attributes such as speed, maximum speed, position, and an independent route. It uses the *continuous model* which means all distances, especially the space on roads, are considered to be continuous while only time is discretized to seconds. At every time step the position of each vehicle within the next time step is calculated based on its current position, speed, acceleration/deceleration, route, and exterior limiting factors such as other vehicles or traffic lights. A SUMO simulation requires three types of input files: vehicle definitions, route definitions (usually included in the vehicle definitions), and a network definition. All these input and SUMO's native output files are in XML format. The combination of vehicles and routes is called *demand*. Every vehicle has a clearly defined starting position on the first edge of its

[1] `http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/`

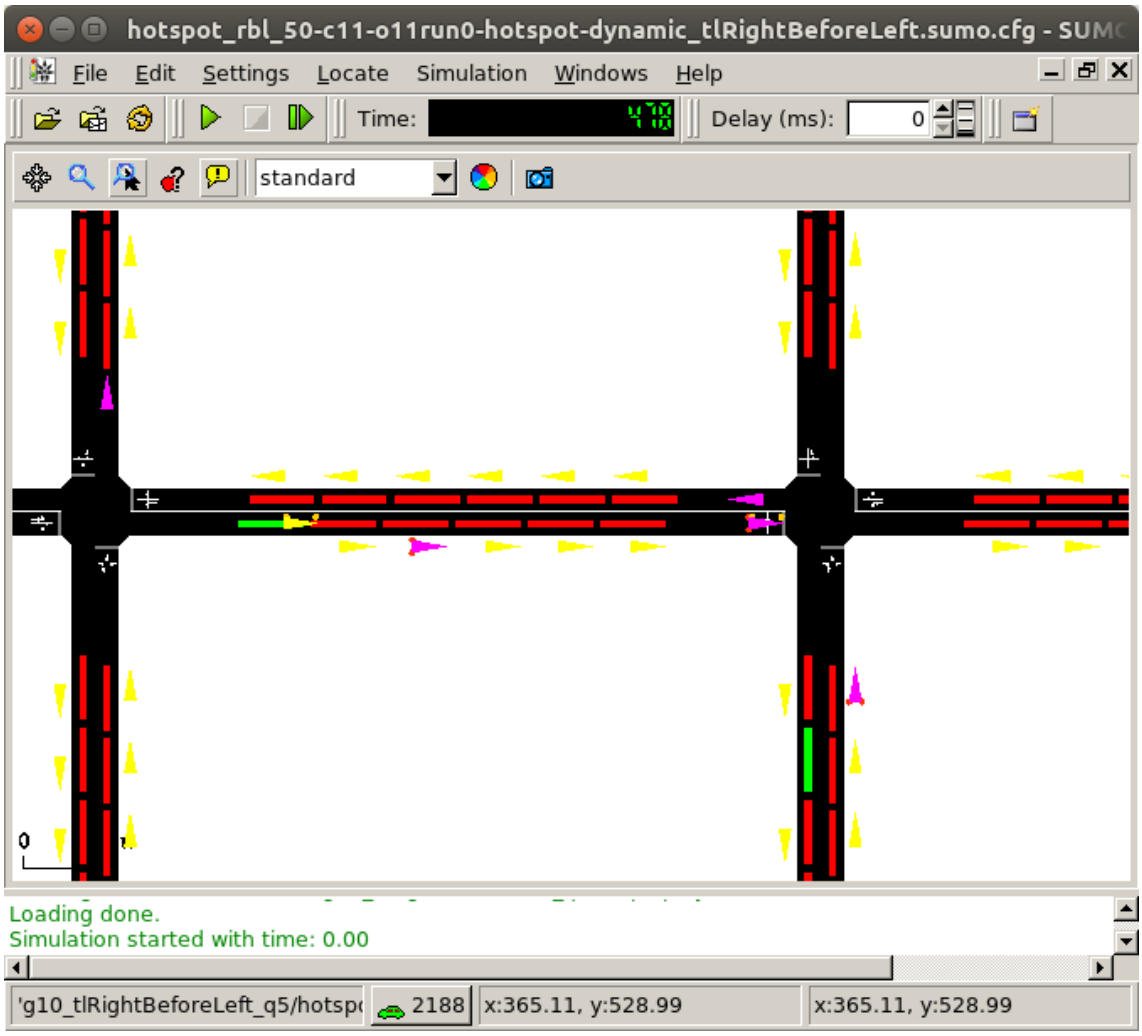


Figure 4.1: Screenshot of the SUMO-GUI. Vehicles are displayed as triangles. Purple vehicles have already switched to the state *lfp*. Parked vehicles are shown next to the road. Red boxes indicate occupied parking spots. Green boxes indicate available parking spots.

route. The route is an ordered list of edges and the end position on the last edge is defined like the starting position. While it is possible to insert vehicles together with their routes into a simulation dynamically through TraCI, in general all demand is defined before SUMO starts the simulation. This means in the usual application a SUMO simulation is deterministic. Since our external steering through TraCI according to the logic explained in Chapter 3 contains randomness, our simulations are not deterministic.

Although SUMO and its accompanied tools support various network import formats (e.g. from OpenStreetMap[2]) for simplicity we generate generic grid road networks with the following specifications:

- 10×10 junctions
- 100 m between junction centers (leads to an actual road length of 90.5 m between junctions)
- only single-lane per direction roads
- maximum speed of 50 km/h
- 6 parking spots per curb (2160 total)
- junction type: right before left / priority to the right

We started off using junctions that were regulated by traffic lights with differing intervals of 45 and 90 seconds but these tests turned out to be difficult to interpret and posed a higher threat of gridlock, which of course always depends on the active vehicle density. The impact of the traffic lights on the parking spot search performance was also hard to determine which is why we use right before left junctions in the experiments presented in this thesis. Kokolaki et al. used roundabouts [36]. Such aspects have to be considered when defining vehicle demand (see Section 4.2).

4.2 Demand Generation

We generate the vehicle demand for our simulations in the following way. There are two initialization parameters: *initial occupancy* and *active vehicle density*. The initial occupancy describes how many parking spots are occupied at the start of the simulation. Depending on the density control mechanism this occupancy remains the same throughout the whole simulation. Referring to the number of parking spots available is obviously equivalent. The active vehicle density constitutes the amount of vehicles that are active (i.e. driving towards their destination) at the beginning of a simulation run v_{init}. Again, depending on the density control mechanism, it stays the same throughout the whole simulation. The starting positions and destinations of all vehicles are randomly distributed according to distributions given below.

[2] `http://www.openstreetmap.org`

Throughout a simulation run the total number of vehicles does not change, only the number of active vehicles fluctuates. SUMO loads all vehicles and their routes prior to the actual simulation.

There are two types of randomness in the simulations. One lies in the static demand generation with the routes and the order in which vehicles activate and the other is dynamic and inherent to the approaches simulated (i.e. random target selection if no information about a free parking spot is available, see Chapter 3). While obviously the dynamic randomness is different between the different approaches, the random routes prior to the simulation and the order in which vehicles get active is the same for all of them. This means that the results presented for all three approaches are based on the same demand definitions.

There are two types of scenarios considering the random vehicle routes:

uniformly distributed Routes are uniformly distributed over the whole grid. When determining the order in which vehicles leave their initial parking spot they are chosen randomly from the whole network. For every vehicle a target position on a road that is at least s meters air-line distance away from the starting point is randomly chosen from the whole network.

hot spot The order in which vehicles leave their initial parking spot is chosen randomly from all vehicles like in the uniformly distributed scenario. However, in the target point generation there is a bias towards the center of the network. The center is defined as a square with side length 270 m mapped around the center point of the grid network (see Figure 4.2). With a probability of 20 % vehicles from the outside will target a point in the center and map it onto a road. With a probability of 80 % they will target a random point that is in the outskirts. Destinations for vehicles that start in the center are generated in the same way as for vehicles in the uniformly distributed scenario. All road positions are sampled uniformly to generate the destinations. In all cases the randomly generated target point has to be at least s meters away from the starting point (note that the actual target point for vehicles starting in the outskirts could be closer to the respective starting point than s after the mapping onto a road). Since the area outside of the center is much larger this leads to sufficiently high competition for free parking spots in the center.

In both scearios s is defined the same way as the center square side length in the hot spot scenario

$$s = \text{netwidth} \times 0.3 = 900 \times 0.3 = 270.$$

In all cases the shortest route calculation is performed using Dijkstra's algorithm [21].

For both scenarios we tried two vehicle density control mechanisms:

constant density The vehicle density is kept constant by starting a parking vehicle every time an active vehicle parks. The density is defined by the number of

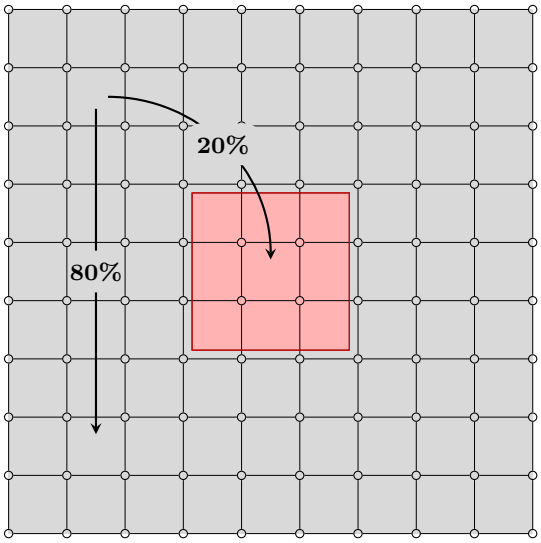


Figure 4.2: The route generation for the hot spot demand scenario. Any points within the red square are considered to be in the *center*. Vehicles starting their route outside of the center have a bias of targeting a random point in the center. They do so with a probability of 20 % while targeting a point outside with a probability of 80 %. Vehicles starting in the center randomly select a target point from the whole network. All routes have a minimum 2D-distance between start and end point of 270 m.

initially active vehicles v_{init}. This number has to be fine-tuned to avoid both grid lock and the impossibility of vehicular communication. Constant vehicle density implies constant free parking spot density. In reality the vehicle density obviously fluctuates depending on the area and time of the day. However, for the same urban areas this active vehicle density is likely similar during the same periods of a week day. Therefore, by changing this parameter we can simulate different real world settings while being able to generate sufficient data for each of them.

constant departure interval In a static interval a parking vehicle is activated. This leads to changing vehicle densities throughout a single simulation run. A problem is that the v_{init} initially active vehicles all reach their destination at around the same time while new vehicles are activated constantly. More importantly, this leads to different vehicle densities for the different approaches that are being compared. While the idea behind such a control mechanism is to penalize approaches that do not manage to keep the traffic fluent, it is not really suitable to compare them. In one case all vehicles might spread through the network efficiently, in another gridlock could occur and freeze the simulation. Our results show that as a consequence of these effects it is not consistent to compare the different approaches in a fair manner using this density control mechanism. We therefore will not discuss it any further.

v_{init}	occupancy		# free parking spots	# vehicles total
uniformly distributed				
20-100	0.99		22	$2138 + v_{init}$
hot spot				
	center	outside		
20-100	0.93	0.995	22	$2138 + v_{init}$
20-100	0.90	0.990	36	$2124 + v_{init}$
20-100	0.80	0.990	50	$2110 + v_{init}$

Table 4.1: Demand definition parameters for both scenarios. The specific values of v_{init} were 20, 25, 30, 40, 50, 60, 70, 80, 90 and 100.

As mentioned before in both cases the order in which vehicles depart is the same for a specific demand definition for all approaches compared. Also, in all cases the simulation ends when all vehicles have been activated once which means that there are exactly v_{init} vehicles remaining. Thereby all vehicles that contribute to the final analysis experience the same network conditions (average number of communication partners, average number of free parking spots available).

To overall reduce the possible effects of randomness (in the route generation as well as the dynamic random target generation inherent to the naïve and distributed approaches) we generate multiple simulation runs and analyze the accumulated data.

Table 4.1 lists all demand scenarios and their respective initial parameters that we analyzed. To quantify the level of competition we can check the ratio between the number of active vehicles $|V| = v_{init}$ and the number of free parking spots $|P|$. Thereby we see, that our simulations of the uniformly distributed scenario cover ratios from 0.91 to 4.55 and for the hot spot scenario from 0.4 to 4.55. As mentioned in the problem statement, situations when $|V| > |P|$ are those of interest.

4.3 Simulation Generator Pipeline

We have developed a range of python scripts that generate SUMO input files according to the aforementioned parameters. While these scripts are in fact all runnable on their own to fulfill a specific task, the *simulationGenerator* combines them into a single pipeline. This section will explain this generator pipeline in general. Figure 4.3 visualizes the *simulation generator pipeline*. The final output is a fully prepared folder containing a python script `startSimulations.py` that can be run without

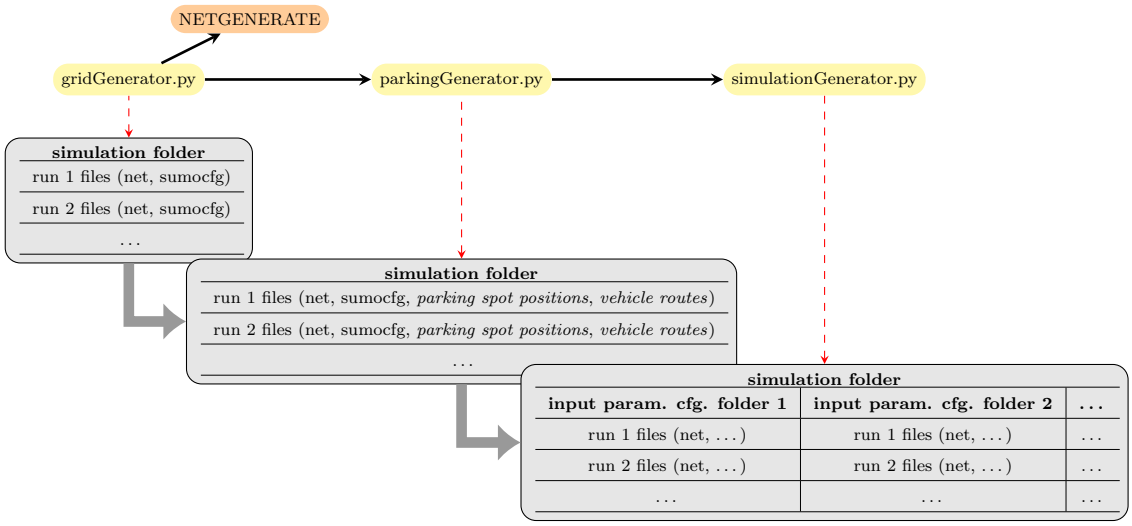


Figure 4.3: Visualization of the simulation generator pipeline.

any parameters to start the whole *simulation pipeline* (see Section 4.4).

First the *simulationGenerator* calls the *gridGenerator* which in turn calls the SUMO tool *NETGENERATE*[3] to create the basic grid network structure. This results in a simulation folder with as many network and SUMO configuration files as the number of runs specified by the user. Note that by default these networks are identical, but the *gridGenerator* provides the option to add even more randomness between individual runs by distributing edge maximum speed limits from a given set of values. However, tests show that this feature leads to gridlock relatively quickly with rising traffic density. This is due to the fact that roads with higher limits are preferred in the route generations but the network infrastructure does not support this obvious bias. Independent of speed limit all roads have one lane per direction and neither right-before-left junctions nor traffic lights with regular intervals for all incoming lanes prioritize any one of them.

Then the *gridGenerator* invokes the *parkingGenerator* which uses the simulation folder from the first pipeline step to add the files needed for the parking logic. Based on the network files it calculates the parking spot positions and writes them to an XML file. It also creates a polygon file that is needed for visualization of the parking spots during graphical inspection when starting SUMO with its GUI. The main function of the *parkingGenerator*, however, is that it generates the vehicle routes for the different demand scenarios described in Section 4.2. In doing so it also writes the vehicle's arrival positions on their target edges and the order in which they depart into files.

As the last step of the simulation generator pipeline the *simulationGenerator* itself performs traffic light adaptions if requested (i.e. it adds the option to simulate and

[3] `http://sumo.dlr.de/wiki/NETGENERATE`

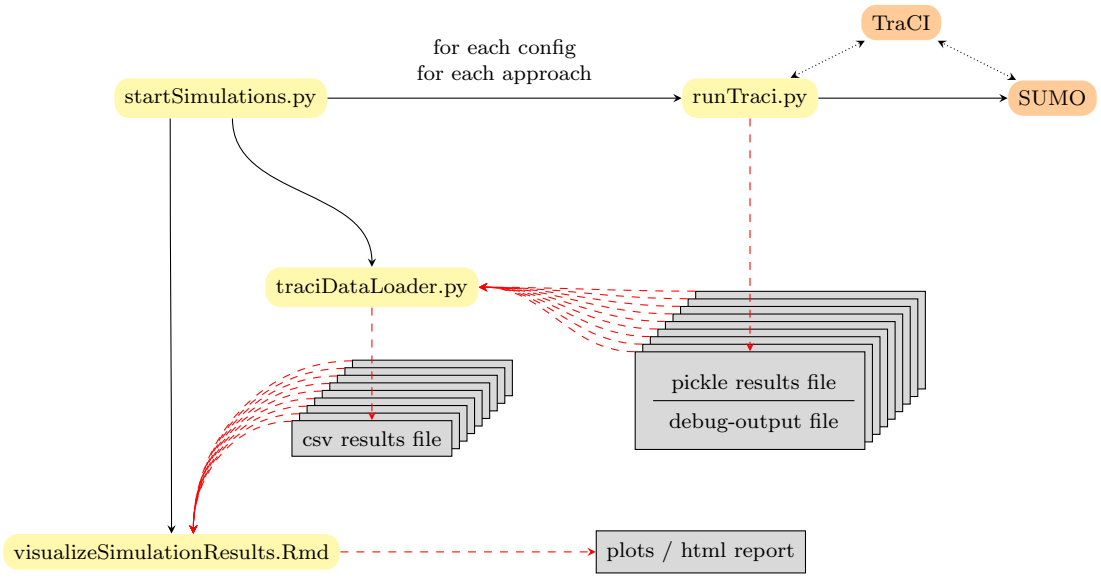


Figure 4.4: Visualization of the simulation pipeline.

compare multiple traffic light intervals for the same simulation setup) and prepares the simulation folder for use. This includes creating sub folders for the different traffic light intervals as well as for the different memory sizes of the distributed approach that are being compared. To avoid recalculation of the global and naïve approaches it then creates dummy result files within these folders. An R-script that generates plots and visualizes the results of the simulations is also copied to the output folder. Finally, the starter script is generated from a template and all parameters set.

4.4 Simulation Pipeline

The previously described simulation generator pipeline outputs a simulation folder. Within this folder the python script `startSimulations.py`, that can be invoked without any parameters, starts the *simulation pipeline*. This simulation pipeline is visualized in Figure 4.4. Its underlying processes will be described in this section.

The start script generates a job list based on the parameters of the simulation folder, checks which of these jobs have already run and generated a result file, and starts the remaining jobs in as many parallel threads as requested. Hence, if aborted, it only loses the data of the currently running jobs and can recover easily. In addition it provides informative status output with an estimate of the remaining calculation time.

One job consists of one call of the python script `runTraci.py`. This script is at the heart of the whole pipeline as it handles the logic of the parking mechanics and the

underlying approaches to information dissemination. Each job starts one instance of SUMO which performs the actual microscopic traffic simulation. The python script then controls SUMO through the TraCI python API which uses TCP/IP sockets. It effectively simulates the application layer of the VANET on top of SUMO. The data that is sent from SUMO to our script consists of vehicle states such as their position, speed, current route, and neighboring other vehicles and parking spots. In turn, the python script sends reroute triggers and stopping signals to SUMO. In all cases, SUMO performs the actual re-routing. When there are less than v_{init} vehicles remaining `runTraci.py` closes the simulation, computes the driving distance between each vehicle's final parking spot and its actual destination, and calculates a few basic statistics for immediate debug output. It then stores the results in a pickle file. By default all output of this script is redirected to a file for further inspection. Depending on user preference this contains only warnings and errors or additional debug information too.

A downside of the socket communication is that it is comparatively slow and should therefore be avoided for performance reasons. Hence, the information retrieval is done in an aggregated form using (context) subscriptions where a predefined set of status variables is retrieved every simulation step. A *context* subscription allows retrieving all objects of interest within a certain radius around the specified object. This is useful to automatically get the vehicles and parking spots around every active vehicle without having to do expensive distance calculations in python.

After all jobs have completed successfully, the *traciDataLoader* loads all pickle files and stores the information in human-readable comma-separated values (CSV) files. This usage of and conversion between two file formats is done for two reasons: First, the result data for each run has multiple types (vehicle specific and simulation specific for the different timesteps) and the pickle format offers a good way to keep them concentrated. Second and more importantly, the final visualization is done in R which cannot read pickle files naturally. The R Markdown script that is responsible for the final output, including visual display in various plots, and statistical analysis is called `visualizeSimulationResults.Rmd`. It loads all single run CSV files, performs some transformations, and saves one comprehensive CSV file for each combination of data type, demand scenario, and vehicle density mechanism. Then it uses the *knitr*[4] R package to generate a comprehensive HTML report of the whole analysis. All plots presented in Chapter 5 were, in their basic form, automatically created by this pipeline.

4.5 Algorithm Parameter Definitions

Table 4.2 summarizes all parameter values for the different approaches that we used in our simulations. This section will explain some of them in more detail.

[4] `https://yihui.name/knitr/`

Some parameters are the same for all three approaches while others are only relevant for particular ones.

search initiation distance ($d_{lfpInit}$) The driving distance left towards the destination at which a vehicle switches to the state *lookingForParking (lfp)*. In this state it will park at any free parking spot it encounters. The reason why $d_{lfpInit}$ is smaller than the parking search radius is that we try to simulate the natural behaviour of a greedy driver who is trying to get as close to his destination as possible before initiating the search. Also, if there is a high availability of free parking spots and $d_{lfpInit}$ is larger, it leads to a greater average distance between the final parking spot and the actual destination.

initial parking search radius (r_{init}) The initial radius around their destinations within which vehicles consider parking spots relevant.

maximum information age ($maxAge$) The maximum age an information tuple may have upon an expected arrival (see Section 3.3.2).

4.6 System

All computations were performed on a PC with the specifications listed in Table 4.3.

Parameter	Value
all	
search initiation distance ($d_{lfpInit}$)	50 m
maximum speed while *lfp*	30 km/h
naïve	
initial parking search radius (r_{init})	100 m
search radius increment function	$r_{init} + t_{lfp}/60 \cdot r_{init}$
global	
best candidate criterion	closest to destination
distributed	
initial parking search radius (r_{init})	100 m
search radius increment function	$r_{init} + t_{lfp}/60 \cdot r_{init}$
communication radius	100 m
merge new data if	partner was not a partner in the last simulation step
$maxAge$	300 s
best candidate criterion	$\arg\max_{p \in R} F(p), \quad \lvert F(p) \rvert < maxAge$
local memory size, in number of information tuples (q)	5, 15, 50

Table 4.2: Algorithm parameters used in our simulations

Component	Value
Operating system	Ubuntu 17.10 64bit
Kernel version	4.13.0-46-generic
CPU	Intel Core i7-6700 3.4GHz × 8
Memory	15.6 GB
Python version	2.7.14
SUMO version	0.32.0
# of parallel runs	8

Table 4.3: System specifications of the platform running the simulation pipeline

5 Simulation Results

We define the following metrics to compare the different approaches to the PSSP and how they perform in the different scenarios:

1. time spent during parking spot search
2. distance driven during parking spot search
3. 2D-distance between final parking spot and actual destination
4. message cost

Additionally, we analyze the effect of the local memory size in the distributed approach by measuring its accuracy. The number of merges and the vehicle density let us compare the effects of the two demand scenarios.

5.1 Time Spent During Parking Spot Search

As stated in Section 2.3 the main quantity one wants to minimize about the PSSP is the time spent during parking spot search. This is the time a vehicle spends in the state *lfp*. Figures 5.1 and 5.2 visualize the averages of these durations for all three approaches in the proposed scenarios. The distributed approaches are shortened to *dist* and *distA* for the advanced variant. The numbers behind these labels indicate the local memory size. The values plotted are the mean values of all vehicles over the respective groups along with 95 %-confidence intervals. Multiple conclusions can be drawn from these plots: First, it is obvious that the global approach clearly outperforms both the naïve and the distributed approaches. Second, with increasing vehicle density the performance of said global approach deteriorates whereas there is no clear trend for the others. Third, the normal distributed approach that only stores information about free parking spots (greenish dashed lines) constantly outperforms its advanced counterpart (dot-dashed lines) that tries to accelerate the effect of information decay by storing information about all parking spots encountered. Fourth and as expected, said advanced distributed approach behaves like the random naïve approach if it is used with a small memory. This is due to the fact that most likely all stored information is about occupied parking spots and, hence, does not help during parking spot search (cf. Section 5.4). Fifth and last, with proper parametrization the distributed approach clearly outperforms the naïve random approach. Interestingly, the parking search times do not differ

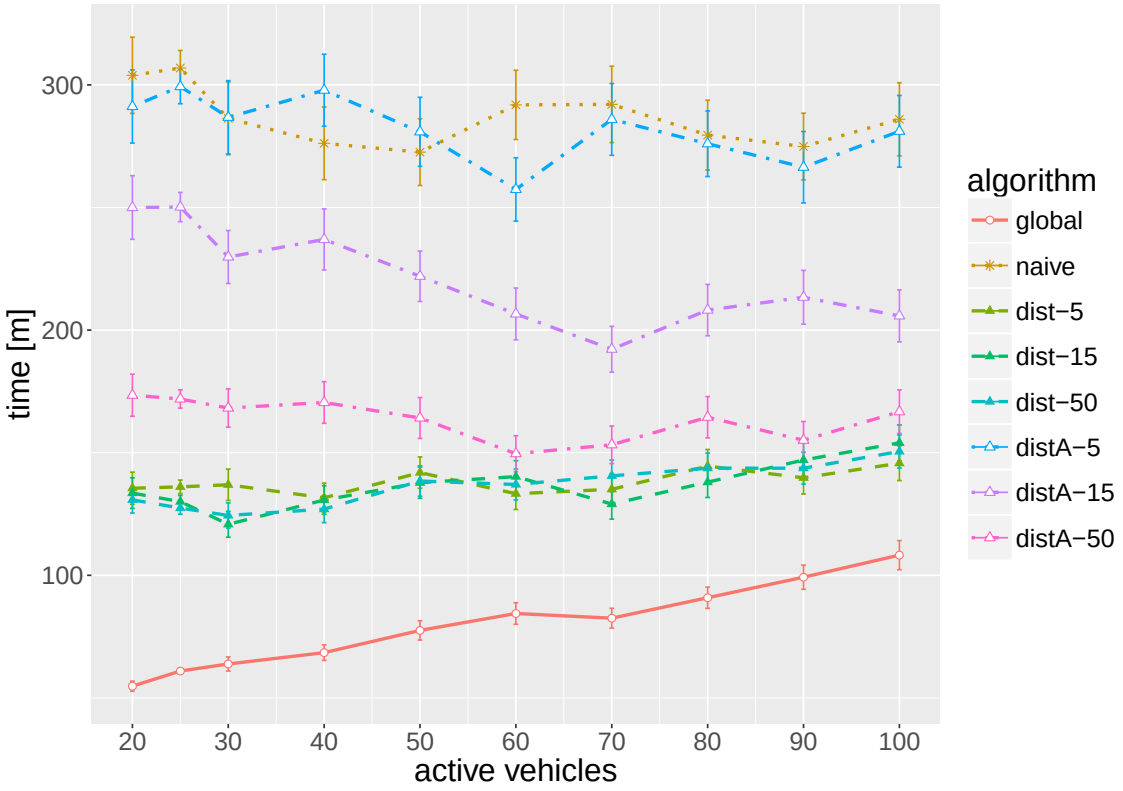
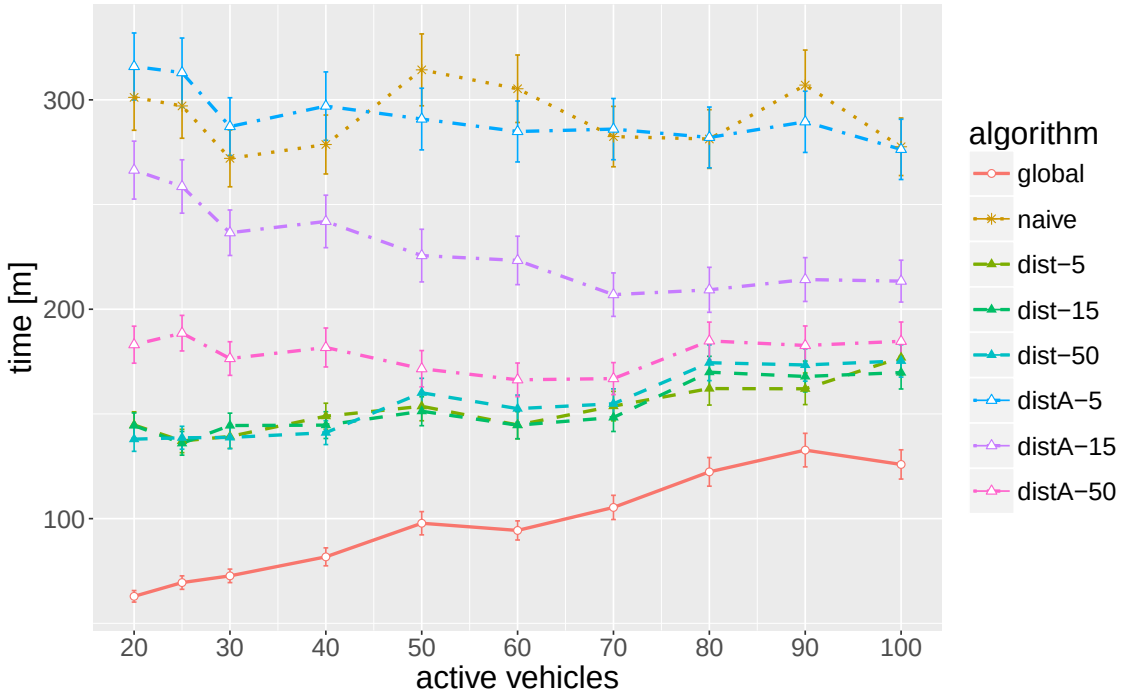


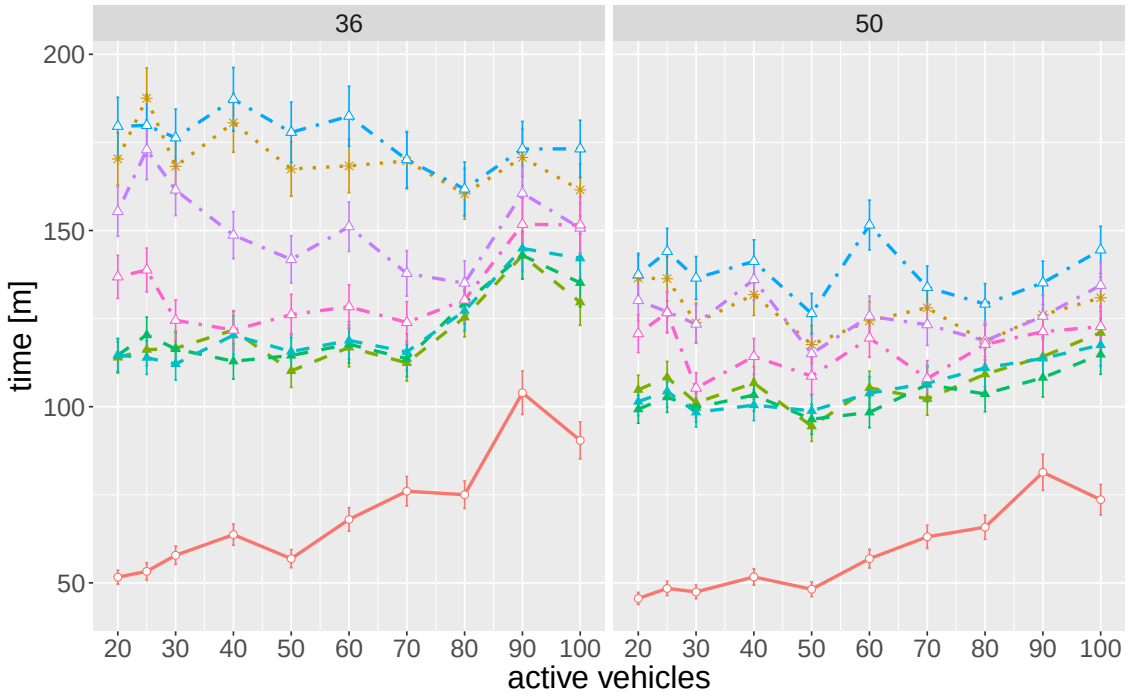
Figure 5.1: Average times spent looking for a parking spot in the uniformly distributed scenario with 22 free spots along with their 95 %-confidence intervals.

significantly between the uniformly distributed and the hot spot scenario given the same number of available parking spots (Figures 5.1 and 5.2a). This might be due to the fact that the majority of the vehicles in the network due to design still does not target the center in the hot spot scenario. Therein, the search times are only slightly higher for the smart algorithms (all but the naïve). However, with an increasing number of active vehicles the performance of the global approach deteriorates faster than in the uniformly distributed scenario. Comparing Figures 5.2a and 5.2b we can see that with a higher number of available parking spots the search times decrease for all algorithms. This is expected. Also, the relative speedup of the global over the distributed approach increases compared to the relative speedup of the global over the naïve approach.

We are plotting mean values and related works have done so too [35, 36], and while the confidence intervals add some value in terms of reliability to these means, it is worth looking at the actual distributions of the data. Let the time spent in the state *lfp* be a continuous random variable X. Figure 5.3 shows the distributions of X exemplarily for the uniformly distributed scenario for selected values of active vehicles as box plots. The same plots for the hot spot scenario look similar which is why we do not show them here. It is obvious that the data has positive skew in



(a) 22 free parking spots



(b) 36 and 50 free parking spots

Figure 5.2: Average times spent looking for a parking spot in the hot spot scenario with different numbers of free parking spots along with their 95 %-confidence intervals. Note the different y-axis scales between (a) and (b).

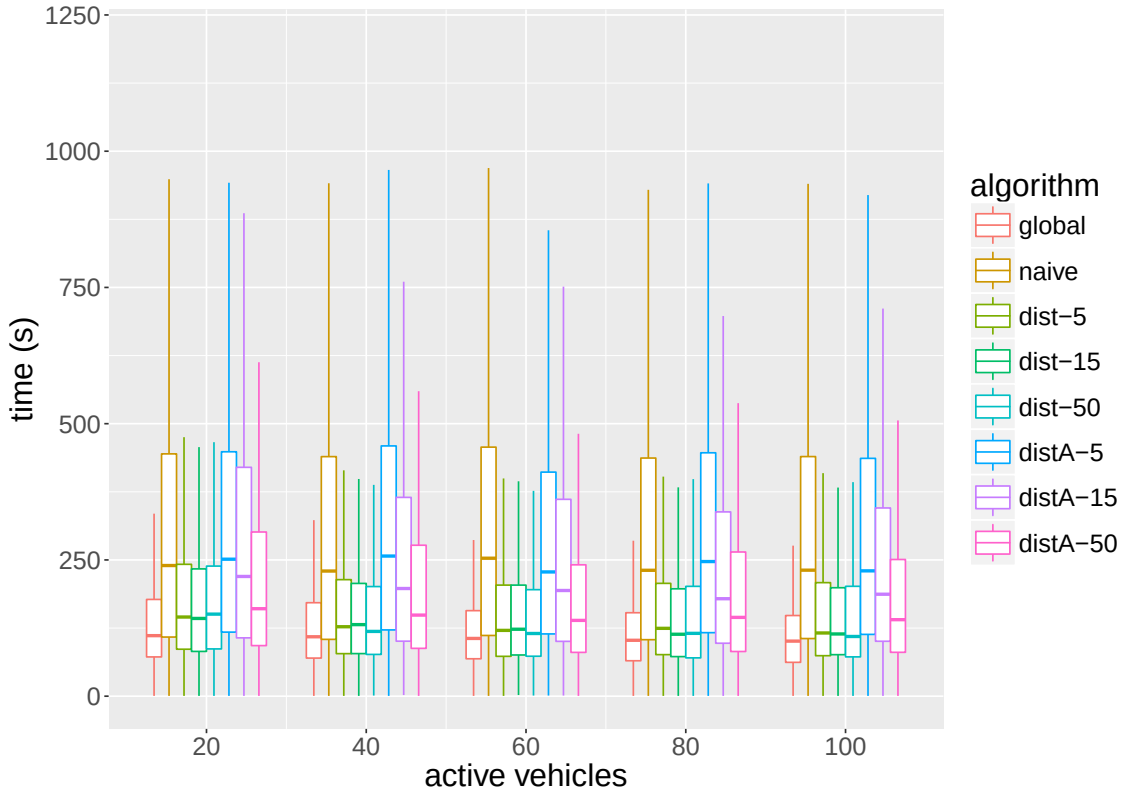


Figure 5.3: Distributions of the time spent looking for a parking spot in the uniformly distributed scenario with 22 free spots. The whiskers of the boxes are defined as $1.5 \times IQR$ from the respective hinge, where IQR is the inter-quartile range, which is the distance between the first and the third quartile. Outliers beyond these whiskers are not shown.

all problem settings. This means most drivers experience reasonable parking spot search times while only a few do not. Since the goal is to reduce the times overall, the mean is a valid metric for comparison. However, since the mean is highly affected by extreme values, improving it means focusing on these few drivers that suffer from long parking spot searches.

5.2 Distance Driven During Parking Spot Search

The effort put into the parking spot search can also be described by the distance driven during the process. In comparison to simply the time this measure is not affected by side-effects that arise from the traffic layer. It is stable in matters of delays through traffic jams or traffic lights. However, these effects do not seem to alter the search times of the different approaches differently as the relative performances, given the distance driven during the search, look similar (Figures 5.4 and 5.5).

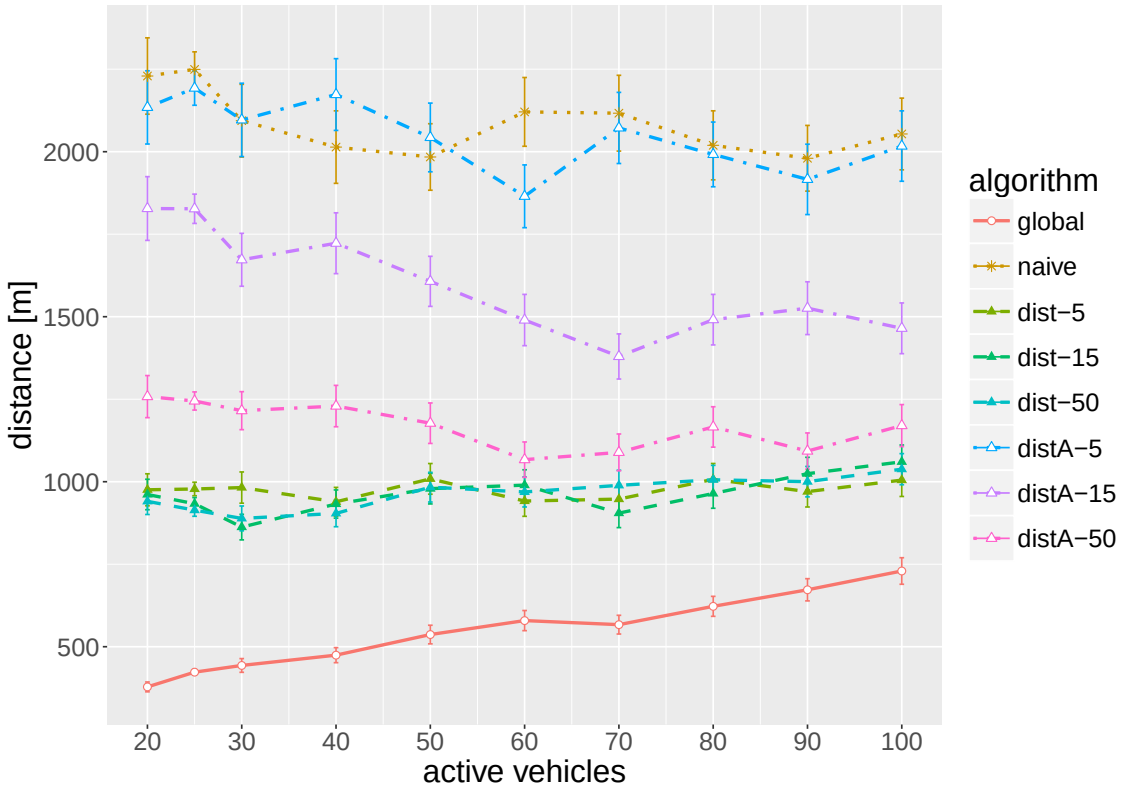
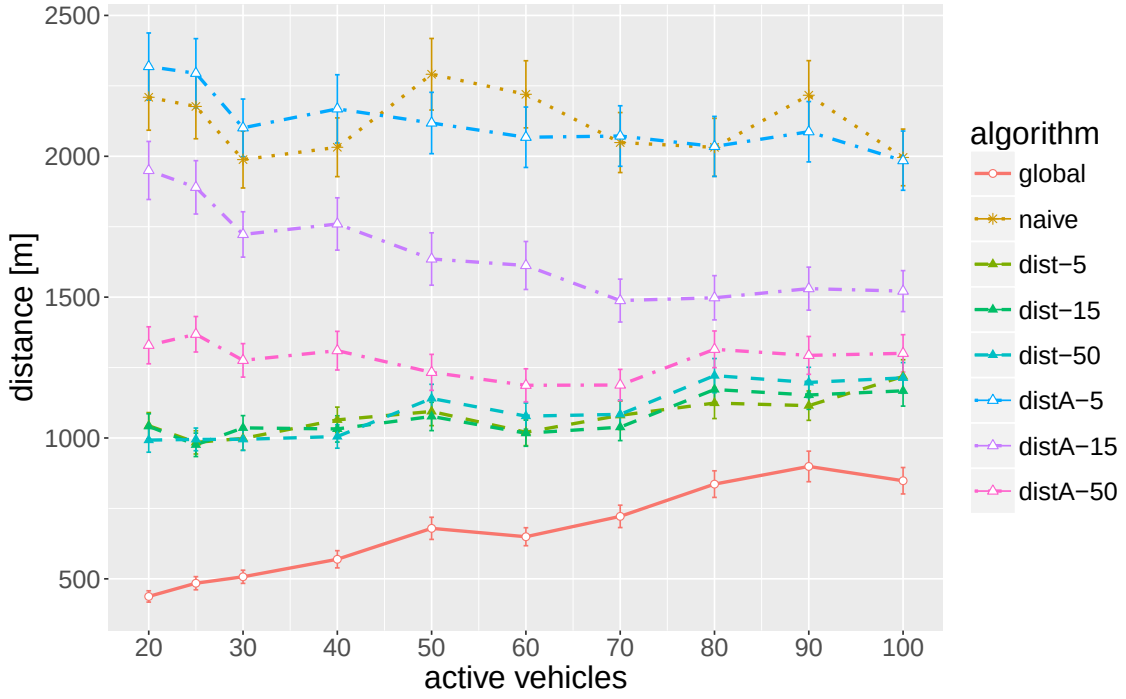
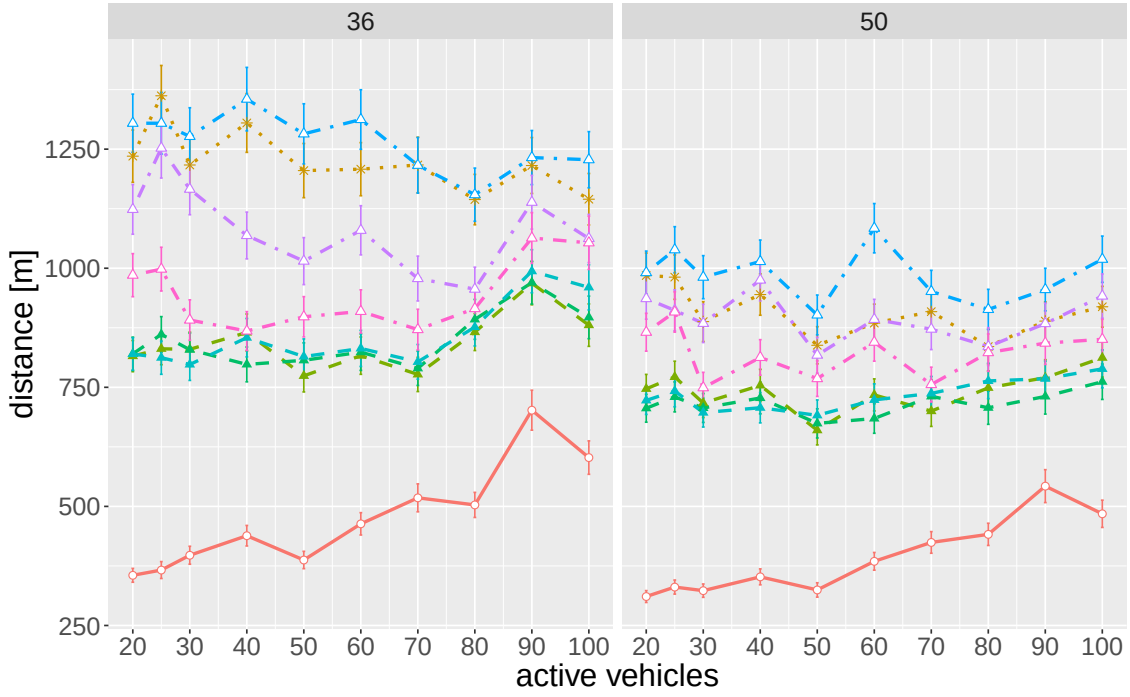


Figure 5.4: Average distances driven looking for a parking spot in the uniformly distributed scenario with 22 free spots along with their 95 %-confidence intervals.



(a) 22 free parking spots



(b) 36 and 50 free parking spots

Figure 5.5: Average distances driven looking for a parking spot in the hot spot scenario with different numbers of free parking spots along with their 95 %-confidence intervals. Note the different y-axis scales between (a) and (b).

5.3 Distance Between Parking Spot and Destination

The second main metric we try to minimize about the PSSP is the distance between the final parking spot and the actual destination of the driver. This distance d_{pd} is the main criterion for parking spot selection in the global approach and factors into the candidate ranking in the distributed approach (cf. Section 3.3.2). Looking at d_{pd} (Figures 5.6 and 5.7) the results regarding relative performance of the three approaches are similar: The global approach constantly outperforms all others, while the distributed approaches outperform the naïve approach (again with the exception of small local memories in the advanced variant). However, we observe an inverse correlation with regard to the number of active vehicles in general—with a growing number of active vehicles d_{pd} decreases. We assume that the reason for this correlation is the fact that with more active vehicles free parking spots are more often created, since vehicles arrive and leave in a quicker interval. This means that with a higher turnover it is more probable that within the parking search area a relevant parking spot is vacated and then detected and its location transmitted to a vehicle on the search.

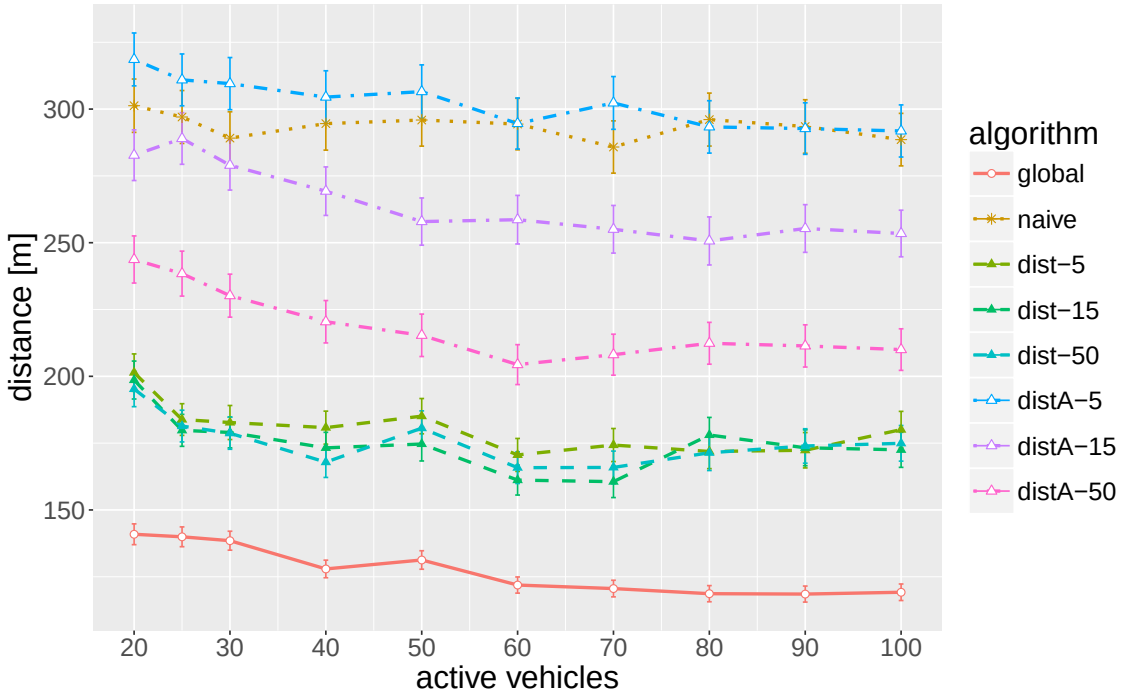
When looking at the average d_{pd} for different numbers of free parking spots in the hot spot scenario (Figures 5.7a and 5.7b), we can see a similar effect as with a changing number of active vehicles. As expected, when more free parking spots are available, the average distance between the final spot and a driver's actual destination decreases.

5.4 Metrics of the Distributed Approach

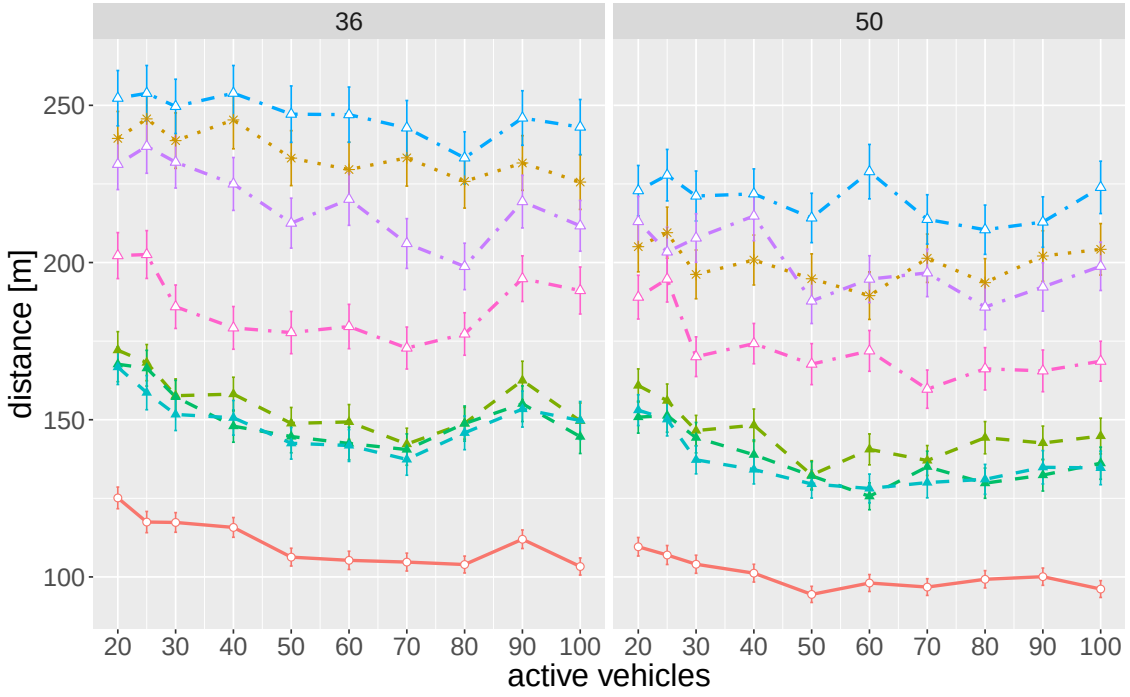
We analyzed the fully distributed approach in more detail to understand and visualize the effect of certain parameters, mainly to be discussed here—the size q of the local memory.

The point in time when the accuracy of the data in the local memory is of utmost importance to the driver is when he initiates the search for a parking spot. Thus, in the following we compare the vehicles' local memories at the moment of this transition to the state *lfp*. Again, all plots show the average values over the specific scenarios along with 95 %-confidence intervals. The results for the hot spot scenario with 22 free parking spots are nearly identical to those of the uniformly distributed scenario with the same number of available spots and, hence, not shown here for the sake of brevity. As previously, if not otherwise stated, all parking spots within r_{init} around a driver's destination are considered *relevant* parking spots. First, we check how many vehicles actually have information about a free relevant parking spot candidate stored locally (Figure 5.8). This shows that with a low active vehicle

Figure 5.6: Average 2D-distances between the final parking spot and the actual destination d_{pd} in the uniformly distributed scenario with 22 free spots along with their 95 %-confidence intervals.



(a) 22 free parking spots



(b) 36 and 50 free parking spots

Figure 5.7: Average 2D-distances between the final parking spot and the actual destination d_{pd} in the hot spot scenario with different numbers of free parking spots along with their 95 %-confidence intervals. Note the different y-axis scales between (a) and (b).

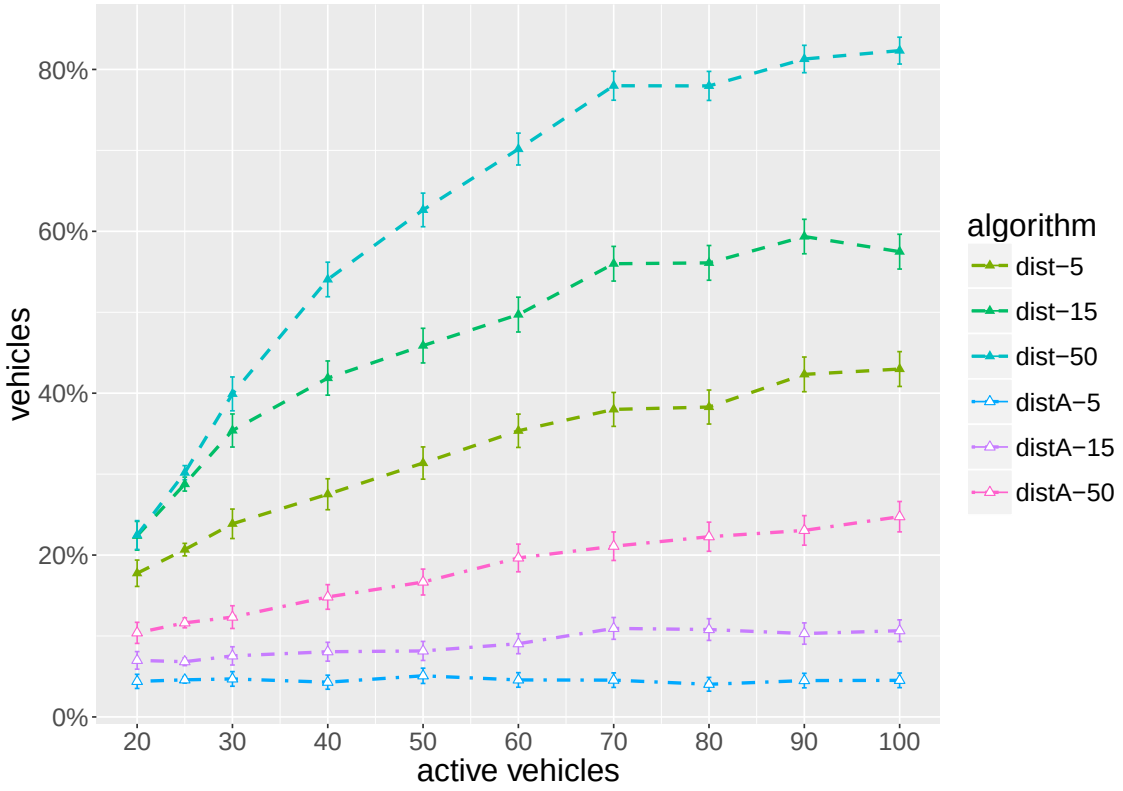


Figure 5.8: Average number of vehicles with a relevant parking spot in their local memory upon parking search start in the uniformly distributed scenario and 95 %-confidence intervals.

density most vehicles initiate their parking spot search with a random guess, like those following the naïve approach. Also, with a larger memory the probability of knowing of a relevant candidate parking spot increases. Interestingly, drivers following the normal distributed approach are more likely to have knowledge of a free relevant parking spot when needed than those following its advanced variant.

In Figure 5.9 we can see how much of the target area the local memory covers for different memory sizes in both distributed variants in the uniformly distributed scenario. It is apparent that the advanced variant that stores information about occupied parking spots too has more knowledge. A vehicle following the normal algorithm could only theoretically achieve 100 % if all parking spots in the target area were free. We can see that, as expected, smaller memory sizes always lead to having less information about the area of interest. Vehicles following the advanced version that only stores information about 5 parking spots seem to always have their memory filled with 5 relevant parking spots. Notably, with an increasing number of active vehicles vehicles gain more knowledge about their destination area.

When looking at the accuracy of this relevant data stored locally (Figure 5.10) we can see three things: (i) The data stored in the advanced variant is almost

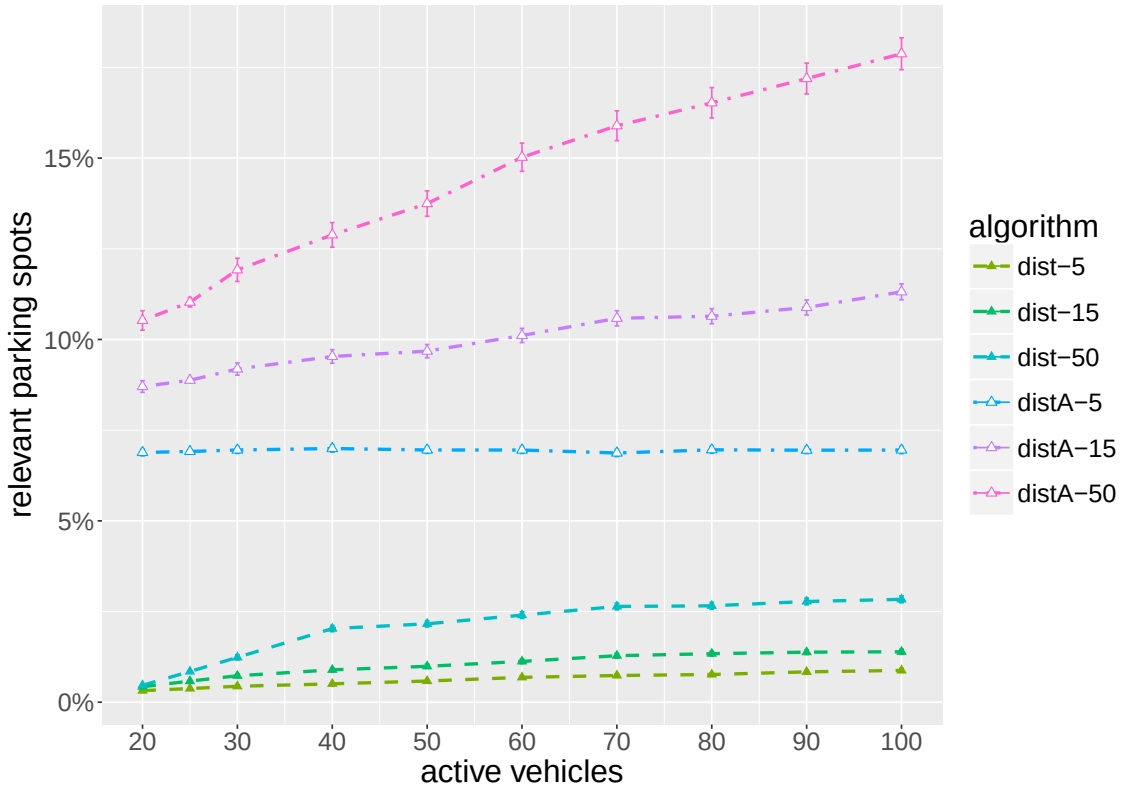


Figure 5.9: Average coverage of relevant parking spots in a vehicle's memory upon parking search start in the uniformly distributed scenario and 95 %-confidence intervals.

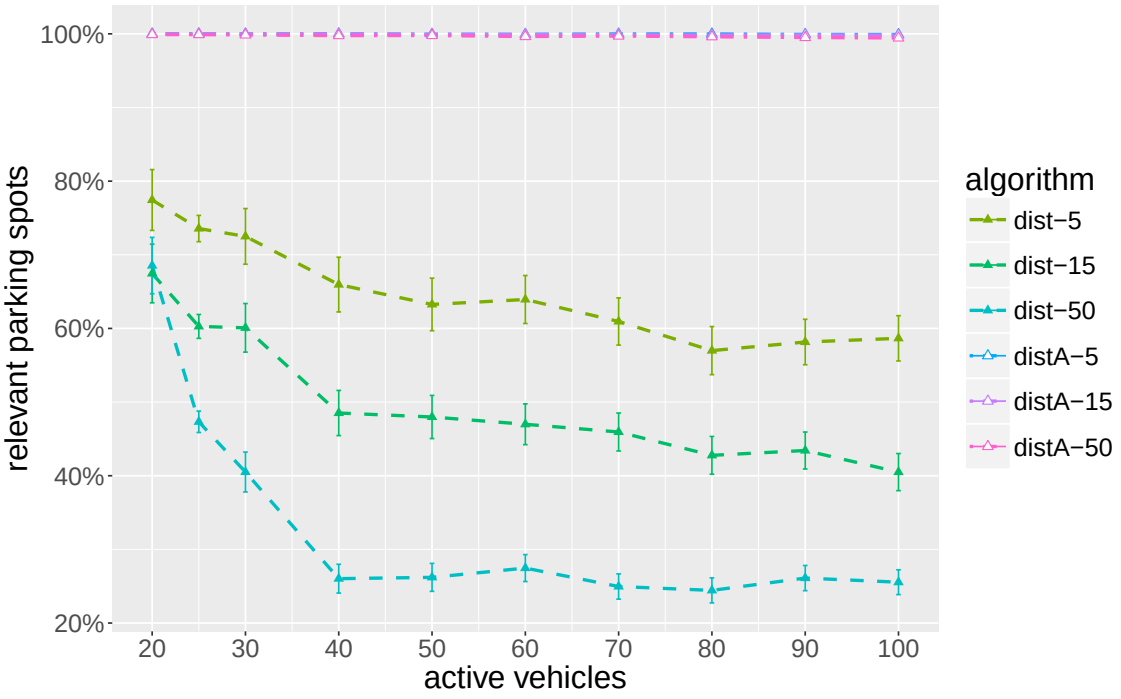


Figure 5.10: Average accuracies of information about all relevant parking spots in a vehicle's memory upon parking search start in the uniformly distributed scenario along with their 95 %-confidence intervals.

completely correct. This is intuitive as most parking spots are occupied and, hence, the information stored is almost exclusively about occupied parking spots. (ii) The larger the local memory is, the more inaccurate information about free parking spots a vehicle stores. However, this plot does not consider the age of the information like the relevance function the on-board system uses before selecting a candidate. This means that it is possible that the additional entries that are stored with a larger memory are already outdated and therefore timewise not relevant for the search. (iii) With an increasing number of active vehicles, the quality of information received about possible parking spot candidates degrades too. However, this effect seems to stall at different levels depending on the memory size. Again, the additional information received through more communication partners is most likely mainly outdated.

If we focus on the actually interesting data on free parking spots within the vehicle's memories (Figure 5.11), we see that vehicles following the standard algorithm achieve higher accuracy than those using the advanced variant. Note the y-axis percent scale—the actual average number of free relevant parking spots stored in local memory is 0.19 in both demand scenarios with 22 free parking spots overall. It rises from 0.13 with 20 active vehicles to 0.25 with 100 active vehicles. The basis for the plotted percent values are only vehicles that actually had information about at least one free parking spot in their memory. In fact, Figure 5.11 shows that, if a vehicle

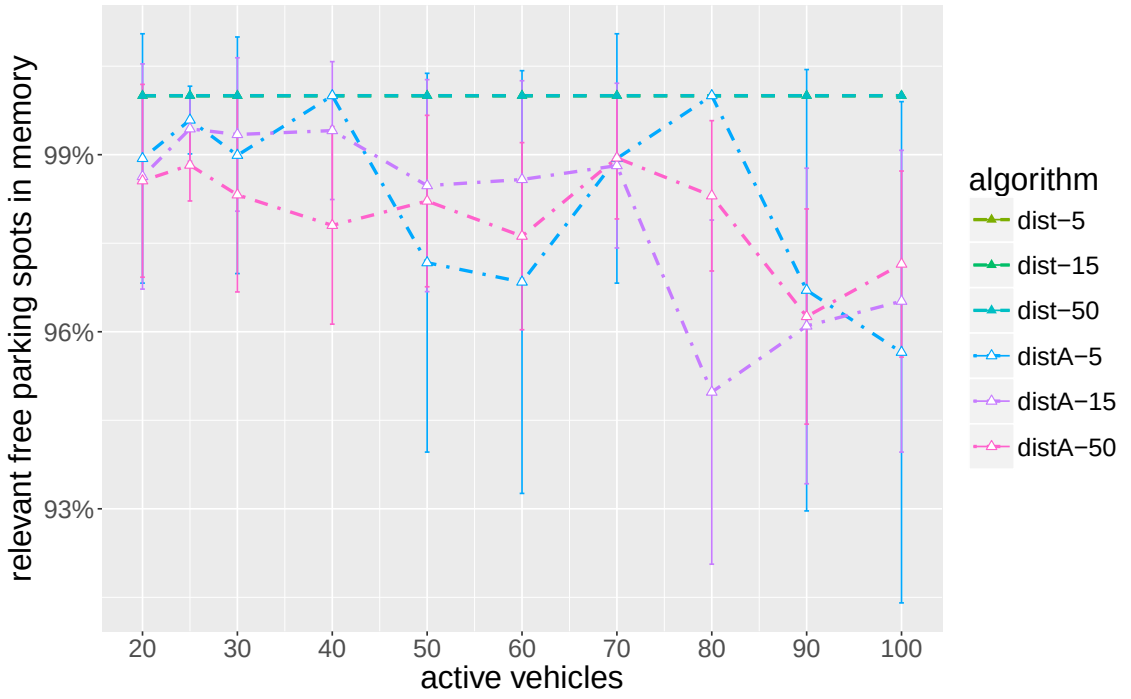


Figure 5.11: Average accuracies of information about free relevant parking spots in a vehicle's memory upon parking search start in the uniformly distributed scenario along with their 95 %-confidence intervals.

following the normal distributed approach has information about a free parking spot in the target area, this information is almost always correct.

Finally, we check how well a vehicle's local memory covers the actually free parking spots within the target area (Figure 5.12). Note that we are plotting percentages and that the actual average number of free parking spots within r_{init} around a vehicle's destination is 0.65, if there are 22 free parking spots overall. This is independent of the demand scenario. We can see that the normal distributed approach achieves a better coverage of the free relevant parking spots than its advanced variant. From Figures 5.9 and 5.12 we can conclude that the increased coverage, that is gained by also storing information about occupied parking spots, does in fact clog the memory with such irrelevant information and thereby lower the coverage of actual interesting potential candidate spots.

5.5 Message Cost

From an algorithmic perspective the cost of the different algorithms can be defined by the *number of messages sent* or, in addition, their respective sizes. We ignore the size and model this number the following way:

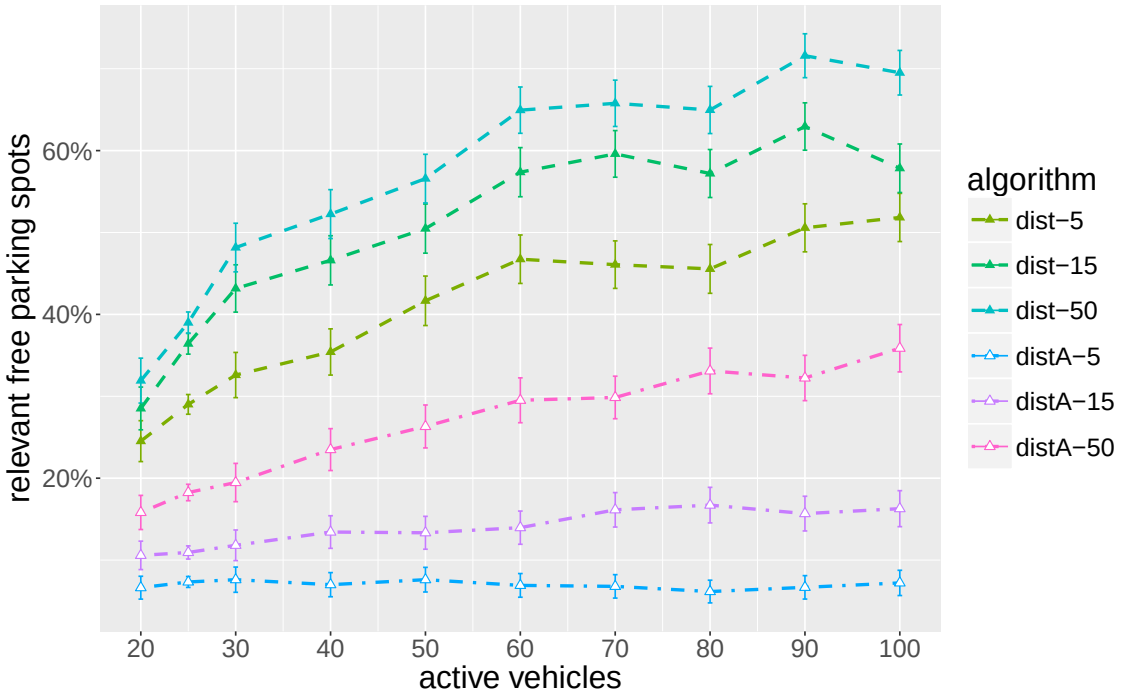


Figure 5.12: Average correct coverage of free relevant parking spots in a vehicle's memory upon parking search start in the uniformly distributed scenario and 95 %-confidence intervals.

- In the global approach, each vehicle notifies the server of the parking spot it is vacating at the start of the trip (this message could also be sent by a stationary sensor) and does the same for the final parking spot upon arrival. These count as one message each. Additionally, each vehicle requests information about a free parking spot once it initiates the search. This is another message sent. It then receives one message every time the central server notifies it that the current target has been taken by sending information about a new target.
- In the distributed approach, during every merge each vehicle sends and receives exactly one message (although such messages usually contain much more data than the messages sent by the central server mentioned above).

In the naïve approach no messages are sent at all. Figure 5.13 visualizes the average number of messages sent or received for both demand scenarios with 22 free parking spots. Note, that this counts messages twice for every distributed approach. We can see that the number of messages increases with the active vehicle density. For the global approach this increase is faster in the hot spot scenario than in the uniformly distributed scenario and results from the fact, that more often a currently targeted parking spot gets taken by a competing vehicle, which in turn triggers an update message from the server. In the distributed case obviously the number of merges increases with the number of neighboring vehicles. There are two ways to argue which of the smart approaches is cheaper in terms of message cost: In the decentralized

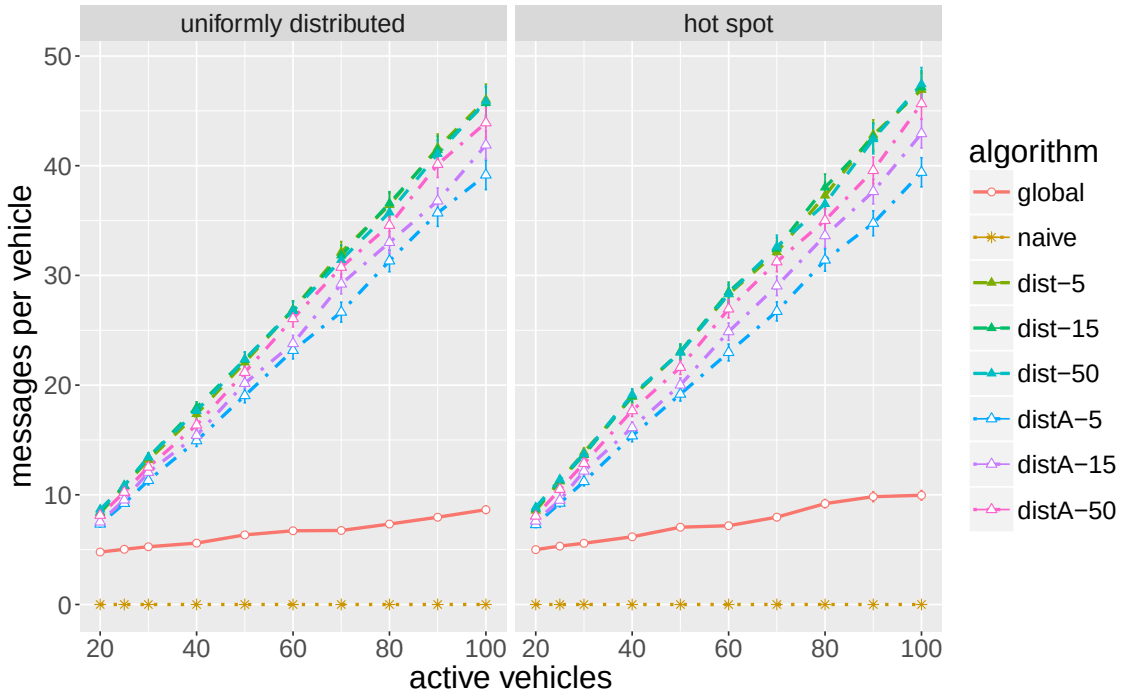


Figure 5.13: Average number of messages sent and received per vehicle in both scenarios with 22 free parking spots and 95 %-confidence intervals.

case the overall number of messages is obviously higher than in the centralized one. However, the load is shared among all participating vehicles. On the contrary, in the global approach the central server—which is a node within the network as well—sends or receives every message of the system.

5.6 Discussion

As stated in Section 2.3.6 we performed a very similar study to [36]. However, although we used similar parameters for our simulations, we were not able to reproduce their results in detail. The authors of [36] did not explicitly state their parameters of the demand generation in the hot spot scenario. This makes it difficult to compare it to ours. Nonetheless, even for the scenario with uniformly distributed destinations our results differ. To us this indicates a pivotal aspect of the topic: *Microscopic traffic simulations of the PSSP are very parameter-dependent.* Therefore it is very important to include the specific values in a publication. In the one discussed [36], the underlying road network was crucially different: While the network bounding box, and hence the covered area, was larger than ours (1 440 000 m^2 vs. 810 000 m^2), the overall road length of the network was much smaller (24 000 m vs. 32 400 m when we consider every lane/direction). This arises from the fact that our grid is more fine-meshed with individual road lengths of 100 m (in comparison to

their 300 m roads). The usage of unregulated priority to the right junctions instead of roundabouts [36] should not impact the simulation results. Both designs are aimed at keeping the traffic flowing.

However, the other very important parameter category next to network topology is the design of vehicle demand. Many studies mentioned, as well as the one conducted in this thesis, use uniformly distributed vehicle routes within a generic network. While we argue for reproducibility, realistic traffic, especially in city centers, does not behave this way. As hinted in Section 4.2, the task of finding a credible vehicle and parking spot density is not straightforward. Either such information is not easily available, or it might not even be suitable to analyze and compare general paradigms. This means every simulation has to be fine-tuned—not just due to the danger of gridlock but also to visualize the desired aspects. It would be preferable to perform such an analysis on a real road network topology with realistic (evidence-based) demand data.

A fundamental difference is how the global approach is modeled in [36] compared to this thesis. It is expected, that a method that uses reservations behaves different to our approaches. For example, it is easy to argue that the centrally-assisted parking search (CAPS) [36] results in short parking search times but long distances between parking spot and destination. This is a direct consequence from the reservation mechanism. However, the way we model a centralized approach in this thesis, it does not suffer the drawbacks of being too rigid. In our model a driver that targets a centrally advertised parking spot immediately parks at a free parking spot he encounters whereas he would ignore this opportunity in the approach presented in [36].

At the core of the very problem that this thesis addresses lies the desire to get the *best* available parking spot for every driver. Hence, one must define what *best* means beforehand. In this thesis we have done so with the cost function given in the problem statement (Equation (1.1)) which is the basis for the relevance function in the distributed approach (see Section 3.3.2). In the global approach we have defined the distance between parking spot candidate and destination to be the criterion. In reality such a ranking would probably be different for different drivers. One does not mind walking a little further, but finding a parking spot more quickly, while another may not even be able to walk long distances and prefers to keep searching until he finds a parking spot close to his destination. As stated, the authors of [74] incorporate these individual parameters into their relevance function (Equation (3.3)).

The way the different approaches were modeled and implemented in this thesis is just one of many. There are multiple aspects that could be modified without changing the general ideas. A globally available server could for example notify vehicles as soon as a better candidate parking spot becomes available. However, this would of course increase the message cost, i.e. the number of messages sent. In the distributed approach our algorithm does not address the competition between vehicles more than not sharing the own target parking spot. Such a mechanism would most likely be

beneficial [19, 20]. In all our algorithms containing parking information dissemination we send atomic information tuples about single parking spots. Others have used aggregates [13, 52] (see Section 2.3.1) and argue that aggregated information is more time stable and, hence, reliable. We tested a very simple method of countering information decay in the advanced version of our distributed approach by using the same local memory to also store information about occupied parking spots. Our results show that this is not beneficial with the memory sizes used. The performance of this advanced variant does improve with increasing memory size and it could outmatch its normal counterpart with even larger memory capacities. However, this increases the message cost and complexity for every merge operation significantly. Another possibility is using two local memories to store information about free and occupied parking spots separately [7]. We leave the analysis of information decay in our approaches to further research. An—also theoretical—analysis of this process can be found in [74].

A crucial detail in the implementation is also the handling of free parking spots on the opposite side of the road. We tried to design our scenarios as realistically as possible—and in reality a driver who sees a free parking spot on the opposite side of the road is very likely to turn at the next intersection and target it. This is how we implemented driver behavior. However, it is difficult to argue how such a "visually detected" parking spot becomes an information tuple in the vehicle's on-board memory. At least, since in such a situation the driver targets it immediately, it does not get shared with others. As for the naïve random algorithm, the algorithm itself does not necessarily resemble realistic driver behavior, since often drivers have some knowledge of areas around their destination with increased probabilities for containing a free parking spot.

Solving the PSSP, a problem that is centered around global up-to-date information, in a completely decentralized manner is not only of interest from a scientific point of view, but poses financial benefits as well. According to [58] each sensor in SFpark [54] costs about US $500 whereas each sensor deployment per vehicle in [42] costs about US $400. When we add the authors' of [42] own argumentation, according to which a deployment on 500 taxi cabs would be enough for a city of the size of San Francisco to have sufficient parking spot detection coverage, we can easily see that the distributed approach is much cheaper in overall sensor installation cost. In numbers, such a distributed approach could be around 10-15 times cheaper than a stationary sensor at every parking spot [42].

In Section 2.3.2 we have addressed the fact that maps of parking spot locations are usually not given, especially those differentiating legal from illegal ones. If we extend on this, we have to remember that we are detecting parking spots on-the-fly in our distributed approach and that in reality this detection is not always 100 % correct. More importantly, *parking spots differ in size* as well as *vehicles differ in size demand*. In our simulations for simplicity we have assumed that all vehicles are of the same type and have the same size. Also, all parking spots have the same length and the

vehicles all agree on these parking spot locations. In reality, especially with curb parking, it is often the case that individual parking spots are not clearly defined and the lane that is used by parking vehicles is used in a dynamic way. This means the locations and sizes of the individual spots move and change depending on the order in which different types of vehicles occupy them.

Important aspects which we have not mentioned yet are privacy and security. The emergence of smart hardware introduces new channels for criminals to harm users. For an overview about "Data Security in Vehicular Communication Networks" refer to [72]. In [60] the authors present a method that treats *locations* as the primary entity of interest, in contrast to users with their current locations attached. This is one way to achieve a certain level of privacy when handling geospatial data. Focusing on the approaches to the PSSP presented in this thesis, there are a few things to consider: In an unassisted naïve search obviously no privacy concerns arise. When communicating with a global entity things are trickier and implementation-dependent. For the sake of the algorithm a driver does not have to share his location with the server. Since we are not using any reservation mechanism there is theoretically no way for the server to know which vehicle parked at which parking spot (with the exception of visual sensors that could read a license plate). Then the vehicle's on-board system would query the server for all parking spot locations (within a larger area) and do the distance calculations by itself. The downside of such an implementation is the increased computational load on vehicles' limited hardware as well as an, from a theoretical standpoint, immensely increased bandwidth usage for the central server (i.e. message cost).

6 Conclusion and Future Work

The parking spot search problem is a problem millions of drivers face each day. Its effects are not only an economic loss due to wasted time but also an immense environmental burden. Smart hardware and software create new possibilities of solving this problem. There are three fundamentally different approaches to do so: (i) The naïve way, where drivers do not use any assisting technologies and randomly search their target area for a free parking spot. (ii) The global way, where all vehicles within the road network communicate with a central instance (e.g. a server on the internet) to gain knowledge about currently free parking spots. (iii) The distributed way, in which vehicles solve the problem completely decentralized by communicating only with their geographical neighbors; vehicles detect parking spots, store them in a local memory, and share their knowledge with others. We have conducted extensive simulations in varying traffic scenarios on a 10×10 grid road network with randomly generated routes. Our results show that smart approaches using ICT infrastructure can significantly reduce search times and distances between the final parking spot and the actual destination. Especially a centralized approach outperforms all others. However, it is the one with the strictest assumptions regarding connectivity and sensor deployment. A decentralized approach still achieves significant performance gains while the cost of sensor deployment and communication infrastructure is delegated to the drivers. In both cases the usage of atomic information about individual parking spots suffices. In the distributed case the data from the local memory has to be ranked by relevance. Our results show that a simple function based on the information age and the position of its origin handle information decay better than also storing information about occupied parking spots in the same memory. Hence, a small local memory capable of storing information about as few as five free parking spots is sufficient to improve driver satisfaction in city centers.

Similar studies have been performed by others [35, 36] and our results do not conform to theirs completely. The fact that their investigated approaches to the PSSP as well as simulation settings differ to ours in the details, shows us that any research done in this field is highly parameter-dependent. Therefore we suggest a more theoretical analysis of the problem as well as the usage of real world data for road network and traffic initialization during simulations in future research. Also, the fact that the attempt to handle information decay in a distributed approach by storing information about occupied parking spots as well actually worsens performance, might indicate that vehicles using this mechanic would need even larger memories. This has to be investigated.

While we have analyzed the PSSP in different problem settings and the three approaches with different initial parameters, there are multiple things left for future work. We have assumed that all vehicles and parking spots are of the same type and size—a more realistic setting uses different types and possibly overlapping parking spot positions that change depending on the order of parking vehicles. It would be interesting to compare the performance of vehicles using a smart approach to vehicles in the same network that do not take part in the system similar to the work in [7], instead of creating separate scenarios for smart and naïve approaches. Since our smart algorithms differ from those presented in [34], it would be desirable to analyze the effect of cheaters on them. The presented algorithms use a multitude of parameters (like the parking search radius r_{init} and its increment function) that we initialized with reasonable defaults in our simulations. However, an extensive analysis of the parameters' effects is left for further research. Also, an individual and dynamic parameter adaption on a per-vehicle basis needs to be investigated. This includes an update of r_{init} if a vehicle detects very high or low occupancy for example. The presented analysis is left to be repeated in a scenario with a real world road network and real traffic demand definitions. As stated in Chapter 3, one could add designated parking lots with a certain capacity to the scenarios in which the different approaches are tested. Also, one could adapt the given algorithms to create a new hybrid approach similar to the one presented in [52], where a vehicle uses the distributed approach if the active vehicle density is high enough, and queries a central server if it is not; or where all vehicles use a central server, but only the sensors for free parking spot detection are vehicle-based. Finally, the idea of solving a problem completely decentralized through sharing of atomic information could be used for similar applications (like sharing road travel times for traffic density estimation).

Bibliography

[1] S. Abidi, S. Krichen, E. Alba, and J. M. Molina. "A New Heuristic for Solving the Parking Assignment Problem". In: *Procedia Computer Science*. Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings 60 (Jan. 1, 2015), pp. 312–321.

[2] F. Ahmed-Zaid, F. Bai, S. Bai, C. Basnayake, B. Bellur, S. Brovold, G. Brown, L. Caminiti, D. Cunningham, H. Elzein, K. Hong, J. Ivan, D. Jiang, J. Kenney, H. Krishnan, J. Lovell, M. Maile, D. Masselink, E. McGlohon, P. Mudalige, Z. Popovic, V. Rai, J. Stinnett, L. Tellis, K. Tirey, and S. VanSickle. *Vehicle Safety Communications – Applications (VSC-A) Final Report*. Final Report DOT HS 811 492A. U.S. Department of Transportation, Sept. 2011.

[3] A. Aliedani, S. W. Loke, A. Desai, and P. Desai. "Investigating vehicle-to-vehicle communication for cooperative car parking: The CoPark approach". In: *2016 IEEE International Smart Cities Conference (ISC2)*. Sept. 2016, pp. 1–8.

[4] R. Arnott, T. Rave, and R. Schöb. *Alleviating urban traffic congestion*. The CESifo book series. OCLC: ocm57749517. Cambridge, Mass: MIT Press, 2005. 240 pp.

[5] P. Basu and T. D. Little. "Wireless ad hoc discovery of parking meters". In: *MobiSys Workshop on Applications of Mobile Embedded Systems (WAMES'04)*. 2004.

[6] F. Bellifemine, A. Poggi, and G. Rimassa. "JADE: A FIPA2000 Compliant Agent Development Environment". In: *Proceedings of the Fifth International Conference on Autonomous Agents*. AGENTS '01. New York, NY, USA: ACM, 2001, pp. 216–217.

[7] N. Bessghaier, M. Zargayouna, and F. Balbo. "Management of Urban Parking: An Agent-Based Approach". In: *Artificial Intelligence: Methodology, Systems, and Applications*. Ed. by A. Ramsay and G. Agre. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 276–285.

[8] R. Bodenheimer, D. Eckhoff, and R. German. "GLOSA for adaptive traffic lights: Methods and evaluation". In: *2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM)*. Oct. 2015, pp. 320–328.

[9] J. L. Boehle. "City-Based Parking and Routing System". Master Thesis. Erasmus University Rotterdam, 2007.

[10] J. L. Boehle, L. J. M. Rothkrantz, and M. v. Wezel. *Cbprs: A City Based Parking and Routing System*. SSRN Scholarly Paper ID 1144292. Rochester, NY: Social Science Research Network, 2008.

[11] I. Bogoslavskyi, L. Spinello, W. Burgard, and C. Stachniss. "Where to park? minimizing the expected time to find a parking space". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2147–2152.

[12] M. Caliskan, A. Barthels, B. Scheuermann, and M. Mauve. "Predicting Parking Lot Occupancy in Vehicular Ad Hoc Networks". In: *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*. Apr. 2007, pp. 277–281.

[13] M. Caliskan, D. Graupner, and M. Mauve. "Decentralized Discovery of Free Parking Places". In: *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks*. VANET '06. New York, NY, USA: ACM, 2006, pp. 30–39.

[14] A. Caragliu, C. Del Bo, and P. Nijkamp. "Smart Cities in Europe". In: *Journal of Urban Technology* 18.2 (2011), pp. 65–82.

[15] D. Caveney. "Cooperative Vehicular Safety Applications". In: *VANET Vehicular Applications and Inter-Networking Technologies*. Ed. by H. Hartenstein and K. P. Laberteaux. John Wiley & Sons, Ltd, 2010, pp. 21–48.

[16] L. Chen, J. Hsieh, W. Lai, C. Wu, and S. Chen. "Vision-Based Vehicle Surveillance and Parking Lot Management Using Multiple Cameras". In: *2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. Oct. 2010, pp. 631–634.

[17] V. Coric and M. Gruteser. "Crowdsensing Maps of On-street Parking Spaces". In: *2013 IEEE International Conference on Distributed Computing in Sensor Systems*. May 2013, pp. 115–122.

[18] B. Defude, T. Delot, S. Ilarri, J.-L. Zechinelli, and N. Cenerario. "Data Aggregation in VANETs: The VESPA Approach". In: *Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Mobiquitous '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, 13:1–13:6.

[19] T. Delot, N. Cenerario, S. Ilarri, and S. Lecomte. "A Cooperative Reservation Protocol for Parking Spaces in Vehicular Ad Hoc Networks". In: *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*. Mobility '09. New York, NY, USA: ACM, 2009, 30:1–30:8.

[20] T. Delot, S. Ilarri, S. Lecomte, and N. Cenerario. "Sharing with Caution: Managing Parking Spaces in Vehicular Networks". In: *Mobile Information Systems* 9.1 (Jan. 18, 2013), pp. 69–98.

[21] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numer. Math.* 1.1 (Dec. 1959), pp. 269–271.

[22] S. Faye, C. Chaudet, and I. Demeure. "A distributed algorithm for adaptive traffic lights control". In: *2012 15th International IEEE Conference on Intelligent Transportation Systems*. Sept. 2012, pp. 1572–1577.

[23] S. Faye, C. Chaudet, and I. Demeure. "A Distributed Algorithm for Multiple Intersections Adaptive Traffic Lights Control Using a Wireless Sensor Networks". In: *Proceedings of the First Workshop on Urban Networking*. UrbaNe '12. New York, NY, USA: ACM, 2012, pp. 13–18.

[24] J. d. Gier, T. M. Garoni, and O. Rojas. "Traffic flow on realistic road networks with adaptive traffic lights". In: *Journal of Statistical Mechanics: Theory and Experiment* 2011.4 (2011), P04008.

[25] V. Gradinescu, C. Gorgorin, R. Diaconescu, V. Cristea, and L. Iftode. "Adaptive Traffic Lights Using Car-to-Car Communication". In: *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*. Apr. 2007, pp. 21–25.

[26] G. Harini and A. G. Selvarani. "Off-Street Parking Guidance System". In: *Indian Journal of Science and Technology* 9.21 (June 16, 2016).

[27] H. Hartenstein and K. Laberteaux, eds. *VANET Vehicular Applications and Inter-Networking Technologies*. Wiley, 2009.

[28] *IEEE 1609 - Family of Standards for Wireless Access in Vehicular Environments (WAVE)*. Sept. 25, 2009.

[29] *Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*. EN 302 663 V1.2.0. European Telecommunications Standards Institute, Nov. 2012.

[30] *Intelligent Transport Systems (ITS); Mitigation techniques to avoid interference between European CEN Dedicated Short Range Communication (CEN DSRC) equipment and Intelligent Transport Systems (ITS) operating in the 5 GHz frequency range*. TS 102 792 V1.2.1. European Telecommunications Standards Institute, June 2015.

[31] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. "Design of 5.9 ghz dsrc-based vehicular safety communication". In: *IEEE Wireless Communications* 13.5 (Oct. 2006), pp. 36–43.

[32] R. Jiang, J.-Y. Chen, Z.-J. Ding, D.-C. Ao, M.-B. Hu, Z.-Y. Gao, and B. Jia. "Network operation reliability in a Manhattan-like urban system with adaptive traffic lights". In: *Transportation Research Part C: Emerging Technologies* 69 (Aug. 1, 2016), pp. 527–547.

[33] E. Kokolaki, M. Karaliopoulos, and I. Stavrakakis. "Value of information exposed: Wireless networking solutions to the parking search problem". In: *2011 Eighth International Conference on Wireless On-Demand Network Systems and Services*. Jan. 2011, pp. 187–194.

[34] E. Kokolaki, G. Kollias, M. Papadaki, M. Karaliopoulos, and I. Stavrakakis. "Opportunistically-assisted parking search: A story of free riders, selfish liars and bona fide mules". In: *2013 10th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*. Mar. 2013, pp. 17–24.

[35] E. Kokolaki, M. Karaliopoulos, G. Kollias, M. Papadaki, and I. Stavrakakis. "Vulnerability of opportunistic parking assistance systems to vehicular node selfishness". In: *Computer Communications* 48 (July 15, 2014), pp. 159–170.

[36] E. Kokolaki, M. Karaliopoulos, and I. Stavrakakis. "Opportunistically assisted parking service discovery: Now it helps, now it does not". In: *Pervasive and Mobile Computing*. Special Issue: Wide-Scale Vehicular Sensor Networks and Mobile Sensing 8.2 (Apr. 1, 2012), pp. 210–227.

[37] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. "Recent Development and Applications of SUMO - Simulation of Urban MObility". In: *International Journal On Advances in Systems and Measurements* 5.3 (Dec. 2012), pp. 128–138.

[38] T. Lin, H. Rivano, and F. L. Mouël. "A Survey of Smart Parking Solutions". In: *IEEE Transactions on Intelligent Transportation Systems* 18.12 (Dec. 2017), pp. 3229–3253.

[39] C. Lochert, A. Barthels, A. Cervantes, M. Mauve, and M. Caliskan. "Multiple Simulator Interlinking Environment for IVC". In: *Proceedings of the 2Nd ACM International Workshop on Vehicular Ad Hoc Networks*. VANET '05. New York, NY, USA: ACM, 2005, pp. 87–88.

[40] R. Lu, X. Lin, H. Zhu, and X. Shen. "SPARK: A New VANET-Based Smart Parking Scheme for Large Parking Lots". In: *IEEE INFOCOM 2009*. Apr. 2009, pp. 1413–1421.

[41] N. Maslekar, M. Boussedjra, J. Mouzna, and H. Labiod. "VANET Based Adaptive Traffic Signal Control". In: *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*. 2011, pp. 1–5.

[42] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe. "ParkNet: Drive-by Sensing of Road-side Parking Statistics". In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 123–136.

[43] MathWorks. *MATLAB*. URL: `https://www.mathworks.com/products/matlab.html` (visited on 09/10/2018).

[44] H. Mousannif, H. A. Moatassime, and I. Khalil. "Cooperation as a Service in VANETs". In: *JUCS - Journal of Universal Computer Science* 8 (Apr. 28, 2011).

[45] K. Nagel and M. Schreckenberg. "A cellular automaton model for freeway traffic". In: *Journal de Physique I* 2.12 (1992), pp. 2221–2229.

[46] R. Panayappan, J. M. Trivedi, A. Studer, and A. Perrig. "VANET-based Approach for Parking Space Availability". In: *Proceedings of the Fourth ACM International Workshop on Vehicular Ad Hoc Networks*. VANET '07. New York, NY, USA: ACM, 2007, pp. 75–76.

[47] W.-J. Park, B.-S. Kim, D.-E. Seo, D.-S. Kim, and K.-H. Lee. "Parking space detection using ultrasonic sensor in parking assistance system". In: 2008 IEEE Intelligent Vehicles Symposium. June 2008, pp. 1039–1044.

[48] *Parking Guidance System*. URL: `http://www.parkeninwien.at/en/Parking%20Guidance%20System.html` (visited on 08/24/2018).

[49] A. Picon. *Smart Cities: A Spatialised Intelligence*. Ad primers. John Wiley & Sons, Nov. 16, 2015. 168 pp.

[50] S. V. Reve and S. Choudhri. "Management of Car Parking System Using Wireless Sensor Network". In: *International Journal of Emerging Technology and Advanced Engineering* 2.7 (July 2012), pp. 262–268.

[51] Rhythm Engineering. *InSync - Advanced Traffic Management for improved safety*. URL: `http://rhythmtraffic.com/insync/` (visited on 06/01/2017).

[52] R. Salpietro, L. Bedogni, M. D. Felice, and L. Bononi. "Park Here! a smart parking system based on smartphones' embedded sensors and short range Communication Technologies". In: *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. Dec. 2015, pp. 18–23.

[53] S. Samadi, A. P. Rad, F. M. Kazemi, and H. Jafarian. "Performance Evaluation of Intelligent Adaptive Traffic Control Systems: A Case Study". In: *Journal of Transportation Technologies* 02.3 (July 23, 2012), p. 248.

[54] San Francisco Municipal Transportation Agency. *SFpark*. URL: `http://sfpark.org` (visited on 08/27/2018).

[55] F. Schmidt-Eisenlohr. *Interference in Vehicle-to-vehicle Communication Networks: Analysis, Modeling, Simulation and Assessment*. KIT Scientific Publishing, 2010. 180 pp.

[56] J.-H. Shin and H.-B. Jun. "A study on smart parking guidance algorithm". In: *Transportation Research Part C: Emerging Technologies* 44 (Supplement C July 2014), pp. 299–317.

[57] D. Shoup. "Cruising for Parking". In: *ACCESS Magazine*. ACCESS Magazine 1.30 (Apr. 1, 2007), pp. 16–23.

[58] L. Stenneth, O. Wolfson, B. Xu, and P. S. Yu. "PhonePark: Street Parking Using Mobile Phones". In: *2012 IEEE 13th International Conference on Mobile Data Management*. July 2012, pp. 278–279.

[59] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu. "Transportation Mode Detection Using Mobile Phones and GIS Information". In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '11. New York, NY, USA: ACM, 2011, pp. 54–63.

[60] K. P. Tang, P. Keyani, J. Fogarty, and J. I. Hong. "Putting People in Their Place: An Anonymous and Privacy-sensitive Approach to Collecting Sensed Data in Location-based Applications". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. New York, NY, USA: ACM, 2006, pp. 93–102.

[61] *The EstiNet Network Simulator*. URL: `http://www.estinet.com/ns/` (visited on 09/18/2017).

[62] *The Matlab/Simulink Application Simulator*. URL: `https://www.mathworks.com/products/simulink.html` (visited on 09/18/2017).

[63] *The NCTUns Network Simulator and Emulator*. URL: `http://nsl.cs.nctu.edu.tw/NSL/nctuns.html` (visited on 09/18/2017).

[64] *The ns-2 Network Simulator*. URL: `https://www.isi.edu/nsnam/ns/` (visited on 09/18/2017).

[65] *The VISSIM Traffic Simulator*. URL: `http://vision-traffic.ptvgroup.com/en-uk/products/ptv-vissim/` (visited on 09/18/2017).

[66] I. J. P. M. Timóteo, M. R. Araújo, R. J. F. Rossetti, and E. C. Oliveira. "TraSMAPI: An API oriented towards Multi-Agent Systems real-time interaction with multiple Traffic Simulators". In: *13th International IEEE Conference on Intelligent Transportation Systems*. Sept. 2010, pp. 1183–1188.

[67] J. D. Turner. *Automotive Sensors*. Vol. 1st ed. Sensors Technology. New York: Momentum Press, 2009.

[68] *Vehicle Safety Communications Project - Final Report*. Final Report DOT HS 810 591. U.S. Department of Transportation, Apr. 2006, p. 1291.

[69] V. Verroios, V. Efstathiou, and A. Delis. "Reaching Available Public Parking Spaces in Urban Environments Using Ad Hoc Networking". In: *2011 IEEE 12th International Conference on Mobile Data Management*. Vol. 1. June 2011, pp. 141–151.

[70] S. Y. Wang and C. C. Lin. "NCTUns 5.0: A Network Simulator for IEEE 802.11(p) and 1609 Wireless Vehicular Network Researches". In: *2008 IEEE 68th Vehicular Technology Conference*. Sept. 2008, pp. 1–2.

[71] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux. "TraCI: An Interface for Coupling Road Traffic and Network Simulators". In: *Proceedings of the 11th Communications and Networking Simulation Symposium*. CNS '08. New York, NY, USA: ACM, 2008, pp. 155–163.

[72] A. Weimerskirch, J. J. Haas, Y.-C. Hu, and K. P. Laberteaux. "Data Security in Vehicular Communication Networks". In: *VANET*. Ed. by H. Hartenstein and K. P. Laberteaux. Wiley-Blackwell, 2009, pp. 299–363.

[73] B. Xie and D. P. Agrawal. *Encyclopedia On Ad Hoc And Ubiquitous Computing: Theory And Design Of Wireless Ad Hoc, Sensor, And Mesh Networks*. Singapore: World Scientific, 2010.

[74] B. Xu, A. Ouksel, and O. Wolfson. "Opportunistic resource exchange in inter-vehicle ad-hoc networks". In: *IEEE International Conference on Mobile Data Management, 2004. Proceedings. 2004.* May 24, 2004.

[75] B. Yang, N. Fantini, and C. S. Jensen. "iPark: Identifying Parking Spaces from Trajectories". In: *Proceedings of the 16th International Conference on Extending Database Technology.* EDBT '13. New York, NY, USA: ACM, 2013, pp. 705–708.

[76] C. Zhang, H. Dong, L. Jia, Y. Qin, and Z. Yang. "Robust vehicle detection and identification with single magnetic sensor". In: *2017 2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE).* Sept. 2017, pp. 94–98.

[77] Z. S. Zhang, H. Q. Yuan, and L. Chen. "A Parking Vehicle Detection Algorithm Using Magnetic Sensor". In: *Applied Mechanics and Materials* 409-410 (Sept. 2013). Ed. by W. Yang and J. Liang, pp. 1353–1356.

[78] Y. Zhao and Z. Tian. "An Overview of the Usage of Adaptive Signal Control System in the United States of America". In: *Applied Mechanics and Materials* 178-181 (2012), pp. 2591–2598.

[79] H. Zhu and F. Yu. "A Vehicle Parking Detection Method Based on Correlation of Magnetic Signals". In: *International Journal of Distributed Sensor Networks* 2015 (July 8, 2015).