



The automated valet parking problem for parking garage service provision

Marc Schmidt¹ · Philipp Staudt¹

Received: 19 January 2023 / Accepted: 26 August 2026
© The Author(s) 2026

Abstract

Automated valet parking is a use case for autonomous driving. It allows customers to leave their vehicles at the entrance of a parking garage. The vehicles then independently steer to a parking space. Besides increasing comfort, automated valet parking also enables parking garage operators to provide additional services, such as charging an electric vehicle, to which the vehicle can be autonomously routed. We formulate automated valet parking service provision as a scheduling problem and propose solution methods that focus on runtime and quality of results. The results show that the proposed optimization approach is capable of slightly outperforming heuristic benchmarks in many scenarios. However, we also find that the First-Come-First-Served and Earliest-Due-Date-First heuristics are often satisfying scheduling approaches. In addition, the potential of optimizing automated valet parking service provision is compared to traditional parking garages, showing at least a 46.4% increase in charging and other services scheduled.

Keywords Automated valet parking · Autonomous vehicles · Electric vehicle charging · Destination charging · Scheduling problems · Smart charging · Constraint programming

Highlights

- We introduce the new problem class of automated valet parking service provision.
- We propose solution methods to the automated valet parking problem based on constraint programming.
- Automated valet parking increases the number of scheduled services by at least 45.4% and up to 339.1%.
- The two-stage constraint program formulated for the automated valet parking problem performs best at low and medium system utilization.
- Using heuristics can provide good results for the automated valet parking problem and scales well.

1 Introduction

Autonomous driving offers promising use cases such as shared mobility or a reduction in vehicle ownership (Baron et al., 2021) and may reduce mobility costs (Bagloee et al., 2016). An early use case is automated valet parking (AVP). Beginning with level 4 autonomy (Sae International, 2021), the driver no longer needs to steer the vehicle in controlled environments (Wang et al., 2021) and can therefore leave the vehicle at the entrance of parking garages. AVP is considered the first use case for autonomous vehicles (Banzhaf et al., 2017) and is already being tested and demonstrated in multiple pilot projects, such as at the International Auto Show in Germany in 2021 (Vda, 2021), the European research project V-Charge (Schwesinger et al., 2016) and the research project AutoPles (Klemm et al., 2016), which analyzed driverless parking and charging. These pilot projects have demonstrated that autonomous vehicle parking is already feasible. There are several advantages associated with AVP, such as the ability to relocate vehicles whenever necessary to improve a parking system's efficiency (Zips et al., 2020), saving the time needed to search for a parking spot (Schu and Riener, 2020) and the potential to reduce the average space occupied per vehicle by double parking vehicles (Orey et al., 2016).

✉ Philipp Staudt
philipp.staudt@kit.edu

Marc Schmidt
marc.schmidt@kit.edu

¹ Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany

This has led to different potential use cases where AVP can increase the comfort of drivers, such as maneuvering an EV to a charging station once it is no longer occupied, taking over the vehicle on arrival at an airport (Bosch, 2020), or as a seamless transition to public transport (Scholliers et al., 2020).

In this study, we formulate the challenge of scheduling services provided to an autonomous vehicle in an AVP parking garage and introduce the AVP service provision problem (hereafter referred to as the AVP problem), which addresses the scheduling of fully automated services performed on vehicles in an AVP parking garage. We assume that customers visit the parking garage (i.e., the AVP system) for other reasons; that is, the AVP service provision is not the primary reason they use the parking garage. That means that service provision is an addition to any other objectives tied to the visit (such as visiting a mall), similar to the concept of destination charging (Schmidt et al., 2020). Examples of such services include EV charging, the delivery of goods, cleaning services such as car washes, or vehicle repairs. The AVP problem can thus be seen as a support to electric mobility as the demand for comfortable charging solutions increases, such as destination charging. Furthermore, the AVP problem benefits service station operators, such as charge point operators, as vehicles do not block the station after the service is performed, which may result in higher utilization rates. Challenges in scheduling for the AVP problem include, for instance, the stochastic arrival of new customers, large problem instances with numerous customers and service stations, and a limited number of service stations. Scheduling is a decision-making process aimed at allocating resources to tasks over a given time period while optimizing for one or more objectives (Pinedo and Dordrecht, 2016). In the context of the AVP problem, a task is a service requested by a customer, and the resources are service stations (as a generalization of charging stations). The objective is to handle as many services as possible within the constraints imposed by the customers' dwell times, as well as the servicing times and the number of service stations.

The AVP problem is unique but related to known scheduling problems, such as the flexible job shop problem (Biaiewicz et al., 1996) in the sense that it has flexible machine allocation, the open shop problem (Gonzalez and Sahni, 1976) regarding the absence of precedence constraints, or the shunting problem (van den Broek et al., 2022) in terms of deadlines. These classes of scheduling problems are well established in the literature, with various extensions, such as the flow shop problem with an identical sequence of tasks for every job (Soukhal et al., 2005). Further, there are various objectives such as minimizing the makespan (e.g., Kis et al. (2010)), minimizing earliness or tardiness (e.g., Grimes and Hebrard (2011)), or minimizing the sum of completion times of jobs (e.g., Anand and Panneerselvam (2015)).

The shunting problem with service requests, on the other hand, is a more specific problem that describes how train components are parked and serviced during off-peak hours. Similarly, to the AVP problem with service requests, they include limited capacity and the possibility of omitting specific requests. However, other than the AVP problem, they are deterministic in the sense that arriving and departing trains, the available processing time, as well as service requests are known in advance (van den Broek et al., 2022). Shop floor scheduling or shunting problems and their derivatives are complex combinatorial problems (Zhang et al., 2019). Similarly, the AVP problem is hard to solve. In an AVP setup, the arrival rate of customers determines the available computing time to find a feasible solution, which can be below one minute (e.g., Abdel-Aal (2020); Klappenecker et al. (2014)). Therefore, an efficient approach is needed to address the trade-off between runtime and quality of results.

Constraint programming (CP) is a possible approach to solving the AVP problem. The advantage of the CP approach is that it provides a feasible schedule quickly and improves the result over time (Pinedo and Dordrecht, 2016). It belongs to the artificial intelligence approaches, intending to effectively reduce the solution space (Zhang et al., 2019). The structure of the AVP problem can be translated into a constraint satisfaction problem, similar to a job shop problem (Da Col and Teppan, 2019), and has the advantage that additional constraints can be easily integrated to address real-world requirements (Beck et al., 2011), such as tardiness (Grimes and Hebrard, 2011). We propose a two-stage solution approach that first reduces the number of pending service requests by finding a set of feasible service requests and then improves the associated schedule.

In this study, we contribute to theory and practice by formalizing the AVP problem and by proposing possible solution approaches. We benchmark our solution method based on constraint programming against different heuristics on a case study. The results further help parking garage operators to quantify the benefit of integrating AVP systems with charging stations and additional services.

2 Problem definition and notation

For the AVP problem, we consider three components with different properties relevant for the problem definition. A **customer** is equivalent to a vehicle, has a departure time, and a set of AVP service requests. An **AVP service request** is associated with a set of compatible service stations. Each AVP service request is non-preemptive and can have individual characteristics, such as the amount of energy requested or the quality of a service, e.g., standard or extensive car wash, which are incorporated into the property **processing time**,

which is specific for each service station associated with the service request. A **service station** provides AVP services at a fixed location within the parking garage and is compatible with a subset of service types. Each service station is capable of handling one AVP service at a time. Figure 2 in Sect. 4 provides an exemplary layout of such a parking garage that we later use in our simulation.

The corresponding scheduling approach has to address three decisions: (1) the allocation, which defines the service station that will handle a customer service request for an AVP service, (2) the order in which the AVP services of a single customer are handled, and (3) the order in which AVP services are handled on a single service station. The three decisions are interrelated, and decisions (2) and (3) can be represented by the single decision variable *start time*.

Based on this introduction, we formally introduce the AVP problem.

$$\max \sum_{J_i \in J} \sum_{O_{i,j} \in G_i} \sum_{M_k \in F(O_{i,j})} a_{i,j,k} \quad (1)$$

$$\text{subject to} \quad \sum_{M_k \in F(O_{i,j})} a_{i,j,k} \leq 1 \quad \forall J_i \in J, \forall O_{i,j} \in G_i \quad (2)$$

$$s_{i,j} + \sum_{M_k \in F(O_{i,j})} (p_{i,j,k} + c) \cdot a_{i,j,k} \leq s_{i,l} \quad \text{or} \quad s_{i,l} + \sum_{M_k \in F(O_{i,l})} (p_{i,l,k} + c) \cdot a_{i,l,k} \leq s_{i,j} \quad \forall J_i \in J, \forall O_{i,j}, O_{i,l} \in G_i \text{ with } j \neq l \quad (3)$$

$$s_{i,j} + p_{i,j,k} \leq s_{a,b} \quad \text{if } a_{i,j,k} = 1 \text{ and } a_{a,b,k} = 1 \quad \text{or} \quad s_{a,b} + p_{a,b,k} \leq s_{i,j} \quad \text{if } a_{i,j,k} = 1 \text{ and } a_{a,b,k} = 1 \quad \forall J_i, J_a \in J, \forall O_{i,j} \in G_i, O_{a,b} \in G_a \text{ with } (i, j) \neq (a, b), \forall M_k \in M \quad (4)$$

$$s_{i,j} + c + \sum_{M_k \in F(O_{i,j})} p_{i,j,k} \cdot a_{i,j,k} \leq d_i \quad \forall J_i \in J, \forall O_{i,j} \in G_i \quad (5)$$

$$s_{i,j} \geq B(M_k) \quad \text{if } a_{i,j,k} = 1 \quad \forall J_i \in J, \forall O_{i,j} \in G_i, \forall M_k \in M \quad (6)$$

$$s_{i,j} \geq C(J_i) \quad \text{if } a_{i,j,k} = 1 \quad \forall J_i \in J, \forall O_{i,j} \in G_i, \forall M_k \in M \quad (7)$$

$$s_{i,j} \geq 0 \quad \forall J_i \in J, \forall O_{i,j} \in G_i \quad (8)$$

$$a_{i,j,k} \in \{0, 1\} \quad \forall J_i \in J, \forall O_{i,j} \in G_i, \forall M_k \in F(O_{i,j}) \quad (9)$$

In every iteration of the AVP problem with a known set of customers (i.e., every time the problem is solved after the arrival of a new customer), there are a set of n customers $J = \{J_i\}_{1 \leq i \leq n}$, a set of m service stations $M = \{M_k\}_{1 \leq k \leq m}$, and a set of AVP service request tuples $G_i = \{O_{i,j}\}_{1 \leq j \leq O(i)}$, where O_i defines the total number of AVP service requests of a customer J_i , which are assumed to be known when the customer enters the parking garage. Within the AVP system, the start times $s_{i,j}$ of service requests are determined. Every service request has a given processing time $p_{i,j,k}$ at service station M_k , and the corresponding customer must drive to the service station, which is assumed to take time c . The overall objective is to schedule as many services as possible during operation while the arrival and characteristics of new customers are unknown. We divide this problem into two stages. First, we determine the feasible service requests. In a second stage, we create slack in the system by trying to schedule feasible service requests identified in stage 1 as early as possible. Service requests can be assigned to the set of compatible service stations (similar to a flexible job shop problem (Zhang et al., 2019)). $a_{i,j,k} \in \{0, 1\}$ denotes whether a service request $O_{i,j}$ is assigned to service station M_k . An overview of all parameters and decision variables is provided in Table 1.

Based on this notation, we formulate the constraints of the AVP problem. We start in stage 1 with objective function (1) that maximizes the number of service requests that are successfully scheduled. First, Constraint 2 ensures that every service request can at most be assigned to one service station. Second, the no-overlap Constraint 3 for customers ensures that the start time $s_{i,j}$ of the j 'th service request of customer J_i , $O_{i,j}$, is scheduled after any previously started AVP services for customer J_i are completed. It is conceptually represented in the optimization model and can be solved using a Big M formulation. This completion time includes the time c needed for driving to and from the station. The start time is always relative to $t = 0$, the time the latest customer entered the system. Constraint 4 then ensures that there is no overlap on any machine M_k from any customers. Here, we do not need to consider the driving times. The order in which AVP services are provided is not important, i.e., there is no precedence constraint requiring that one AVP service be completed as an input for another service.

A service request $O_{i,j}$ can be assigned to any service station M_k within the set of compatible service stations $F(O_{i,j}) \subseteq M$.

Every customer within the AVP system is parked for a limited time only, which is why a deadline for completing all services for customer J_i is needed. We assume that the departure time (i.e., dwell time) d_i is known in advance. Such deadline constraints are also used in related scheduling problems such as JSP and FJSP (e.g., Balas et al. (1998, 2008)) where the objective is to minimize the sum of delayed ser-

Table 1 Nomenclature for AVP problem

Indices and parameters	
$J = \{J_i\}_{1 \leq i \leq n}$	Set of customers
$M = \{M_k\}_{1 \leq k \leq m}$	Set of service stations
$G_i = \{O_{i,j}\}_{1 \leq j \leq O_i}$	Set of AVP service requests of customer J_i
$G'_i = \{O_{i,j}\}_{1 \leq j \leq O_i}$	Set of AVP service requests of customer J_i that are successfully scheduled in optimization stage 1
O_i	Total number of service requests of customer J_i
$O_{i,j}$	Service request j of customer J_i
$F(O_{i,j}) \subseteq M$	Set of compatible service stations for AVP service request $O_{i,j}$
$p_{i,j,k}$	Processing time of service request $O_{i,j}$ of customer J_i on service station M_k
d_i	Departure (dwell) time of customer J_i
c	Constant driving time to service station
Decision variables	
$s_{i,j}$	Preliminary start of service request $O_{i,j}$ of customer J_i after optimization stage 1
$s'_{i,j}$	Start of service request $O_{i,j}$ of customer J_i
$a_{i,j,k} \in \{0, 1\}$	Allocation of service request $O_{i,j}$ to service station M_k
$a'_{i,j,k} \in \{0, 1\}$	Allocation of service request $O_{i,j}$ to service station M_k in stage 2
Derived variables	
$B(M_k)$	Remaining blocked time of service station M_k at arrival of new customer
$C(J_i)$	Remaining servicing time of customer J_i at arrival of new customer

vices (e.g., Bahroun et al. (2018)) or the aggregated tardiness of services (e.g., Zhu and Heady (2000)), while all services have to be performed. In the AVP problem, not every service request is necessarily completed at departure, as the customer may have more service requests than the departure deadline allows, or service requests from other customers may block the service stations. Constraint 5 ensures that only those service requests get scheduled that can be performed before the dwell time is over.

A challenge of the AVP problem is the lack of foresight regarding customer arrivals and the demand for service requests. We assume that customers arrive randomly at the AVP system and that there is uncertainty regarding their arrival, dwell time, and service requests. This creates characteristics of online optimization, requiring us to consider the state of the AVP system at the time of optimization. As a consequence, the AVP problem has to be solved every time a customer arrives. To address this requirement, we include two blockage constraints that represent prior online commitments of the AVP system in the AVP problem formulation. As services are non-preemptive, service requests cannot be interrupted or modified until they are completed once started. This includes driving toward the service. Thus, when a car has started to move toward a service, it can no longer be interrupted. While service requests that have not yet begun processing can be rescheduled or canceled altogether, started service requests must be completed. When new customers arrive in this situation, the service requests of the currently serviced customer and the corresponding

service station are not available at $t = 0$ when the AVP problem is solved with updated information. This is ensured in Constraint 6 for the service station and Constraint 7 for the customer. The function $B(*)$ returns the time until which the service station M_k is blocked with successfully scheduled and already begun service requests, i.e., the time it still takes to complete the service request allocation $a_{i,j,k}$. $C(*)$ does the same for customer J_i that is being serviced at the time of optimization.

In our proposed formulation of the AVP problem, the main objective is to successfully schedule as many services as possible, i.e., to maximize the number of executed services indicated in the objective function (1). Other objectives are conceivable, such as maximizing the total execution time of services or the total value of services scheduled by assigning a weight to each service.

Since there is no information available on future arrivals, we cannot accurately anticipate future demand in the AVP problem. Therefore, even if we find an optimal solution given the currently available information under the current state of the system (i.e., scheduling all service requests of all customers present in the parking garage), this does not guarantee an optimal operation of the AVP system under the circumstances of new arrivals (e.g., once the next 100 customers have arrived in the system). For instance, scheduling a service request later during a customer's dwell time, when it could also have been scheduled earlier, results in the same number of completed services. However, delaying the service request could lead to blockage of the service station when

new customers with new service requests arrive. To address this, we use an iterative approach. In optimization stage 1, we determine starting times $s_{i,j}$ of all feasible service requests as an initial solution. These starting times are then improved in the second optimization stage to increase slack in the system, resulting in starting times $s'_{i,j}$. Therefore, in the second stage, we consider a reduced set of service requests G'_i for customer J_i that only contains service requests $O_{i,j}$ with $\sum_{M_k \in F(O_{i,j})} a_{i,j,k} = 1$. Using this set, we then minimize the completion time of all service requests by determining $s'_{i,j}$ and $a'_{i,j,k}$ through the objective function (10). It ensures that every service request is scheduled as early as possible, maximizing the available slack in the AVP system. Other than that, the constraints are the same, with one adjustment: In Constraint 11, $a'_{i,j,k}$ now has to be equal to one for each tuple of customer, service request, and machine to ensure that all service requests in the sets G'_i for customer J_i are successfully scheduled.

$$\min \sum_{J_i \in J} \sum_{O_{i,j} \in G'_i} \sum_{M_k \in F(O_{i,j})} (s'_{i,j} + c + p_{i,j,k}) \cdot a'_{i,j,k} \quad (10)$$

$$\text{subject to} \quad \sum_{M_k \in F(O_{i,j})} a'_{i,j,k} = 1 \quad \forall J_i \in J, \forall O_{i,j} \in G'_i \quad (11)$$

$$s'_{i,j} + \sum_{M_k \in F(O_{i,j})} (p_{i,j,k} + c) \cdot a'_{i,j,k} \leq s'_{i,l} \\ \text{or} \\ s'_{i,l} + \sum_{M_k \in F(O_{i,l})} (p_{i,l,k} + c) \cdot a'_{i,l,k} \leq s'_{i,j} \\ \forall J_i \in J, \forall O_{i,j}, O_{i,l} \in G'_i \text{ with } j \neq l \quad (12)$$

$$s'_{i,j} + p_{i,j,k} \leq s'_{a,b} \quad \text{if } a'_{i,j,k} = 1 \text{ and } a_{a,b,k} = 1 \\ \text{or} \\ s'_{a,b} + p_{a,b,k} \leq s'_{i,j} \quad \text{if } a'_{i,j,k} = 1 \text{ and } a_{a,b,k} = 1 \\ \forall J_i, J_a \in J, \forall O_{i,j} \in G'_i, \\ O_{a,b} \in G'_a \text{ with } (i, j) \neq (a, b), \forall M_k \in M \quad (13)$$

$$s'_{i,j} + c + \sum_{M_k \in F(O_{i,j})} p_{i,j,k} \cdot a'_{i,j,k} \leq d_i \quad \forall J_i \in J, \\ \forall O_{i,j} \in G'_i \quad (14)$$

$$s'_{i,j} \geq B(M_k) \quad \text{if } a'_{i,j,k} = 1 \quad \forall J_i \in J, \\ \forall O_{i,j} \in G'_i, \forall M_k \in M \quad (15)$$

$$s'_{i,j} \geq C(J_i) \quad \text{if } a'_{i,j,k} = 1 \quad \forall J_i \in J, \forall O_{i,j} \in G'_i, \\ \forall M_k \in M \quad (16)$$

$$s'_{i,j} \geq 0 \quad \forall J_i \in J, \forall O_{i,j} \in G'_i \quad (17)$$

$$a'_{i,j,k} \in \{0, 1\} \quad \forall J_i \in J, \forall O_{i,j} \in G'_i, \forall M_k \in F(O_{i,j}) \quad (18)$$

This defines the AVP problem. To further contextualize the problem, Table 2 summarizes the relevant constraints and compares them with the flexible job shop and open shop problems.

Table 2 Properties of the optimization problems

Property	Flexible job shop	Open shop	AVP problem
Precedence constraints	✓	×	×
No overlap	✓	✓	✓
Prior online commitments	×	×	✓
Deadline constraints	×	×	✓
Flexible machine allocation	✓	×	✓

3 Solution approach and benchmarks

As described, we use two subsequent objective functions in the formulation of the AVP problem. The rationale of this approach is not only grounded in concerns for the quality of the solution, which is potentially improved as the second objective ensures a higher system flexibility when new customers arrive. It is also based on runtime considerations. As customers continuously arrive and as we re-optimize the system every time new information becomes available in the form of a new customer (which might similarly be understood as additional requested services), solutions need to be quickly determined before the next arrival. In our case study, we award not more than 15 seconds to each of the optimization stages. This is necessary to ensure that a new solution is duly available for new customers as they arrive.

The two-stage approach contributes to this, as it splits the decision into determining the set of feasible service requests and determining a time-efficient schedule. Determining the feasible services is a less complex decision. On the other hand, a valid schedule can be challenging to compute, which is facilitated by excluding services that cannot be scheduled anyway.

Besides the design of the AVP optimization problem, the choice of scheduling algorithm and its implementation also have an impact on the quality of the results and the computational speed of the system. Therefore, the objective is to find a relatively good feasible solution in a reasonable time. For such applications, constraint programming (CP) has garnered considerable attention in recent research. The advantage of the CP approach is that it provides a feasible schedule quickly and improves the result over time. A schedule is feasible if it complies with all constraints of the AVP problem, but it may not be the optimal solution of the objective functions. For solving the AVP problem, this means that a time limit can be used to provide a faster response from the scheduling approach and to ensure the successful operation of the system. In our evaluation, we also include a set of runs with a warm start based on the result of the Earliest-Due-Date-First heuristic to make the best use of computational resources (Ball, 2011) and to determine whether this approach can improve results.

3.1 Benchmarks

As the optimization of the AVP problem with CP cannot guarantee an optimal solution, both overall and in each optimization stage, we use benchmarks to assess the quality of the solution. These benchmarks are two static heuristics (First-Come-First-Served and random) based on simple rules to schedule services, two dynamic heuristics that are recalculated every time new customers arrive (Earliest-Due-Date-First and Least-Slack-First), and two technical benchmarks that serve as upper and lower bounds. To provide a lower bound for the AVP problem solution, we calculate the scenario where no automation is implemented, representing current parking garages where vehicles do not move after being parked. That means they can use at most one service and then block the service station for the remainder of their dwell time. For the upper bound, we implement an optimization with complete information about customer arrivals and dwell times, as well as their service requests. Using these approaches, we can calculate a lower benchmark called *no-avp* and an upper benchmark with *complete information* to assess the quality of the AVP problem solution. In addition, using the four heuristics, we can evaluate if and in which scenarios the AVP problem formulation can improve the AVP system operation beyond computationally less expensive solutions. Note that we assume the use of autonomous driving in the parking garage, i.e., the repositioning of vehicles, for all benchmarks except for *no-avp*. When running the heuristic approaches, the available machine with the shortest processing time is selected for service requests with different processing times at different service stations. For example, if a fast or slow charger can perform the service and both are currently available, fast charging is selected. Using the heuristic approaches, if not otherwise specified, the service requests are scheduled in the order they appear in the set of service requests G_i of customer J_i .

First-come-first-served In the First-Come-First-Served (FCFS) heuristic, customers arriving first are assigned to service stations in the order they arrive. The strength of the FCFS heuristic lies in its ability to create dense schedules, as arriving vehicles are instantly scheduled if possible, or the first available service station is assigned to the next service request. This approach has been proven to provide good results in smart charging scenarios without information on future trips (Flath et al., 2012) and is therefore a promising benchmark solution. The FCFS heuristic follows a similar objective to the second objective function of the AVP problem formulation, which is to schedule services as early as possible, thereby providing flexibility for future solutions. In our implementation of the FCFS heuristic, the algorithm first determines the set of service stations with the earliest possible start time and then chooses the station with the shortest completion time. This decision is particularly relevant for

services where multiple machines are capable of providing the service at different speeds, such as charging stations with both slow and fast charging capabilities.

Random The random heuristic serves as a lower benchmark for the scheduling algorithms, addressing the question of whether any optimization of the AVP problem, as proposed in this paper, is useful or whether being able to move vehicles autonomously is already sufficient. In this scheduling approach, a random possible start for the service request is selected, where every time $t \leq t_{\max}$ has the same probability of being selected, as long as the service request is completed before the customer's departure and both the customer and service stations do not have overlapping services. The random approach does not prioritize any service station.

Earliest-due-date-first The Earliest-Due-Date-First (EDD) approach is a dynamic heuristic. In contrast to the FCFS heuristic, it recalculates the schedule every time new customers arrive at the AVP system. It is similar to the FCFS heuristic in the sense that it provides dense schedules. Every time a service station completes a service request, the customer with the earliest departure time and demand for that service is scheduled next. Therefore, there are no gaps in the schedules as long as there are feasible service requests for the service stations. As for the FCFS heuristic, the service station with the shortest processing time is selected if several service stations are available to process a service request.

Least-slack-first The Least-Slack-First (LSF) approach is a second dynamic heuristic. It also recalculates the schedule every time new customers arrive at the AVP system, but instead of the customer with the earliest departure time, the customer with the least slack (i.e., difference between dwell time and total required servicing time) is scheduled next. Instead of servicing requests in the order of appearance in the set of service requests, as for the other heuristics, we scheduled the shortest service from this set first to address the objective of maximizing the number of scheduled services. This is intended to provide a comparison between heuristics and ensure that we have not overlooked any quick-win improvements. The service station with the shorter processing time is selected if several service stations are available to process a service request.

No AVP While the three heuristics are used to benchmark the quality of the AVP problem's solution, we use the *no-avp* benchmark as a lower bound for the AVP system overall. It quantifies whether an AVP system has any value at all compared to today's parking reality. Without an AVP system, the customer performs the first feasible service request in the list, possibly among a set of service requests, if a corresponding service station is available upon arrival. If a set of possible service stations is available, the customer chooses the station with the shortest service processing time. The customer is then parked at the service station and blocks the station until departure. In case no service station is available upon

arrival, the customer parks in a regular parking lot. Note that this approach allows the customer to satisfy only one service request.

Complete information Unfulfilled service requests can be caused by poor scheduling decisions or simply system constraints. To distinguish between these, we implement the *complete information* benchmark. We apply a variation of the AVP problem. We limit the optimization to the first objective function as the system needs no flexibility, and use Constraint 7 to account for the time until arrival, during which customers cannot fulfill any service requests. The complete information optimization is, of course, not limited to 15 seconds of computing time, as it is meant to serve as a benchmark. However, even though the optimization is limited to the first objective function, not all scenarios can be computed in a reasonable time. We therefore impose a time constraint of four hours and use the solution found within that time frame. We also use CP to solve the complete information benchmark. A detailed analysis of the runtime, including a justification for the four-hour restriction, is provided in Sect. 5.2.

4 Case study

To evaluate the AVP problem and the proposed solution approaches, we design a case study. We include three different types of service requests, which we call charging, washing, and loading a vehicle, for illustrative purposes.

4.1 Simulation

Setup The simulation is based on the process-based discrete-event simulation framework *SimPy* implemented in *Python* (Team SimPy, 2021). We use an event-based simulation setup, i.e., the starting times are continuous. The simulation is thus advancing from event to event. Every time new information becomes available in the form of a new customer, the AVP problem is solved again, and the dynamic heuristics (EDD, LSF) are reapplied, or, in the case of the static heuristics, just applied to the new customer. Every time the event *service request completed* occurs, the service request is counted as successful. To implement the CP solution approach for the AVP problem, including the warm start, we use OR tools (Da Col and Teppan, 2019).

Every execution of the CP for the AVP problem is limited to 15 seconds for each of the two optimization stages (i.e., finding and rescheduling feasible service requests). In other words, if not more than two vehicles arrive per minute, the AVP system is capable of recalibrating its solutions.

The simulation process is illustrated in Fig. 1. Whereas in our AVP problem formulation, we have included the driving time between services as a constant, we calculate the actual driving path in the simulation.

4.1.1 Design

To define the scenarios, we differentiate between parameters that specify the infrastructure at the parking garage and those that describe the customers.

Infrastructure Every parking garage is designed differently, and no standard layout is available. This has led to the development of multiple techniques to find the optimal layout for a particular parking lot (Young, 1988). To provide generalizable findings for the AVP problem and to reduce the impact of specific parking garage designs, we base our design on a structural element generally found in parking garages, where vehicles are parked in islands of two, with each vehicle having its own access to the driveway. We scale this design to incorporate 10 bays with 10 vehicles parked in each row. An illustration of the parking garage is shown in Fig. 2, where the gray nodes represent driveways and the green nodes represent parking lots. In total, this results in a parking garage with 200 parking spaces. The entrance is located on the lower left (brown node), and the exit is on the upper left (blue node).

We position the charging and washing service stations in the outer left bay. The number of service stations varies per scenario. In scenarios with more than 10 service stations, we repeat the same allocation, moving one bay to the right, until all service stations are located in the parking garage. As some service stations may be located in a remote location, we positioned the loading area at the bay on the right side.

In the basic setup, we include twelve charging stations, out of which four are fast chargers (50 kW) and eight regular chargers (11 kW). Additionally, we add ten service stations for vehicle washing and four loading stations. Whereas the charging stations provide a single service with varying speeds, the washing and loading stations can offer services of different quality. We differentiate between a small and a large service, where the washing takes 15 or 30 minutes, and the loading takes 5 or 10 minutes, respectively. The services provided in this simulation are fictional and could be replaced with others. The combination of different services with variable processing times is an important requirement in an AVP system, and hence, it is included in the case study.

User behavior We base our simulation on a medium-sized parking garage and use scenarios to evaluate the impact of different utilization rates. Unless otherwise stated in the scenarios, we simulate 1000 vehicles with an arrival rate of 60 vehicles per hour. However, the arrival rate is modified in some scenarios, in which case we refer to it as λ . On average, each vehicle has a dwell time of 90 min, with a minimum of 60 and a maximum of 120 min. The dwell time is a random variable drawn from a uniform distribution. We assume a 30% chance of a customer requesting each service, with a 50% chance for the small and large services, respectively. The amount for each charging request is randomly drawn from a uniform distribution within a 4 kWh interval centered

Fig. 1 Interaction of a vehicle with the AVP system

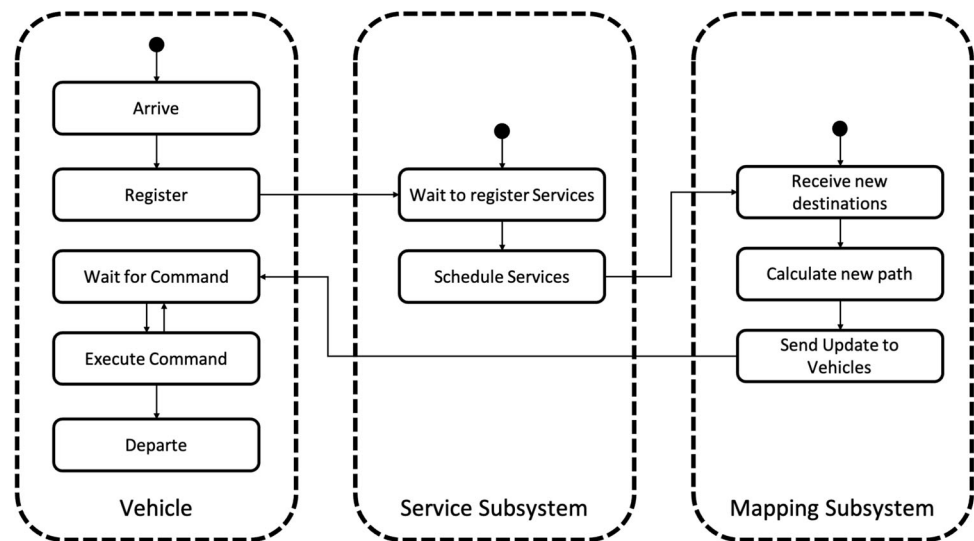
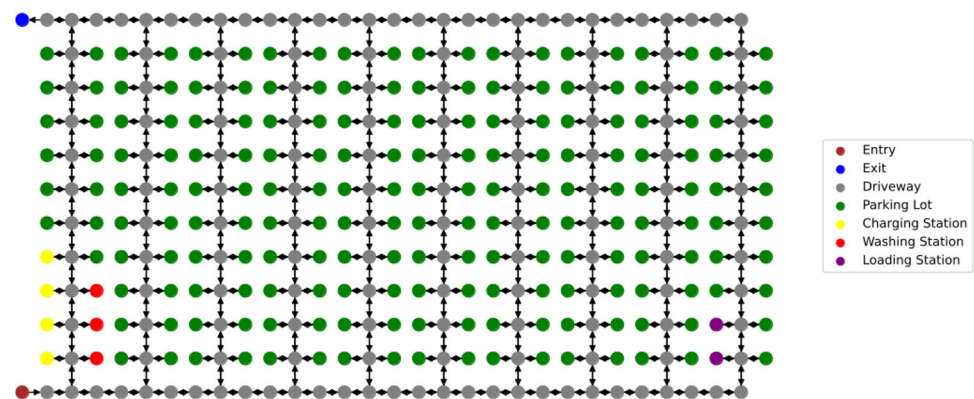


Fig. 2 Exemplary parking garage Layout with 200 parking lots



on the mean of 13.3 kWh (Electrive and Net, 2020). The realization of the simulation environment then determines the total duration of the simulation, $t_{\max}$, which extends until the last vehicle has left the parking garage.

4.1.2 Scenarios

The utilization of parking garages varies from day to day and within a single day. This raises the question of how scheduling approaches can handle different utilization rates and whether there are advantages for some approaches given certain utilization levels. The customer set is generated at the beginning of the simulation, including an arrival and departure time for each customer, as well as the set of service requests for each customer. This also allows us to calculate $t_{\max}$. To reduce the driving time within the parking garage before the subsequent service request is fulfilled, the closest idle parking spot to the next scheduled service station is assigned. We use constant driving times to the service stations. This can also be achieved with a lookup table that stores the driving times from all parking spaces to all service stations, thereby improving the representation and complexity

of the system. We implement three sets of scenarios that vary concerning the following parameters.

Arrival rate Parking garages can experience on- and off-peak utilization, where at certain times, a high number of vehicles enter the parking garage. To analyze how the different scheduling approaches handle these situations, we vary the average number of vehicles within the system. To this end, we use Little's Law to determine the average number of vehicles L in the parking garage in a steady state. For a given arrival rate and dwell time, Little's Law provides the average number of vehicles within the AVP system. The exact formulation is shown in Eq. 19a.

The capacity utilization C^{Util} is then calculated as the ratio of the number of vehicles L and the total number of parking lots in the parking garage $|PL|$. With a constant dwell time d inside the AVP system, we increase the average number of vehicles arriving per time unit λ to represent a utilization of 10% to 100% of the parking spaces.

$$L = \lambda \cdot d \quad (19a)$$

$$C^{\text{Util}} = \frac{L}{|PL|} \quad (19b)$$

Between the scenarios, the customers have identical probability distributions of having a specific service request. Overall, we analyze 10 individual scenarios, $P_{C^{Util}}$, with $C^{Util} \in [10\%, 20\%, \dots, 100\%]$, where C^{Util} describes the share of parking spaces utilized in a steady state.

Flexibility of customers With an increasing acceptance of AVP, the number of service requests from customers will likely increase. As there is no reason to assume that the dwell time will grow similarly, the time flexibility for each customer within the AVP system decreases. Time flexibility in the context of this simulation refers to the ratio of the aggregated processing time of all service requests to the total dwell time of all customers, and it is a relevant factor, for example, in charging EVs (Huber et al., 2020). In the following, f_i calculates the flexibility of an individual customer J_i by dividing the sum of the fastest possible processing times of all services by the dwell time before subtracting this ratio from 1. In other words, if the customer has no service requests, the flexibility is 100%, and when the total number of service requests matches or exceeds the dwell time, the flexibility is 0%. We then calculate the average over all customers to determine the system's flexibility, f^{avg} .

$$f_i = \max(1 - \frac{\sum_{j=1}^{O_i} \min_{k \in F(O_{i,j})} P_{i,j,k}}{d_i}, 0) \quad (20a)$$

$$f^{avg} = \frac{\sum_{i=1}^n f_i}{n} \quad (20b)$$

By decreasing the average time flexibility from 80% to 0%, we gain insight into how the scheduling approaches cope with an increasing number of service requests from customers, including situations where the infrastructure is insufficient to handle every request. This leads to 9 individual scenarios $F_{f^{avg}}$ where $f^{avg} \in [80\%, 70\%, \dots, 0\%]$ describes the flexibility of an average customer (Eq. 20b).

Utilization of service stations Due to spatial or financial limitations, it may not be possible to increase the number of service stations to provide an adequate supply for customers. Within these scenarios, we evaluate how different scheduling approaches cope with varying utilization rates of the service stations by reducing the number of service stations for a constant demand. The utilization rate across all service stations is determined by the average aggregated processing time of service requests across all service stations that can service the request, divided by the maximum supply the service stations can provide during a time period t_{max} . We calculate the virtual average processing time for a service by dividing the sum of the processing times of the service across all corresponding service stations by the count of service stations available for the service. As we assume all types of service stations to have equal utilization rates, we can combine them in one equation and use the total possible processing time $|M| \cdot t_{max}$ of all available service stations, regardless of the

specific requested service.

$$u_M = \frac{\sum_{i=1}^n \sum_{j=1}^{O_i} \sum_{m \in F(O_{i,j})} P_{i,j,m} \cdot \frac{1}{|F(O_{i,j})|}}{|M| \cdot t_{max}} \quad (21a)$$

In total, we evaluate 10 utilization rates, U_{u_M} , where $u_M \in [10\%, 20\%, \dots, 100\%]$ describes the utilization of all different service stations as the ratio of the time needed for service requests to the time available for service requests. An overview of all scenarios and the parameters is provided in Table 3.

5 Results

The results presented in this section are calculated on virtual machines allocating four threads and 2GB RAM to each CP solver instance. The setup enables the independent simulation of each scenario without any interference during runtime. The CP solver algorithm benefits from more threads for parallelized execution, but the heuristics can only utilize a single CPU thread. Because the simulation outcome is based in part on random sequences, the results are based on 10 independent runs per scenario. Between those runs, we calculated the standard deviation of the share of successfully performed service requests to show that the 10 independent runs are sufficient. The maximum variance between the 10 independent runs across all solution approaches and scenarios was 3.13%.

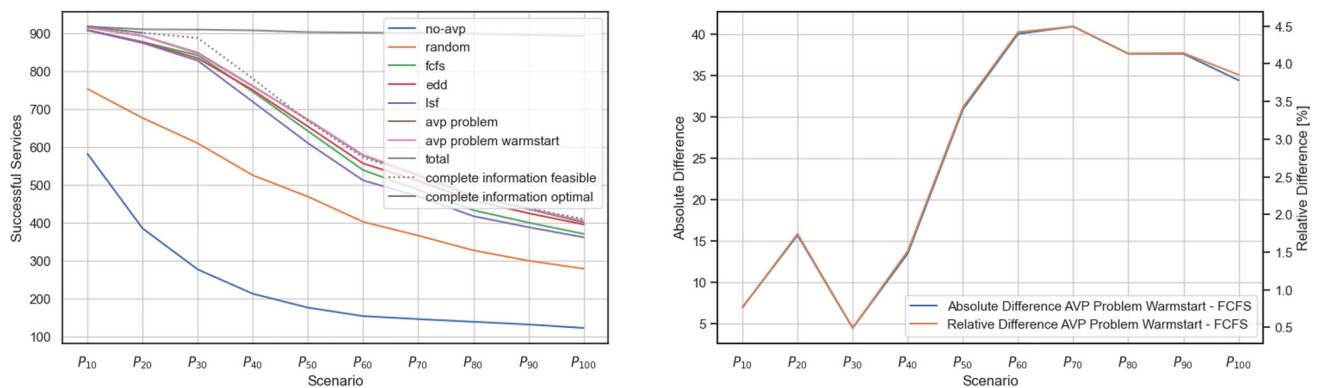
5.1 Scenarios

We compare the impact of increasing arrival rates, decreasing customer flexibility, and increasing utilization of the service stations. The results of each scenario for each solution approach (FCFS, random, EDD, LSF, no-avp, complete information, and the proposed AVP optimization with and without warm start) are presented in Figs. 3, 6, and 7. Each figure is split into two subfigures. The left figure illustrates the average number of successful services over all 10 independent runs for each scenario and scheduling approach. The data for complete information are split into scenarios where every run provided an optimal solution (continuous line) and scenarios where at least one independent run was not solved to optimality (dotted line) within the four-hour time limit. To highlight the performance of the AVP problem solution using CP with a warm start based on EDD in comparison with FCFS, the right subfigures show the absolute and relative differences between the two. Positive values signal the superiority of the AVP optimization.

Arrival rate In the scenarios $P_{C^{Util}}$ with $C^{Util} \in [10\%, 20\%, \dots, 100\%]$, the average utilization of parking lots in the parking garage is raised by increasing the arrival rate of customers. The results show that as the arrival rate increases, the

Table 3 Parameters of all scenarios

Scenario	Infrastructure				User behavior			
	Slow charging	Fast charging	Washing station	Loading station	Arrival rate[veh/h]	Prob. charging[%]	Prob. washing[%]	Prob. loading[%]
P_{10}	12	4	10	4	20	30	30	30
P_{20}	12	4	10	4	40	30	30	30
P_{30}	12	4	10	4	60	30	30	30
P_{40}	12	4	10	4	80	30	30	30
P_{50}	12	4	10	4	100	30	30	30
P_{60}	12	4	10	4	120	30	30	30
P_{70}	12	4	10	4	140	30	30	30
P_{80}	12	4	10	4	160	30	30	30
P_{90}	12	4	10	4	180	30	30	30
P_{100}	12	4	10	4	200	30	30	30
U_{10}	40	25	50	20	60	30	30	30
U_{20}	30	10	30	10	60	30	30	30
U_{30}	40	4	20	7	60	30	30	30
U_{40}	20	4	15	5	60	30	30	30
U_{50}	14	4	12	4	60	30	30	30
U_{60}	10	4	11	3	60	30	30	30
U_{70}	10	3	10	3	60	30	30	30
U_{80}	7	3	9	3	60	30	30	30
U_{90}	5	3	8	2	60	30	30	30
U_{100}	4	3	7	2	60	30	30	30
F_{80}	12	4	10	4	60	18	19	19
F_{70}	12	4	10	4	60	27	27	27
F_{60}	12	4	10	4	60	36	36	36
F_{50}	12	4	10	4	60	45	45	45
F_{40}	12	4	10	4	60	54	54	54
F_{30}	12	4	10	4	60	63	63	63
F_{20}	12	4	10	4	60	72	72	72
F_{10}	12	4	10	4	60	81	81	81
F_0	12	4	10	4	60	90	90	90

**Fig. 3** Results of scenarios $P_{C^{Util}}$ with $C^{Util} \in [10\%, 20\%, \dots, 100\%]$

number of service requests that are scheduled successfully decreases for all scheduling approaches. The worst performance is observed in the *no-avp* benchmark, with 64.6% of service requests successfully scheduled in scenario P_{10} and approximately 13.7% in scenario P_{100} . With an increase in arrival rate, the service stations are occupied more often at the arrival of new customers. The same trend is observed for the other scheduling approaches, but with a slower decline. The random heuristic provides better results than the *no-avp* approach, but lags behind the FCFS and AVP problem optimization approaches. With the random heuristic, vehicles can be relocated in the AVP system, and hence, more than one service request per customer can, in theory, be scheduled. Nevertheless, due to the random nature of the scheduling approach, gaps occur between services, resulting in time intervals where the service stations are not handling customers, despite remaining service requests. The reason for this is the non-preemptive service requests. As some gaps between scheduled service requests are shorter than the shortest service request processing time, there is no possibility to utilize the gap.

This is illustrated in Figs. 4 and 5, where two exemplary Gantt diagrams of a schedule using the FCFS and the random heuristic are illustrated using 100 vehicles. The figures show that the FCFS heuristic creates a “dense” schedule. This allows for a better utilization of the service stations, resulting in more handled service requests. As a consequence, the FCFS heuristic, as well as the EDD and LSF heuristics that generate similarly dense schedules, provide good results when compared to the random heuristic. They often perform similarly well as the AVP problem optimization approach without complete information. The differences between the FCFS and EDD heuristics, as well as the AVP problem, are minor. However, the AVP approach consistently outperforms the heuristics. Here and in the following, we observe that the AVP optimization with a warm start based on EDD slightly outperforms the AVP optimization without a warm start. The results show that the schedules created provide close to optimal results. In addition, Fig. 3 shows that the relative difference between the FCFS heuristic and the AVP solution with warm start increases with an increasing arrival rate. With more vehicles in the parking garage, the optimization has more potential to re-prioritize single service requests to create an optimal schedule. In contrast, the FCFS does not rearrange customers over time. The EDD heuristic shares this advantage with the AVP optimization, which might contribute to its better performance. Interestingly, the LSF heuristic does not perform equally well as EDD. One possible explanation is that the least slack as an indicator prioritizes the execution of longer services, whereas EDD will always try to add services for customers who depart soon. It is notable that in most scenarios with higher arrival rates, the AVP optimization is on par or outperforms the optimization with complete

information. This is, of course, caused by the non-optimality of the complete information optimization after the four-hour optimization window, but it remains notable that incremental optimization can compete with this comprehensive approach that has advantages in terms of information availability. The same applies to the following scenarios.

Flexibility of customers In the scenarios $F_{f_{avg}}$ with $f_{avg} \in [80\%, 70\%, \dots, 0\%]$, we increase the number of service requests of customers. Figure 6 shows that the total number of service requests increases linearly, whereas the successfully scheduled services in the *no-avp* approach stay the same. This is caused by the fact that for each customer, at most one service can be scheduled as described before. Using the random heuristic, on the other hand, the AVP system relocates vehicles, thereby increasing the number of service requests that are successfully scheduled. Due to the capacity limitations of the service stations, the benefit of additional service requests decreases, and the heuristics eventually reach a plateau. The results of the FCFS and EDD heuristics (which yield virtually the same performance) and the AVP problem optimization occur in three clusters. The first cluster covers scenario F_{80} and F_{70} . Here, the customer’s flexibility is high, so that both heuristics and the optimization provide close to optimal solutions. In these scenarios, the service stations are underutilized; hence, even a simple heuristic, such as the FCFS approach, provides optimal results. In the second cluster, covering scenarios F_{60} to F_{20} , the AVP problem optimization is capable of improving upon the heuristics. It creates a schedule that covers up to 2.5% or 25 service requests more than the FCFS and EDD heuristics. In the third cluster, covering F_{10} and F_0 , more service requests cannot be handled by the service stations and remain in the queue. Whereas in theory, the AVP problem optimization should expand its lead over the FCFS heuristic, the opposite is observed (even though only with a slight deterioration). Within the scenarios with the lowest customer flexibility, the AVP problem optimization loses some of its lead compared to the FCFS heuristic. This can be traced back to the runtime limitations of the CP optimization for the AVP problem, which are further analyzed in Sect. 5.2. Similar to the utilization scenarios, the LSF heuristic underperforms the other heuristics.

Utilization of service stations In the third set of scenarios U_{u_M} with $u_M \in [10\%, 20\%, \dots, 100\%]$, we decrease the number of service stations. The results of the *no-avp* benchmark and random heuristic follow a similar trend as in the previous scenarios. Whereas the *no-avp* benchmark provides the worst results, the random heuristic performs between the other heuristics and the *no-avp* benchmark. It decreases with the reduction of service stations. The FCFS and EDD heuristics, along with AVP problem optimization, provide an optimal result in the first two scenarios, U_{10} and U_{20} . The large number of service stations is capable of handling all service requests. Whereas the complete information illustrates



Fig. 4 Gantt diagram of a schedule based on the random heuristic

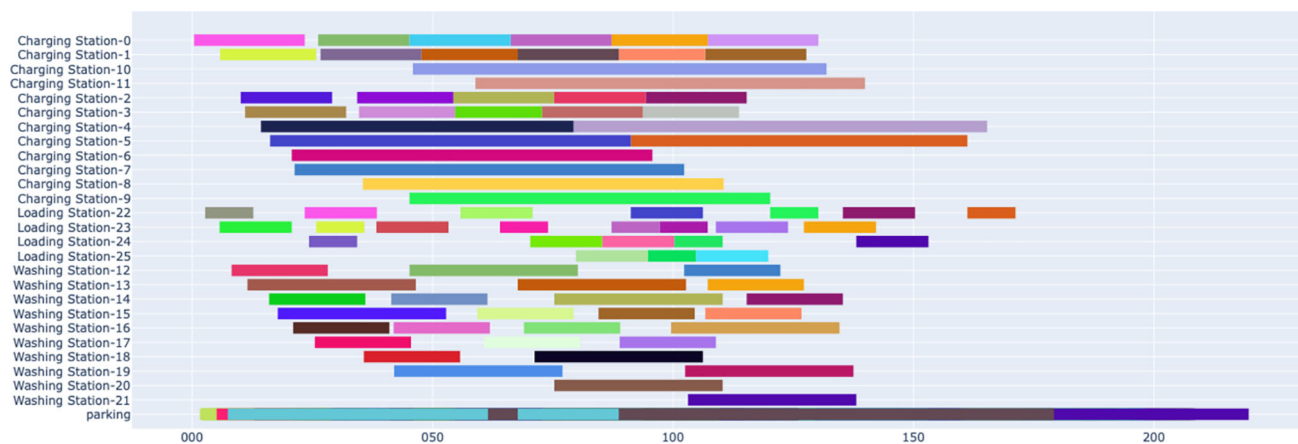


Fig. 5 Gantt diagram of a schedule based on the FCFS heuristic

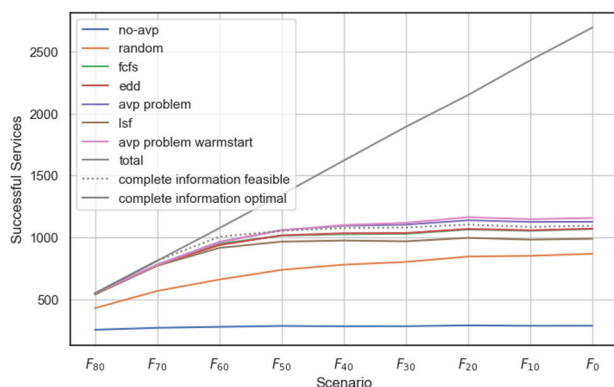
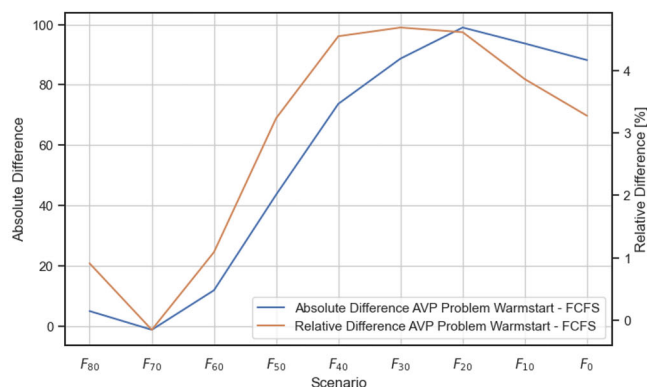


Fig. 6 Results of scenarios $F_{f^{avg}}$ where $f^{avg} \in [80\%, 70\%, \dots, 0\%]$

that up to scenario U_{50} , the infrastructure is capable of handling almost every service, the FCFS and EDD heuristics, as well as the AVP optimization, fail to schedule several services starting in scenario U_{30} . This trend continues with a further reduction of service stations, but the magnitude to which the two approaches are impacted differs. As illustrated in Fig. 7, the gap increases with a maximum above 30 scheduled ser-



vice requests or more than 4%, respectively, in favor of the AVP optimization. The graph on the right shows that the AVP with warm start consistently outperforms the FCFS heuristic, as the difference between scheduled services is always positive. This can be explained by looking at the input data of the simulation. In this set of scenarios, the number of customers and thus the service requests always enter the system at the

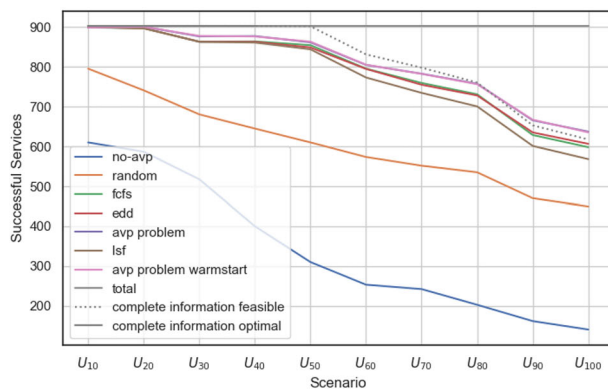


Fig. 7 Results of scenarios U_{u_M} with $u_M \in [10\%, 20\%, \dots, 100\%]$

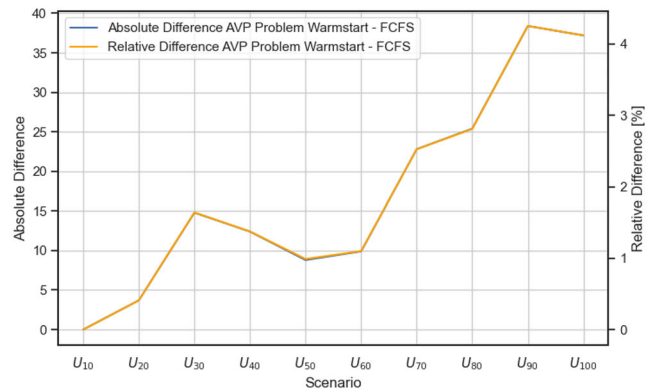
same rate, but the number of service stations decreases. As a consequence, two factors impact the quality of the AVP problem optimization. The number of services in the queue increases, leading to a corresponding increase in complexity, as discussed in the previous paragraph. However, the reduction in service stations simplifies the optimization problem. Additionally, our two-stage approach helps reduce the number of service requests that need to be scheduled in the second stage, as services that are technically infeasible are excluded after the first stage. Apparently, this trade-off works in favor of the AVP problem optimization, and the number of additional service requests scheduled increases compared to the FCFS and EDD heuristics.

5.2 Evaluation of solutions

Within each scenario, the CP-based AVP problem optimization is computed multiple times as customers arrive at the parking garage. Figure 8 shows the share of AVP problem instances with optimal and feasible solutions for the scenarios $P_{C_{Util}}$ within one simulation run. It illustrates that the CP can compute the optimal result only in scenarios with a long inter-arrival time. For scenarios P_{10} and P_{20} , more than 99% of optimization instances are optimal as only a few customers are present at the parking garage and the utilization of the service stations is low. This changes when customers arrive in quicker succession. As a consequence, only a fraction of AVP problem instances lead to optimal solutions. These instances occur at the beginning of the simulation, when the number of customers is still low. As parking garage utilization increases, the results no longer become optimal even within four hours of computation.

To illustrate this in further detail, we focus on scenario P_{50} , which represents a balanced number of customers, and analyze both the quality of the results and the improvements over time.

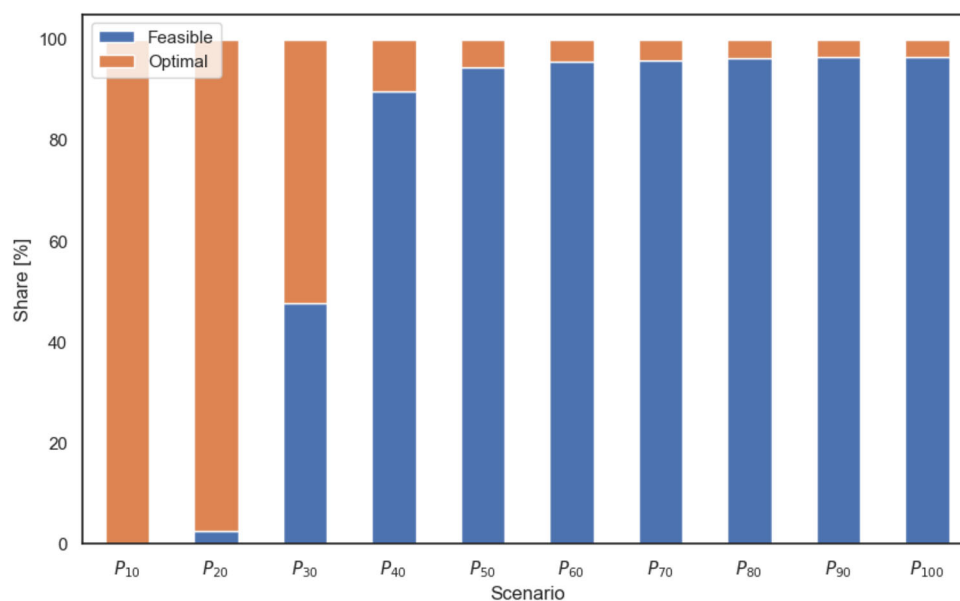
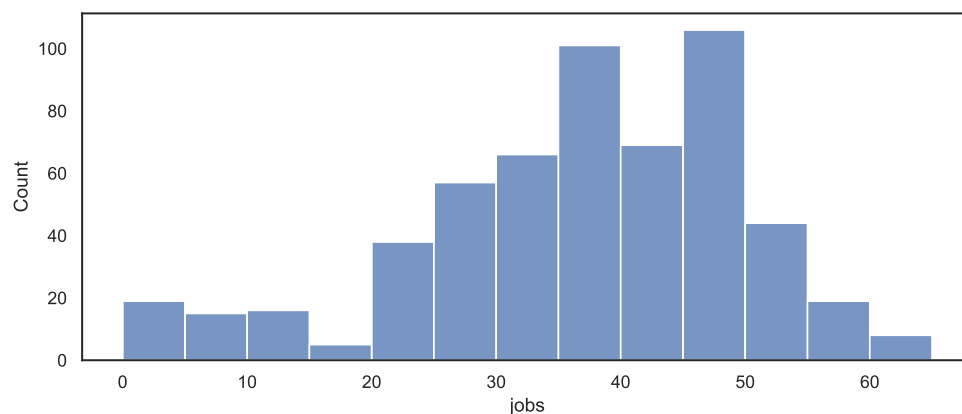
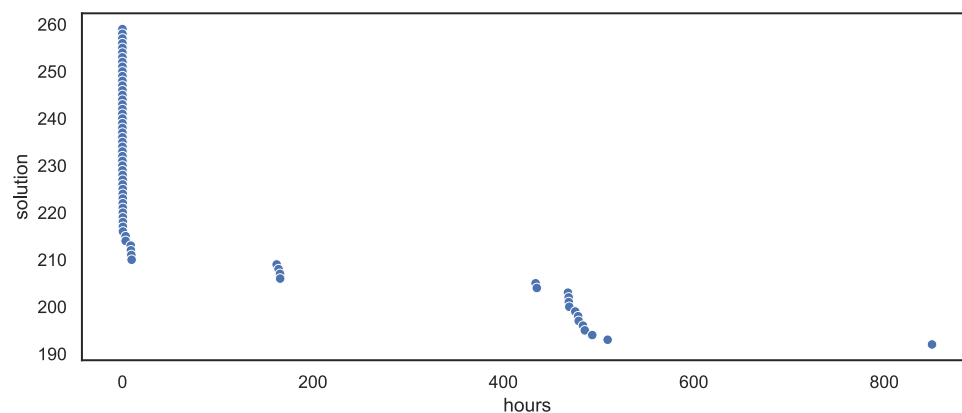
Quality of results Figure 9 presents a histogram of the number of customers handled within each AVP problem instance.



With an average of 45 and a maximum of 71, the number of customers is well above benchmarks for jobs in related scheduling tasks described in the literature, such as the job shop problem. It hence explains why the CP cannot find an optimal solution for the AVP problem within the 15-second time limit. This indicates why the FCFS and EDD heuristics, as well as the AVP problem optimization, perform similarly in specific scenarios. For the operation of a parking garage using an AVP system, this implies that when the number of customers reaches a threshold, the benefits of the AVP problem optimization decrease, and FCFS might be a viable alternative, as shown in scenario F_0 . In scenario P_{50} , this threshold is at a maximum of 11 customers in the queue, where the objectives of both stages of the AVP problem reach an optimal result. After 63 customers in the queue, even the first objective of our two-stage approach can no longer be solved to optimality. It illustrates the challenges of the AVP problem as it does not scale well with an increasing number of customers.

Impact of time The CP approach to the AVP problem finds feasible solutions and improves them over time. This is illustrated in Fig. 10, where every solution found by the algorithm along the timeline is shown for Scenario P_{50} with complete information. The figure highlights that the CP approach does not improve the result linearly but rather in steps with no improvement in between. This makes any statements on the quality of the result challenging, as no asymptotic behavior is visible.

Particularly at the beginning, CP can constantly improve the result, and in total 45 solutions were found after four hours, with an improvement from 259 to 214 missed service requests. Whereas the next 4 hours provide no new solution, the objective is down to 210 after 10 hours and to 192 after 850 hours. Consequently, we need to determine a cut-off point, as an optimal solution was not found within the first 1000 hours. We used 4 hours for our AVP problem optimization with complete information, as it provides sufficient results with a tolerable runtime.

Fig. 8 Quality of results**Fig. 9** Distribution of the number of customers handled in scenarios**Fig. 10** Result improvement of CP optimization of AVP problem with complete information over time

6 Discussion and outlook

In our formulation and evaluation of the AVP problem, we make several assumptions, which are discussed below. We focus on the successful scheduling of service requests. Other objectives are conceivable, such as the monetary value of completed services. In our setup, shorter services might be

prioritized over longer services as this increases the objective. There are arguments for and against this approach, and it might be interesting to compare the optimization for different objectives. More importantly, under the CP optimization of the AVP problem (as well as the dynamic EDD and LSF heuristics), customers do not receive a guarantee that their service requests will be completed when they enter the park-

ing garage (this is different from the FCFS heuristic, for example). This might not be acceptable in actual service environments. Improvements to the problem could consider scheduled services as required to be completed, but movable within the schedule. Due to the innovative nature of AVP and the extension with service requests, there is limited insight into user acceptance of such systems. Further research is necessary to gain a deeper understanding of customer needs and acceptance. Further, we assume that customers cannot remove services from their set of service requests (while adding service requests could be implemented as a new arrival). Furthermore, we assume the dwell time to be known and deterministic, which is not always the case in practice. Future research could address this issue and develop a stochastic extension of the AVP problem. In the second stage of our two-stage solution approach, we minimize the total completion time. This might potentially lead to an over-reliance on faster service stations. Ideally, these would not be blocked by customers who, given their dwell time, could also use slower service stations when new customers arrive that potentially need fast service completion. Further research is required to better integrate this additional requirement into the second-stage optimization problem. We use CP to determine a valid and optimal schedule for the AVP problem. We selected CP because it has been successfully applied to related scheduling problems (Da Col and Teppan, 2019) and due to its ability to quickly return feasible solutions, while also being flexible enough to include further constraints (Beck et al., 2011). We also used warm starts based on the EDD heuristic to understand whether this would improve the results. However, we acknowledge that other solution approaches may lead to equal or better results and should be analyzed in future research.

7 Conclusion

In this study, we introduce the automated valet parking problem, which describes the scheduling of service requests for customers with autonomous vehicles within a parking garage. This is one of the earliest applications of autonomous driving, which can increase the utilization of parking infrastructure, including service stations providing services such as electric vehicle charging. To demonstrate the problem, we use constraint programming in a case study to solve the automated valet parking problem and benchmark the scheduling algorithm against First-Come-First-Served, Earliest-Due-Date-First, Least-Slack-First, and random heuristics. Additionally, we provide theoretical upper and lower bounds for the results through simulation of a parking garage without automated valet parking and a complete information optimization. The results show that constraint programming-based automated valet parking problem optimization with

a warm start provides the best results in most scenarios, especially when the optimization becomes more complex; however, it does not scale well in terms of optimality. The runtime of the First-Come-First-Served, Earliest-Due-Date-First, and Least-Slack-First heuristics is not affected by an increasing number of service requests and provides results in the range of the AVP optimization, with Earliest-Due-Date-First and First-Come-First-Served outperforming Least-Slack-First. Comparing the results against the lower and upper benchmarks reveals that incorporating automated valet parking into a parking garage increases the number of scheduled service requests, indicating the potential of automated valet parking systems. Using warm starts based on the Earliest-Due-Date-First heuristic did improve the results. Furthermore, there is ample potential to improve the automated valet parking modeling and scheduling approaches, for example, by including forecasting techniques or stochastic approaches.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Abdel-Aal, M. M. M. (2020). Survey-based calibration of a parking entry as a single-server mathematical queuing model: A case study. *Alexandria Engineering Journal*, 59, 829–838. <https://doi.org/10.1016/j.aej.2020.02.016>
- Anand, E., & Panneerselvam, R. (2015). Literature review of open shop scheduling problems. *Intelligent Information Management*, 07, 33–52. <https://doi.org/10.4236/iim.2015.71004>
- Bagloee, S. A., Tavana, M., Asadi, M., & Oliver, T. (2016). Autonomous vehicles: Challenges, opportunities, and future implications for transportation policies. *Journal of Modern Transportation*, 24, 284–303. <https://doi.org/10.1007/s40534-016-0117-3>
- Bahroun, Z., Shamayleh, A., Zakaria, R.S. (2018). Dynamic and flexible scheduling for a single machinetchnologies. In *2018 5th international conference on control, decision and information* <http://ieeexplore.ieee.org/servlet/opac?punumber=8362241>.
- Balas, E., Lancia, G., Serafini, P., & Vazacopoulos, A. (1998). Job shop scheduling with deadlines. *Journal of Combinatorial Optimization*, 1, 329–358.
- Balas, E., Simonetti, N., & Vazacopoulos, A. (2008). Job shop scheduling with setup times, deadlines and precedence constraints. *Journal*

- of Scheduling, 11, 253–262. <https://doi.org/10.1007/s10951-008-0067-7>
- Ball, M. O. (2011). Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, 16, 21–38. <https://doi.org/10.1016/j.sorms.2010.07.001>
- Banzhaf, H., Nienhuser, D., Knoop, S., Zollner, J.M. (2017). The future of parking: A survey on automated valet parking with an outlook on high density parking. In *2017 IEEE intelligent vehicles symposium (IV)* (pp. 1827–1834). IEEE. <https://doi.org/10.1109/IVS.2017.7995971>
- Baron, O., Berman, O., & Nourinejad, M. (2021). Introducing autonomous vehicles: Adoption patterns and impacts on social welfare. *Manufacturing and Service Operations Management*. <https://doi.org/10.1287/msom.2020.0955>
- Beck, J. C., Feng, T. K., & Watson, J. P. (2011). Combining constraint programming and local search for job-shop scheduling. *INFORMS Journal on Computing*, 23, 1–14. <https://doi.org/10.1287/ijoc.1100.0388>
- Błażewicz, J., Domschke, W., & Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93, 1–33. [https://doi.org/10.1016/0377-2217\(95\)00362-2](https://doi.org/10.1016/0377-2217(95)00362-2)
- Bosch. (2020). Stuttgart airport set to welcome fully automated and driverless parking. <https://www.bosch-presse.de/pressportal/de/en/press-release-219648.html>
- van den Broek, R., Hoogeveen, H., van den Akker, M., & Huisman, B. (2022). A local search algorithm for train unit shunting with service scheduling. *Transportation Science*, 56, 141–161.
- Da Col, G., & Teppan, E. C. (2019). Industrial size job shop scheduling tackled by present day cp solvers. In T. Schiex & S. de Givry (Eds.), *Principles and Practice of Constraint Programming* (pp. 144–160). Cham: Springer International Publishing.
- d'Orey, P.M., Azevedo, J., & Ferreira, M. (2016). Exploring the solution space of self-automated parking lots: An empirical evaluation of vehicle control strategies. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)* (pp. 1134–1140). IEEE. <https://doi.org/10.1109/ITSC.2016.7795699>
- Electrive.net. (2020). Die flotte als hebel für den erfolg der elektromobilität. <https://www.electrive.net/2020/05/28/die-flotte-als-hebel-fuer-den-erfolg-der-elektromobilitaet/>
- Flath, C. M., Ilg, J. P., & Weinhardt, C. (2012). Decision support for electric vehicle charging. In *Proceedings of the eighteenth Americas conference on information systems*. Seattle, Washington
- Gonzalez, T., & Sahni, S. (1976). Open shop scheduling to minimize finish time. *Journal of the ACM (JACM)*, 23, 665–679.
- Grimes, D., & Hebrard, E. (2011). Models and strategies for variants of the job shop scheduling problem. In *International conference on principles and practice of constraint programming* (pp. 356–372).
- Huber, J., Dann, D., & Weinhardt, C. (2020). Probabilistic forecasts of time and energy flexibility in battery electric vehicle charging. *Applied Energy*, 262, 114525. <https://doi.org/10.1016/j.apenergy.2020.114525>
- Kis, T., de Werra, D., & Kubiak, W. (2010). A projective algorithm for preemptive open shop scheduling with two multiprocessor groups. *Operations Research Letters* 38, 129–132. <http://www.sciencedirect.com/science/article/pii/S0167637709001308>, <https://doi.org/10.1016/j.orl.2009.10.007>
- Klappenecker, A., Lee, H., & Welch, J. L. (2014). Finding available parking spaces made easy. *Ad Hoc Networks*, 12, 243–249. <https://doi.org/10.1016/j.adhoc.2012.03.002>
- Klemm, S., Essinger, M., Oberlander, J., Zofka, M.R., Kuhnt, F., Weber, M., Kohlhaas, R., Kohs, A., Roennau, A., Schamm, T., & Zollner, J.M., (2016). Autonomous multi-story navigation for valet parking. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 1126–1133). IEEE. <https://doi.org/10.1109/ITSC.2016.7795698>
- Pinedo, M. (2016). Scheduling: Theory, algorithms, and systems. Fifth edition ed., Springer, Cham and Heidelberg and New York and Dordrecht and London. <https://doi.org/10.1007/978-3-319-26580-3>
- SAE International. (2021). J3016c: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles - sae international. https://www.sae.org/standards/content/j3016_202104/
- Schmidt, M., Staudt, P., & Weinhardt, C. (2020). Evaluating the importance and impact of user behavior on public destination charging of electric vehicles. *Applied Energy*, 258, 114061. <https://doi.org/10.1016/j.apenergy.2019.114061>
- Scholliers, J., Pyrkönen, P., Virtanen, A., Aittoniemi, E., Malin, F., Federley, M., & Nikolaou, S. (2020). Automated valet parking using iot. In *Proceedings of TRA2020, the 8th Transport Research Arena*, (pp. 68).
- Schulz, M., Riener, A. (2020). "Give me the keys, i'll drive!": Results from an exploratory interview study to assess public's desires and concerns on automated valet parking. In *12th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (pp. 16–18). <https://doi.org/10.1145/3409251.3411712>
- Schwesinger, U., Burki, M., Timpner, J., Rottmann, S., Wolf, L., Paz, L. M., Grimmett, H., Posner, I., Newman, P., Hane, C., Heng, L., Lee, G. H., Sattler, T., Pollefeys, M., Allodi, M., Valenti, F., Mimura, K., Goebelsmann, B., Derendarz, W., Muhlfellner, P., Wonneberger, S., Waldmann, R., Grysczyk, S., Last, C., Bruning, S., Horstmann, S., Bartholomäus, M., Brummer, C., Stellmacher, M., Pucks, F., Nicklas, M., & Siegwart, R. (2016). Automated valet parking and charging for e-mobility. In *2016 IEEE intelligent vehicles symposium (IV)* (pp. 157–164). IEEE. <https://doi.org/10.1109/IVS.2016.7535380>
- Soukhal, A., Oulamara, A., & Martineau, P. (2005). Complexity of flow shop scheduling problems with transportation constraints. *European Journal of Operational Research*, 161, 32–41. <https://doi.org/10.1016/j.ejor.2003.03.002>
- Team SimPy. (2021). Documentation for simpy. <https://simpy.readthedocs.io/en/latest/contents.html>
- VDA. (2021). Premiere at the IAA mobility 2021: The future of parking is autonomous.
- Wang, J., Huang, H., Li, K., & Li, J. (2021). Towards the unified principles for level 5 autonomous vehicles. *Engineering*. <https://doi.org/10.1016/j.eng.2020.10.018>
- Young, W. (1988). A review of parking lot design models. *Transport Reviews*, 8, 161–181. <https://doi.org/10.1080/01441648808716682>
- Zhang, J., Ding, G., Zou, Y., Qin, S., & Fu, J. (2019). Review of job shop scheduling research and its new perspectives under industry 4.0. *Journal of Intelligent Manufacturing*, 30, 1809–1830. <https://doi.org/10.1007/s10845-017-1350-2>
- Zhu, Z., & Heady, R. B. (2000). Minimizing the sum of earliness/tardiness in multi-machine scheduling: A mixed integer programming approach. *Computers and Industrial Engineering*, 38, 297–305. [https://doi.org/10.1016/S0360-8352\(00\)00048-6](https://doi.org/10.1016/S0360-8352(00)00048-6)
- Zips, P., Banzhaf, H., Quast, G., Kugi, A. (2020). Increasing the capacity for automated valet parking using variable spot width. In *IEEE 23rd international conference on intelligent transportation systems (ITSC)* (pp. 1–6). <https://doi.org/10.1109/ITSC45102.2020.9294185>