Article

Optimization of On-Demand Shared Autonomous Vehicle Deployments Utilizing Reinforcement Learning

Karina Meneses-Cime, Bilin Aksun Guvenc * and Levent Guvenc

Automated Driving Lab, Ohio State University, Columbus, OH 43210, USA
* Correspondence: aksunguvenc.1@osu.edu

Abstract: Ride-hailed shared autonomous vehicles (SAV) have emerged recently as an economically feasible way of introducing autonomous driving technologies while serving the mobility needs of under-served communities. There has also been corresponding research work on optimization of the operation of these SAVs. However, the current state-of-the-art research in this area treats very simple networks, neglecting the effect of a realistic other traffic representation, and is not useful for planning deployments of SAV service. In contrast, this paper utilizes a recent autonomous shuttle deployment site in Columbus, Ohio, as a basis for mobility studies and the optimization of SAV fleet deployment. Furthermore, this paper creates an SAV dispatcher based on reinforcement learning (RL) to minimize passenger wait time and to maximize the number of passengers served. The created taxi-dispatcher is then simulated in a realistic scenario while avoiding generalization or over-fitting to the area. It is found that an RL-aided taxi dispatcher algorithm can greatly improve the performance of a deployment of SAVs by increasing the overall number of trips completed and passengers served while decreasing the wait time for passengers.

Keywords: shared autonomous vehicles; mobility; traffic-in-the-loop simulation; optimization; reinforcement learning

1. Introduction

Recent research suggests that the deployment of shared autonomous vehicles (SAV) in the form of fleets of robo-taxis will be a natural and economically feasible way of introducing autonomous vehicles (AV) to the public that will help solve mobility problems of under-served areas and further positively impact network flow [1–4]. While current deployments of robo-taxis exist, much of the current literature utilizes simple models of AVs, or AV features such as adaptive cruise control (ACC) and cooperative adaptive cruise control (CACC), to extrapolate on the effects of penetration rates for AVs [5]. Furthermore, most reported work in the literature uses grid-based simulations [6] or analytical models [7] to decrease the computational demands of the simulation.

Emulating the European project, CityMobil2 [8], the early deployment of low-speed shuttles aimed as last- and first-mile solutions have taken place in Ann Arbor, Michigan [9], Arlington, Texas [10], San Ramon, California [11], and the Metro Phoenix Area [12]. In Columbus, Ohio, the smart circuit deployment of the Smart Columbus project operated in the Scioto Mile downtown area from December 2018 through September 2019 and provided over 16,000 rides with more than 19,000 miles traveled in [13]. The subsequent Linden AV deployment of the Smart Columbus project aimed at closing transportation gaps and providing access to public transportation, affordable housing, healthy food, childcare, recreation, and education was launched in February 2020 and was halted two weeks, due to an on-board incident and continued again afterwards. This project was named Linden LEAP (Linden Empowers All People) and distributed close to 3600 food boxes and more than 15,000 masks during the pandemic [13]. The Linden LEAP deployed autonomous shuttles had a static route and circled between the stops without on-demand capabilities. The static
The main goal and contribution of this paper is the development of a method of modeling and planning of SAV service in a geo-fenced urban area that does not have any public transportation. There are many minor contributions that serve this overall major contribution which are explained next. The developed method presented in this paper first models the traffic network and calibrates it based on available historical traffic data.
This is used to generate realistic other traffic in the simulation model of the geo-fenced area where the SAVs will operate. An SAV dispatcher, called the taxi dispatcher here, is then developed to optimize the operation of the SAV service using reinforcement learning to maximize the number of SAV trips and the number of passengers served while minimizing wait times in the presence of the other vehicle traffic which acts as a constraint. This overall approach is both scalable and replicable to other areas of operation and can also be used in a simulation study to determine the optimal size of the SAV fleet and the expected performance of a chosen fleet size based on the metrics mentioned above.

The organization of the rest of the paper is as follows. The relevant literature review is presented in Section 2. This is followed in Section 3 by the modeling of the urban area traffic network and its calibration, to be used for validation, along with the simpler traffic network, to be used for training. Section 4 presents the reinforcement-learning-based optimization method, namely the taxi dispatcher of this paper. The effectiveness of this reinforcement learning taxi dispatcher is shown in Section 5 using the simulation environment developed for validation. The paper ends with conclusions in Section 6.

2. Literature Review

The literature on SAVs can be divided into (1) literature regarding studies based on surveys and opinion papers, and (2) studies conducted utilizing simulation for analysis. The former concentrates on public opinion regarding the state of AVs, their adoption, and further possible implications by extrapolating from the information collected through surveys or workshops. Reference [20] provides ample background on the current state of literature regarding SAVs. The literature on adoption of AV technology shows that the way the public will be ready to adopt AV technologies is through the use of shared autonomous vehicles (SAVs). In turn, the factors that will guide the rate of adoption are the level of comfort, time, and cost. Reference [21] is a study composed of a survey of 2588 households with an over-represented demographic of participants between 18 and 24 and under-represented demographic of 65 and above. From their survey, it can be concluded that although young Americans are not confident with the use of AV technologies, opinions about the subject show a promise of rise in shared rides over time. Meanwhile, Reference [22] shows a promise of public acceptance to trade-off their current vehicles with shared autonomous electric vehicles (SAEVs). Likewise, Reference [23] shows partial support for successful adoption of AVs as part of mobility solutions via SAVs. It was also found that the optimization of comfort, cost, and time will increase the demand of SAV use. Reference [24] further agrees on the adoption of the technology favoring SAVs, and it was also found that the important factors to the users for adoption of SAVs are the determination of routes and co-passengers.

Meanwhile, studies regarding simulation of SAVs vary in variables of interest such as vehicle miles traveled (VMT), emissions, price, etc., as well as simulation type and penetration rate. In general, it is understood that the total replacement of vehicles with AVs generates higher road capacities. Furthermore, the performance of SAVs highly relies on the dispatching algorithm. Reference [25] assumed an SAV penetration rate of 3.5% and simulated a gridded city based on Austin. Furthermore, variables such as trip generation rates, and trip relocation strategies were varied. It was found that each SAV can replace 11 conventional vehicles while increasing travel distance on each SAV 10% more than a typical vehicle trip. Reference [26] utilizes a SUMO simulation to evaluate road capacity increase based on a 100% penetration rate of AVs. It was shown in that study that replacement of vehicles with AVs leads to an increase in capacity for existing traffic infrastructures. Furthermore, congestion and loss of time, which consequently increases the quality of the traffic flow, are reduced. However, even a single non-AV is able to create a state of congestion. Reference [27] shows a mixed mobility study with SAV integration. It was found that the mobility demand can be served with 10% of the vehicles. Furthermore, Reference [28] replaces all vehicles in Berlin with AVs and shows that a fleet of 100,000 vehicles is required to serve the demand currently served by 1.1 million vehicles. However, the
prices scale higher as the service is given to outskirt locations and remote locations with a single passenger. On a follow-up study, Reference [29] evaluates the potential for shared rides and the introduced algorithm is tested against a real-world data set of taxi requests in Berlin. It is found that vehicle miles traveled (VMT) can be reduced by 15–20%, while travel time increases can be kept at less than three minutes per person. Meanwhile, Reference [30] shows improvement of accessibility with the implementation of shared services as well as the reduction in transport costs. Regarding the performance parameters, Reference [31] shows that there is a strong correlation of performance and fleet size as well as shared or individual use deployment strategy.

On the optimization of the SAV dispatch, the work for optimizing the SAV dispatcher in the literature is limited. Reference [30] simulates an optimized SAV dispatcher by implementing a minimum set cover algorithm. Their algorithm converts the map of Lisbon to a grid simulation and considered all off-street parking facilities less than 5 min away from a station as parking stations for the SAVs. The dispatcher dispatched to multi-modal forms of transport and ran a local search algorithm maximizing the number of passengers’ completed requests of times and modes. Furthermore, the taxi dispatcher tried to insert passengers using the minimum insertion Hamiltonian path. Next, Reference [32] sought to optimize the data collected from vehicle rides in Florida by matching and maximizing number of passengers per vehicle trip while maintaining trip times. In their approach, the authors use on demand dynamic ride sharing in routing algorithms to maximize shared rides. In general, their taxi dispatching approach took the form of allocating as many passengers en route as possible while maintaining time restrictions. Reference [33] utilized the set-up of the interval scheduling problem and the use of the earliest finishing time (EFT) to extend these heuristics to car-sharing. The resulting formulation uses set-up costs, and time windows for jobs and shows that car-sharing allocation is an NP-hard problem. Lastly, Reference [34] used a simple dispatching algorithm that works well enough for simulation purposes, but the authors admittedly showcase room for improvement.

3. Modeling

There are many modeling features necessary to implement the realistic performance of a SAV fleet and its dispatcher. These include, the network, the SAV agents, their surrounding traffic, and requests. Additionally, the approach taken in this paper is to train a taxi dispatcher based on reinforcement learning principles. Thus, a second simulation environment must be created where the agent is able to learn and practice dispatching. Thus, first a simulation environment for validation will be created, and then, a simulation environment for training will also be introduced.

3.1. Validation Modeling

The Linden network of the Smart Columbus Linden LEAP autonomous shuttle deployment is used here to develop and demonstrate the RL taxi dispatcher. The approach for the validation model of the Linden network is to use an agent-based micro-simulation. Thus, the environment is first built; then, the environment is adapted with a fleet of SAVs, and consequently, the environment is adapted with a taxi dispatcher. The process to build the network is outlined in [35]. OpenStreetMap information was used to create a network version of the Linden area and with further work on intersections and areas of conflict, the network was ready for simulation. Ring and barrier systems were used to control the traffic lights. A ring and barrier system is a way to encode safe pairings of movements and define phases for control of the intersection. It was important to model such behavior as the network used for simulating was prone to traffic jams without the aid of traffic lights.

The first step was to divide the intersection phases into two groups which determine the placement of the first “barriers”. This can be performed by separating the phases into north- south and east–west phases. Next, the phases in each group were sorted based on the compatibility of the phases. That is, phases which are not compatible were further sorted into sequences which are called rings. The beauty of ring and barrier
diagrams is that adjusting the timing for the traffic signals becomes easy as the length of
the phases is adjusted at will as long as they are maintained inside their corresponding
barriers. This allows for more traffic flow by allowing different phases to proceed and to
be mix-and-matched. The next task is to decide the length of each phase. For this, the
county was contacted for information regarding their traffic lights and these phases were
adjusted accordingly. As an example, the diagram for the Cleveland and Eleventh Avenue
intersection of the Linden area can be seen in Figure 2. In this way, the entire Linden area
was adapted with ring-and-barrier signal intersections based on timings provided by the
city on the main road. The map can be seen in Figure 3.

![Figure 2. Full Ring and Barrier diagram for Cleveland Avenue and Seventeenth Street.](image1)

![Figure 3. Simulated Area of Deployment.](image2)

Furthermore, the network was calibrated based on vehicle capacity, route choice,
and system performance. The necessary field measurements were obtained through the
Mid-Ohio Regional Planning Commission (MORPC) site [36]. The data, however, were
limited so many decisions had to be made to create a tuned network. The method for
calibration used was to interpolate the observed information to populate base traffic for
the whole network. The necessary vehicle inputs were placed along Cleveland avenue,
Bonham Avenue, and Seventeenth Avenue. Next, routing decisions were placed along the
network so as to minimize the GEH statistic observed where observed data were provided.
The GEH statistic is a goodness-of-fit measure that quantifies the acceptable differences
between field observations and simulation observations. An acceptable GEH statistic in a
network is <5. Furthermore, temporal dependencies were introduced into the network by
interpolating from provided Peak Hour to Design Hour Factor report [37]. This adjusts the
total traffic in the network depending on time of day. An example of this distribution for an Urban Principal Arterial road can be observed in Figure 4.

**Total Urban Principal Arterial Hourly Percentage**

![Figure 4. Plot of percentage vehicles vs. time of day.](image)

Next, the network was adapted with six SAV pick-up/drop-off locations distributed along the network. The base of the SAVs is assumed to be Stephen’s Community center as this is where the SAVs recharged and stayed during non-working hours during the real deployment. Furthermore, an extension of the simulation was built using the COM interface as is described in [35]. The main working modules include the demand module, which allows the simulation of passengers placing requests; the routing module, which creates the paths for SAV travel; and the SAV management module, which models the dispatchment of SAVs according to different algorithms. These algorithms will be discussed in a later section. The main three modules are interconnected by the traffic and SAV simulation, which updates at the same rate as the rest of the modules. This structure can be seen in Figure 5.

![Figure 5. Extension Package Library written in Python [35].](image)

It is important to mention the level of SAVs simulated. In this paper, autonomous taxi (AT) is used often as an umbrella term for any kind of services involving an AV as a means to move passengers between two locations. However, it is recognized that there are many variations of ATs and each one deserves its own detailed model. Hence, in this paper, when referring to autonomous shuttles, or ASs, it is understood that an AS consists of an AV with a fixed set of locations. The order and paths taken to get to and from may vary, but the locations are fixed as it has been indicated in the past by placing “drop-off/pick-up” locations across Linden. Often, the implementations of ASs are on fixed routes due to the
limitations of the resulting geo-fencing of the regions. On the other hand, an AT is an AV that may stop at any desired spot through the area. These AVs have to inherently be more advanced than the current level 3 deployments mentioned in the beginning of this paper. Furthermore, an AS may have different kinds of requests. The deployed Linden shuttle was a shuttle operating on a fixed route. By comparison, the types of shuttles modeled in this paper are “on-demand” shuttles, which adapt their on-coming destination based on the passenger demand. A final distinction is that of single party occupancy AS and SAVs. Single party occupancy ASs may serve more than one passenger, but the driving logic will pick up any new passengers along the way. On the other hand, SAVs will plan their routes according to both picked-up and to-be-picked-up passengers along the way. Both single and shared occupancy AS will be compared in this paper. A summary of these distinctions is show in Figure 6.

![Figure 6. Variations in ATs.](image)

The simulation of SAVs begins by modeling an SAV agent. Each agent is regarded as an AV with an upper-level controller. The upper level controller receives requests from passengers with a time-stamp of when they are received. The upper-level controller is regarded as the SAV dispatcher and is tuned with parameters such as maximum wait time for pick-up and maximum wait time for drop-off. At each simulation time-step, requests are placed and the information is updated in the SAV dispatcher. The job of the SAV dispatcher is to send AVs to their destinations such that the number of passengers moved from stop to stop is maximized while the wait times are minimized. In this way, there exists one SAV dispatcher controlling a fleet of AVs and the SAV dispatcher hypothesized is a centralized dispatcher. An example of the regarded SAV dispatcher for three SAVs is shown in Figure 7.

![Figure 7. Three SAVs regarded as extensions of AVs by adding a centralized upper-level controller.](image)
Furthermore, the behavior of interest for the micro-simulation validation model is the parameterized Wiedemann’ 99 model. The Wiedemann’ 99 is a psycho-physical extension model of the stimulus-response model, or the Gazis–Herman–Rothery (GHR) family of models [38]. The GHR models propose that the follower’s acceleration is proportional to the speed of the follower, the speed of difference, and the space headway. The Wiedemann’ 99 model also defines further driver behaviors and parameters to account for the influence of the leading vehicle. Further research has been dedicated to tuning the behavior parameters by analyzing data from deployed AVs. In this paper, the recommended parameters for “Cautious,” “Normal,” and “Aggressive” behaviors outlined in [39] are used. Furthermore, the theorized vehicle model used in this paper is the same model utilized in deployments that were carried out by EasyMile [40]. This model has a passenger capacity per vehicle of 12. In the simulations, the capacity of passengers is extended to 15. A picture of the shuttle in mind is shown in Figure 8.

Figure 8. EasyMile Shuttle.

3.2. Training Environment Modeling

A second simulation model is introduced which is used for reinforcement training of the dispatch controller. The simulation model for RL is much simpler than the previously introduced environment. Python along with libraries such as Pygame were utilized to build a library in the style of OpenAI Gym [41]. This library was built on top of [42]. The library utilizes a different kind of microscopic vehicle simulator. This model considers the relationship of a lead-following vehicle. The model in question is the Intelligent Driver Model (IDM), which was introduced by [43], and allows for the computation of the acceleration of following vehicle based on the gap between vehicles. As a reference, consider Figure 9, which defines the names of the variables for leader and follower. Table 1 defines the necessary parameters for the IDM model.

The difference between AVs and regular surrounding traffic is that the simulated AV vehicle will behave carefully in relationship to the surrounding traffic. The minimum distance, \( s_{0,i} \), is increased, and the reaction time, \( T_r \), is decreased. Each AV introduced into the simulation was also adapted with a handle such that its behavior can be controlled from outside the simulation. The typical gymAI RL environments provides handles step, reset, and render while declaring the action and observation space. The simulation environment created provides these functions as object handles. Furthermore, pick-up/drop-off spots were also placed along the network as well as traffic light intersections to facilitate the flow of traffic. This resulted in the traffic simulator shown in Figure 10.
Table 1. IDM parameters.

| Variable | Definition |
|----------|------------|
| $s_{0,i}$ | Minimum distance between vehicle $i$ and $i-1$ |
| $v_{0,i}$ | Maximum velocity of vehicle $i$ |
| $\theta$ | Acceleration smoothness |
| $T_i$ | Reaction time of vehicle $i$ |
| $a_i$ | $i$th vehicle’s comfortable acceleration |
| $b_i$ | $i$th vehicle’s comfortable deceleration |
| $s^*$ | Desired distance between vehicles $i$ and $i-1$ |

Figure 9. Reference diagram for defined variables.

Figure 10. Created RL environment.

4. Optimization of Taxi Dispatcher

The approach to optimizing the SAV dispatcher began by gamifying it. The problem is first set-up as a Markov decision process (MDP). First, assume there is a fleet of $m$ SAVs ready for deployment. The network in question can be modeled as a directed connected graph $N = (V, E)$. Throughout the network, setup $n$ pick-up/drop-off stops. These are
enumerated $B = \{b_1, \ldots, b_n\}$ and can be located by placing it at a distance $d$ from the source node, as indicated by Figure 11.

![Figure 11. Mathematical representation of SAV pick-up areas placement in network.](image)

Next, throughout the hours of operation, the SAV travels between pick-up areas $b_i$ attending to the requests of passengers. The state of a taxi, $s$, is then representative of the base in which it is in. Furthermore, to provide the dispatcher with more information, the state $s$ is augmented with the state of the requests at pick-up/drop off areas. The pick-up-areas are enumerated $i \in [m]$ and a one-hot encoding representation is utilized where for vector $\theta$ each entry $\theta_i$ is given by

$$\theta_i = \begin{cases} 1, & \text{if there is at least one request coming from base } b_i \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Furthermore, let $N_p$ denote the number of passengers in vehicle $i$ and $P_{max}$ be the maximum number of seats in the vehicle. Then, the state of the vehicle at time $t$ is represented by

$$s_t = (b_t, \theta, N_p) \text{ where } b_t \in B, N_p \in [P_{max}] \quad (2)$$

while updating the simulation, it may be the case that the vehicle is driving. In this case, denote the driving behavior by setting $b_t = -1$. It will be assumed here that the dispatching process can be modeled as a Markov process so that a Markov decision process can be used later. We are assuming that the next dispatching request attended to depends only on the previous request attended to, which can be expressed mathematically in terms of the given probability state transition $P$

$$P(s_t|s_{t-1}, \ldots, s_0) = P(s_t|s_{t-1}) \quad (3)$$

being the same at every point in time or

$$P(s_t|s_{t-1}) = P(s_t'|s_t' - 1), \forall t'. \quad (4)$$

by establishing the dispatching process as a Markovian process, where it is easy to see that this leads to a natural transition into a Markovian decision problem. A Markovian decision problem requires an augmentation of a Markovian process with a set of actions and rewards functions. In the case of the dispatch problem, the set of actions is an array containing all the possible routes from the current pick-up spot.

Though not trivial, it is possible to assume that the path chosen from one pick-up spot to another is always the shortest path with respect to distance. This path is found by creating a directed graph representation of the network, where nodes equal intersections and edges equal roads. Then, Dijkstra’s algorithm is applied to find the shortest path. Dijkstra’s algorithm is well-known in the computer science field and creates a tree of paths by calculating the shortest path from one node to every other node [44]. This simple algorithm runs with time complexity $O(|E| + |V| \log |V|)$, where $|E|$ is the number
of edges and $|V|$ is the number of vertices of the graph representation and suffices to calculate a shortest path. This action is performed at the beginning to plan routes.

Next, the action function is represented by

$$a(s) = s'; \ s, s' \in S$$

(5)

where $S$ is the set of all possible states. The reward function is represented by

$$r(s, a) = r_{s, a}; \ r_{s, a} \in R$$

(6)

The reward function is further crafted as follows. At every step of the Markovian decision problem, the agent decides to take action $a$. This action represents a destination with the knowledge that the SAV is standing at pick-up stop $i$ and has requests vector representation $\theta$. Furthermore, denote a time limit per request, $T_{\text{max}}$. This is the maximum amount of time a passenger is willing to wait and is assumed to be fixed. Note that the list of requests is updated at each time step to eliminate requests that have exceeded the maximum wait time. Let $t_{\text{req}}$ be the time the SAV took to serve a request. During a simulation, it is possible that an outdated request pends for some small amount of time if the SAV is deployed before $T_{\text{max}}$ arrives. Thus, this case is also addressed in the mathematical formulation by considering when $t_{\text{req}} \geq T_{\text{max}}$. Then, the request reward function for a single taxi $i$, is

$$r_i = \begin{cases} 
-1 & \text{if } \theta_i = 0 \\
0 & \text{if vehicle } i \text{ is driving} \\
0 & \text{if } \theta_i = 1 \text{ and } t_{\text{req}} \geq T_{\text{max}} \\
N_p \frac{T_{\text{max}}}{100} \log \left( \frac{T_{\text{max}}}{t_{\text{req}}} \right) & \text{if } \theta_i = 1 \text{ and } t_{\text{req}} < T_{\text{max}} 
\end{cases}$$

(7)

Each reward function only rewards vehicles when they complete the request before the set maximum amount of time. If the vehicle arrives at a destination with no requests, there is a negative reward as this indicates the vehicle has wasted time driving to a place with no requests. While vehicle is driving, it cannot accumulate new rewards. Furthermore, higher rewards go to requests completed earlier and with more passengers picked up. The rewards function for $N_p = 1$ is shown in Figure 12.

![Single Reward Function for AT](image)

**Figure 12.** Reward function for taxi dispatcher.

The total reward is the average of the reward collected by every vehicle or,

$$r_{\text{tot}} = \frac{1}{N} \sum r_i$$

(8)
Coupling the Markovian decision problem with the reward and action function, the dispatch problem can be represented as an RL problem. The applied formulation is a Q-learning algorithm [45]. Q-learning can be applied to a Markovian decision problem, and it has been shown earlier in this section that a taxi dispatcher agent can be modeled as a Markovian decision problem. Q-learning will then find an optimal policy for the taxi dispatcher according to the specified reward function. Consider a standard Q-learning algorithm with variables reward discount $\gamma$, learning rate $\alpha$ and exploration rate $\epsilon$. The Q-learning update rule is

$$ Q(s, a) = (1 - \alpha)Q(s, a) + \alpha[r_{t+1} + \gamma E_{\pi}(\max Q^*(s', a'))] \quad (9) $$

The corresponding algorithm is shown in Algorithm 1.

**Algorithm 1 Q Learning Algorithm**

1. Initialize Q-table of size (states, actions)
2. Choose reward discount, learning rate and exploration rate
3. Do
4. Choose random number between 0 and 1
5. If random number is less than exploration rate
6. Choose random action
7. Else
8. Choose maximum Q-action
9. Perform chosen action
10. Observe
11. Update Q entry based on previously defined Q-update
12. Until
13. Reward threshold is achieved

$\gamma > 0$ is the reward discount and represents how much weight current rewards have vs. past rewards. $\gamma$ is set to 0.4 for the beginning iterations since the algorithm will count the immediate rewards much heavier than long-term rewards. Once training reaches above 400 episodes, it slowly increases $\gamma$ until $\gamma = 0.95$. This can be observed in Figure 13. $\alpha > 0$ is the learning rate and represents how fast the new Q-value observed is adopted. $\alpha$ is slowly decreased over time from 0.9 to 0.2. This can be observed in Figure 14. $\epsilon > 0$ chooses the odds in which a random action is chosen, i.e., the exploration vs. exploitation rate. Figure 15 shows the chosen learning rate over episodes.
5. Simulation

The introduced RL algorithm was trained in the Python environment before transferring the learning to the Linden network. The algorithm was trained over 1000 episodes which results in about 500,000 steps. The normalized training rewards for the first and last episodes are shown in Figure 16. The RL algorithm was trained for \( m = 4 \) and 10 SAV stops. Then, it was extended to encompass the case of six stops, and two- and four-size fleets. The SAV dispatcher can be regarded as a function \( f \) that maps current requests at time \( \text{req}_i \) to next deployment destinations for each vehicle \( i, b_{i,\text{next}} \). The variable \( \text{req}_i \) consists of a three-tuple containing the time the request was placed, \( T_{\text{req}_i} \), the pick-up spot, \( b_{\text{up}_i} \), and the drop-off spot \( b_{\text{off}_i} \). Thus, under policy \( \pi \)

\[
 f_\pi : (T_{\text{req}_i}, b_{\text{up}_i}, b_{\text{off}_i}) \to (b_{1,\text{next}}, \ldots, b_{m,\text{next}}) 
\]  

Policy \( \pi \) is dependent on the number of vehicles and shuttle stops. The Linden network was built such that it contains six SAVs. For the case when four shuttles are deployed, it is assumed that for four stops, the state of the one-hot encoding representation is \( \theta_i = 0 \) at all times. Thus, the SAV dispatcher never sends SAVs to these locations.
When two SAVs are deployed, at each time step, the state of two SAVs is regarded as driving. In other words, for two chosen indices,

\[ s_t = (-1, \theta, N_p), \forall t \]  \hspace{1cm} (11)

When six SAV are deployed, two RL dispatchers are used. Initializing dispatchers \( f_1 \) and \( f_2 \), the overall dispatcher, function \( f \) under policy \( \pi \), is extended such that each one tends toward requests that will benefit each vehicle’s reward score. Let \( f_1 \) control vehicle set \( v_1 = \{v_1, v_2, v_3\} \) and \( f_2 \) control vehicle set \( v_2 = \{v_4, v_5, v_6\} \). This choice was made to make the problem more tractable and to introduce cooperation within vehicle sets. In a multi-agent reinforcement learning approach, each SAV (agent) would be controlled separately, being motivated by its own rewards and taking actions for its own interests as opposed to the interests of the other SAVs.

The handling of the network presented gives the abstract version of the Linden network shown in Figure 17. This is a directed graph with weighted edges equal to the distances between pick-up/drop-off spots in the network. The distance between a vehicle \( v_i \) and a request \( req_{t,j} \) given its source stop, \( b_{up} \), \( d(req_{t,j}, v_i) \) is given by the weight of the connected edge from the vehicle’s current pick-up/drop-off spot to \( req_{t,j} \). This is always defined as it is computed a priori, at the beginning of the day. Furthermore, it is also assumed that there is always a path between nodes in the graph as shown previously.

The requests \( req_t \) are divided into two queries. First, define a radius of \( r = 10 \text{m} \); then, for every request \( req_{t,j} \in req_t \), compute set of vehicles \( V_r \) such that

\[ V_r(req_{t,j}) = \{v_i | d(req_{t,j}, v_i) \leq r\} \]  \hspace{1cm} (12)

Then, create lists \( req^1 \) and \( req^2 \) such that

\[ \exists v_i \in (V_r(req_{t,j}) \cup v_{f_1}) \Rightarrow req_{t,j} \in req^1 \]

\[ \exists v_i \in (V_r(req_{t,j}) \cup v_{f_2}) \Rightarrow req_{t,j} \in req^2 \]  \hspace{1cm} (13)

for enumerated requests \( req_{t,j} \in req_t \). For requests \( req_{t,j} \) such that \( V_r(req_{t,j}) = \emptyset \), set

\[ i \mod 2 = 1 \Rightarrow req_{t,j} \in req^1 \]

\[ i \mod 2 = 0 \Rightarrow req_{t,j} \in req^2 \]  \hspace{1cm} (14)

Based on the defined distribution of requests, an extended version of policy \( \pi \) can be described by

\[ \pi_f = \begin{cases} f_1 & \text{if } req_{t,j} \in req^1 \\ f_2 & \text{if } req_{t,j} \in req^2 \end{cases} \]  \hspace{1cm} (15)

To obtain a deployable algorithm, the list of requests is appended with the drop-offs. Furthermore, a new variable \( T_{off,MAX} \) is defined as a soft drop-off deadline. In the simulations used, half the number of dispatchers as the number of deployed vehicles were used. The completed dispatcher can be seen in Figure 18.
The SAV dispatcher is compared to a simple first-come, first-serve algorithm, which provides service to passengers based on the order of requests placed. The decision-making process for serving the passengers can be seen in Figure 19. This simple dispatcher is
used for benchmarking purposes. The motivation behind its use for benchmarking in this paper is as follows. The Linden area of Columbus, Ohio, which was modeled in this study, has no public transportation. Two autonomous shuttles operated at very low speeds on a fixed route as a circulator service meaning that they only served one main route shown in Figure 1. This autonomous fixed route circulator autonomous shuttle service, despite its limitation in service area and very long wait and trip times, was found to be very useful in this transportation underserved urban area. Even this simple on-demand first-come, first-serve algorithm that serves the whole geo-fenced area around Linden shown in Figure 3 offers a significant upgrade to mobility as compared to just a fixed route circulator service. It should also be noted that AV technology companies currently offer mobility service that is either a fixed route circulator or on demand first-come, first-serve service on fixed routes. Since our aim is to develop an SAV dispatch algorithm that can be compared with current practice, a first come first serve algorithm was chosen as the benchmark.

Figure 19. First-come, first serve SAV dispatcher.

The simple dispatcher has no option of shared passengers. Furthermore, the simulations were conducted over a 24 h day period. The first simulation results show the average
wait time over all number of trips. Due to the small size of the Linden network, it was found that higher number of AVs deployed could easily cause a traffic jam if the AVs deployed were acting cautiously. This is due to the low-speed of the vehicles. However, this speed was not changed within the “cautiously” deployed AVs since this behavior resembles the currently deployed Linden LEAP shuttles. Thus, it is imperative to pay close attention to the introduced size and behavior of the fleet.

For the performance of the dispatcher, it can be observed that the number of completed trips double with the addition of the RL SAV dispatcher in comparison with a simple dispatcher. The dispatcher tends to flatten the performance of the distributed trips. This is shown in Figure 20. The average wait time is also optimized by the RL SAV dispatcher. The average wait time decreases to 10–20 min while the regular dispatcher sees wait times of up to an hour but of an average of 25 min. The fleet wait time performance linearly improves with the size of the fleet. See Figure 21 for details. In Figure 21, the average wait time of 2SAVs of simple dispatch is observed to be less than that of SAV RL dispatch in cautious and normal driving modes. The RL SAV dispatcher tries to maximize number of trips while minimizing wait times. Please note that for the two cautious and normal SAVs, the number of completed trips were much higher as seen in Figure 20 for the RL dispatcher as compared to the simple dispatcher. Since the cautious and normal SAVs have slow speeds, it takes them much longer to serve more trips trying to increase ridership with only two SAVs because of the larger travel distance per SAV that is needed. As the number of SAVs is increased, the higher number helps as the travel distance required per SAV to service more passengers is lower and the wait times are less for the RL dispatcher. It can be observed that the drop-off wait time of the fleet also linearly improves with the size of the fleet. The average number of passengers served is also linear with the size of the fleet. These observations are shown by the numbers in Figure 22.

![Figure 20. Number of trips of simple dispatcher vs. SAV dispatcher.](image-url)

![Figure 21. Wait time of simple dispatcher vs. SAV dispatcher.](image-url)
was shown that the SAV dispatcher optimization could reduce passenger wait times and which trained an agent to become a SAV dispatcher. The variability introduced two to the wait time for passengers. Affect the vehicle routes or obstacles such as the ones discussed before introduce further jams or closed lanes where the dispatching agent cannot perform decisions. Future work of over-fitting. Further limitations include obstacles in the travel links such as traffic jams and slow down times for every vehicle in the network, including other SAVs.

Furthermore, the number of passengers served improves mostly linearly with the improved driving logic. It was shown that the SAV dispatcher optimization could reduce passenger wait times and increase the number of served taxi requests, thus benefiting the passengers utilizing the service. The number of trips double with optimized logic while wait times also decrease to an average of 15 min. The fleet size is linearly proportional to the wait times, while the average completed number of trips mostly increases by 60 with better driving logic. Furthermore, the number of passengers served improves mostly linearly with the improved dispatching logic. Any gaps and jumps observed within the data can be attributed to the driving logic of the vehicles and the size of the fleet. With a larger size of fleet, the driving logic should side further with an aggressive AV driver as too many too slow vehicles cause traffic jams and slow down times for every vehicle in the network, including other SAVs.

The dispatching logic can further be improved by introducing temporal dependencies. Since the RL agent was trained over variable traffic flow conditions, it is possible to improve the performance by categorizing temporal dependencies and by further creating multiple temporal-dependent behaviors. The independence of training under different network geometries could also introduce further improvements by decreasing the chance of over-fitting. Further limitations include obstacles in the travel links such as traffic jams or closed lanes where the dispatching agent cannot perform decisions. Future work includes extending the dispatching logic to networks where high levels of traffic density affect the vehicle routes or obstacles such as the ones discussed before introduce further choices thereby introducing dynamic route assignments. However, overall an RL-aided taxi dispatcher algorithm can greatly improve the performance of a deployment of SAVs by increasing the overall number of trips completed and passengers served while decreasing the wait time for passengers.

In this paper, the SAVs were modeled at a higher level based on their cautious, normal, or aggressive autonomous driving systems. Our future work will also take a look at

Figure 22. More stats on SAV dispatcher.

Using image processing, it is possible to extract useful information about roadways and vehicles travelling on roadways. CAVs and UAVs can benefit from this information in terms of fuel economy, ride comfort, and mobility. Since information regarding traffic flow, average vehicle speed, presence of work zones, as well as queues and accidents can be detected using cameras in conjunction with UAVs, a companion computer setup with a camera was prepared.

6. Conclusions, Limitations, and Future Work

Deployment of SAVs can greatly increase the mobility of the people affected by the technology. However, great attention should be paid to details such as size of fleet and expected driver behavior. Furthermore, these technologies should be optimized for deployment based on constraints that affect the passengers. Without well-developed and tested technologies, the deployments of SAVs can harm the perception of new technologies and delay the adoption of such. In this paper, a Q-learning based algorithm was developed, which trained an agent to become a SAV dispatcher. The variability introduced two to six SAVs with improved dispatching logic while also varying the AV driving logic. It was shown that the SAV dispatcher optimization could reduce passenger wait times and increase the number of served taxi requests, thus benefiting the passengers utilizing the service. The number of trips double with optimized logic while wait times also decrease to an average of 15 min. The fleet size is linearly proportional to the wait times, while the average completed number of trips mostly increases by 60 with better driving logic.
these SAVs at a lower level and focus on the controls [46], collision avoidance [47], safety evaluation and energy efficiency [48] of their autonomous driving systems. Robust control methods that have been used for autonomous driving [49–52] being successful in other demanding applications [53–57] can be used to handle uncertainties in modeling and external disturbances. Safety evaluation will need to use model-in-the-loop, hardware-in-the-loop and vehicle-in-the-loop simulation evaluation.

Reinforcement learning instead of deep reinforcement learning was applied in this paper as deep reinforcement learning was taking an extremely long time to train in the computer used. Our future work will also investigate the use of deep reinforcement learning for solving the optimal SAV taxi dispatcher problem. Some recent examples of deep reinforcement learning applied to the taxi dispatching problem are presented in references [58,59]. It should be noted that these references treat the ride hailing taxi dispatching problem for regular taxis driven by a driver and do not consider autonomous taxis with consideration for their autonomous driving (cautious, normal, and aggressive) nor do they treat the shared autonomous vehicle case with large passenger capacity. The RL SAV dispatcher of this paper solves the problem of increasing the number of passengers served while minimizing wait times for shared rides with passengers picked up from nearby locations while the taxi dispatcher in the abovementioned references does not use ride sharing.

Author Contributions: Conceptualization, K.M.-C., B.A.G. and L.G.; methodology, K.M.-C.; software, K.M.-C.; validation, K.M.-C., B.A.G. and L.G.; formal analysis, K.M.-C.; investigation, K.M.-C.; resources, B.A.G. and L.G.; data curation, K.M.-C.; writing—original draft preparation, K.M.-C.; writing—review and editing, K.M.-C., B.A.G. and L.G.; visualization, K.M.-C.; supervision, B.A.G. and L.G.; project administration, B.A.G. and L.G.; funding acquisition, B.A.G. and L.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors thank and acknowledge the support of the Automated Driving Lab at the Ohio State University.

Conflicts of Interest: The authors declare no conflict of interest.

References
1. Horváth, M.T.; Lu, Q.; Tettamanti, T.; Török, Á.; Szalay, Z. Vehicle-in-the-Loop (VIL) and Scenario-in-the-Loop (SCIL) Automotive Simulation Concepts from the Perspectives of Traffic Simulation and Traffic Control. Transp. Telecommun. 2019, 20, 153–161. [CrossRef]
2. Liu, M.; Wu, J.; Zhu, C.; Hu, K. A Study on Public Adoption of Robo-Taxis in China. J. Adv. Transp. 2020, 2020, 8877499. [CrossRef]
3. Meurer, J.; Pakusch, C.; Stevens, G.; Randall, D.; Wulf, V. A Wizard of Oz Study on Passengers’ Experiences of a Robo-Taxi Service in Real-Life Settings. In Proceedings of the 2020 ACM Designing Interactive Systems Conference, Eindhoven, The Netherlands, 6–10 July 2020; pp. 1365–1377.
4. Ambadipudi, A.; Heineke, K.; Kampshoff, P.; Shao, E. Gauging the Disruptive Power of Robo-Taxis in Autonomous Driving. In Automotive & Assembly; McKinsey and Company: Atlanta, GA, USA, 2017.
5. Jerath, K.; Brennan, S.N. Analytical Prediction of Self-Organized Traffic Jams as a Function of Increasing ACC Penetration. IEEE Trans. Intell. Transp. Syst. 2012, 13, 1782–1791. [CrossRef]
6. Wu, J.; Huang, L.; Cao, J.; Li, M.; Wang, X. Research on grid-Based Traffic Simulation Platform. In Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC’06), Guangzhou, China, 12–15 December 2006; pp. 411–420.
7. Jordan, W.C. Transforming Personal Mobility; WorldPress: San Francisco, CA, USA, 2012.
8. Alessandrini, A.; Cattivera, A.; Holguín, C.; Stam, D. CityMobil2: Challenges and Opportunities of Fully Automated Mobility. In Road Vehicle Automation; Springer: Berlin/Heidelberg, Germany, 2014; pp. 169–184.
9. Moorthy, A.; De Kleine, R.; Keoleian, G.; Good, J.; Lewis, G. Shared Autonomous Vehicles as a Sustainable Solution to the Last Mile Problem: A Case Study of Ann Arbor–Detroit Area. SAE Int. J. Passeng. Cars-Electron. Electr. Syst. 2017, 10, 328–336. [CrossRef]
10. City of Arlington. Milo Pilot Program Closeout Report. 2017. Available online: https://www.urbanismnext.org/resources/milo-pilot-program-closeout-report (accessed on 1 October 2022).

11. abc 7 News. California’s First Driverless Bus Hits the Road in San Ramon. 2018. Available online: https://www.bishopranch.com/californias-first-driverless-bus-hits-the-road-in-san-ramon/ (accessed on 1 October 2022).

12. abc News Arizona. Waymo Expanding Driverless Vehicle Rides to Downtown Phoenix. Available online: https://abc15.com/news/region-phoenix-metro-central-phoenix/waymo-expanding-driverless-vehicle-rides-to-downtown-phoenix (accessed on 1 October 2022).

13. SMART Columbus 2022. Available online: https://smart.columbus.gov/ (accessed on 1 October 2022).

14. Guvenc, L.; Guvenc, B.A.; Li, X.; Doss, A.C.A.; Meneses-Cime, K.; Gelbal, S.Y. Simulation Environment for Safety Assessment of CEAV Deployment in Linden. arXiv 2020, arXiv:2012.10498.

15. Community of Caring. Self-Driving Shuttle, Linden LEAP. 2020. Available online: https://callingallconnectors.org/announcements/self-driving-shuttle-linden-heap-coming-soon/ (accessed on 1 October 2022).

16. Smith, B.W. Human Error as a Cause of Vehicle Crashes; Centre for Internet and Society: Bangalore, India, 2013.

17. NSTC, USDOT. Ensuring American Leadership in Automated Vehicle Technologies: Automated Vehicles 4.0; NSTC, USDOT: Washington, DC, USA, 2020.

18. US DOT. Preparing for the Future of Transportation: Automated Vehicles 3.0. 2018. Available online: https://www.transportation.gov/av/3 (accessed on 1 October 2022).

19. NCLS. Autonomous Vehicles | Self-Driving Vehicles Enacted Legislation. In Proceedings of the National Conference of State Legislatures, Indianapolis, IN, USA, 10–13 August 2020.

20. Narayan, S.; Chaniotakis, E.; Antoniou, C. Shared Autonomous Vehicle Services: A Comprehensive Review. Transp. Res. Part C Emerg. Technol. 2020, 111, 255–293. [CrossRef]

21. Gurumurthy, K.M.; Kockelman, K. Modeling Americans’ Autonomous Vehicle Preferences: A Focus on Dynamic Ride-Sharing, Privacy & Long-Distance Mode Choices. Technol. Forecast. Soc. Chang. 2020, 150, 119792. [CrossRef]

22. Webb, J.; Wilson, C.; Kularatne, T. Will People Accept Shared Autonomous Electric Vehicles? A Survey before and after Receipt of the Costs and Benefits. Econ. Anal. Policy 2019, 61, 118–135. [CrossRef]

23. Stoiber, T.; Schubert, I.; Hoerler, R.; Burger, P. Will Consumers Prefer Shared and Pooled-Use Autonomous Vehicles? A Stated Choice Experiment with Swiss Households. Transp. Res. Part D Transp. Environ. 2019, 71, 265–282. [CrossRef]

24. Philipsepin, R.; Brell, T.; Zielle, M. Carriage Without a Driver—User Requirements for Intelligent Autonomous Mobility Services. In Procedings of the Advances in Human Aspects of Transportation; Stanton, N., Ed.; Springer International Publishing: Cham, Switzerland, 2019; pp. 339–350.

25. Fagnant, D.J.; Kockelman, K.M. The Travel and Environmental Implications of Shared Autonomous Vehicles, Using Agent-Based Model Scenarios. Transp. Res. Part C Emerg. Technol. 2014, 40, 1–13. [CrossRef]

26. Friedrich, B. Verkehrliche Wirkung autonomer Fahrzeuge. In Autonomes Fahren: Technische, Rechtliche und Gesellschaftliche Aspekte; Maurer, M., Gerdes, J.C., Lenz, B., Winner, H., Eds.; Springer: Berlin /Heidelberg, Germany, 2015; pp. 331–350. [CrossRef]

27. International Transport Forum. Urban Mobility System Upgrade. Int. Transp. Forum Policy Pap. 2015, 6, 1–36. [CrossRef]

28. Bischoff, J.; Maciejewski, M. Autonomous Taxicabs in Berlin—A Spatiotemporal Analysis of Service Performance. Transp. Res. Procedia 2016, 19, 176–186. [CrossRef]

29. Bischoff, J.; Maciejewski, M.; Nagel, K. City-Wide Shared Taxis: A Simulation Study in Berlin. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 275–280. [CrossRef]

30. Martinez, L.M.; Viegas, J.M. Assessing the Impacts of Deploying a Shared Self-Driving Urban Mobility System: An Agent-Based Model Applied to the City of Lisbon, Portugal. Int. J. Transp. Sci. Technol. 2017, 6, 13–27. [CrossRef]

31. Vosooghi, R.; Puchinger, J.; Jankovic, M.; Vouillon, A. Shared Autonomous Vehicle Simulation and Service Design. Transp. Res. Part C Emerg. Technol. 2019, 107, 15–33. [CrossRef]

32. Gurumurthy, K.M.; Kockelman, K.M. Analyzing the Dynamic Ride-Sharing Potential for Shared Autonomous Vehicle Fleets Using Cellphone Data from Orlando, Florida. Comput. Environ. Urban Syst. 2018, 71, 177–185. [CrossRef]

33. Masoud, N.; Jayakrishnan, R. Autonomous or Driver-Less Vehicles: Implementation Strategies and Operational Concerns. Transp. Res. Part E Logist. Transp. Rev. 2017, 108, 179–194. [CrossRef]

34. Alazzawi, S.; Hummel, M.; Kordt, P.; Sickenberger, T.; Wieseotte, C.; Wohak, O. Simulating the Impact of Shared, Autonomous Vehicles on Urban Mobility—A Case Study of Milan. In Proceedings of the SUMO 2018—Simulating Autonomous and Intermodal Transport Systems, Berlin-Adlershof, Germany, 14–16 May 2018; Wießner, E., Lücken, H., Hilbrich, R., Flötteröd, Y.P., Erdmann, J., Bieker-Walz, L., Behrisch, M., Eds.; EPiC Series in Engineering. EasyChair: Stockport, UK, 2018; Volume 2, pp. 94–110. [CrossRef]

35. Cime, K.M.; Cantas, M.R.; Fernandez, P.; Guvenc, B.A.; Guvenc, L.; Joshi, A.; Fishelson, J.; Mittal, A. Assessing the Access to Jobs by Shared Autonomous Vehicles in Marysville, Ohio: Modeling, Simulating and Validating. SAE Int. J. Adv. Curr. Pract. Mobil. 2021, 3, 2509–2515.

36. MORPC. Mid-Ohio Open Data. Available online: https://www.morpc.org/tool-resource/mid-ohio-open-data/ (accessed on 1 October 2022).
37. ODOT. Ohio Traffic Forecasting Manual (Module 2): Traffic Forecasting Methodologies. Available online: https://transportation.ohio.gov/static/Programs/StatewidePlanning/Modeling-Forecasting/Vol-2-ForecastingMethodologies.pdf (accessed on 1 October 2022).

38. Olstam, J.J.; Tapani, A. Comparison of Car-Following Models; Swedish National Road and Transport Research Institute: Linköping, Sweden, 2004; Volume 960.

39. Sukennik, P.; Group, P. Micro-Simulation Guide for Automated Vehicles. COEXIST 2018, 3, 1–33.

40. Mehmet, S. Smart Columbus Launches Three Mobility Technology Pilots; Intelligent Transport: Kent, United Kingdom, 2020.

41. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai Gym. arXiv 2016, arXiv:1606.01540.

42. Himite, B.; Patmalnieks, V. Traffic Simulator. Available online: https://traffic-simulation.de/ (accessed on 1 October 2022).

43. Treiber, M.; Hennecke, A.; Helbing, D. Derivation, Properties, and Simulation of a Gas-Kinetic-Based, Nonlocal Traffic Model. Phys. Rev. E 1999, 59, 239. [CrossRef]

44. Huang, C.Y.R.; Lai, C.Y.; Cheng, K.T.T. Fundamentals of Algorithms. In Electronic Design Automation; Elsevier: Amsterdam, The Netherlands, 2009; pp. 173–234.

45. Watkins, C.J.; Dayan, P. Q-learning. Mach. Learn. 1992, 8, 279–292. [CrossRef]

46. Guvenc, L.; Aksun-Guvenc, B.; Zhu, S.; Gelbal, S.Y. Autonomous Road Vehicle Path Planning and Tracking Control, 1st ed.; Wiley/IEEE Press: New York, NY, USA, 2021; ISBN 978-1-19-74974-9.

47. Wang, H.; Tota, A.; Aksun-Guvenc, B.; Guvenc, L. Real Time Implementation of Socially Acceptable Collision Avoidance of a Low Speed Autonomous Shuttle Using the Elastic Band Method. IFAC Mechatron. J. 2018, 50, 341–355. [CrossRef]

48. Ma, F.; Yang, Y.; Wang, J.; Li, X.; Wu, G.; Zhao, Y.; Wu, L.; Aksun-Guvenc, B.; Güvenç, L. Eco-Driving-Based Cooperative Adaptive Cruise Control of Connected Vehicles Platoon at Signalized Intersections. Transp. Res. Part D Transp. Environ. 2021, 92, 102746. [CrossRef]

49. Emirler, M.T.; Wang, H.; Aksun Guvenc, B.; Guvenc, L. Automated Robust Path Following Control based on Calculation of Lateral Deviation and Yaw Angle Error. In Proceedings of the ASME Dynamic Systems and Control Conference, DSC 2015, Columbus, OH, USA, 28–30 October 2015.

50. Aksun Guvenc, B.; Guvenc, L. Robust Steer-by-wire Control based on the Model Regulator. In Proceedings of the Joint IEEE Conference on Control Applications and IEEE Conference on Computer Aided Control Systems Design, Glasgow, UK, 18–20 September 2002; pp. 435–440.

51. Aksun Guvenc, B.; Guvenc, L. The Limited Integrator Model Regulator and its Use in Vehicle Steering Control. Turk. J. Eng. Environ. Sci. 2002, 26, 473–482.

52. Aksun Guvenc, B.; Guvenc, L.; Ozturk, E.S.; Yigit, T. Model Regulator Based Individual Wheel Braking Control. In Proceedings of the IEEE Conference on Control Applications, Istanbul, Turkey, 23–25 June 2003.

53. Demirel, B.; Guvenc, L. Parameter Space Design of Repetitive Controllers for Satisfying a Mixed Sensitivity Performance Requirement. IEEE Trans. Autom. Control. 2010, 55, 1893–1899. [CrossRef]

54. Guvenc, L.; Srinivasan, K. Force Controller Design and Evaluation for Robot Assisted Die and Mold Polishing. J. Mech. Syst. Signal Process. 1995, 9, 31–49. [CrossRef]

55. Orun, B.; Necipoglu, S.; Basdogan, C.; Guvenc, L. State Feedback Control for Adjusting the Dynamic Behavior of a Piezo-actuated Bimorph AFM Probe. Rev. Sci. Instrum. 2009, 80, 063701. [CrossRef]

56. Guvenç, L.; Srinivasan, K. Friction Compensation and Evaluation for a Force Control Application. J. Mech. Syst. Signal Process. 1994, 8, 623–638. [CrossRef]

57. Cebi, A.; Guvenc, L.; Demirci, M.; Kaplan Karadeniz, C.; Kanar, K.; Guraslan, E. A Low Cost, Portable Engine ECU Hardware-In-The-Loop Test System, Mini Track of Automotive Control. In Proceedings of the IEEE International Symposium on Industrial Electronics Conference, Dubrovnik, Croatia, 20–23 June 2005.

58. Liu, Z.; Li, J.; Wu, K. Context-Aware Taxi Dispatching at City-Scale Using Deep Reinforcement Learning. IEEE Trans. Intell. Transp. Syst. 2022, 23, 1996–2009. [CrossRef]

59. Haliem, M.; Mani, G.; Aggarwal, V.; Bhargava, B. A Distributed Model-Free Ride-Sharing Approach for Joint Matching, Pricing, and Dispatching Using Deep Reinforcement Learning. IEEE Trans. Intell. Transp. Syst. 2021, 22, 7931–7942. [CrossRef]