Learning-Based Routing for Autonomous Shuttles Under Stochastic Demand in Urban Mobility Systems Using Generative Adversarial Imitation Learning and Reinforcement Learning

Hyun Kim * and Branislav Dimitrijevic

Posted Date: 11 February 2026

doi: 10.20944/preprints202602.0835.v1

Keywords: autonomous mobility-on-demand (AMoD); autonomous shuttles; Dial-a-Ride Problem (DARP); intelligent transportation systems (ITS); reinforcement learning (RL); stochastic and dynamic vehicle routing problem (SDVRP)

 doi:10.20944/preprints202602.0835.v1

Article

Learning-Based Routing for Autonomous Shuttles Under Stochastic Demand in Urban Mobility Systems Using Generative Adversarial Imitation Learning and Reinforcement Learning

Hyun Kim * and Branislav Dimitrijevic

Department of Civil and Environmental Engineering, New Jersey Institute of Technology, Newark, New Jersey, USA
* Correspondence: hk385@njit.edu

Abstract

Extensive research has been conducted to develop technologies that enable paratransit systems to operate autonomously, including advanced sensing technologies and associated software. However, there remains a significant gap in research addressing the development of adaptive operational algorithms for such systems in urban environments. Autonomous Shuttles (AS) represent an emerging technology that has gained attention from industry, government, and academia as a novel public transit solution. AS hold the potential to enable Ride-shared Autonomous Mobility on Demand (RAMoD), which can improve accessibility and service equity in transportation-disadvantaged populations across urban and surrounding regions. To address this gap, this study applies an imitation-learning-assisted Deep Reinforcement Learning (DRL) approach to develop a routing method for AS under stochastic and dynamic passenger demand conditions. The proposed framework integrates Generative Adversarial Imitation Learning with Proximal Policy Optimization to enable real-time pickup and drop-off decision-making without centralized re-optimization. The DRL agent was trained over approximately 1.5 million training steps and evaluated across twenty episodes with stochastic passenger generation. Its performance was benchmarked against a deterministic Dial-a-Ride Problem (DARP) solver implemented using Google's OR-Tools, which employs a Cheapest Insertion heuristic with Local Search refinement. Comparative analysis showed median percentage differences of 37%, –6%, 20%, and 44% in passenger wait time, in-vehicle time, total service time, and episode completion time relative to the DARP baseline. The OR-Tools implementation was selected as a benchmark due to the lack of established step-wise evaluation methods for dynamic routing optimization in simulation environments. These findings demonstrate the potential of learning-based routing policies to support scalable, demand-responsive autonomous mobility services and future smart urban transportation systems.

Keywords: autonomous mobility-on-demand (AMoD); autonomous shuttles; Dial-a-Ride Problem (DARP); intelligent transportation systems (ITS); reinforcement learning (RL); stochastic and dynamic vehicle routing problem (SDVRP)

1. Introduction

Autonomous vehicles (AVs), and in particular shared autonomous shuttles, have received increasing attention from industry, government agencies, and academia as a potential next-generation public transportation solution. Shared autonomous mobility systems are expected to improve operational efficiency, expand service coverage, and enhance accessibility for transportation-disadvantaged populations by enabling flexible, demand-responsive transit operations [1–3]. As a result, autonomous shuttles have been actively piloted in campus environments, first–last mile services, and paratransit applications, including recent deployments in New Jersey and other regions worldwide [4].

 doi:10.20944/preprints202602.0835.v1

From an urban systems perspective, autonomous shuttles represent a form of demand-responsive public transportation that must continuously adapt to uncertain passenger arrivals, heterogeneous trip patterns, and real-time network conditions. Unlike conventional fixed-route transit, such services operate as sequential decision-making systems embedded within complex urban environments, where routing policies directly influence service reliability, passenger equity, and overall network efficiency. Recent studies have demonstrated the applicability of reinforcement learning for urban mobility control problems such as transit operations and shared mobility coordination, highlighting the potential of learning-based methods to support adaptive transportation systems [5,6]. These control-oriented applications can be viewed as special cases of sequential decision-making under uncertainty, with routing emerging as a higher-dimensional extension that requires jointly optimizing pickup, drop-off, and movement decisions over time. However, most existing approaches focus on isolated operational tasks or simplified service settings, leaving a gap in end-to-end routing frameworks that explicitly model both pickup and drop-off decisions under stochastic urban demand.

Despite this growing interest, much of the existing research has focused on vehicle technology, safety, and system-level impacts, while comparatively less attention has been devoted to the development of *operational routing and scheduling algorithms* suitable for shared autonomous shuttles operating under *stochastic and dynamically evolving passenger demand*. Most demand-responsive transit and paratransit systems rely on variants of the Dial-a-Ride Problem (DARP), which typically assume deterministic or fully known demand and often require centralized re-optimization as new requests arrive [7,8]. These assumptions limit their applicability to real-time autonomous shuttle operations, where passenger requests occur continuously and routing decisions must be updated online.

Recent advances in machine learning, particularly deep reinforcement learning (DRL), offer a promising alternative for addressing dynamic routing problems. By learning policies through interaction with the environment, DRL agents can adapt routing decisions in real time without repeatedly solving large-scale combinatorial optimization problems [9,10]. However, training stable and effective DRL policies for routing under stochastic demand remains challenging due to sparse rewards, high-dimensional state spaces, and the need to coordinate pickup and drop-off decisions across time.

To address these challenges, this study proposes a learning-based routing framework for autonomous shuttles operating under stochastic passenger demand. The proposed approach integrates *Generative Adversarial Imitation Learning (GAIL)* [11] with *Proximal Policy Optimization (PPO)* [10] to accelerate policy convergence and improve training stability. The learned routing policy is evaluated in a dynamic simulation environment and benchmarked against a deterministic DARP solution implemented using Google's OR-Tools. Performance is assessed using passenger waiting time, in-vehicle time, service completion time, and overall service efficiency. This work contributes to smart city transportation research by demonstrating how learning-based routing policies can support demand-responsive autonomous mobility systems under stochastic urban travel demand. The proposed framework enables real-time decision-making without centralized re-optimization, offering a scalable pathway toward intelligent, sustainable, and inclusive urban mobility services. Such approaches are particularly relevant for future electric and shared autonomous shuttle deployments in transportation-disadvantaged communities.

The main contributions of this paper are threefold: (1) formulation of a stochastic autonomous shuttle routing problem within a reinforcement learning framework; (2) development of an imitation-learning-assisted DRL training pipeline for dynamic pickup and drop-off routing; and (3) systematic benchmarking of the learned policy against a conventional DARP solver under controlled stochastic demand scenarios.

2. Literature Review

The routing and scheduling of shared autonomous shuttles is closely related to the classical Dial-a-Ride Problem (DARP), which seeks to determine optimal pickup and drop-off sequences subject to time

 doi:10.20944/preprints202602.0835.v1

windows, vehicle capacity constraints, and passenger service quality objectives. Early formulations of DARP focused on deterministic settings with fully known demand, using exact methods such as dynamic programming and branch-and-bound, as well as heuristic approaches including insertion heuristics, local search, and interchange methods [12–14]. These methods have been successfully applied to applications such as paratransit services, school bus routing, and company fleet operations.

Subsequent research extended DARP formulations to incorporate stochastic elements such as travel time variability, request cancellations, and vehicle disruptions [15]. Although these extensions improved robustness, most approaches continue to rely on centralized optimization and assume that passenger requests are known in advance or arrive at discrete re-optimization intervals. As a result, they remain computationally expensive and poorly suited for continuous, real-time routing decisions required by autonomous shuttle systems operating in highly dynamic environments.

In parallel, studies on shared autonomous vehicles (SAVs) and dynamic ride-sharing have explored simulation-based and optimization-based frameworks to assess system-level impacts and operational feasibility. Many of these studies impose simplifying assumptions, such as restricting ride-sharing to passengers with closely aligned spatiotemporal characteristics or aggregating pickup and drop-off locations into zones [16,17]. While these assumptions reduce computational complexity, they may compromise service equity and limit applicability to paratransit populations requiring door-to-door service [18].

More recently, reinforcement learning and Markov decision process (MDP) formulations have been investigated for stochastic vehicle routing problems. Learning-based approaches offer the advantage of producing real-time decisions without repeated global optimization [1,19]. However, many existing applications focus primarily on pickup decisions, neglect explicit modeling of passenger destinations, or are limited to single-vehicle or simplified service scenarios. Moreover, training instability and sparse reward signals remain significant barriers to practical deployment.

Despite growing interest in learning-based routing, a critical gap remains in the development of reinforcement learning frameworks that explicitly model *both pickup and drop-off decisions*, operate under *stochastic passenger arrivals*, and are evaluated against established optimization-based benchmarks. This study addresses this gap by integrating imitation learning with deep reinforcement learning to train an autonomous shuttle routing policy and by systematically comparing its performance to a deterministic DARP solver under controlled experimental conditions.

3. Research Motivation

Routing and scheduling shared autonomous shuttles under stochastic and dynamically evolving passenger demand represents a fundamental challenge for future demand-responsive transit systems. In its most general form, this challenge can be characterized as a MSDVRP (*Multi-agent Stochastic and Dynamic Vehicle Routing Problem*), in which multiple autonomous shuttles must make coordinated, real-time routing and pickup–drop-off decisions as new ride requests continuously arrive. Unlike conventional dial-a-ride formulations that assume deterministic or pre-known demand, such systems require online decision-making under uncertainty and evolving system states.

Despite recent advances in optimization and learning-based methods, developing stable and scalable solutions for this class of problems remains difficult. The combination of high-dimensional state spaces, delayed and sparse rewards, and non-stationarity arising from interactions among multiple vehicles poses significant challenges for DRL. Moreover, limited prior research exists on learning-based routing frameworks that explicitly address these issues in autonomous shuttle settings.

Motivated by these challenges, this study adopts a progressive research strategy in which the core decision-making problem is first examined in a simplified setting. By focusing on a relaxed formulation of the autonomous shuttle routing problem, this work seeks to establish a robust methodological foundation that can be extended to multi-agent and large-scale scenarios in future research.

 doi:10.20944/preprints202602.0835.v1

4. Scope and Problem Definition

This study addresses a relaxed formulation of the Autonomous Shuttle Problem (ASP), which concerns the dynamic routing of autonomous shuttles operating under stochastic passenger demand. While the broader ASP aligns with a multi-agent stochastic and dynamic vehicle routing problem, the present work focuses on a deliberately constrained setting to enable systematic analysis and stable policy learning.

Specifically, the scope of this study is limited to a *single autonomous shuttle agent* operating within a 10×10 grid-based network under stochastic passenger arrivals. This abstraction represents a simplified street network and allows for reproducible experimentation and step-wise evaluation of routing decisions in a dynamic environment. By excluding multi-agent coordination, the formulation avoids non-stationarity introduced by agent interactions and isolates the fundamental challenges associated with dynamic pickup and drop-off routing.

The objective of the agent is to learn adaptive routing policies that minimize passenger waiting time, in-vehicle time, and overall service completion time. Policy performance is evaluated using operational metrics relevant to paratransit and shared autonomous mobility systems and is benchmarked against a conventional Dial-a-Ride Problem (DARP) formulation implemented using Google's OR-Tools.

The scope of this study focuses on a simplified formulation of the Autonomous Shuttle Problem to enable systematic analysis and stable policy learning under stochastic passenger demand. Specifically, the single-agent, grid-based environment serves as an abstract representation of urban street topology, allowing controlled evaluation of dynamic pickup and drop-off routing decisions without the confounding effects of large-scale network complexity or multi-vehicle interactions. While cooperative or competitive multi-shuttle coordination and full city-scale deployment considerations are not explicitly modeled, the results presented here demonstrate the feasibility of learning-based routing policies for demand-responsive autonomous mobility. The proposed framework is designed with extensibility in mind, and the single-agent formulation serves as a foundational step toward scalable, multi-agent autonomous shuttle routing systems applicable to smart urban transportation contexts.

5. Materials and Methods

The existing DARP heuristics and optimization algorithms provide the optimal route and schedule (sequence) to pick up passengers, considering their desired times of pick up and arrival at destinations. This is achieved by optimizing the DARP objective function with constraints and all variables known in advance, such as the number of passengers, the time windows of their desired pick up and drop off times, and their pick up and drop off locations. To exemplify the limitation of the traditional DARP algorithm, consider a simple 10 by 10 grid representing a street network, with known passengers' origin and destination information, as shown in Figure 1. In Figure 1, the small circles represent origins of the passengers requesting rides, and crosses represent their destination. The matching origin and destination (i.e., for the same passenger trip) are shown in the same color. For example, P1 represents the origin of passenger 1, located at the intersection H3 and represented by red circle; the destination of passenger 1 is at intersection C7, represented by a cross shown in red color. The origin and destination pairs for different passengers are shown in different colors. As mentioned previously, depending on the time window of passengers' desired pickup and arrival time, traditional DARP methods can be used to determine the optimal spatiotemporal departure point and the sequence of pickups and drop offs of these passengers such as the Figure 1 below.

 doi:10.20944/preprints202602.0835.v1

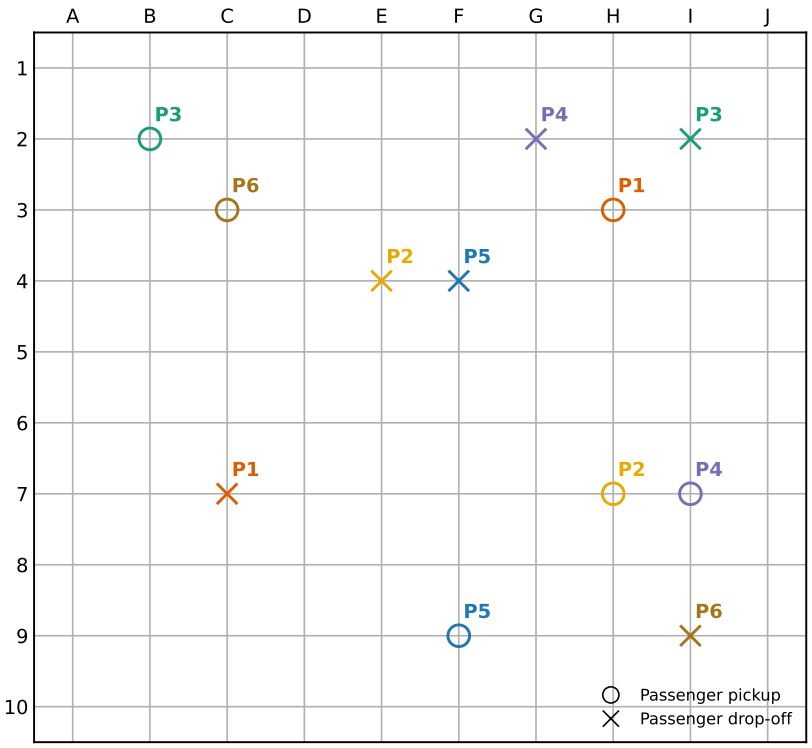

Figure 1. An illustration of a hypothetical street network with passenger pick up and drop off locations.

Now let us consider a scenario in which the passenger trip requests and trip information is not available in advance. The three grid plots in Figure 2 represent the locations of passenger trip requests and the location of an autonomous shuttle (AS1) in the street network at three different times: the left plot is at time zero T = 0, the middle plot is at time one T = 1, and the right plot is at time T = 2. At T = 0, only one passenger, P1, requested a service and the shuttle AS1 is just 2 vertices away. At time T = 1, AS1 is moving towards P1. However, at T = 1, another passenger, P2, requests service and is four vertices away from the origin of P1. Then, at T = 2, another passenger, P3, requests service. At T=2, AS1 has to decide between different options for servicing the passengers. For example, one option (let us call it Option 1) would be to pickup P2, then drop off P1, then drop off P2, then pickup P3, and then drop off P3. Another option (say, Option 2) would be to pick up P2, then drop off P2 on the way to P3, pick up P3, then drop off P1, and lastly drop off P3. As a combinatorial problem, the passenger service plan has multiple solutions, but the traditional route planning optimization methods would not be able to dynamically add and subtract passengers while the shuttle is on the move and for multiple shuttles at once.

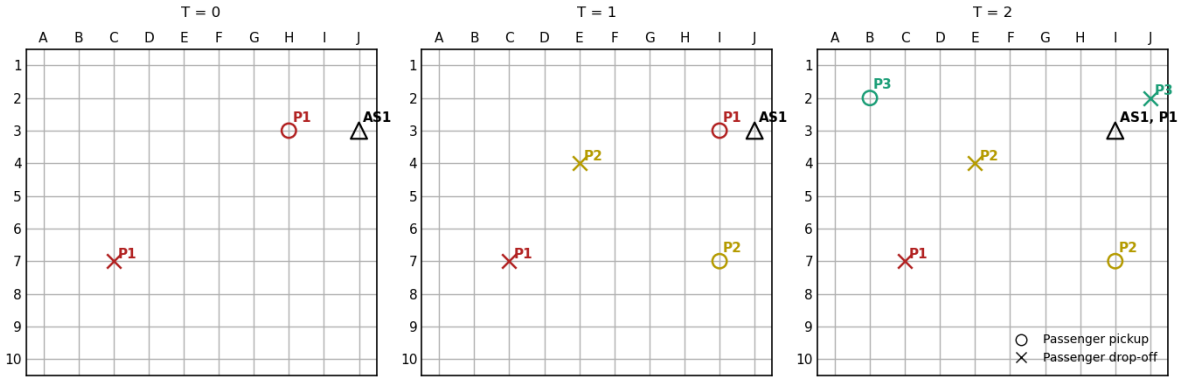

Figure 2. An illustration of a hypothetical grid street network with passenger pick up and drop off locations, and the location of an AS

 doi:10.20944/preprints202602.0835.v1

5.1. ASP Problem Formulation as a Markov Decision Process (MDP)

Though not specific to optimizing routing algorithms, Nijs et al's [20] research on Constrained Multiagent Markov Decision Process (CMMDP) seems invaluable in designing and conceptualizing the ASP using MDP. CMMDP is to aid decisions for multi-agents sharing resources to make decisions with uncertainties in the domain. They determine which variations of CMMDP will serve the best depending on the objective, type of constraint, and observability of the domain. MDP can be formulated as tuples that include state, action, transition function, reward, and discount factor,as shown in Equation 1,

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle \tag{1}$$

$$\mathcal{S} : \text{state space}, \tag{2}$$
$$\mathcal{A} : \text{action space}, \tag{3}$$
$$P(s' \mid s, a) : \text{state transition probability}, \tag{4}$$
$$R(s, a, s') : \text{reward function}, \tag{5}$$
$$\gamma \in [0, 1) : \text{discount factor}. \tag{6}$$

In MDP, the agent must decide an appropriate action that will produce the highest reward depending on the current state. Also, depending on the weight of the discount factor, the agent can value immediate reward more than the future reward or vice versa.

In the context of ASP, the state represents the positions of all AS, whether the shuttles have passengers onboard, the locations of all passengers and their elapsed wait time, travel time, and their destinations, including both the passengers waiting for a pick up and those already onboard the shuttles. The action space is defined as movements of the shuttle that can maneuver the shuttle on a grid including moving forward or turning left and right.

In a conventional MDP, the transition probability is used to model the probability of advancing to the next state from the current state if there are multiple scenarios possible by taking a single action. The probabilistic outcome of taking an action introduces some randomness, which is appropriate to be used in modeling stochastic scenarios. However, in the context of this research problem, the transition probability is not necessary for use because picking up and dropping off passengers are deterministic and there is no other outcome to performing each action. If a shuttle picks up passengers, then they are picked up with no other possible outcomes to that action.

The reward function is set to guide the agent to reach the overall objective. A careful and thorough reward shaping process is necessary to prevent erratic behavior while promoting actions that align with objectives such as finding the shortest path and optimally routing to maximize efficiency of transporting passengers. A more comprehensive setting of the state information, the action space, and reward function for the ASP are covered in the subsequent chapters.

5.2. Proposed Method: Imitation Learning–Assisted Deep Reinforcement Learning

This study proposes a two-stage learning framework that integrates imitation learning and deep reinforcement learning (DRL) to train an autonomous shuttle agent for dynamic routing under stochastic passenger demand. The proposed approach combines the rapid policy initialization capabilities of imitation learning with the long-term optimality and adaptability of reinforcement learning, resulting in a more stable and efficient training process than using reinforcement learning alone.

5.2.1. Overview of the Learning Pipeline

Training begins with imitation learning using Generative Adversarial Imitation Learning (GAIL), where the agent learns an initial routing policy by imitating expert behavior. This stage provides the agent with a structured understanding of the task objective and reduces the exploration burden associated with sparse and delayed rewards. Once the policy converges under imitation learning,

 doi:10.20944/preprints202602.0835.v1

the learned network weights are transferred to a deep reinforcement learning agent. The agent then continues training using direct environmental feedback, allowing further policy refinement through exploration and exploitation. This two-stage pipeline leverages the strengths of both learning paradigms while mitigating their individual limitations.

5.2.2. Imitation Learning via Generative Adversarial Imitation Learning (GAIL)

GAIL formulates imitation learning as an adversarial optimization problem involving two components: a policy (agent) and a discriminator (adversary) [11]. The discriminator is trained to distinguish between state–action pairs generated by the agent and those obtained from expert demonstrations recorded by a human operator. Concurrently, the agent learns to generate trajectories that are indistinguishable from expert behavior.

Formally, GAIL solves the following minimax optimization problem:

$$\min_{\pi_\theta} \max_{D_\phi} \; \mathbb{E}_{(s,a)\sim\pi_E}\big[\log D_\phi(s,a)\big] + \mathbb{E}_{(s,a)\sim\pi_\theta}\big[\log(1 - D_\phi(s,a))\big], \tag{7}$$

where π_E denotes the expert policy, π_θ is the agent policy, and $D_\phi(s,a)$ is the discriminator output indicating whether a state–action pair is expert-generated.

During training, the discriminator provides a learned reward signal to the agent, defined as:

$$r^{\text{GAIL}}(s,a) = -\log\big(1 - D_\phi(s,a)\big), \tag{8}$$

which replaces the environment reward. By learning from this adversarial feedback, the agent rapidly acquires expert-like routing behavior without directly interacting with the environment. While this mechanism provides an effective initialization, GAIL training is inherently limited by the quality of expert demonstrations and may lead to premature convergence if used alone.

5.2.3. Policy Refinement via Proximal Policy Optimization (PPO)

To overcome the limitations of imitation learning and enable further performance improvements, the GAIL-trained policy is transferred to a deep reinforcement learning agent and refined using Proximal Policy Optimization (PPO) [10]. PPO is selected due to its robustness, sample efficiency, and strong empirical performance in stochastic and dynamic environments.

PPO constrains policy updates by optimizing a clipped surrogate objective, which limits the deviation between successive policy iterations. The probability ratio used to compare the updated policy with the previous policy is defined as:

$$r_t(\Theta) = \frac{\pi_\Theta(a_t \mid s_t)}{\pi_{\Theta_{\text{old}}}(a_t \mid s_t)}. \tag{9}$$

The PPO clipped surrogate objective is given by:

$$L^{\text{CLIP}}(\Theta) = \mathbb{E}_t\Big[\min\Big(r_t(\Theta)\,\hat{A}_t,\; \text{clip}\big(r_t(\Theta),\,1-\varepsilon,\,1+\varepsilon\big)\hat{A}_t\Big)\Big], \tag{10}$$

where $\hat{A}_t$ is the advantage function measuring the relative quality of an action compared to the expected value of the current state, and ε is a hyperparameter that controls the allowable magnitude of policy updates. By clipping the probability ratio, PPO prevents excessively large updates that may destabilize learning while maintaining sufficient exploration.

In addition to policy optimization, PPO jointly updates a value function to improve estimates of future returns and incorporates entropy regularization to prevent premature convergence to deterministic policies. Through continued interaction with the environment, the agent refines the imitation-initialized policy and learns to adapt to situations beyond the expert demonstrations. This combined imitation learning and reinforcement learning framework enables efficient convergence and robust policy learning for autonomous shuttle routing in dynamic environments.

 doi:10.20944/preprints202602.0835.v1

5.3. ASP Environment Set Up

Python packages including Gym and Stable Baselines3 were used to set up a custom environment suitable to simulate the ASP in a discrete simulation space using a computer with AMD Ryzen 9 5900HS, NVIDIA GeForce RTX 3080 Laptop GPU, and 32 GB of RAM. Gym, developed by OpenAI, is an open-source standardized toolkit for developing and comparing DRL algorithms in a customizable environment involving an agent. The custom ASP simulation environment was developed initially as a 10x10 grid street network, with an agent controlling a shuttle placed at the coordinate point (5,5) in the grid, facing north, with three randomly positioned passengers. Also, after every 10 steps (a step being the measure of time in the simulation), a passenger appears at a random location, until reaching a total of six passengers. Each simulation, equivalent in this experiment to an episode, terminates when either 500 steps are reached, or all six passengers have been transported (dropped off) to their destinations. The simulation environment has a predefined space orientation to prevent the shuttle from simply traversing in all directions without changing its heading, similar to how vehicles operate on streets. The orientation of the agent can be north, south, east, and west.

On the other hand, despite this structured environment design, early training experiments indicated that the agent struggled to learn coherent routing behaviors when exposed to the full state space of the grid network and passenger information. In particular, the agent frequently exhibited unstable behaviors such as circling, idling near passenger locations, or failing to follow shortest paths toward pickup and drop-off points, even after extensive training. This suggested that the agent was unable to reliably infer short-horizon navigational structure solely from sparse reward signals in a highly stochastic setting.

To address this challenge, a lightweight environment-level guidance mechanism based on the Breadth-First Search (BFS) algorithm was incorporated. BFS was used to compute shortest-path distances between the shuttle's current position and candidate passenger pickup or drop-off locations, allowing the environment to dynamically identify target locations that were most relevant at each decision step. Rather than directly prescribing actions, this mechanism reduced the effective state complexity by exposing only target-relative information and restricting feasible movements to those consistent with shortest-path traversal. This approach preserved the agent's autonomy in decision-making while providing sufficient structural guidance to enable stable and efficient policy learning.

5.4. DRL Observation Space

The observation space available to the agent includes: location, distance, and orientation towards the nearest passenger; location, distance, and orientation towards the nearest drop off location; current shuttle position and orientation. Though the observation space may seem overly simplified for the ASP, this configuration empirically yielded the best performance. Until the problem was relaxed with a custom guidance system and the simplification of the observation space, the agent was not able to learn or produce coherent actions, contrary to the theories and expectations. The ASP DRL environment was first configured similarly to how a DARP would be solved, providing as much information as possible, then waiting for a solution to be developed. In the initial stage of the model development, the observation space included all passengers' elapsed waiting time, elapsed in-vehicle time, upper limit of all passengers' waiting time and arrival time, and the number of passengers onboard the shuttle, along with the current observation space, which summed up to 60 dimensions in the observation space array. Upon incrementally reducing the number of observations, the current observation space showed signs of improvement over time and therefore was adopted.

5.5. DRL Action Space

To prevent the agent from traversing simply up and down in the grid environment, the orientation mechanism was implemented that introduced more realistic movement of shuttles. Therefore, the discrete action space for the agent can be formulated as Equation 11.

 doi:10.20944/preprints202602.0835.v1

$$\mathcal{A} = \begin{cases} 0, & \text{move forward,} \\ 1, & \text{turn left then move forward,} \\ 2, & \text{turn right then move forward.} \end{cases} \tag{11}$$

As described, the agent can go forward, or turn left then go forward, or turn right then go forward. "Going forward" was added to the turning movements because situations were observed in the trials where the agent was only turning and not producing coherent maneuver within the grid to pick up or drop off passengers. To design a more robust learning opportunity and an action space that is aligned with the reward function, making a turn without moving forward was not included as an action choice.

5.6. DRL Reward Function

The reward function helps the agent achieve the objective by providing guidance in the form of reward or punishment. The reward function must match the observation space and the objective of the simulation environment for effective learning. Therefore, in the ASP DRL environment, the reward function is engineered to minimize overall passenger waiting and traveling time by following (an optimal) passenger pickup and drop-off sequence, and traveling along the shortest path when en route to pick up or drop off passengers. There are several types of rewards that have been considered in the ASP DRL model: base rewards, vicinity rewards, inverse travel time rewards, and an early completion rewards. The base rewards for picking up and dropping off are set to be 0.2 and 0.15 respectively to encourage pickups more than drop-offs, aligned with the transit user cost theory that people value wait time, up to two to three times more than in-vehicle time [21]. In this initial research experiment the pickup base reward was only slightly higher than the drop-off reward, but these values can be adjusted after examining the performance of the trained agent. Vicinity rewards are calculated by setting up vicinities of 5 distance units around passenger pickup and drop-off locations, and giving incrementally greater reward as the agent gets closer to the locations. The vicinity reward calculation is shown below in Equation 12.

$$\text{vicinity reward} = \frac{\text{VICINITY\_REWARDS}}{\text{distance} + 1} \tag{12}$$

where:

$$\text{VICINITY\_REWARDS} = \begin{cases} \text{pickup} & = 0.5, \\ \text{drop-off} & = 0.5. \end{cases}$$

As shown in Equation 12, the value of is inverse to the distance between a pickup or a drop-off location, and the closer the agent gets to one of these locations, the more rewards it will receive. Since traveling along the shortest path is equally important for both pickups and drop-offs, their vicinity rewards are the same. In a transit system it is important to transport all passengers as quickly as possible. Thus, the agent is given rewards for completing pick and drop off of passengers quickly. This is accomplished with the inverse travel time reward, or speed bonus, which is calculated using the following Equation 13.

$$\text{Speed bonus} = \text{base reward} \times \left(2 - \frac{\text{elapsed time}}{\text{current time}}\right) \tag{13}$$

where:

$$\text{elapsed time} = \begin{cases} \text{elapsed wait time,} & \text{if status = waiting,} \\ \text{elapsed in vehicle time,} & \text{if status = onboard.} \end{cases}$$

The is calculated for each passenger and includes elapsed wait time or elapsed in-vehicle time depending on the status of passengers. If a passenger is in waiting status, then the elapsed time measures the time since the passenger demand was generated. On the other hand, if the status of a passenger is onboard (a shuttle vehicle), then the elapsed time measures their current in-vehicle travel time. For example, if a passenger was generated at step 0, picked up at step 7, and dropped off at step

 doi:10.20944/preprints202602.0835.v1

20, then their elapsed wait time would be 7 and elapsed in-vehicle time would be 13. Therefore, the agent will receive more rewards the faster it is able to pickup and drop off passengers by shortening passenger elapsed wait time and elapsed in-vehicle time. Also, at the completion of either a pickup or a drop-off, the will be added to the so the agent can receive both a base reward and inverse travel time reward for picking up or dropping of passengers. Lastly, since the objective of the agent is to transport all the passengers generated in an episode as early as possible, additional reward is given for ending the episode as quickly as possible. The early completion reward is calculated as follows:

$$\text{early completion bonus} = \text{remaining time} \times 0.01 \tag{14}$$

where:

$$\text{remaining time} = 500 - \text{episode end time}$$

An episode would terminate upon either reaching 500 steps or transporting all six passengers. The agent will not receive the early completion bonus if it was not able to transport all passengers in an episode. As shown from the equation, the early completion bonus is calculated by multiplying .01 to the remaining time and the remaining time is the difference of 500 and the episode end time. Therefore, the agent can receive more rewards by decreasing the episode end time thereby increasing the remaining time to receive high early completion bonus.

6. Results

Following the two-stage training procedure described above, 125 expert demonstration episodes were collected using the same simulation environment and configuration. The resulting policy learned through GAIL was subsequently transferred to a deep reinforcement learning agent and further refined using PPO. The hyperparameters used across both training stages are summarized in Table 1.

Table 1. Training configuration and hyperparameters across learning stages.

Training Sequence	Training Algorithm	# of Steps	Learning Rate	n Steps	Batch Size	Clip Range	Ent. Coef.
1	GAIL	200,000	1.0×10^{-4}	1024	4096	0.2	0.02
2	GAIL	200,000	1.0×10^{-4}	1024	4096	0.2	0.01
3	GAIL	400,000	1.0×10^{-4}	1024	4096	0.2	0.001
4	DRL	200,000	1.0×10^{-4}	1024	4096	0.2	0.02
5	DRL	200,000	1.0×10^{-4}	1024	4096	0.2	0.01
6	DRL	400,000	1.0×10^{-4}	1024	4096	0.2	0.001

The entropy coefficient was annealed to slowly discourage exploration over time, while encouraging the agent to exploit the policy it learned to refine the optimal routing policy based on the locations of passenger pickups and drop-offs. Different combinations of the hyperparameters were tested, such as the learning rate ranging from 10^{-3} to 10^{-5}, n steps from 64 to 1024, and batch_size from 256 to 8192. However, the hyperparameter values presented in Table 1 demonstrated the best performance. Given the stochastic environment, the hyperparameters were configured to ensure sufficiently long horizons (n steps and batch size) for the frequency of update and a robust data collection. Furthermore, a low learning rate and constrained update values were employed to promote effective generalization by the agent.

To evaluate the effectiveness of the trained policy and the environment-level guidance mechanism in the absence of established benchmarks for the Autonomous Shuttle Problem (ASP), a deterministic Dial-a-Ride Problem (DARP) formulation was implemented using Google's OR-Tools. In this benchmark, passenger demand was assumed to be fully known in advance, providing an upper-bound reference for routing performance. Because the proposed DRL-based approach optimizes routing decisions online without prior knowledge of future passenger requests, it is expected to yield inferior objective values relative to the deterministic DARP solution. Consequently, performance proximity to the DARP benchmark is interpreted as an indication of effective real-time decision-making.

 doi:10.20944/preprints202602.0835.v1

After training, key operational statistics—including passenger pickup and drop-off locations and times, waiting times, and in-vehicle travel times—were recorded at the end of each validation episode. The same passenger demand realizations were then applied to a DARP model implemented using the Google OR-Tools Python package, employing the `ortools.constraint_solver` and `pywrapcp.routing_enums_pb2` modules, to generate benchmark solutions for comparison.

In each episode, a total of six passengers were generated at random locations within the service network. Three passengers (P1–P3) were generated at the start of the episode, followed by the arrival of one additional passenger every 10 time steps until all six passengers had been introduced. To ensure consistency between the online DRL setting and the offline DARP benchmark, pickup time constraints were imposed in the DARP formulation such that passenger P4 could not be picked up before step 10, P5 before step 20, and P6 before step 30. No explicit constraints were placed on maximum waiting time or in-vehicle travel time.

The service network was modeled as a 10×10 grid, where movement between adjacent nodes incurred a unit cost of one time step. Travel costs and shortest paths were computed using the Manhattan distance metric, consistent with the grid-based environment used for DRL training.

For the DARP benchmark, OR-Tools was configured by specifying paired pickup and drop-off locations, pickup time constraints, vehicle capacity, and pickup–drop-off precedence constraints, with the objective of minimizing total travel cost across all passenger services. An initial routing solution was generated using the `PARALLEL_CHEAPEST_INSERTION` heuristic, followed by refinement using the `GUIDED_LOCAL_SEARCH` metaheuristic. These general vehicle routing problem (VRP) strategies iteratively improve solution quality through cost-based insertion and local neighborhood exploration [22].

Table 2 presents a comprehensive comparison between the DRL agent and the corresponding DARP solution obtained using Google's OR-Tools. The performance metrics used for comparison include average passenger waiting time, average in-vehicle time, the episode completion time, and total service time, defined as the sum of average waiting time and average in-vehicle time.

Table 2. Comparison of episode-level performance metrics between the DRL agent and the DARP solution using Google OR-Tools. W denotes average passenger waiting time, and IV denotes average in-vehicle time.

Episode	DRL				DARP			
	Avg. Wait Time	Avg. In-Vehicle Time	Episode End Time	Serviced Time (W+IV)	Avg. Wait Time	Avg. In-Vehicle Time	Episode End Time	Serviced Time (W+IV)
1	30	27	96	57	17	21	59	38
2	25	8	73	33	8	22	49	30
3	20	8	70	28	24	17	59	41
4	34	12	102	46	8	14	43	22
5	21	11	72	32	16	8	52	24
6	32	14	98	46	18	22	62	40
7	27	26	107	53	19	12	58	31
8	20	16	69	36	22	21	64	43
9	37	14	101	51	26	11	61	37
10	24	14	78	38	31	19	73	50
11	31	14	95	45	28	16	68	44
12	18	18	65	36	26	17	60	43
13	17	26	87	43	32	15	67	47
14	16	49	98	65	9	13	52	22
15	31	22	84	53	17	22	57	39
16	19	10	72	29	26	13	61	39
17	22	30	78	52	25	10	57	35
18	24	16	77	40	8	24	49	32
19	16	17	66	33	18	17	58	35
20	36	13	98	49	12	21	58	33

 doi:10.20944/preprints202602.0835.v1

Table 3 summarizes the percentage differences between the DRL agent and the DARP benchmark across twenty validation episodes. The percentage difference for each performance metric is computed by subtracting the corresponding DARP value from the DRL value and normalizing by the DARP value. In addition to service-related metrics, cumulative passenger trip distance and average passenger trip distance are reported to examine whether trip length characteristics are associated with variations in relative performance.

Because the DRL agent makes routing decisions online as passenger requests arrive, longer cumulative passenger trip distances may increase decision complexity and potentially affect passenger waiting time or in-vehicle travel time. To investigate this relationship, the percentage differences in performance metrics are examined as a function of aggregated passenger travel distance.

Figure 3 illustrates the relationship between the percentage difference in average passenger waiting time and aggregated passenger trip distance. The resulting coefficient of determination ($R^2 = 0.0708$) indicates a weak association between these variables. Figure 4 presents the corresponding relationship for average in-vehicle travel time, yielding an R^2 value of 0.1831, which likewise suggests a limited correlation. Finally, Figure 5 shows the relationship between the percentage difference in overall passenger service time and aggregated passenger travel distance, with an R^2 value of 0.012, indicating a negligible association.

Overall, these results suggest that variations in cumulative passenger trip distance do not strongly explain the observed performance differences between the DRL-based routing policy and the deterministic DARP benchmark under the evaluated scenarios.

Table 3. Episode-level percentage differences and passenger trip distance statistics.

Episode	% Difference of Wait Time	% Difference of In-Vehicle Time	% Difference of Service End Time	Cumulative Passenger Trip Distance	Avg. Passenger Travel Distance
1	76%	29%	63%	43	7
2	213%	-64%	49%	38	6
3	-17%	-53%	19%	35	6
4	325%	-14%	137%	39	7
5	31%	38%	38%	35	6
6	78%	-36%	58%	51	9
7	42%	117%	84%	42	7
8	-9%	-24%	8%	39	7
9	42%	27%	66%	40	7
10	-23%	-26%	7%	46	8
11	11%	-13%	40%	51	9
12	-31%	6%	8%	44	7
13	-47%	73%	30%	54	9
14	78%	277%	88%	54	9
15	82%	0%	47%	40	7
16	-27%	-23%	18%	39	7
17	-12%	200%	37%	42	7
18	200%	-33%	57%	27	5
19	-11%	0%	14%	35	6
20	200%	-38%	69%	45	8

 doi:10.20944/preprints202602.0835.v1

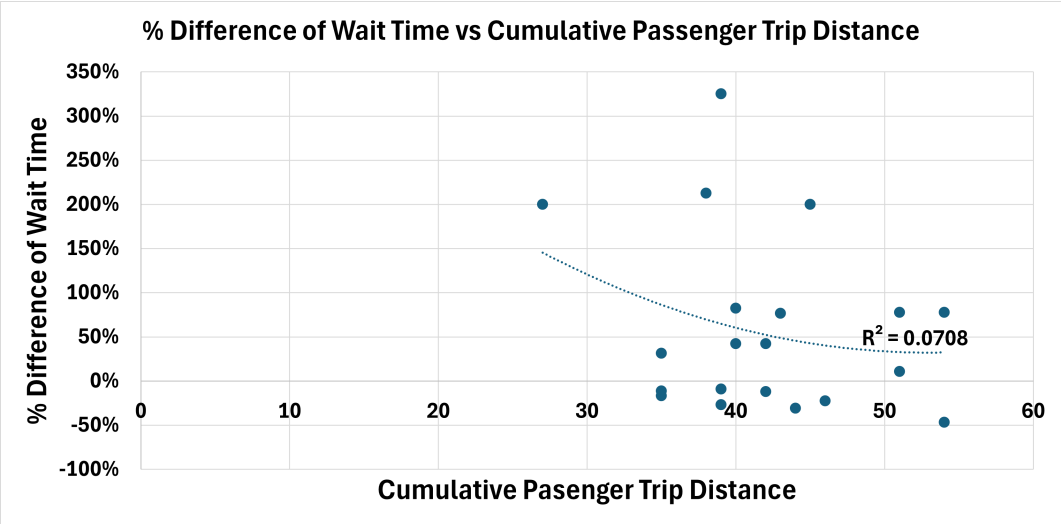

Figure 3. Difference of Wait Time and Aggregated Passenger Travel Distance.

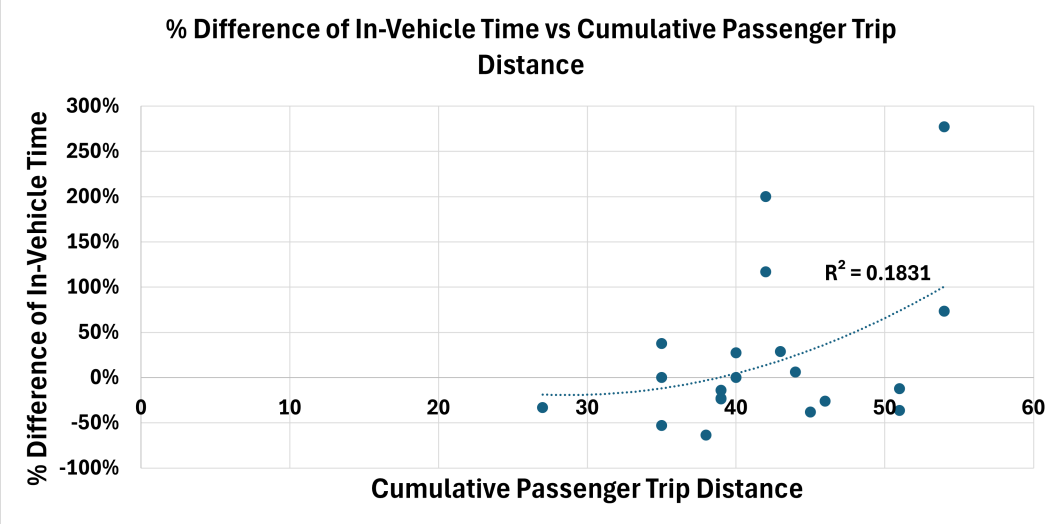

Figure 4. Difference of In-Vehicle Time and Aggregated Passenger Travel Distance.

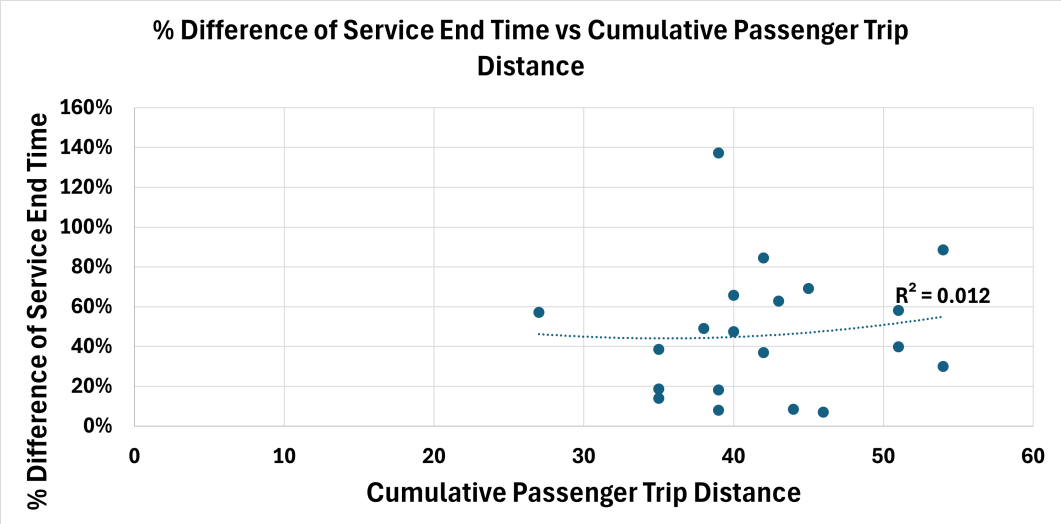

Figure 5. Difference of Service End Time and Aggregated Passenger Travel Distance.

 doi:10.20944/preprints202602.0835.v1

Table 4 reports the mean, median, minimum, and maximum values of key performance metrics across twenty validation episodes for both the DRL agent and the DARP benchmark. Table 5 presents the corresponding summary statistics of percentage differences between the two approaches, computed using DARP as the baseline.

Across episodes, substantial variability is observed in both methods, as reflected by the wide ranges between minimum and maximum values. While the maximum performance values are consistently lower for the DARP benchmark, indicating more stable outcomes, the DRL agent exhibits greater dispersion due to its online, stochastic decision-making process. On average, the DARP solution outperforms the DRL agent by 60% in passenger waiting time, 22% in in-vehicle time, 47% in episode end time, and 28% in overall serviced time.

Despite these aggregate trends, the results also reveal instances in which the DRL agent achieves lower passenger waiting times and in-vehicle travel times than the DARP benchmark. These cases highlight the potential of learning-based routing policies to perform competitively under certain demand realizations, even when compared against a deterministic solution with full prior knowledge of passenger requests.

Table 4. Comparison of average episode-level performance metrics for the DRL agent and the DARP benchmark across twenty validation episodes.

Statistic	DRL				DARP			
	Avg. Wait Time	Avg. In-Vehicle Time	Episode End Time	Serviced Time	Avg. Wait Time	Avg. In-Vehicle Time	Episode End Time	Serviced Time
Mean	25	18	84	43	20	17	58	36
Median	24	15	81	44	19	17	59	38
Min.	16	8	65	28	8	8	43	22
Max.	37	49	107	65	32	24	73	50

Table 5. Summary statistics of percentage differences between the DRL agent and the DARP benchmark across twenty episodes. Positive values indicate higher values for the DRL agent relative to DARP.

	% Diff. Wait Time	% Diff. In-Vehicle Time	% Diff. Episode End Time	% Diff. Serviced Time	Cumulative Trip Distance
Mean	60%	22%	47%	28%	42
Median	37%	-6%	44%	20%	41
Min.	-47%	-64%	7%	-32%	27
Max.	325%	277%	137%	195%	54

7. Discussion

In the initial experiments, several components of the proposed modeling framework were intentionally relaxed—including the number of shuttles—to enable stable policy learning and to validate the feasibility of the learning-based routing approach. While the presented results demonstrate that the proposed framework can learn coherent and adaptive routing behavior under stochastic passenger demand, the original objective of developing a scalable routing algorithm for multiple autonomous shuttles motivates several extensions to the current formulation.

First, the quality and scale of expert demonstrations used during the imitation learning stage can be substantially improved. In the present study, expert trajectories were generated by a human operator due to the lack of existing datasets capable of capturing optimal pickup and drop-off routing under stochastic demand. While this approach proved sufficient to initialize the learning process and yielded meaningful initial results, it introduces limitations related to both routing optimality and dataset size. To address these limitations, future work will focus on developing an automated expert data generation pipeline capable of producing optimal observation–action pairs based on passenger locations and request times. This can be achieved by integrating a discrete-event simulation

 doi:10.20944/preprints202602.0835.v1

framework, such as SimPy [23], with a dynamically updated DARP solver to generate step-by-step shuttle movements under evolving passenger demand.

Second, the training process can be enhanced through the incorporation of a self-play mechanism. Self-play has been shown to enable agents to discover strategies beyond those demonstrated by human experts in complex decision-making tasks, including AlphaGo, AlphaStar, AlphaZero, and OpenAI Five [24–27]. After acquiring baseline routing behavior through imitation learning, self-play can be employed to allow agents to compete against previous versions of themselves, encouraging the emergence of more efficient routing, pickup sequencing, and ridesharing strategies based on spatiotemporal information.

Third, the framework can be extended to a fully multi-agent setting by adopting a multi-agent reinforcement learning algorithm. Candidate approaches include QMIX (Value Decomposition Networks) [28] and Multi-Agent Proximal Policy Optimization (MAPPO) [29], both of which follow the centralized training and decentralized execution (CTDE) paradigm. Under CTDE, agents leverage shared global information during training while relying solely on local observations during execution. QMIX combines individual agent value functions through a mixing network to approximate a global action-value function, whereas MAPPO employs decentralized policies with a centralized value function. Both approaches are well suited to the multi-vehicle Autonomous Shuttle Problem, and future empirical evaluation will determine their relative effectiveness in this domain.

8. Conclusions

Beyond methodological contributions, this work highlights the role of learning-based routing in enabling demand-responsive autonomous shuttle services within urban transportation systems. By eliminating the need for repeated centralized re-optimization and allowing policies to adapt online to stochastic passenger demand, the proposed framework offers a scalable foundation for future smart city mobility deployments. Despite operating without prior knowledge of future passenger requests, the proposed learning-based policy achieved performance comparable to a deterministic DARP benchmark in key service metrics, particularly in in-vehicle travel time and overall service duration. Such capabilities are particularly relevant for first–last mile connectivity, paratransit operations, and service provision in transportation-disadvantaged communities. While the present study focuses on a single-agent formulation, the framework is designed to extend naturally to multi-shuttle environments, providing a pathway toward coordinated autonomous transit systems that support sustainable, efficient, and inclusive urban mobility.

Author Contributions: Conceptualization, Hyun Kim; methodology, Hyun Kim; software, Hyun Kim; validation, Hyun Kim and Branislav Dimitrijevic; formal analysis, Hyun Kim and Branislav Dimitrijevic; investigation, Hyun Kim; resources, Hyun Kim; writing—original draft preparation, Hyun Kim; writing—review and editing, Hyun Kim and Branislav Dimitrijevic; visualization, Hyun Kim; supervision, Branislav Dimitrijevic; project administration, Branislav Dimitrijevic; funding acquisition, N.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data and source code supporting the findings of this study are publicly available at https://github.com/kimhyun1018/gail_ppo_ridesharing.

Acknowledgments: During the preparation of this manuscript/study, the authors used ChatGPT for the purposes of checking grammar and refining sentences. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

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

 doi:10.20944/preprints202602.0835.v1

Abbreviations

The following abbreviations are used in this manuscript:

ASP	Autonomous Shuttle Problem
DRL	Deep Reinforcement Learning
GAIL	Generative Adversarial Imitation Learning
MDP	Markov Decision Process
MSDVRP	Multi-agent Stochastic and Dynamic Vehicle Routing

References

1. Burger, A. What Might Autonomous Public Transit Look Like? https://www.vta.gov/blog/what-might-autonomous-public-transit-look, 2021. Accessed: 2024-08-07.
2. Imhof, S.; Fr¨olicher, J.; von Arx, W. Shared Autonomous Vehicles in Rural Public Transportation Systems. *Research in Transportation Economics* **2020**, *83*, 100925.
3. Trubia, S.; Curto, S.; Severino, A.; Arena, F.; Zuccalà, Y. Autonomous Vehicles Effects on Public Transport Systems. In Proceedings of the AIP Conference Proceedings, 2021, Vol. 2343.
4. New Jersey Transit. What Might Autonomous Public Transit Look Like? https://www.njtransit.com/Avatar.
5. Farazi, N.P.; Zou, B.; Ahamed, T.; Barua, L. Deep reinforcement learning in transportation research: A review. *Transportation research interdisciplinary perspectives* **2021**, *11*, 100425.
6. Feng, S.; Duan, P.; Ke, J.; Yang, H. Coordinating ride-sourcing and public transport services with a reinforcement learning approach. *Transportation Research Part C: Emerging Technologies* **2022**, *138*, 103611.
7. Cordeau, J.F. A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. *Operations Research* **2006**, *54*, 573–586.
8. Hiermann, G.; Puchinger, J.; Ropke, S.; Hartl, R.F. The Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Recharging Stations. *European Journal of Operational Research* **2016**, *252*, 995–1018.
9. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2 ed.; The MIT Press, 2018.
10. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* **2017**.
11. Ho, J.; Ermon, S. Generative Adversarial Imitation Learning. In Proceedings of the Advances in Neural Information Processing Systems, 2016, Vol. 29.
12. Wilson, N.H.M.; Colvin, N.J. Computer Control of the Rochester Dial-a-Ride System. Technical Report Report 77-22, Massachusetts Institute of Technology, Center for Transportation Studies, 1977.
13. Psaraftis, H.N. A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem. *Transportation Science* **1980**, *14*, 130–154.
14. Jaw, J.J.; Odoni, A.R.; Psaraftis, H.N.; Wilson, N.H.M. A Heuristic Algorithm for the Multi-Vehicle Advance Request Dial-a-Ride Problem with Time Windows. *Transportation Research Part B: Methodological* **1986**, *20*, 243–257.
15. Xiang, Z.; Chu, C.; Chen, H. The Study of a Dynamic Dial-a-Ride Problem under Time-Dependent and Stochastic Environments. *European Journal of Operational Research* **2008**, *185*, 534–551. https://doi.org/10.1016/j.ejor.2007.01.007.
16. Brownell, C.; Kornhauser, A. A Driverless Alternative: Fleet Size and Cost Requirements for a Statewide Autonomous Taxi Network in New Jersey. *Transportation Research Record* **2014**, *2416*, 73–81.
17. Fagnant, D.J.; Kockelman, K.M. Dynamic Ride-Sharing and Fleet Sizing for a System of Shared Autonomous Vehicles in Austin, Texas. *Transportation* **2018**, *45*, 143–158.
18. Goralzik, A.; König, A.; Alčiauskaitė, L.; Hatzakis, T. Shared mobility services: an accessibility assessment from the perspective of people with disabilities. *European transport research review* **2022**, *14*, 34.
19. Hildebrandt, F.D.; Thomas, B.W.; Ulmer, M.W. Opportunities for Reinforcement Learning in Stochastic Dynamic Vehicle Routing. *Computers & Operations Research* **2023**, *150*, 106071.
20. De Nijs, F.; Walraven, E.; De Weerdt, M.; Spaan, M. Constrained Multiagent Markov Decision Processes: A Taxonomy of Problems and Algorithms. *Journal of Artificial Intelligence Research* **2021**, *70*, 955–1001.
21. ECONorthwest and Parsons Brinckerhoff Quade & Douglas. Estimating the Benefits and Costs of Public Transit Projects: A Guidebook for Practitioners. Technical Report TCRP Report 78, Transportation Research Board, 2002. https://doi.org/10.17226/13787.
22. Google. Routing Options in OR-Tools. https://developers.google.com/optimization/routing/routing_options. Accessed: 2024-11-22.

 doi:10.20944/preprints202602.0835.v1

23. Team SimPy. SimPy: Discrete-Event Simulation for Python. https://simpy.readthedocs.io/, 2024. Accessed: 2025-03-08.
24. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* **2016**, *529*, 484–489.
25. Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, J.; Hashme, S.; Hesse, C.; et al. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680* **2019**.
26. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature* **2019**, *575*, 350–354.
27. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *nature* **2017**, *550*, 354–359.
28. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Journal of Machine Learning Research* **2020**, *21*, 1–51.
29. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *Advances in Neural Information Processing Systems* **2022**, *35*, 24611–24624.