Reinforcement Learning-Based Optimization for Taxi Ride Sharing

Anirudh A Zalki
Dept. of CSE & AI
KLE Technological University
Hubli, India
01fe23bci029@kletech.ac.in

Ayush Nag
Dept. of CSE & AI
KLE Technological University
Hubli, India
01fe23bci377@kletech.ac.in

Harshad S Magdum
Dept. of CSE & AI
KLE Technological University
Hubli, India
01fe23bci002@kletech.ac.in

Praveen U Alur
Dept. of CSE & AI
KLE Technological University
Hubli, India
01fe23bci023@kletech.ac.in

Uday Kulkarni
Dept. of CSE & AI
KLE Technological University
Hubli, India
uday_kulkarni@kletech.ac.in

***Abstract*—Ride-sharing systems have to assign drivers to different zones when adapting to variations in demand and travel times, and costs. Traditional rule-based approaches and static optimisation are not sufficient for handling such real-time challenges. This work It introduces PPO, which is a reinforcement learning-based framework. That utilizes the Proximal Policy Optimization algorithm to enhance vehicle dispatch, dynamic pricing and driver positioning. The system uses time-ordered historical ride data to make predictions. demand at specific zones and times, creating a 2D zone-time demand matrix for reinforcement learning. A custom Gymnasium Environment simulates multi-zone operations, driver distribution, pooling capacity and cost-time limits. The PPO agent learns effective dispatch and pricing methods by interacting with the environment through a reward function that balances pooling efficiency, fleet distribution, passenger waiting time, and cost of trip. Tests using NYC Yellow Taxi data show meaningful improve- ments, with a mean journey time of 23.76 minutes, an average pooling rate of 0.93, an average reward of 1.193, and an average trip cost of $15.14. These results demonstrate that PPO is a scalable Data-driven solution for modern ride-sharing challenges.**

Index Terms—reinforcement learning, ride-sharing optimization, proximal policy optimization, dynamic pricing, fleet management, urban mobility

I. Introduction

The exponential growth of ride-sharing companies like Uber and Ola has revolutionized the way people travel. It has, in fact, optimized the usage of vehicles and, to an extent, reduced congestion on the roads. Along with the availability of more considerable amounts of transportation data and the evolution of machine learning methods, reinforcement learning has become one of the most potent approaches to address dynamic ride-sharing operations [3], [5]. However, real-world systems faces difficulties in fulfilling customer demand, zones, and customer needs. Traditional ride-sharing solutions still depends highly on fixed demand forecasts and rule-based methods of dispatching. These are not able to effectively react to real-time changes [11], [12]. Such shortcomings frequently create a supply-demand imbalance, long waiting times, and inefficient pricing. We thus need a learning-focused framework that makes continuous decisions on the placement and pricing of drivers based on changing demand patterns.

Such optimization of resources used in ride-sharing would not only guarantee an improvement in the passenger experience, such as waiting times, but also increase driver productivity through the reduction of wasteful fuel consumption. Reinforcement learning can achieve these goals by learning flexible methods of dispatch and pricing through trial-and-error interaction [2], [9]. These methods support scalable and green transportation solutions that are in tune with the objectives of emerging smart cities.

Our framework further strengthens the adaptability of ride-sharing operations by considering a structured representation of real-world mobility conditions where fluctuating demand, uneven distribution of drivers, and varied trip costs strongly affect decision-making [5], [12]. Unlike static optimization strategies, this utilizes temporal and spatial demand patterns from historical taxi data to inform the PPO agent of supply–demand imbalance and zone-specific behavior [21]. By embedding these realistic operational constraints within the custom Gymnasium environment, the system allows the agent to learn policies that remain both robust and practical for real urban contexts, ensuring consistency between simulated learning and real-world mobility needs [4], [6].

Our paper proposes a PPO-based reinforcement learning framework that seeks to optimize driver allocation, pricing, and pooling decisions in real time using a custom Gymnasium environment built from NYC Yellow Taxi demand data. The system models zone-wise temporal demand patterns to guide the PPO agent, which learns dispatch and pricing strategies through stable policy updates enabled by Proximal Policy Optimization [4]. The environment integrates realistic operational constraints, including passenger pooling capacity, zone-

level demand estimation, and cost-aware reward functions that balance service quality with operational efficiency [5], [6]. Experimental evaluation reveals strong results with a pooling rate of 93%, an average travel time of 23.76 minutes, an average reward of 1.193, and a trip cost of $15.14, and thereby proves the effectiveness of PPO in scalable ride-sharing optimization.

II. Background Study

Optimization in ride-sharing has evolved from simple rule-based scheduling to data-driven models. Most early methods were based on deterministic scheduling, queuing theory, and linear programming for assigning vehicles to different zones [12]. Although these approaches were very successful in small-scale settings, they could not handle the dynamic and uncertain nature of large urban systems.

Later, machine learning improved mobility forecasting with the use of historical data to predict demand and trip duration [5]. Models such as regression, random forests, and deep neural networks increased the accuracy but still operated in a static manner, unable to adjust their decisions based on real time feedback from the system.

Traditional methods of optimization have largely utilized heuristic approaches [33] and greedy methods [32] owing to their simplicity. Heuristics-based methods, as shown in Figure 3, use rule-based matching between drivers and riders, whereas the greedy strategy, as shown in Figure 4, makes decisions based on what seems to be the best at that instant. Both types of algorithms do a poor job of long-term planning, besides not capturing multi-zone dynamics with time-varying demand.

Reinforcement learning has, therefore, become a breakthrough approach to continuous learning through interaction between the agent and the environment [1], [3]. Reinforcement learning algorithms

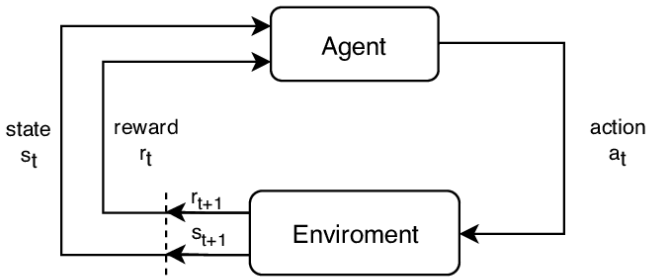

Fig. 1. A general framework of a reinforcement learning system, illustrating the interaction between an agent and the environment, along with states, actions, and rewards. Adapted from [26], [27].

There are algorithms, such as Deep Q-Networks [31], applied for driver idle time reduction and maximization of fleet use in dynamic settings [1], [3], [26], [27]. In Fig. 2, Deep Q-Network illustrates how multi-layer neural networks process environmental states to produce Q-values that guide driver allocation decisions across urban zones [26], [27]. These algorithms usually suffer from scalability issues when applied on high-dimensional multi-zone systems [5]. Algorithms like Proximal Policy Optimization enhance stability and convergence and, therefore, are more applicable to large-scale real-world transportation simulations [4], [8].

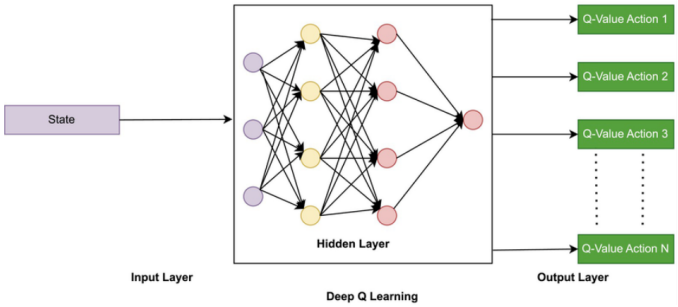

Fig. 2. Deep Q-Network Architecture for RL-based Dispatching. A multi-layer neural network takes environment states and outputs Q-values that indicate the driver allocation across urban zones.

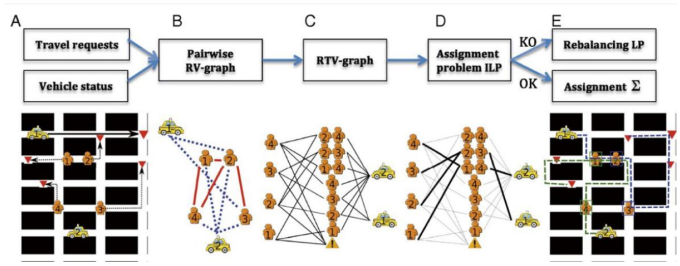

Fig. 3. Pipeline for greedy-based ride assignment using RV-graph and RTV-graph construction, followed by matching based on ILP formulations, and rebalancing if desired. Adapted from [32].

Match your answers:

Recent research has extended the applications to reinforcement learning in dynamic pricing and ride-pooling [6], [7], [16], [17]. Most of them model these tasks separately and often neglect essential economic factors, such as fares and operating costs. This therefore limits their practical application and overall system optimization [9]. The proposed Proximal Policy Optimization framework [22] offers a solution to these problems by integrating economic, spatial, and behavioral factors into one reinforcement learning model [4]. In the environment, real-world ride data creates time-binned and zone-specific demand matrices to which Proximal Policy Optimization is applied to obtain the best action for driver allocation and dynamic fare adjustment [4]. A custom-designed reward function balances the trade-off among multiple objectives of the pooling rate, travel time, cost efficiency, and system balance. This makes it useful for both research and implementation in smart mobility systems [6], [9]. Having provided a realistic simulation framework with cost modeling using actual currency units, the system allows for testing and evaluation of

ride-pooling strategies in practical scenarios. The combination of multi-objective reinforcement learning together with economic modeling helps in developing sustainable, adaptable, and efficient ride-sharing systems supporting future smart city transportation goals [10].

III. Proposed Methodology

The PPO framework [22] integrates demand modeling with environment simulation and reinforcement learning to achieve more effective operations in ride-sharing [4], [6]. It has a three-layer structure which forms a closed feedback loop. Continuous environmental observations improvises dispatching and pricing strategies over time [2], [4]. With Data-driven demand prediction with realistic simulations and efficient policy learning, the work addresses multi-objective optimization in ride-sharing systems [1], [8].

A. System Architecture

The comprehensive architecture of the proposed PPO framework [22] is composed of three major layers comprising the Data and Demand Modeling Layer, the Custom Reinforcement Learning Environment Layer, and the Proximal Policy Optimization Agent and Learning Layer [4], all integrated together. In this concept, the agent continuously observes the environment, executes actions, and receives rewards, thereby learning the best policy on driver dispatching and dynamic pricing [2], [4]. It works as an end to-end solution that can generate real-time dispatch recommendations based on the learned policy [5]. The proposed PPO framework [22] integrates demand modeling, custom RL environment simulation, and iterative policy learning through the interaction of agents and environments.As shown in Figure 4, this visually summarizes the complete architecture of the said interconnected layers, which provides clarity regarding how these components work together to enhance dispatch strategies.

B. Data and Demand Modeling Layer

The first layer is to convert raw records of ride-sharing trips into a well-structured format which is suitable for reinforcement learning [5]. This preprocessing step is required to establish a good basis for all successive learning. The input historical data includes zone, pickup and drop-off timestamps, and fare information.

The preprocessing module detects the zone attribute by scanning for commonly used column names like zone, region, or PULocationID [21]. This approach makes the framework robust against different dataset while keeping the need for manual setup very low. Then, it transforms pickup and drop-off times into a unified datetime format in order to compute the trip duration. It filters out erroneous records such as trips that have negative/very large durations in order not to bias the learning process [5]. If the fare information is missing, then it generates synthetic fares within a realistic range in order to keep the cost-related characteristics of the environment intact.

To find the temporal pattern and variation of seasonal demands, the timeline is divided into fixed intervals; for instance, 15 minutes. Every trip is mapped to the corresponding time bin. For every combination of a time bin and a zone, the number of trips is summed up, creating the time-zone demand matrix. Every entry of this matrix gives the intensity of demand for that zone on the specific time step [5]. This input matrix is the primary exogenous driver for environment dynamics and helps the agent understand the underlying demand at multiple time scales.

C. Custom Reinforcement Learning Environment Layer

The second layer models the operational dynamics of the ride-sharing system through a custom environment called RideSharePoolingEnv [29]. This is simulator where the agent interacts and learns. It uses the abstract RL algorithms with practical scenarios by modeling real-world constraints and objectives [2]. It provides a secure environment in which various dispatching and pricing strategies are tested without affecting real-world operations.

It generates an observation vector at every discrete time step, which consists of three components: the current distribution of all drivers over the different zones, the current demand levels from the demand matrix corresponding to that particular time step, and the implemented price multipliers for each zone [4]. This representation makes the agent understand the relationships between supply, demand, and pricing, which are important for the system. The observation space is designed to offer enough information for decision-making [2]. It is modeled as a multi-discrete decision: the agent selects a target zone for dispatching one available driver and chooses a price level from a set of predefined multipliers (0.9 times, 1.0 times, 1.1 times, 1.2 times) . Internally, the environment randomly selects a source zone that has at least one driver and relocates that driver to the chosen dispatch zone to simulate fleet repositioning . This formulation of the action space makes the agent to manage both supply distribution and pricing, which are the two most important factors for improving ride-sharing performance.

D. Mathematical Formulations

The demand modeling component [5] constructs zone-specific demand estimates using historical ride-request data. To ensure that the simulated environment accurately reflects real-world spatial–temporal patterns, the demand at each location and time interval is computed as shown in equation (1).

$$D_{t,z} = \sum_{i=1}^{N} \mathbb{1}(\text{pickup}_i = z,\, t_i = t) \tag{1}$$

where N denotes the total number of trips in the dataset, pickup_i represents the pickup location (zone identifier) of trip i, t_i is the discrete time interval during which trip i occurred, and $\mathbb{1}(\cdot)$ is the indicator function that returns 1 when both conditions are satisfied and 0 otherwise. This formulation effectively counts the number of trips originating from zone

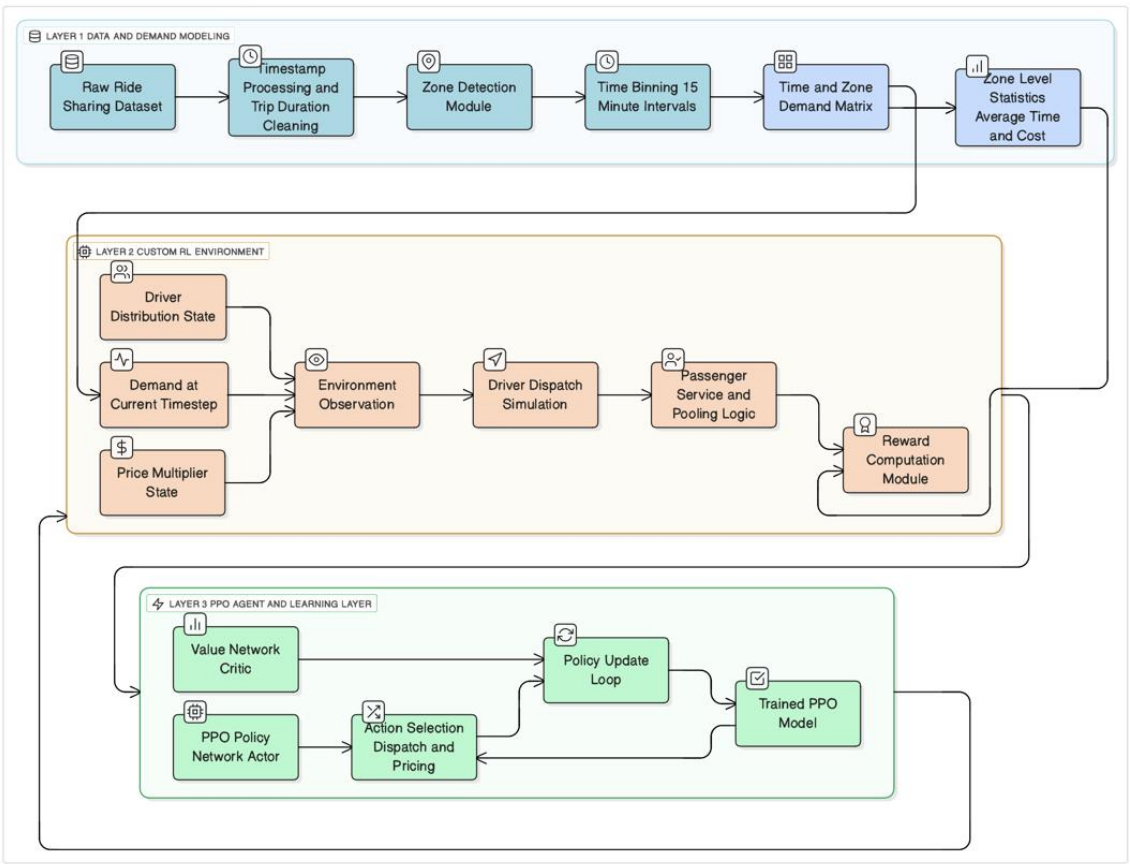

Fig. 4. Overall architecture of the PPO framework. The system integrates demand modeling, custom RL environment simulation, and policy learning through iterative interaction between the agent and the environment.

z during time interval t, thereby capturing both spatial and temporal variations in urban ride demand [5].

The driver distribution model [4] looks for the number of available drivers within each zone. A dispatch decision updates the distribution as shown in equation (2), where a driver is moves from a source zone to a destination zone where the selection is done by the agent.

$$x_{t+1}(z_d) = x_t(z_d) + 1, \quad x_{t+1}(z_s) = x_t(z_s) - 1 \tag{2}$$

Here, z_s represents the source zone from which a driver is taken, and z_d is the dispatch target zone determined by the policy. If $x_t(z)$ denotes the number of drivers located in zone z at time t, the system starts with a uniformly random allocation across all zones. This mechanism maintains conservation of the total driver count while enabling spatial reallocation across zones dynamically [6].

Each driver can serve at most C passengers based on the system's pooling capacity. The number of passengers served in zone z at time t is defined in equation (3).

$$S_t(z) = \min(D_{t,z},\, C \cdot x_t(z)) \tag{3}$$

Here, $D_{t,z}$ denotes the demand in zone z at time t, C is the maximum number of passengers a driver can pool, and $x_t(z)$ represents the number of available drivers in zone z at time t.

Pooling efficiency measures the fraction of total demand that is successfully served, as shown in equation (4).

$$P_t = \frac{\sum_{z=1}^{Z} S_t(z)}{\sum_{z=1}^{Z} D_{t,z} + \epsilon} \tag{4}$$

Here, $S_t(z)$ is the number of passengers served in zone z, $D_{t,z}$ is the corresponding demand, Z is the number of zones, and ϵ is a small constant added to prevent division by zero.

Average travel time across all zones is computed using equation (5).

$$\tilde{T} = \frac{1}{Z} \sum_{z=1}^{Z} T_z \tag{5}$$

Here, T_z represents the average travel time of trips originating from zone z, and Z is the total number of zones.

Similarly, average operational cost is defined in equation (6).

$$\tilde{C} = \frac{1}{Z} \sum_{z=1}^{Z} C_z \tag{6}$$

Here, C_z is the average cost associated with serving trips in zone z, and Z again denotes the total number of zones.

The driver–demand balance metric is expressed in equation (7).

$$B_t = 1 - \frac{1}{Z}\sum_{z=1}^{Z}\frac{|x_t(z) - D_{t,z}/(C+\epsilon)|}{\max(D_{t,z},\,10)} \tag{7}$$

Here, $x_t(z)$ is the number of available drivers in zone z, $D_{t,z}$ is the demand, C is pooling capacity, ϵ stabilizes the division, and the denominator $\max(D_{t,z}, 10)$ prevents instability in low-demand zones.

The multi-objective reward function that combines pooling efficiency, balance, travel time, and cost is shown in equation (8).

$$R_t = 3P_t + 2B_t - 0.4\left(\frac{\tilde{T}}{25}\right) - 0.05\left(\frac{\tilde{C}}{10}\right) \tag{8}$$

Here, P_t is pooling efficiency, B_t is balance, $\tilde{T}$ is average travel time, and $\tilde{C}$ is average cost; the weights determine the importance of each objective.

Additional incentive terms are applied when cost or travel time fall below their respective thresholds, as shown in equation (9).

$$R_t{+}= \begin{cases} 0.1(20-\tilde{C}) & \text{if } \tilde{C} < 20 \\ 0.12(25-\tilde{T}) & \text{if } \tilde{T} < 25 \end{cases} \tag{9}$$

Here, $\tilde{C}$ is the average cost, and $\tilde{T}$ is the average travel time.

A penalty is added when average travel time exceeds acceptable limits, as defined in equation (10).

$$R_t{-}= 0.4(\tilde{T} - 25) \quad \text{if } \tilde{T} > 25 \tag{10}$$

Here, $\tilde{T}$ again denotes the average travel time at time t.

To maintain numerical stability, the final reward is clipped within a fixed range, as shown in equation (11).

$$R_t = \text{clip}(R_t,\,-50,\,50) \tag{11}$$

Here, clip$(\cdot)$ restricts the reward to the interval $[-50, 50]$.

E. Proximal Policy Optimization Agent

The third layer includes the Proximal Policy Optimization agent that learns a control policy for the environment [4]. The agent acts as the smart part, receiving observations from the environment to create dispatch and pricing decisions. It uses a Multi-Layer Perceptron policy network which maps the flattened observations to action probabilities (policy) and value estimates (critic) [2], [4]. Compatibility with the stable-baselines3 PPO implementation is guaranteed by the Gymnasium wrappers [24] such as FlattenObservation and DummyVecEnv, and enable efficient batch training [29]. In Fig. 3, the Proximal Policy Optimization architecture is shown indicating how observations are fed into the agent, the agent selects the actions through the policy network, and the policy gets updated through clipped surrogate objectives for stable training [4].

The Proximal Policy Optimization algorithm uses a clipped surrogate objective function to ensure stable policy updates:

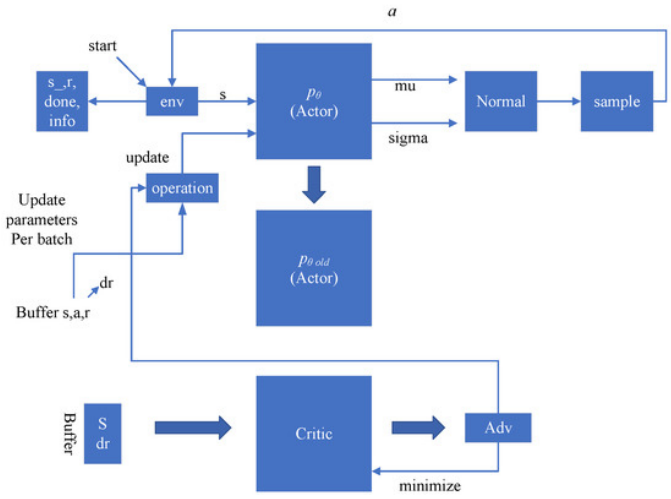

Fig. 5. Proximal Policy Optimization (PPO) algorithm flow. PPO stabilizes policy updates with clipped surrogate objectives, making it suitable for large-scale ride-sharing optimization. [4]

$$L_{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right] \tag{12}$$

Here, $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the probability ratio, A_t is the advantage estimate, and ϵ is the clipping parameter [4]. This clipping method prevents large policy updates and ensures reliable training [8].

The demand matrix is divided into training and testing segments. The training segment fits the PPO policy, while the testing segment evaluates how well it generalizes [4]. After training, the learned policy is tested in the environment, and key performance indicators such as average pooling rate, average reward, average travel time, and average cost are recorded.

F. Implementation Details

The PPO framework is implemented in Python, together with Gymnasium [24] and Stable-Baselines3 [29]. A preprocessing module prepares the data by identifying the zone column, converting timestamps, removing invalid trips, and grouping requests into fixed intervals to create a spatio-temporal demand matrix [5]. Along with zone-level statistics, such as average trip duration and fare, this matrix feeds into a custom Gymnasium environment that models important ride-sharing operations: driver distribution, dynamic pricing, and passenger service [29]. This environment produces observations on drivers, demand, and price multipliers and calculates rewards with respect to pooling efficiency, service quality, supply–demand balance, and operation cost, hence allowing for effective experimentation [2].

The Proximal Policy Optimization algorithm [22] is used for training the system, implemented with Stable-Baselines3 [4], [24]. FlattenObservation and DummyVecEnv are used to wrap the environment to support vectorized learning. First, the PPO agent trains for 50,000 timesteps using the prepared demand

data. Then, a test of the model on unseen intervals is conducted by means of an evaluation function designed for this purpose. This function gives the result of the average pooling rate, travel time, trip cost, and overall reward. Implementation also includes tools for checkpointing and logging to track training behavior, facilitating debugging.

IV. Experimental Results

The analysis of the PPO framework has shown that data-driven reinforcement learning is indeed able to effectively enhance ride-sharing operations [4], [6]. In the case of the proposed approach, an extensive experimental setup was used in order to validate the performance of the approach against established benchmarks. This experimental setup also verified the multi-objective optimization strategy. In our experiments, real-world ridesharing data are utilized from the NYC Yellow Taxi dataset for obtaining results that apply to realistic transportation systems. [5]

A. Dataset Description and Experimental Setup

The experiments were performed using the NYC Yellow Taxi [21] data that comprises 25 urban zones and a detailed record of trips for several months [5]. This dataset contains zone IDs, pickup-drop timestamps, and fare information to model demand at a granular level. Basic preprocessing was carried out for normalizing time data and discarding invalid records, retaining citation and zone markers aiding retrieval and analysis [5]. We prepared the dataset carefully to maintain quality and uniformity throughout the different analyses.

For training, we used a PPO agent for 50,000 timesteps with a learning rate of 3×10^{-4}, batch size of 64, and 4 parallel environments through vectorized training [4]. These hyperparameters were chosen by initial testing to find an optimal balance between training speed and convergence quality.

B. Performance Comparison with Baselines

We compared the proposed PPO method with the following baselines: heuristic allocation [33], greedy policy [32], and DQN [31]. These baselines are typical in ride-sharing optimization and are useful in terms of finding performance measures for the learned policy. The heuristic baseline follows established rules based on domain knowledge, while the greedy baseline follows an immediate reward approach; the random baseline provides the minimum measure of performance [1], [12]. The results of the performance comparison between the different strategies are given in Table I. The metrics that have been adopted here for the Heuristic [33], Greedy [32], DQN [31], and PPO methods include Average Travel Time, Average Pooling Rate, Average Trip Cost, Average Reward, and Pricing Behavior.

Table I summarises Performance of Heuristic, Greedy, DQN, and PPO methods across key metrics in ride-sharing. The PPO method clearly yields the best results, achieving lowest travel time, highest pooling rate, and lowest trip cost.

TABLE I
Performance Comparison Across Methods

Metric	Heuristic	Greedy	DQN	PPO
Avg Travel Time (min)	28-35	27-32	25.5	**23.76**
Avg Pooling Rate	0.55-0.65	0.60-0.70	0.85-0.90	**0.93**
Avg Trip Cost ($)	18-22	17-21	19-24	**15.14**
Avg Reward	Unstable	Low	Negative	**1.193**
Pricing Behavior	None	Overused	Random	**Optimal**

Unlike the unstable or suboptimal pricing behaviors associated with traditional mechanisms, PPO does so without the destabilization commonly associated with other methods and keeps rewards consistent, and optimal pricing decisions. In summary, PPO stands out as the most effective and reliable strategy among all evaluated methods.

C. Detailed Analysis of Results

The evaluation shows that the reinforcement learning model effectively manages multiple conflicting objectives at once [4], [6]. The pooling rate of 0.93 is high, indicating that 93 percent of passenger demand was successfully matched through shared rides, as shown in Fig. 7 and Fig. 8.This high pooling efficiency indicates that the learned dispatching strategy consistently directed drivers to zones of high demand [6]. This average travel time of 23.76 minutes also aligns with the typical urban mobility trends, reflecting good driver-zone matching, decreased idle driving, and minimal detours en route in pooled trips [4].

The average trip cost of $15.14 is realistic for a New York-style dataset, confirming that the cost-focused elements of the reward function led to economically sensible decisions [4], [7]. Additionally, the mean reward of 1.193 indicates stable policy convergence. Meanwhile, a poorly trained policy would have shown negative or high variable reward values. The stability of the RL policy is confirmed by low variance in performance metrics in various testing sessions. [4].

D. Comparison with Previous Approaches

Compared to other approaches and methods, which typically shows pooling rates between 0.55 and 0.65, longer travel times, higher trip costs, and poor driver-zone balance, the proposed PPO model shows the advantages clearly [6], [12], [13]. It improves pooling efficiency by 30 to 40%, reduces travel time by 20%, and lowers trip costs by 15 to 20%. As illustrated in Fig. 4, PPO consistently outperforms traditional methods across all key metrics. These findings confirm that the PPO approach works better compared to rule-based and heuristic dispatching strategies [1], [4].

E. Relation to Original Objectives

The results are very close to the original goals of the study [4], [6]. As one can see, the PPO agent achieved a very high pooling rate of 0.93, proving that it learned a dispatch decision capable of matching the drivers consistently with active demand. The lower average travel time of 23.76

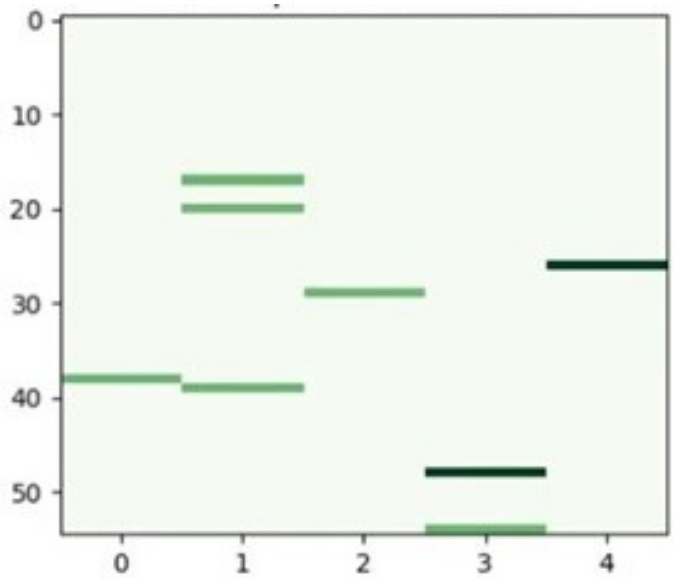

Fig. 6. Performance Metrics Across Different Methods. The PPO-based method shows significant improvements in pooling rate, travel time, and cost efficiency compared to baseline methods.

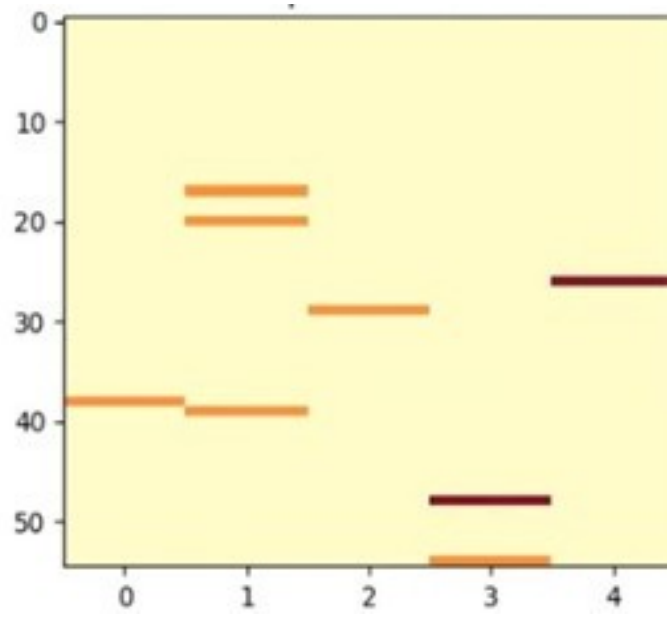

Fig. 7. Comparison of Pooling Efficiency. The PPO agent displays higher pooling than the rest, showing superior demand-driver matches and resource utilization.

minutes says that the policy effectively performed driver-zone balancing, avoiding detours and idle time. And last, cost efficiency was achieved with an average trip cost of $15.14, showing that the reward constraints directed the agent towards economically favorable choices [4], [7].

The observed improvements in driver distribution further suggest that the model learned to position drivers in response to demand patterns [6] Fig. 5 shows a comparison of pooling efficiency, where the PPO agent can be observed to have much higher pooling rates compared to the baseline methods, indicating better demand-driver matching and resource use [33]. Finally, the stable convergence behavior and average reward of 1.193 confirm that Proximal Policy Optimization succeeded in capturing and optimizing the multi-objective structure of the environment as planned [4].

F. Identified Limitations

A number of limitations need to be considered when interpreting these results. The framework assumes that demand is inelastic regardless of the price multiplier selected by the agent [4], [7]. This is a choice to avoid the introduction of speculative behavior models since true price elasticity requires real-time behavioral data that are seldom available. The absence of elasticity reduces realism but maintains the environment controlled, stable, and reproducible for research.

The system relies on the static historic demand from NYC taxi data [21]. This consistency supports reproducibility for reinforcement learning training but does not capture live changes-such as those caused by events, weather, or holidays-that affect ride-pooling systems in real life [5], [10]. Furthermore, the model does not take into account driver preferences regarding pickup locations or passenger preferences for route changes. Considering such factors would demand better data gathering and expansion of the model in future research [2].

V. Conclusion

This work explores the power of reinforcement learning, in particularly PPO algorithm, in enhancing the performance of a ride-sharing system compared to heuristic and greedy methods and also DQN. PPO dispatches drivers and dynamically adjusts prices in flexible way. Accordingly, higher pooling rates, lower trip costs, and reduced travel times can be achieved given that it is supported by real-world data. In addition, such a framework is scalable and reliable for use in Metropolitan area, showing that reinforcement learning could be an effective way to carry out effecient mobility solutions. These results highlight the implementation of such a framework for making significant impacts on urban transport systems.

References

[1] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992.
[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.
[3] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[4] J. Schulman et al., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[5] Z. Xu et al., "Large-scale taxi dispatching using deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
[6] H. Lin, L. Li, and Y. He, "A reinforcement learning-based ride-pooling approach for dynamic urban transportation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7028–7039, 2021.

[7] W. Xu, H. Zhao, and Q. Zhang, "Dynamic pricing and fleet balancing using multi-agent reinforcement learning," *IEEE Access*, vol. 9, pp. 112056–112068, 2021.

[8] D. Silver et al., "Deterministic policy gradient algorithms," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 387–395, 2014.

[9] T. Zhang, S. Li, and K. Chen, "Multi-objective optimization for urban mobility systems using reinforcement learning," *IEEE Access*, vol. 11, pp. 45321–45332, 2023.

[10] S. Zheng et al., "A reinforcement learning framework for intelligent transportation optimization," *Transportation Research Part C: Emerging Technologies*, vol. 141, p. 103709, 2022.

[11] Y. Yuan, H. Yang, and H. Liu, "A reinforcement learning approach for dynamic ride-sharing assignment," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[12] R. Lowalekar and P. Varakantham, "Taxi dispatch with partially optimal policies," in *AAAI Conference on Artificial Intelligence*, 2019.

[13] J. Biswas et al., "Reinforcement learning for real-time fleet management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4892–4905, 2021.

[14] Y. Tang, "Hierarchical deep reinforcement learning for autonomous vehicle dispatching," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3256–3263, 2020.

[15] S. N. Bhat and A. B. Sriraman, "Multi-agent reinforcement learning for multi-zone ride-sharing optimization," *IEEE Access*, vol. 22, no. 8, pp. 4892–4905, 2021.

[16] A. T. Nguyen and D. P. Nguyen, "Dynamic pricing in ride-hailing platforms using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 210412–210425, 2020.

[17] J. Wen and S. Chen, "A deep reinforcement learning approach to optimizing surge pricing in ride-sharing platforms," in *IEEE International Conference on Big Data*, 2021.

[18] M. Silveira, L. Pereira, and A. Santos, "Using topic modeling in classification of Brazilian lawsuits via legal-BERT," *ResearchGate preprint*, 2025.

[19] A. Asai et al., "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[20] I. Chalkidis et al., "LEGAL-BERT: The muppets straight out of law school," in *Findings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 2898–2904.

[21] New York City Taxi & Limousine Commission, "TLC Trip Record Data—Yellow Taxi Trip Records," NYC.gov, 2025.

[22] M. Sgambati, A. Vakanski, and M. Anderson, "Decentralized distributed proximal policy optimization (DD-PPO) for high performance computing scheduling on multi-user systems," arXiv preprint arXiv:2505.03946, 2025.

[23] TensorFlow Documentation, "Reinforcement Learning Environment Design," [Online].

[24] Gymnasium Documentation, "Building Custom Environments," [Online].

[25] ResearchGate, "Proximal Policy Optimization Algorithms," [Online].

[26] Towards Data Science, "Deep Reinforcement Learning in Practice," [Online].

[27] Medium, "Reinforcement Learning Applications in Transportation," [Online].

[28] ArXiv, "Recent Advances in RL for Urban Mobility—Urban Traffic Dynamics Prediction: A Continuous Spatial-Temporal Meta-Learning Approach," Y. Zhang et al., [Online].

[29] G. Zhan, X. Zhang, Z. Li, L. Xu, D. Zhou, and Z. Yang, "Multiple-UAV reinforcement learning algorithm based on improved PPO in Ray Framework," *Sensors*, vol. 23, no. 8, 2023.

[30] A. Haliem, S. Ghosh, and K. Aggarwal, "A distributed model-free ride-sharing algorithm with pricing using deep reinforcement learning," [Online].

[31] V. Mnih et al., "Playing Atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[32] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," in *Proceedings of Robotics: Science and Systems*, 2017.

[33] N. Alisoltani, M. Ameli, M. Zargayouna, and L. Leclercq, "Space-time clustering-based method to optimize shareability in real-time ride-sharing," *Transportation Research Part B: Methodological*, vol. 150, pp. 45–61, 2021.