E-VaaS: Edge-Enabled Vehicle-as-a-Service for Smart Transportation Systems

Priyanshu Jogdand[1], Anjani Kumar[1], Ayan Mondal[1], and Erkki Harjula[2]

[1] Department of Computer Science and Engineering, Indian Institute of Technology Indore, India
[2] Centre for Wireless Communications, University of Oulu, Oulu, Finland

Abstract. Traffic congestion and low vehicle occupancy rates pose significant challenges in urban areas, highlighting the need for efficient, shared transportation systems. Hence, in this work, we introduce an Edge-Enabled Vehicle-as-a-Service scheme, named E-VaaS, designed to improve traffic flow, reduce travel times, and enhance user satisfaction in intelligent transportation networks. By combining the *Gale-Shapley stable matching* algorithm with edge computing, the proposed E-VaaS system effectively schedules and matches riders to vehicles based on metrics that include travel time, distance, and cost. With the help of edge computing, E-VaaS enables low-latency processing, ensuring that ride-matching decisions are made swiftly and improving system responsiveness and user experience. Experimental results validate the effectiveness of the model, showing improved vehicle occupancy, reduced travel times and costs, and a balanced distribution of shared vehicles across the transportation network. This research advances the development of stable, scalable, and efficient ride-sharing solutions for urban environments.

Keywords: Edge networks · Ride-Sharing · Stable matching · Smart transportation · Game Theory

1 Introduction

In modern urban environments, the challenges of traffic congestion and inefficient vehicle usage are becoming increasingly critical. With growing urban populations, the traditional model of private vehicle ownership has led to substantial under-utilization of transportation resources, where individual vehicles frequently operate with low occupancy rates. This contributes to environmental issues and economic inefficiencies, as congested roads result in wasted time and higher fuel consumption. For example, according to the 2021 TomTom Traffic Index [2], cities such as Istanbul, Moscow, and Kyiv, suffer from severe traffic congestion, leading to hours of delay for commuters annually. Addressing these challenges requires a shift from individual vehicle ownership toward shared transportation solutions that optimize vehicle occupancy and improve traffic flow.

We argue that ride-sharing can be one of the promising solutions to these issues by encouraging individuals with similar destinations or routes to share a

single vehicle. Various digital platforms have facilitated this practice, but traditional ride-sharing models often rely on centralized cloud-based systems, which can struggle to process requests efficiently during peak times. This also contributes heavily to a high carbon footprint. These systems typically use basic matching algorithms that focus on geographic proximity rather than a more holistic measure of user satisfaction, which includes factors like travel time, cost, and route efficiency. Consequently, they may fall short of providing the quick response and optimal matches that urban travelers expect.

To address these limitations, we introduce an Edge-Enabled Vehicle-as-a-Service (E-VaaS) system that combines the Gale-Shapley stable matching algorithm with edge computing to deliver a more efficient and user-centered ride-sharing experience. This novel E-VaaS framework is designed to improve traffic efficiency by maximizing vehicle occupancy, minimizing travel costs, and optimizing travel time, thereby enhancing traveler satisfaction. By integrating edge computing, the E-VaaS system reduces latency by processing data closer to end-users at edge nodes rather than solely relying on centralized cloud resources. This distributed architecture enables faster response times, allowing the system to match riders and vehicles in real time, essential in dynamic and high-demand urban contexts. In summary, our primary contributions are as follows:

1. We introduce a novel E-VaaS system that combines the Gale-Shapley stable matching algorithm with edge computing to achieve stable, real-time ride-sharing.
2. We present a satisfaction-driven matching framework that considers travel time and cost, enabling personalized matches that improve vehicle occupancy and user satisfaction.
3. We propose an edge-based ride request transmission mechanism, which reduces latency and ensures timely, responsive service for riders and drivers alike.
4. We evaluated and validated experimentally the effectiveness of the model, improvement of vehicle occupancy, reduced travel times and costs, and a balanced distribution of shared vehicles across the transportation network.

2 Related Works

There are a few existing works that focus on the concept of ride-sharing. Additionally, the integration of edge computing into public vehicle (PV) systems has garnered significant attention in recent years, aiming to enhance traffic efficiency, optimize vehicle occupancy ratios, and reduce overall congestion. Zhang et al. [6] presented a comprehensive Edge Computing Based Public Vehicle (ECPV) system designed to address these challenges by leveraging edge devices for real-time ride-sharing and vehicle scheduling. Their approach emphasizes the reduction of decision-making latency through distributed computing, thereby improving the quality of experience (QoE) for travelers. Previous research on the matching problem in ride-sharing systems has primarily addressed optimization challenges in route planning, dynamic pricing, and cost-sharing mechanisms. For instance,

Guan *et al.* [4] examined ride-sharing trade-offs from a multi-objective perspective, aiming to maximize trip-sharing willingness and minimize total vehicle usage costs. Other studies have focused on dynamic pricing schemes, highlighting how fluctuating demand can result in pricing imbalances, leading to inefficiencies in matching and incentivizing drivers. Notably, Xie *et al.* [5] proposed a cost-sharing approach utilizing a double auction mechanism to maximize cost savings for drivers and passengers within specific timing constraints. Troyan *et al.* [1] proposed another approach that contrasts with standard commercial models by focusing on stable matching. Using the Gale–Shapley algorithm, adapted to accommodate different driver and passenger capacities, the authors demonstrate that the algorithm always reaches a stable matching. The experimental simulations of their model reinforce its effectiveness by validating stable matches across multiple randomly generated test cases, emphasizing the model's stability and scalability.

The proposed E-VaaS builds on the iterated version of the Gale-Shapley algorithm by introducing refined utility functions tailored to the cost-time trade-offs in ride-sharing systems, enhancing matching efficiency. By leveraging edge computing, we also reduce latency in real-time matching processes, addressing a critical gap in existing systems where rapid response times are essential for optimal user experience and system performance. This unique approach aims to deliver a robust, low-latency matching solution that improves stability and operational speed.

3 System Model

We consider a road network abstracted as a directed multi-graph $G = (N, E)$, where N and E represent the set of nodes corresponding to intersection points and the set of directed edges representing road segments between these nodes, respectively. Each user u, traveling along a path in G, follows a sequence

$$P_{0k} = n_0 \xrightarrow{e_{01}} n_1 \xrightarrow{e_{12}} \cdots \xrightarrow{e_{jk}} n_k$$

where $n_i \in N$, $e_{jk} \in E$, and $0 \leq i, j, k \leq k$. e_{jk} represents the directed edge connecting two nodes n_j and n_k. A sub-path $P_{ij} = n_i \cdots \xrightarrow{e_{ij}} \cdots n_j$ is considered a proper segment of a path P_{0k} if $\{e_{ij}\} \subseteq \{e_{0k}\}$. A weighting function $\omega : E \to \mathbb{R}^+$ is defined to represent the cost, e.g., travel time and distance, associated with each edge of the graph. The total cost of traversing the path P_{ij} is given by

$$\omega(P_{ij}) = \sum \omega(e_{ij})$$

In E-VaaS, we consider the roadside units (RSUs) as edge devices that ensure operational efficiency. Vehicles exchange real-time route and scheduling information with the closest RSUs over wireless networks. The cloud data center communicates with these edge devices via wired connections for global data aggregation and analysis. This distributed architecture allows edge devices to process and respond to ride requests while reducing latency significantly.

Each node $n \in N$ in the road network hosts an edge device or RSU, ensuring comprehensive coverage. These edge devices facilitate real-time communication with vehicles and handle localized data processing, thereby alleviating the load on central cloud servers and improving the scalability and responsiveness of the system.

4 E-VaaS: The Proposed Edge-Enabled Vehicle-as-a-Service System

4.1 Justification of Using Gale-Shapley Stable Matching Algorithm

A central component of the proposed E-VaaS system is adapting the Gale-Shapley stable matching algorithm [3]. It is traditionally used to solve matching problems with participants having individual preferences. Hence, it is particularly effective in situations requiring stable matches, ensuring that no rider-vehicle pair would prefer a different matching over the one assigned. In ride-sharing, the Gale-Shapley algorithm is adapted to consider the preferences of both riders and vehicles based on a utility function that incorporates travel time, distance, and cost. Each rider and vehicle has preferences calculated using a weighted sum of these factors, where time and cost often have the highest weights. The goal is to create stable matches that align with the preferences of both parties, reducing the likelihood of mid-trip cancellations or disruptions.

4.2 Mathematical Model Formulation

The utility function in E-VaaS aims to balance travel time and cost. For instance, travelers with more flexibility in timing may prefer routes with lower costs, while those on tight schedules might prioritize shorter travel times. This preference-based approach allows the system to generate personalized matches that increase user satisfaction. Additionally, using the Gale-Shapley algorithm enables the system to dynamically adjust matches as new requests are received, which is particularly beneficial during peak times when demand is high. By optimizing for both time and cost, the E-VaaS model increases the likelihood of achieving stable matches and incentivizes more users to participate in the ride-sharing service, as their preferences are accounted for.

To further enhance efficiency, our E-VaaS system leverages edge computing to handle the processing and decision-making closer to the users, thus reducing the dependency on a central server. In traditional cloud-based ride-sharing systems, requests are sent to a centralized server, which may result in significant latency due to network delays and heavy computational loads. Edge computing mitigates these issues by distributing the computational tasks across multiple edge nodes near end-users. This proximity allows edge nodes to quickly process ride requests and make matching decisions locally, thus enabling real-time response and reducing the load on cloud resources. With edge nodes handling the bulk of the processing, latency is minimized, which is critical for a responsive ride-sharing system that operates efficiently under fluctuating demand.

Dynamic Zone Management with Edge Devices A specific edge device manages each geographic area or zone within the city. The edge device serves as a local computational hub, responsible for handling the matching requests, processing waiting lists, and monitoring vehicles within its designated zone.

Vehicles entering or leaving a zone inform the respective edge device of their status. When a vehicle (driver) enters a zone, it sends an entry message to the edge device, signaling that it is now available for matching within that zone. Similarly, when a vehicle exits, it sends a departure message informing the edge device, which removes it from the list of available drivers in that zone.

Matching Process and Edge Computation The matching algorithm is run locally on each edge device, which handles requests from passengers and drivers within its specific zone. Each edge device represents the system as a bipartite graph, where nodes represent passengers and drivers in the zone, and edges signify potential matches with weights based on preference and utility levels.

4.3 Handling the Waiting List with Edge Devices

In cases where passengers are not matched within a few iterations, they are placed on a waiting list within the edge device's queue. Suppose a passenger remains on the waiting list without a match after a set number of iterations. The waiting list data is forwarded to nearby edge devices to explore broader matching options. This forwarding mechanism leverages the proximity of other edge devices, which may have nearby drivers that could pick up the unmatched passenger. In a hierarchical or peer-to-peer configuration, the closest edge devices can include the waiting passenger in their following matching process.

Inter-Zone Coordination via Edge Devices Edge devices maintain lightweight communication with neighboring edge devices, allowing passengers on the waiting list to have a greater chance of finding a driver even if their zone lacks available vehicles. When a new driver enters a zone and registers with the local edge device, the system checks if any passengers are still on the waiting list, locally or from nearby zones. Inter-zone coordination can help resolve cases where passengers in low-density areas might have experienced delays.

5 Theoretical Framework for E-VaaS

In our study, we introduce an **enhanced utility function** to assess and optimize ride-sharing matches between drivers and passengers. The utility function is a core concept in economic and mathematical optimization, representing a measure of preference or satisfaction. Here, our utility function quantifies the collective benefit or satisfaction that a potential ride-sharing arrangement brings to both drivers and passengers. The main aim of this utility function is to ensure that each match not only reduces costs and travel times for both parties

but also respects individual rationality, where both driver and passenger benefit positively from the shared ride. This optimization is further supported by theoretical constraints and efficiency principles, ensuring the proposed matching process is practical and effective.

5.1 Utility Function Definition

The utility function $U(d, p)$ captures the net benefit of a shared ride between a driver d and a passenger p by considering cost savings and time efficiencies. Formally, this function is represented as follows:

$$U(d, p) = w_c \cdot [C_d(p) + C_p(d)] + w_t \cdot [T_d(p) + T_p(d)]$$

where w_c and w_t are weight coefficients for cost and time, constrained by $w_c + w_t = 1$ and $w_c, w_t \in [0, 1]$; $C_d(p)$ and $C_p(d)$ represent the cost-related functions for the driver and passenger, respectively; and $T_d(p)$ and $T_p(d)$ represent the time-related functions for the driver and passenger, respectively.

By balancing cost and time through these weighted terms, the function allows us to quantify the **overall utility** of a ride-sharing match. This model aims to maximize $U(d, p)$ for pairs of drivers and passengers to determine optimal, mutually beneficial matches.

5.2 Cost Savings Analysis

Cost savings are crucial for drivers and passengers considering ride-sharing, as both are motivated by potential reductions in travel expenses.

Driver's Cost Function $C_d(p)$ For a driver, the cost function $C_d(p)$ represents the difference between the revenue received from the passenger $R_d(p)$ and any additional operational costs incurred due to the shared ride, denoted $O_d(p)$. Mathematically:

$$C_d(p) = R_d(p) - O_d(p)$$

where $R_d(p) = \beta \cdot d_p \cdot r$, with β as a fare share coefficient that defines the proportion of revenue received by the driver, d_p as the trip distance of the passenger, and r as the base fare rate; and $O_d(p) = (d_{det} \cdot c_r) + (t_{det} \cdot c_t)$, where d_{det} is the detour distance incurred to accommodate the passenger, c_r is the cost per distance unit, and $t_{det} = \frac{d_{det}}{v}$ is the detour time at average velocity v, multiplied by a time-based cost c_t.

Passenger's Cost Function $C_p(d)$ For a passenger, the cost function $C_p(d)$ quantifies the savings achieved by sharing the ride instead of bearing the full cost of a solo trip. It is represented as:

$$C_p(d) = C_s - C_r$$

where $C_s = d_p \cdot r$ is the cost of a solo trip; and $C_r = d_p \cdot r \cdot (1 - \delta)$ represents the reduced cost in the shared trip, with δ as a discount factor.

Thus, the passenger's utility from cost savings stems from the difference between the solo trip cost and the discounted shared trip cost.

5.3 Latency Analysis

In addition to cost, both drivers and passengers seek time efficiencies in ride-sharing. Time functions capture the impact of shared travel on each party's total journey duration.

Driver's Time Function $T_d(p)$ The driver's time-related benefit is calculated based on any additional time incurred from the detour and waiting time for the passenger:

$$T_d(p) = -(t_{det} + t_{wait})$$

where t_{wait} represents the waiting time for the passenger. A negative value indicates that these elements reduce the driver's utility.

Passenger's Time Function $T_p(d)$ The passenger's time savings function compares the time for a solo trip with that of a shared ride:

$$T_p(d) = T_s - T_r$$

where $T_s = \frac{d_p}{v}$ represents the solo travel time. $T_r = \frac{d_p + d_{det}}{v} + t_{wait}$ is the shared trip time, factoring in the detour and waiting time. This function captures how much travel time the passenger saves (or loses) by sharing a ride with the driver.

5.4 Theoretical Analysis

To ensure the effectiveness of the utility-based ride-sharing model, we establish two primary theorems [3]: **Individual Rationality** and **Route Efficiency**.

Theorem 1: Individual Rationality A ride-sharing match between driver d and passenger p is deemed rational if both benefit positively from the shared ride. Formally, this condition is satisfied if:

$$U(d, p) > 0 \iff U_d(p) > 0 \land U_p(d) > 0$$

Proof:

1. **Forward Direction** ($\Rightarrow$): If $U(d, p) > 0$, then both the driver's and passenger's components of utility must also be positive.
2. **Reverse Direction** ($\Leftarrow$): If both driver and passenger utilities are positive, then the overall utility $U(d, p)$ is also positive.

Theorem 2: Route Efficiency Route efficiency ensures minimal detours, keeping the shared route convenient and cost-effective. This is characterized by the **detour ratio** $\frac{d_{det}}{d_p}$, which should be bounded by a constant ϵ:

$$\exists\, \epsilon > 0 : \frac{d_{det}}{d_p} \leq \epsilon \iff \text{match is route-efficient}$$

Proof:

1. **Forward Direction** ($\Rightarrow$): Given $\frac{d_{det}}{d_p} \leq \epsilon$, the total time remains within acceptable bounds, ensuring route efficiency.
2. **Reverse Direction** ($\Leftarrow$): If the match is route-efficient, then $\frac{d_{det}}{d_p} \leq \epsilon$ must hold to maintain bounded travel times.

Corollary 1: Cost-Time Trade-off Balancing cost and time effectively within the utility function is crucial for maximizing matches. The optimal weights w_c^* and w_t^* for cost and time, respectively, can be derived as:

$$w_c^* = \frac{T}{C+T}, \quad w_t^* = \frac{C}{C+T}$$

where $C = C_d(p) + C_p(d)$ and $T = T_d(p) + T_p(d)$ are the combined cost and time components, respectively. This yields the highest possible utility for the given weight constraints.

This utility-based model provides a structured, theoretically sound framework for maximizing mutual benefit in ride-sharing arrangements. By balancing cost savings and time efficiencies, and applying constraints for rationality and route efficiency, this model guides an optimal ride-sharing matching process, effectively implemented through the Gale-Shapley algorithm. This approach improves user satisfaction and enhances system efficiency, encouraging broader adoption of shared mobility solutions.

5.5 Algorithm

This work proposes a new algorithm to address the stable matching problem in ride-sharing systems, adapting the well-known Gale-Shapley algorithm to a unique context. The proposed algorithm iterates over the Gale-Shapley method, modifying the traditional constraints on matching. Here, unlike in the original algorithm, the number of drivers and passengers need not be equal, and the algorithm can accommodate incomplete or non-strict preference lists, which more realistically reflect real-world ride-sharing environments.

To set up the matching framework, we define the users in the system as either passengers (type p) or drivers (type d). Each driver d has a limited number of seats available in their vehicle, denoted by s, and each passenger p seeks to occupy a seat. A total seat count, N_{seats}, represents all available seats in the ride-sharing system, and the number of passengers n_p should not exceed this capacity.

The algorithm introduces a "waiting list" for unmatched passengers, ensuring that any unmatched passenger is given a final option to join this list. If a passenger p is on the waiting list, it implies they could not find a driver, and in the model, this waiting list serves as a fallback with minimal priority for matching purposes.

To implement this matching process, the proposed algorithm represents the system as a bipartite graph, where nodes represent users (passengers and drivers), and edges represent possible pairings. Each edge between a passenger p_i and a driver d_j has a weight, which reflects the utility or preference level that p_i has for d_j, and vice versa. This setup allows us to represent the relationships between passengers and drivers as directed edges with weights for each direction, capturing each user's utility.

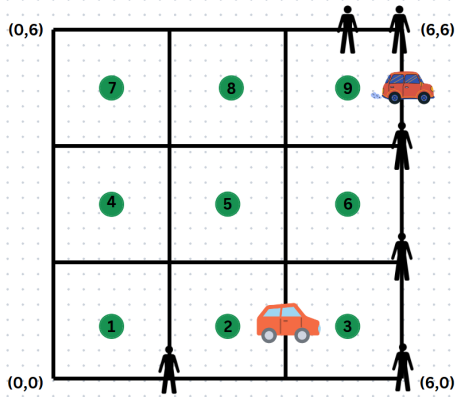
In every iteration of the modified Gale-Shapley process, a matching is conducted between passengers and drivers based on these preferences and available seats. If a stable matching is not yet achieved, the algorithm iterates further, updating the bipartite graph and reducing the set of edges by eliminating unstable or lower-priority pairings.

The algorithm continues to iterate until no further pairings improve stability, ensuring a convergent, stable matching for the ride-sharing system. By allowing for a flexible number of drivers and passengers and the option for passengers to wait if they are unmatched, this approach adapts Gale-Shapley to the complexities of real-world ride-sharing, prioritizing efficient seat allocation and maximizing utility for users.

6 Performance Evaluation

In this study, we designed a 3×3 grid comprising nine equal squares, with edges representing roads, as shown in Fig. 1. Each square has an edge device as a local computational hub for efficient communication and processing. Once a vehicle enters a zone, it notifies the edge device of its presence and the number of available seats, enabling real-time coordination for ride-sharing requests. All passengers and drivers are positioned on the edges, facilitating quick interactions with the corresponding edge device. This setup allows a comprehensive comparison of solo trips versus ride-sharing, with ride-sharing shown to be more cost-effective due to optimized shared expenses.

For the analysis, we examined two scenarios — one with a fixed number of passengers and varying drivers, and the other with a fixed number of drivers and varying passengers, as presented in Fig. 2. Results highlight that ride-sharing significantly reduces costs compared to solo trips by distributing expenses across multiple passengers. Additionally, if a zone lacks an available driver for a ride request, the edge device forwards the request to adjacent zones anticlockwise until a match is found. This dynamic, grid-based edge-device coordination ensures that passengers experience minimal delays even when no drivers are initially in their starting zone.



(a)

	SOURCE	DESTINATION
P1	(6,4)	(0,2)
P2	(6,6)	(2,2)
P3	(5,6)	(0,0)
P4	(6,0)	(2,6)
P5	(2,0)	(0,6)
P6	(6,2)	(0,4)
D1	(6,5)	
D2	(6,2)	

(b)

Fig. 1. Simulation Setup

We visualize the comparative performance of solo and shared trips for both scenarios. The findings underscore the efficiency and cost benefits of the ride-sharing model, primarily as the grid-based system supports rapid matching and minimizes redundant detours by systematically forwarding unmet requests across the grid. With zone-specific edge devices, this grid design demonstrates a scalable and responsive approach to optimizing urban ride-sharing networks.

We take a case of *six* passengers and *two* drivers. The current coordinates of passengers and drivers are given in Fig. 1(b). We evaluate the performance of ride-sharing in terms of cost and time efficiency. The distances of all passengers from the driver are calculated as shown in the figure. For solo riding drivers, D1 and D2 will pick up the nearest passenger and drop them at their location. The initial distances of drivers from passengers are given in Fig. 2(a). The one who drops first will then look on for the nearest passengers near them, and so on. The total time and cost for passengers are 43 seconds and Rs. 159(Considering unit distance on grid costs Rs. 3). In ride sharing, drivers broadcast their location and seat availability to their nearest edge device. Similarly, the ride request of passengers is also taken on the nearest edge device governing that area. If any driver or passenger is located equidistant from two edge devices, the requests are taken to the edge devices anticlockwise.

For example, in the given case, P1 is on (6,4), hence its request is forwarded to edge device 6. If any edge device has a request from a passenger, but there is no driver in the area, then the request is forwarded to the next edge device until the area covered by some edge device has a driver in it. For example, the ride request of P5 is taken on E1, but as there is no driver there, it is forwarded to E2, which

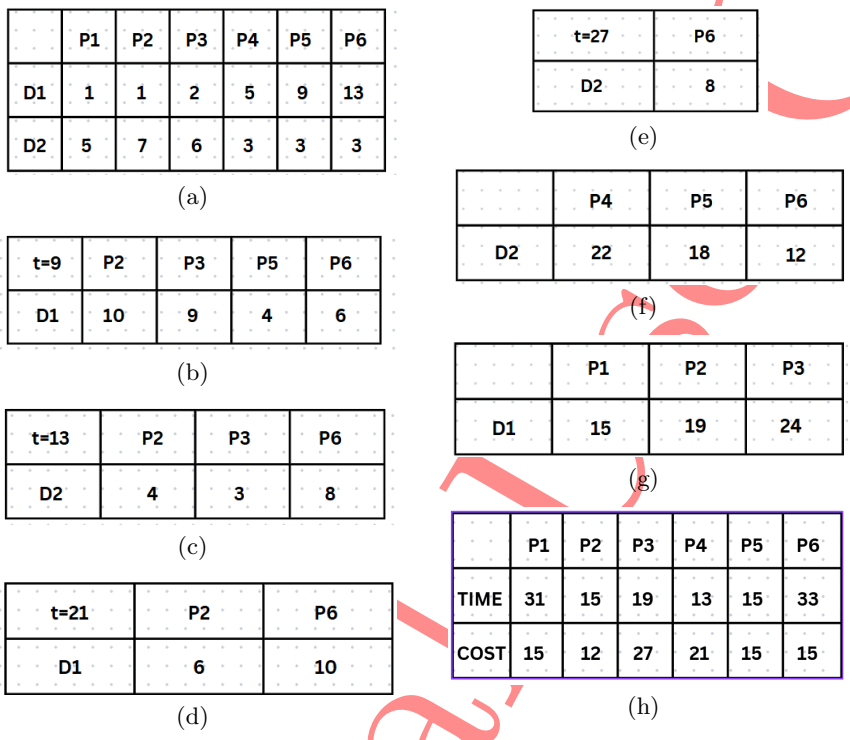


Fig. 2. Evaluation Scenarios

has a driver in that area. After this consideration, D1 received the requests from P1, P2, and P3 as shown in Fig. 2(g). Based on utility functions and applying the Gale-Shapley algorithm (considering cost coefficients as 0.7 and time as 0.3, discount factor as 0.5) on this, D1 will take P3 and P2 with it; similarly, D2 will take P4 and P5. There will be cost reduction for passengers as there are some common subroutes. Fig. 2(h) shows the time at which each passenger is dropped and their effective cost. Finally, after calculations, the cumulative time and cost for passengers are 33 secs and Rs. 107.

7 Conclusion

This study introduces a novel, utility-optimized ride-sharing framework that leverages edge computing for dynamic, responsive matching between drivers and passengers. By integrating enhanced utility functions into a modified Gale-Shapley algorithm, the proposed system adapts traditional stable matching principles to better address the unique challenges of real-world ride-sharing, such as variable passenger-driver ratios and non-strict preferences. Adding edge devices in a distributed grid configuration improves matching efficiency and latency,

enabling faster, localized decision-making and inter-zone coordination. In some cases, we demonstrate that ride-sharing using this edge-computing approach significantly reduces both costs and travel time when compared to solo trips. This system improves users' cost-effectiveness and offers a scalable solution that minimizes resource strain on central cloud servers. As urban mobility demands continue to evolve, the proposed model provides a practical, efficient solution that can enhance the adoption of shared mobility services, benefiting both individual users and the broader transport infrastructure.

Acknowledgment

This research was partially supported by the 6G Flagship (grant number 369116) funded by the Research Council of Finland.

References

1. Essentially stable matchings. Games and Economic Behavior **120**, 370–390 (2020). https://doi.org/https://doi.org/10.1016/j.geb.2020.01.009, https://www.sciencedirect.com/science/article/pii/S0899825620300105
2. Beedham, M.: Here are the world's most congested cities according to the 2021 tomtom traffic index (2022), https://www.tomtom.com/newsroom/explainers-and-insights/the-most-congested-cities-in-2021/
3. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. The American Mathematical Monthly **69**(1), 9–15 (1962), http://www.jstor.org/stable/2312726
4. Guan, L., Pei, J., Liu, X., Zhou, Z., and, P.M.P.: Ridesharing in urban areas: multi-objective optimisation approach for ride-matching and routeing with commuters' dynamic mode choice. International Journal of Production Research **60**(5), 1439–1457 (2022). https://doi.org/10.1080/00207543.2020.1859635
5. Xie, H., Yan, P., Bai, M., Chen, Z.: An efficient parking-sharing program through owner cooperation with robust slot assignment and incentive revenue distribution. Transportation Research Part E: Logistics and Transportation Review **191**, 103697 (2024). https://doi.org/https://doi.org/10.1016/j.tre.2024.103697, https://www.sciencedirect.com/science/article/pii/S1366554524002886
6. Zhu, M., Liu, X.Y., Tang, F., Qiu, M., Shen, R., Shu, W., Wu, M.Y.: Public vehicles for future urban transportation. IEEE Transactions on Intelligent Transportation Systems **17**(12), 3344–3353 (2016). https://doi.org/10.1109/TITS.2016.2543263