Online Decentralised Mechanisms for Dynamic Ridesharing

Nicos Protopapas  
Archimedes/RC Athena  
Athena, Greece  
n.protopapas@athenarc.gr

Enrico H. Gerding  
University of Southampton  
Southampton, United Kingdom  
eg@ecs.soton.ac.uk

Vahid Yazdanpanah  
University of Southampton  
Southampton, United Kingdom  
V.Yazdanpanah@soton.ac.uk

Sebastian Stein  
University of Southampton  
Southampton, United Kingdom  
ss2@ecs.soton.ac.uk

ABSTRACT

Ridesharing services promise an exciting new future for urban mobility. A carefully designed ridesharing system will decrease congestion levels and increase air quality. However, an effective system needs to capture online demand, where users do not schedule their trips in advance, but instead appear and ask for a ride right away. Such online demands require rerouting to be efficient. We call this setting with online demands and available rerouting “dynamic ridesharing” and propose a market-based mechanism where the prospective riders are provided with a menu of choices between several available cars. Our algorithm incentivises users to share their rides and guarantees riders’ utility by properly compensating riders whose routes change during their journey. We provide numerical results, comparing our algorithm against natural benchmarks representing real-world ridesharing services for several cases and with respect to efficiency, fairness, and environmental impact.

KEYWORDS

Ridesharing; Mechanism Design; Dynamic Algorithms

1 INTRODUCTION

Shared mobility services have the potential to revolutionise urban transportation by offering a more convenient and efficient way for people to get around. Efficient implementations of such systems can reduce traffic congestion and carbon emissions [2, 15], providing substantial economic, environmental and societal benefits in large urban areas. As such, it can substantially aid the efficient implementation of Sustainable Development Goals [31]. However, for such a system to truly be successful, it must be able to handle the unique challenges that come with dynamic, on-demand requests from strategic participants. This demanding, yet inevitable flow of requests can put a strain on the system, making it difficult to match riders with available cars in a way that meets their preferences and ensures an acceptable level of compromise [7, 10, 35].

To address the challenges of matching cars and riders in ridesharing, auction-based and mechanism design approaches [6, 11, 20, 21, 27, 28, 39] have been proposed in the literature. However, integrating the vehicle-rider matching side of the dynamic ridesharing problem [1] with the routing side is an open challenge.

**Related Work on On-demand Routing.** Dynamic ridesharing as a concept refers to an automated system matching riders with cars at very short notice [1]. The problem has received much attention from an algorithmic point of view (see, e.g., [3, 5, 19, 22, 37, 38]), but the literature is not so rich when participants’ incentives need to be addressed. A series of papers [13, 18, 27] consider the online allocation of passengers, but with the crucial assumption that all requests arrive before a car starts to service these requests. Asghari and Shahabi [4] propose a system capable of handling online demands for allocating riders to drivers, but merely focused on the incentives of drivers, while ignoring incentives for riders. In contrast, Shen et al. [29] recognise the imminent transition to autonomous cars and present an online mechanism with a focus on riders’ incentives. However, the riders can merely accept or reject a given allocation, without having the option to evaluate various proposals and choose the one that maximises their utility. Incentives for both the rider and the driver are considered in [30, 34], with the catch that a driver serves at most one rider at a time.

Against this background, in this paper, we propose a mechanism to address dynamic re-routing in ridesharing by offering riders a menu of choices presented by multiple cars. Rather than focusing on profit maximization (like, e.g., in [8]), we follow the perspective of a government-designed not-for-profit service which aims to increase ridesharing among riders. Our framework focuses on the challenging task of handling online arrivals, without reliance on any prior knowledge about the arrivals, with new routes computed on-the-fly to accommodate new passengers. Our main contribution is incentivising ridesharing from the rider’s point of view by aggregating their preferences and compensating them for any delay caused by changes to their routes. To the best of our knowledge, we are the first to propose the use of a menu of choices for riders in the ridesharing allocation process. The menu of choices allows

---

1For detailed surveys, the reader is referred to [14, 25].

2A similar approach is also used by [32] under a slightly different context. The main difference with our approach is that their system offers only a single proposal to the users.
riders to express their preferences and priorities, which are then used to optimise the allocation of riders to cars.

We evaluate our proposed mechanism against two natural benchmarks. The first benchmark is a decentralised, peer-to-peer allocation of riders to cars, similar to a taxi service – serving riders one by one on a first-come, first-served basis. The second benchmark is based on discounting, a tactic used by real-world ridesharing services³ where the riders are charged a discounted price for choosing to share a ride with other passengers. We compare the performance of our proposed mechanism against these benchmarks in terms of various metrics, such as the total utility of the riders, fairness between the riders, and the total travelling time. The results of our evaluation show that our proposed mechanism outperforms the benchmarks in most of these metrics; even when it does not, its performance is still close to the benchmarks.

Sustainability. Our work on dynamic ridesharing is aligned with the implementation of Sustainable Development Goals (SDGs) 11, dedicated to sustainable cities and communities, primarily, and to SDG 13, dedicated on climate action, secondarily. Following the approach of [23], we focus mostly on social and environmental aspects of the ridesharing problem: our work provides a starting point for the analysis of ridesharing strategies from the point of view of a central planner, with the goal to promote ridesharing services through a self-sustainable service that is robust to strategic behaviour.

Technical Contributions. Our work introduces a novel, decentralized, market-based mechanism that dynamically allocates riders to shared cars. Unlike other proposals in the literature, our mechanism does not require riders to schedule their journeys in advance, offers the flexibility of dynamic route re-scheduling, and provides riders with a menu of choices that includes estimated travel time and cost. Crucially, we have opted for a model-free approach, and our proposed mechanism does not require any prior knowledge in order to be efficiently implemented. Our approach provides proper incentives for riders to utilize the system, and we have experimentally evaluated it for efficiency, fairness, and environmental impact, using two natural benchmarks for comparison.

The rest of the paper is structured as follows. Section 2 describes the mathematical modelling of the dynamic ridesharing system. Section 3 presents two mechanisms which work as an introduction to the model and as benchmarks for our main mechanism, presented in Section 4 and formally analysed in Section 5. In Section 6, we present an experimental evaluation of our mechanism compared to the mechanisms from Section 3, and we conclude with some final remarks in Section 7. Due to space restrictions, some proofs appear in an extended version of the paper.⁴ The code used in the experiments is given in this repository.⁵

2 SYSTEM MODEL

In this section, we present in detail the mathematical formulation of our model. We introduce formal notation and definitions, and we explain our goals and assumptions.

⁴Available at https://eprints.soton.ac.uk/486964/
⁵https://github.com/NicosProtopapas/DynamicRidesharing.git

Time steps: Let \( \{k\} \) denote the set \( \{1, \ldots, k\} \) for any \( k \in \mathbb{N} \). We assume discrete time-steps \( t \in \mathbb{N} \), for some \( T \in \mathbb{N}_{\geq 0} \). A discrete time-step is the minimum time for which a rider could decide to use a vehicle instead of walking.

City Map: Let the directed graph \( G = (L, E) \) denote a city map. The set \( L \) denotes pick-up and drop-off locations. For simplicity, we assume that any location can be used for the pick-up and drop-off of passengers, but our results work independently of this assumption. The set of edges \( E \) denotes directed links between the locations, and we assume that each location has at least one incoming and one outgoing edge. We assume a highly discretised map where each edge is traversed in a single time-step. This can be done by identifying pick-up and drop-off locations and combining historical data and expert advice. Let \( \delta : L \times L \rightarrow \mathbb{N}_{>0} \) be a function which returns the shortest distance in time-steps between two locations \( a, b \in L \). Using this abstraction, we can ignore unnecessary details, such as the vehicle’s average speed and road congestion, and focus on the travelling time. Formally, \( \delta(a, b) := \min_{p \in P(a, b, G)} |p| \), where \( P(a, b, G) \) denotes all paths connecting \( a \) to \( b \) in \( G \). Without loss of generality, we can assume that \( \delta \) satisfies the triangle inequality, i.e., \( \delta(x, y) \leq \delta(x, z) + \delta(z, y) \) for any locations \( x, y, z \in L \). We assume that everyone has oracle access to this function and that the shortest paths and shortest times returned by this function are accurate.

Riders: Riders are represented by strategic, autonomous agents. Let \( N \) denote the set of all riders. This set is unknown beforehand, and the riders arrive one by one in an unknown order. Each rider is described by her type, which is a tuple consisting of both private and public information. The private information includes (i) the arrival time \( t_i \in [T] \) and (ii) the value of time (VoT), denoted by \( v_i \in \mathbb{R}_{\geq 0} \). This value denotes a patience weight for the rider: a VoT of \( x \) implies that the rider suffers a cost of \( x \cdot y \) for travelling \( y \) more time steps, compared to the optimal option. The type also includes a publicly known pair \((a_i, b_i) \in L \times L\), denoting departure and arrival locations for the rider. We assume that the riders always truthfully report their preferred pick-up and drop-off locations \( a_i \) and \( b_i \). For the pick-up location, this can be ensured when the system refuses to accept passengers appearing in other locations than the reported. Similarly, the system can ensure truthful reporting of the destination \( b_i \), by enforcing the rider to disembark at the location they report.⁶

The variable \( s_i \) denotes the status of a rider. Each rider first appears as unallocated, it is then allocated to a vehicle, and once picked-up becomes a travelling rider. A rider who reaches her destination is a finished rider.

Vehicles and Routes: Let \( M \) denote the set of all cars in the system. Vehicles start their trips from designated locations in the city, which we call car depots and denote with the set \( D \subseteq L \). A live vehicle is one with allocated riders, and is either (i) already carrying one or more riders or (ii) empty, but moving to pick up a rider. A vehicle returning empty to the depot is not considered live, but can accept new requests without the need to return. We use \( m_t \) to denote the number of the live vehicles at time \( t \), and \( M_t \) to denote the set of these vehicles. Each vehicle may have a professional driver or it is

⁶It could be possible for a rider to report some locations \( a'_i \neq a_i \) and \( b'_i \neq b_i \), and then use other means to get to their final destination. Here, we consider \( a_i \) and \( b_i \) to mean the source and destination of rider \( i \) when using the ridesharing service.
autonomous. In both cases, our main assumption is that cars are not strategic. Vehicle \( j \in M \) is described by a set of variables: \( C_j \) denotes the vehicle’s capacity, which is a constant, small integer number. Variable \( f_j \in L \) denotes the car’s location at time \( t \), and \( c_j \) is a scalar value denoting the cost for travelling a unit of time.

The set \( R^A_j \) denotes the allocated riders in car \( j \) at time \( t \), while the set \( R^F_{j,t} \subseteq R^A_j \) denotes the set of travelling riders.

The tuple \( r \) denotes an ordering of visiting riders’ pick-up and drop-off locations. Given a set of riders \( R \), let \( L(R) := \{a_i, b_i\}_{i\in R} \). An order \( r \) is defined as a sequence of pick-up and drop-off locations of the riders in \( R \). We use the notation \( r_i \in L(R) \) to denote the location in the \( i \)-th element of the order \( r \), and \( r^{-1}_R \in |r| \) to denote the position of location \( x \in L(R) \) in the same order.

For an order to be valid, it must be that (i) \( b_i \in r \) for all \( i \in R \), and (ii) \( a_{i+1} \) < \( b_i \). The second condition is always met when \( a_i \notin L(R) \). As an example, consider the set of riders \( R = \{x, y, z\} \).

Then \( (a_x, b_x, a_y, b_y, a_z, b_z) \) are valid orders, while, \( (a_x, b_y, a_z, b_x, a_y, b_z) \) and \( (b_z, b_x) \) are not valid orders for \( R \).

Given a vehicle \( j \in M \), a state \( t \in T \), and a valid order \( r \) over the set \( R^A_j \), let \( r^{j,t}_k := e_{j,t}, r^{j,t}_k = r_k \) for \( k \in |r| \) and \( r^{F}_{j,t+1} \in D \). Then the route \( (r^{j,t}, \theta^j_t) \) is defined as a concatenated path, created by the shortest paths between the locations \( r^{j,t}_k \) and \( r^{j,t}_{k+1} \), for \( k \in \{0, ..., |r|\} \). For simplicity, we use the simplified notation \( r^{j,t} \) to denote a route.

Let \( \tau_i := \delta(a_i, b_i) \) be the ideal route cost in time for rider \( i \), i.e., the best case scenario for rider \( i \), e.g., using a private ride. Given a route \( r^{j,t} \), let \( \tau(r^{j,t}) := \sum_{k=1}^{r^{j,t} - |r^{j,t}|} \delta(r_k, r_{k+1}) \) be the total travelling time for the route to complete, without measuring the time for a car to return to a depot. Let \( \tau_i(r^{j,t}) := \sum_{k=1}^{r^{j,t} - |r^{j,t}|} \delta(r_k, r_{k+1}) \) be the total travelling time of rider \( i \), including time spent embarked and the waiting time, starting at \( t_i \). Similarly, we define \( \phi^j_t := \delta(t_i, r_1) + \tau(r^{j,t}) + t_i \) as the finishing time of rider \( i \), and \( \phi^j_t := t^*_i + t_i \) as the ideal finishing time for rider \( i \).

**Preferences:** Now that we have introduced the notions of routes and cars, we discuss the preference model of the riders. The riders have quasi-linear preferences over their travelling time. The riders are utility maximisers, and the utility of the rider diminishes as the sum of travelling time and the price increases. Formally, given a route \( p_i \) and a car-dependent price \( p^i_j \), the utility for rider \( i \) at time \( t \) is defined as: \( u_i(r^{j,t}; p_i^j) := -v_i \cdot (\phi^j_t - \phi_i^j) - p_i^j \), where the scalar value \( v_i \in \mathbb{R}_{\geq 0} \) denotes the sensitivity of rider \( i \) to deviations from the optimal route from its own perspective.

In the following, we compare the utility of a rider according to two different points of view. The projected utility refers to the utility of a rider at the allocation time, before the rider actually travels. In contrast, the ex-post utility refers to the utility of the rider after she completes her journey. Hence, the former is an estimation of the latter. A rider is called myopic if she cares only about maximising her utility at the time of the arrival without considering future arrivals of other riders.

**Mechanisms:** A mechanism works in a two-step procedure: First, it receives reports from the riders on their (privately known) types, e.g., using a mobile app. Then, it produces multiple proposals, one for a subset of all available cars. More concretely, a mechanism \( F \) is defined using an algorithm and payment component. Upon the arrival of a rider \( i \), each car \( j \in M' \subseteq M \) proposes an expansion of its current route to accommodate the new rider, using an allocation policy \( a^j \). The proposed route is accompanied by a proposed payment, calculated according to a payment policy \( \pi^j \). Then, each rider will accept one of these proposals. Importantly, the allocated car cannot change, and the allocation decision must be made immediately. The selected route and payment might change due to future arrivals, however.

Formally, at time \( t \), and upon the appearance of rider \( i \), each car \( j \in M' \) proposes a route \( r^{j,t} = a^j(\theta_{i,j}; r^{j,t}) \) and a respective payment \( p^j_t = \pi^j(\theta_{i,j}; r^{j,t}) \). Then the rider selects a new route and price pair such that: \((z^{j,t}, q^{j,t}) \in \arg \max_{j \in M'} u_i(a^j(\theta_{i,j}; r^{j,t}), \pi^j(\theta_{i,j}; r^{j,t})) \).

The pair \((z^{j,t}, q^{j,t}) \) is the outcome of mechanism \( F(\theta; t) \) at time \( t \). The final outcome is \( F(\theta) \).

**Properties:** To present desirable properties of the mechanisms, we recall formal definitions rooted in the literature on mechanism design [26]. The first property follows the classic notion of incentive-compatibility.

**Definition 2.1.** A mechanism \( F \) is Dominant Strategy Incentive Compatible (DSIC) if \( u_i(F(\theta_i, \theta_{-i}; t)) \geq u_i(F(\theta'_i, \theta_{-i}; t)) \) for every rider \( i \in N \), and rider’s \( i \) types \( \theta_i, \theta'_i \) and any profile of other riders’ types \( \theta_{-i} \).

A weaker notion of incentive compatibility is concerned only with myopic agents. Under this notion, any rider cannot gain by any misreport, if no other rider is allocated to the same car in the future.

**Definition 2.2.** A mechanism \( F \) is Dominant Strategy Incentive Compatible for myopic riders if \( u_i(F(\theta_i, \theta_{-i}; t)) \geq u_i(F(\theta'_i, \theta_{-i}; t)) \) for every rider \( i \in N \), and rider’s \( i \) types \( \theta_i, \theta'_i \) and arrival time \( t_i \), and any profile of other riders’ types \( \theta'_{-i} \), which have arrived before \( i \).

Finally, we need to know whether our mechanisms generate enough income to cover the operation costs of the rides. To be fair to the riders, we only care about the cost a car suffers while it is allocated at least one rider.

**Definition 2.3.** A mechanism \( F \) is Budget-Balanced if \( \sum_{i \in N} p_i \geq \sum_{j \in M} c_j \cdot \chi_j \) where \( p_i \) is the ex-post payment of rider \( i \) and \( \chi_j \) is the total travelling time of vehicle \( j \), while allocated with riders, once all riders are delivered.

3 DECENTRALISED MECHANISMS FOR DYNAMIC RIDESHARING

We will explain the functionality of our proposed family of mechanisms with a few simple examples. We first consider a case with only three riders, Alice, Bob, and Carlos, and a single depot. Alice asks for a ride at time \( t_A \), to get from her current location \( a_A \) to location \( b_A \). When Alice requests a ride, only a single car is live: Car 1, with a maximum capacity for 4 passengers, and already allocated with Bob and Carlos. Bob is already on-board, while the car is

---

The set \( M' \) is a subset of all available cars which precludes cars that are not valid options, e.g., because the distance between the car and the rider’s pick-up location is too large.
moving to pick up Carlos from location $a_C$. To be more precise, at the time-step before Alice’s request, the car is following the route $r^{1/a_C} = (t_{1/a_C}, a_C, b_C, b_B)$; Carlos will be picked-up and delivered, and finally Bob will arrive to his destination. Another identical car is available empty at the depot. Based on this instance, we will illustrate two simple mechanisms. We will use these mechanisms in Section 6 as benchmarks for our main contribution.

**Fixed Pricing FIFO:** Probably the simplest solution to allocate riders in a car is a simple first-in first-out strategy, where each car adds the new rider’s pick-up and drop-off locations at the end of their route, upon their arrival on the system, and the rider selects the best available car.\(^6\) The rider is charged exactly the cost of the ride. We call this the FIFO mechanism, and the car proposal procedure is shown in Algorithm 1.

We will explain the FIFO mechanism using the aforementioned scenario. Once Alice arrives, Car 1, which is travelling towards location $a_C$, proposes the route $r^{1/a_C} = (t_{1/a_C}, a_C, b_C, b_B, a_B)$, hence scheduling Alice behind Carlos. Alice should pay $p^{1/a_C} = c_1 \cdot (\delta(t_{1/a_C}, a_A) + \delta(a_A, b_A))$. The car from the depot will propose the route $r^{2/a_C} = (t_{2/a_C}, a_A, b_A)$, at a price $p^{2/a_C} = c_2 \cdot (\delta(t_{2/a_C}, a_A) + \delta(a_A, b_A))$. Alice then selects the route that maximises her utility.

Despite its simplicity, the Fixed Pricing FIFO satisfies some important properties. The mechanism is truthful, budget-balanced and trivially scalable. The truthfulness of the mechanism is straightforward to establish. The mechanism cares only about the starting time of the rider; hence, the only possibility of misreporting from the rider is to propose a fake arrival time in the future, but this makes little sense to the rider: this might allow future riders to be scheduled before her. On the contrary, by simply proposing her true arrival time, she ensures she is served first. The mechanism is also trivially budget-balanced, as each rider pays exactly the cost of the ride. It is also computationally attractive. A car only needs to compute two shortest paths, to propose a route and price. On the downside, this mechanism does not promote ridesharing at all.

**Discount mechanism.** A description of the proposal phase is given in Algorithms 2 and 4.

Following the previous example, when Alice arrives, car 1 proposes a route which minimises the travelling time of the car, e.g., $r^{1/a_C} = (t_{1/a_C}, a_C, b_C, b_B, a_B)$. The payment for Alice to use car 1 is now $p^{1/a_C} = (1 - \alpha) \cdot c_1 \cdot \tau(r^{1/a_C})$. The car from the depot will propose the route $r^{2/a_C} = (t_{2/a_C}, a_A, b_A)$, at a price $p^{2/a_C} = c_2 \cdot (\delta(t_{2/a_C}, a_A) + \delta(a_A, b_A))$. Hence Alice may have an incentive to use car 1, even if this car would increase her finishing time.

It is not hard to see that this mechanism is not truthful. Indeed, a rider who knows that a new rider with a similar itinerary will arrive shortly can delay her arrival to get a trip at a discounted price. However, as we will see in Section 6, this mechanism behaves quite well with respect to total operational time.

### 4 DYNAMIC MARKET-BASED ALLOCATION

This section proposes our main mechanism for allocating a rider in a car, which we call the Compensation mechanism. The main procedure works roughly as follows: when a rider appears in the system, it requests a ride and provides her origin $a_i$, destination $b_i$, arrival time $t_i$ and VoT $v_i$ to the system. Then, a group of selected cars are notified about the presence of the new rider, and each car proposes a route to accommodate the request, accompanied by a payment. Notably, the new route should include the already allocated riders’ destinations and, for riders not ready picked up, the origins. The payment should allow for compensating any already allocated passenger so that the utility of the already allocated riders does not decrease, and the payment should cover any additional

---

\(^6\)This is a common approach in the social choice literature [9] and, in particular, is similar to the well-known Random Priority mechanism.

---

**Algorithm 1** Vehicles’ proposal for FIFO.

1. **input:** rider $i$, current route $r$ at time $t$, set of current riders $R_j$. 
2. **output:** new route $w$, payment $p^{i,j}$.
3. **procedure** `ComputeProposal(i, j, t)`
   4. $w \leftarrow (r, a_i, b_i)$  \quad $\triangleright$ append the new rider to the end
   5. $p^{i,j} \leftarrow c_i \cdot (\delta(t_{i,t}, w_i) + \tau(w))$
   6. **end procedure**

**Algorithm 2** Vehicles’ proposal for Discount.

1. **input:** rider $i$, current route $r$ at time $t$, set of current riders $R_j$, discount rate $\alpha$.
2. **output:** new route $w$, payment $p^{i,j}$.
3. **procedure** `ComputeProposal(i, j, t)`
   4. for $r \in RouteGenerator(i, j, r)$ do
   5. $s^r \leftarrow \tau(\hat{r})$  \quad $\triangleright$ compute total travelling time
   6. **end for**
   7. $w \leftarrow \arg\max (s^r)$
   8. $p \leftarrow c_i \cdot (1 - \alpha) \cdot (\tau(w) + t)$
   9. **end procedure**

**Figure 1:** Example of route proposals from Algorithm 3.
Whenever the new route increases the finishing times of the already allocated riders, the new rider should pay compensation to each of them. The new rider should cover any extra travelling costs deriving from the re-routing. We present the following example to build intuition on how the routes are proposed.

Consider the simple case with two riders who need to move on a grid. The grid is served by two car depots located at locations (0, 0) and (10, 10). Adjacent locations in the grid are connected with vertical and horizontal roads, traversable by a car in a single unit of time. All cars have identical costs and capacities, and all riders have the same VoT.

A rider appears at time $t = 0$ in location (2, 2) and wants to move to location (5, 7). The rider will then receive two proposals: A car from the first depot would propose the route (0, 0) → (2, 2) → (5, 7), for 12 pounds, and a projected finishing time at 12. A car from the second depot will similarly propose to the rider the route (10, 10) → (2, 2) → (5, 7) for 24 pounds. The ideal route length for the rider is $r_1^* = 8$; at time 0, no car is moving, and the only available options for the rider are cars not in use from the two depots. This implies that rider 1 has a utility of $- (12 - 8) = - 16$ for the first proposal, and a utility of $- (24 - 8) = - 16$ for the second proposal. Hence, the first car is selected and starts its trip, with the first stop at (2, 2).

At time $t = 5$, a new rider, say rider 2, appears at location (6, 5) and wants to travel to location (2, 6), with an ideal cost of 5 units of time. The first car has now arrived in location (3, 2), with rider 1 embarked (at time 4, from location (2, 2)). The travelling car now has three available options to propose to the new rider. This situation is depicted in Figure 1.

**Option 1**: the path (3, 2) → (6, 5) → (2, 6) → (5, 7). This route firstly serves rider 2, and then drops-off rider 1. The new route will conclude at time $t = 20$, while the car’s current route is expected to finish at time $t = 12$. Rider 2 needs to cover this difference, and must also compensate rider 1 for all the extra time it needs to travel. Indeed, rider 1 is projected to finish at time $t = 20$ while the previous route had proposed a finish for rider 1 at time $t = 12$. Hence, rider 2 should pay a price of $8 + 8 = 16$ pounds, and since rider 2 needs 10 units of time to reach its destination, its projected utility is $− (10 + 5) = (8 + 8) = - 21$.

**Option 2**: The path (3, 2) → (6, 5) → (5, 7) → (2, 6). This route picks up rider 2, then moves to drop off rider 1, then drops rider 2. This route finishes at time $t = 18$, and rider 1 needs to travel for 2 more units of time than the current route. Thus rider’s 2 payment is $6 + 2 = 4$ pounds. Rider 2 gets a projected finishing time of 13 units of time from its appearance. This implies a projected utility of $− (13 + 5) = (6 + 2) = - 16$.

**Option 3**: Finally, the path (3, 2) → (5, 7) → (6, 5) → (2, 6). In this route, rider 1 is first dropped off, then the second rider is served. Rider 2 pays 8 pounds in total for the part (5, 7) → (6, 5) → (2, 6) and no compensation is needed. Under this route, rider 2 finishes 15 units of time after its appearance, which implies a utility of -18.

The mechanism proposes the second route to rider 2, as it is the best route from rider’s 2 perspective (shown with a solid line in Figure 1). Alongside this proposal, rider 2 also gets proposals from the cars parked in the two depots: one passing through (0, 0) → (6, 5) → (2, 6), with a utility of -27, and one using the path (10, 10) → (6, 5) → (2, 6) for a utility of -21. Hence, the rider will eventually choose to travel alongside rider 1.

## 5 Formal Properties and Analysis

In this section, we analyse the performance of the Compensation mechanism. The first property formally shows the main idea of the mechanism: that each rider is guaranteed not to lose on their utility because of the other riders.

**Theorem 5.1.** For the Compensation mechanism, the ex-post utility is always higher than the projected utility for all riders.

**Proof.** First, observe that the utility of rider $i$ can change (compared to the (promised) projected utility by the mechanism) only when a new rider is allocated in the car after rider’s $i$ allocation. For the sake of contradiction, assume that some rider $k$ exists for which the utility of rider’s $i$ decreases when $k$ is allocated to the same car, say $j$, at time $t$. Let $\phi_i$ and $p_j$ denote the finishing time and the proposed payment for the already allocated rider $i$ at time $t$, before the allocation of rider $k$. Let also $\phi'_i$ denote the finishing time for rider $i$, after the allocation of rider $k$. Furthermore, let $R$ be the set of allocated riders in car $j$, allocated after rider $i$, but before rider $k$. Hence, the difference in the rider’s $i$ utility, after and before the allocation of $k$ equals to:

$$- u_i \cdot (\phi'_i - \tau_i^*) - p_i + \sum_{l \in R \cup \{k\}} h_l = \left( - u_i \cdot (\phi_i - \tau_i^*) - p_i + \sum_{l \in R} h_l \right) = u_i \cdot (\phi_i - \phi'_i) + h_k.$$ 

When $\phi_i - \phi'_i < 0$, observe that $h_k$ is exactly equal to $u_i \cdot (\phi_i - \phi'_i)$, hence the theorem holds. When $\phi_i - \phi'_i \geq 0$, i.e., when rider $i$ finishes earlier, the theorem holds trivially since the compensation $h_k$ received is never negative.

In the next theorem, we show that using the Compensation mechanism, a myopic new rider has no incentive to manipulate the mechanism.

**Theorem 5.2.** The Compensation mechanism is DSIC for myopic riders.
Figure 2: Comparison of the Compensation (solid, blue), FIFO (dashed, orange) and Discount(dotted, green) mechanisms. Simulations are performed a 50 × 50 grid, with average time between arrivals $\lambda = 5$, VoT between $B_I = 1$ and $B_U = 10$, and at most 10 identical cars departing from 2 depots. Results are averaged over 20 iterations showing 95% confidence intervals.

The full proof of the theorem is presented in the extended version of the paper. Here, we briefly explain the high-level intuition. The main idea is to associate any route proposed by a car when a rider acts non-truthfully with a different route, grounded on the riders’ true valuations. This new route either provides a better utility for the rider or generates the proper compensation to make this new route a better choice.

Note also that this mechanism is no longer truthful in the ex-post sense. This can be proven using the following counter-example:

**Example 5.3.** Consider the line graph $(0, 1, ..., k)$, with unit cost on the edges, a single depot at 0 and one available car at the depot. Rider $i$, who needs to get from location 1 to $k$, knows that another rider, namely $i'$, will appear at time $t_i = t_i' + 1$, at location 2, and also needs to go to the same destination, $k$. When rider $i$ reports truthfully, she will be allocated in the single available car, with the route $r_i = (0, 1, k)$. Since this is the direct route, the utility of rider $i$ is equal to $-(k + 1)$, i.e., the cost of the ride. This is, in fact, the ex-post utility of rider $i$. When rider $i'$ appears, the car will propose the route $r_{i'} = (0, 2, k)$. Note that the total payment for rider $i'$ is 0 as the route cost is already covered by rider $i$, and no compensation is needed for rider $i$ since the rider does not lose utility with the new ride.

When rider $i$ reports untruthfully $t_i' = t_i + 1$, the car will first assign rider $i'$ with the route $r_{i'} = (0, 2, k + 1)$. In the following time-step, when rider $i'$ appears, then route will change to $r_i' = (3, 1, k)$. This route needs $k + 2$ time-steps to finish, 2 time-steps longer than the previous route. The rider also incurs an extra cost of $2 \cdot \delta_i$ to rider $i$, for which she should compensate her. Hence the payment for rider $i$ becomes $2 \cdot (1 + \delta_i)$. As a consequence, the utility is now $-2 \cdot \delta_i - 2 \cdot (1 + \delta_i)$, which is at least equal to $-(k + 1)$ with specific values for $k$, $\delta_i$ and $\delta_i$. This holds for $\delta_i = \delta_i = 1$ and any $k > 6$.

We also note that the Compensation mechanism is Budget-Balanced, i.e., the travelling cost of the riders is always covered by the payments.

**Algorithm 4 Route Generator.**

**Input:** potential rider $i$, current route $r$ at time $t$, set of current riders $B_r$. **Output:** Set of new routes $P$

1. **procedure** RouteGenerator($i, j, r$)  
2. \hspace{1cm} $t = |r| + 2$ \hspace{1cm} // set the length of the new routes  
3. \hspace{1cm} for $k \in \{0, ..., t\}$ do  
4. \hspace{2cm} for $l \in \{0, ..., t\}$ do  
5. \hspace{3cm} $r_0 = r_0$ \hspace{2cm} // set the car location  
6. \hspace{3cm} $r_{1, ..., k-1} = r_{1, ..., k-1}$  
7. \hspace{3cm} $r_{k} = a_l$ \hspace{2cm} // Add pick-up location  
8. \hspace{3cm} $r_{k+1, ..., l-1} = r_{k, ..., l-1}$  
9. \hspace{3cm} $r_l = b_l$ \hspace{2cm} // Add drop-off location  
10. \hspace{3cm} $P = r_{1, ..., l}$  
11. \hspace{2cm} end for  
12. \hspace{1cm} end for  
13. end procedure
The Compensation mechanism is Budget-Balanced.

Proof. This is trivially derived by line 8 in Algorithm 3, which states that any new rider will cover the travelling cost of the car for the whole route. Note that the computed compensations are not needed to cover the travelling cost. □

Finally, we discuss the computational traceability of our proposal. The most crucial part of the algorithm is the RouteGenerator procedure, which computes valid routes for the car to accommodate a new rider. The number of these routes is quite large, exponential to the number of allocated riders, in general. To cope with this, we use a restricted class of routes, which respects the order of the previous riders, generated as follows: given a previous order of size \( k \), the pick-up location index is \( i \in \{1, k + 1\} \) in the new order. Then the drop-off locations is \( j \in \{2, k + 2\} \setminus \{i\} \). The remaining \( k \) slots are filled according to the previous order. This leaves us with substantially fewer possible orders to explore, squared in the number of allocated riders. The pseudocode for the described implementation is presented in Algorithm 4. An interesting open problem is to quantify how this restriction affects the efficiency of the algorithm and to review our mechanism with more sophisticated route recommendation methods (see, e.g., [36]).

6 EXPERIMENTAL EVALUATION
This section evaluates the performance of our mechanism using a series of simulations.

6.1 Experimental Setting
In our experiments we use a simple \( k \)-grid graph where each node \((i, j) \in [k]^2\) is connected to its (at most four) neighbours \((i \pm 1, j)\), \((i, j \pm 1)\), \((i \pm 1, j)\), \((i, j \pm 1)\), whenever \(i \pm 1\) and \(j \pm 1\). Each edge is traversed in a single time step. The graph is created using the NetworkX Python library [16], and shortest paths are computed using the well-known A* algorithm [17], using the library’s native functions. In each experiment, we use a fixed number of depots whose locations in each iteration are drawn uniformly over the graph without replacements. Whenever a new rider requests a ride, it receives quotes from all live vehicles (vehicles moving in the grid) and one vehicle from each depot.

The time between arrivals is simulated by a Poisson distribution with mean \( \lambda \). The riders’ VoT is uniformly distributed in the interval \([B_L, B_U]\), \(B_U \geq B_L \geq 0\). Finally, riders’ origin and destinations are uniformly and independently distributed. The cost of the Compensation and FIFO mechanisms routes is equal to their length, multiplied by the car cost per unit of time. For the Discount mechanism, the payment for a new passenger is discounted by 20% when a rider is allocated in the car with 1 other allocated riders on board, and 40% when a rider is allocated in a car with 2 passengers and more.

We evaluate our mechanisms according to various criteria. First, in terms of efficiency, we compare our mechanism according to the (mean) ex-post utility, the utility of the riders when they finish their ride, and the (mean) projected utility, the utility the riders are “offered”, at the time of allocation. In terms of fairness, we evaluate

\[ \lambda \text{mean of } \frac{1}{\text{number of riders scheduled per time unit}} \]

\[ \lambda \text{mean of } \frac{1}{\text{number of riders scheduled per time unit}} \]

\[ \lambda \text{mean of } \frac{1}{\text{number of riders scheduled per time unit}} \]

6.2 Results and Analysis
This section briefly analyses our experiments regarding efficiency, fairness and environmental impact.

Efficiency: Apart from guaranteeing the riders their projected utility at the allocation, the Compensation mechanism yields approximately 14% higher ex-post utility compared to the second-best choice, the FIFO strategy in the first experiment. This margin increases to approximately 50% in the second experiment. In contrast, the Discount strategy promises the highest utility at allocation time. Still, it fails to deliver on these promises at the end of the journey. Observe here that the identical graphs for the FIFO mechanism in Figures 2a and 2b (and in 3a and 3b) are not coincident since this mechanism yields the same utility at allocation and after the end of the journey. The Compensation and FIFO mechanisms perform best in commute time, approximately 20% better than the Discount policy in the first experiment and 40% better in the second, where the Compensation mechanism performs slightly better. This illustrates that the good performance of the Compensation mechanism in terms of utility does not happen because a large amount of compensation is received, but it occurs organically.

Fairness: Regarding fairness, the Compensation and FIFO mechanisms also cope quite well in all experiments. The Compensation mechanism performs slightly better in both experiments on the
rather pessimistic criterion of minimum ex-post utility. This trend also continues with respect to the Gini Index.

**Environmental Impact:** Our final measure compares the mechanisms for online ridesharing, focusing on the riders’ incentives. Our model focuses on riders with on-demand requests, in the sense that they can appear at unknown times and request a ride that should be served as soon as possible. Our model is decentralised in the sense that riders receive quotes from available cars, and they choose one of them to complete their trip. Our main contributions are the Compensation mechanism and the approach to presenting a set of options to riders. The Compensation mechanism is truthful for myopic riders, budget-balanced and outperforms the two other methods in most experiments, and it is never critically overrun by any of the benchmarks. An interesting open problem regarding our approach is whether meaningful truthful mechanisms for non-myopic riders exist in the model-free setting. We believe that only mechanisms similar to FIFO can be truthful under this demanding environment, and we aim to show this in future work. We also note that to avoid concurrency issues, we have assumed that every single rider appears at each time step, which is, of course, non-realistic. To safely drop this assumption, our work could be enhanced by being analysed from the perspective of distributed mechanism design (see, e.g., [12]).

**ACKNOWLEDGMENTS**

The authors acknowledge the use of the IRISDIS High Performance Computing Facility and associated support services at the University of Southampton in completing this work. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through a Turing AI Fellowship (EP/ V022607/1) on Citizen-Centric AI Systems (https://ccais.ac.uk) and the platform grant entitled “AutoTrust: Designing a Human-Centered Trusted, Secure, Intelligent and Usable Internet of Vehicles” (EP/R029563/1). This work has been partially supported by project MS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

---

**Figure 3:** Comparison of the Compensation (solid, blue), FIFO (dashed, orange), and Discount(dotted, green) mechanisms. Simulations are performed on a 50 x 50 grid, with average time between arrivals $\lambda = 2$, VoT between $B_L = 1$ and $B_U = 20$, and at most 20 identical vehicles departing from 2 depots. Results are averaged over 20 iterations, showing 95% confidence intervals.
ETHICAL STATEMENT

This work facilitates the ethical use of AI in smart mobility by promoting diversity, fairness, and flexibility. Our proposal for dynamic ridesharing caters to the preferences of users by allowing a diverse range of individuals with varying needs to choose from ridesharing options and prices, as opposed to imposing a single fixed price by service providers. Furthermore, our approach provides flexibility for users to join the service dynamically, allowing them to join during its operational hours. We also advocate for accessibility and equity in mobility services by introducing a fair and manipulation-resistant method for efficient ridesharing. This method aims to optimise total travel time, thereby supporting the sustainability of mobility systems and efforts to mitigate climate change.

REFERENCES


