A Polynomial-time, Truthful, Individually Rational and Budget Balanced Ridesharing Mechanism

Tatsuya Iwase[1] , **Sebastian Stein**[2] , **Enrico H. Gerding**[2]

[1]Toyota Motor Europe NV/SA, Zaventem, Belgium
[2]University of Southampton, Southampton, United Kingdom
tiwase@mosk.tytlabs.co.jp, {ss2, eg}@ecs.soton.ac.uk

Abstract

Ridesharing has great potential to improve transportation efficiency while reducing congestion and pollution. To realize this potential, mechanisms are needed that allocate vehicles optimally and provide the right incentives to riders. However, many existing approaches consider restricted settings (e.g., only one rider per vehicle or a common origin for all riders). Moreover, naive applications of standard approaches, such as the Vickrey-Clarke-Groves or greedy mechanisms, cannot achieve a polynomial-time, truthful, individually rational and budget balanced mechanism. To address this, we formulate a general ridesharing problem and apply mechanism design to develop a novel mechanism which satisfies all four properties and whose social cost is within 8.6% of the optimal on average.

1 Introduction

Ridesharing can meet more traffic demand with fewer vehicles than existing taxi services or privately-owned vehicles. At the same time, it can reduce traffic congestion, energy consumption and pollution. Moreover, with the introduction of autonomous vehicles, unused vehicles become active transportation resources, and ridesharing can further increase utilization of such vehicles [Spieser *et al.*, 2014].

Broadly, there are two research streams addressing ridesharing problems: the first one focuses on algorithms for assigning vehicles to riders while solving the routing problem in an optimal way [Cordeau and Laporte, 2007; Hasan *et al.*, 2018; Chen *et al.*, 2019]. However, none of these studies consider incentives and whether it is in the best interest of riders to follow the proposed solution and to report their preferences truthfully. Especially when the origins and destinations of the riders are different, they are sometimes required to detour to share the vehicle with others. Hence, there may be an incentive to misreport time constraints and preferences to prevent detours.

To address this, the second research stream focuses on mechanism design approaches, which incentivize truthful reporting and avoid strategic manipulation. In addition to this

property, which is called incentive compatibility (IC), other desirable properties include (weak) budget balance (BB, i.e., the system does not require an outside subsidy), individual rationality (IR, i.e., an individual is never worse off participating in the mechanism) and scalability (i.e., the allocation and any payments can be computed in polynomial time). An early approach uses a second price auction [Kleiner *et al.*, 2011], but their work is limited to single riders per vehicle and does not allow cooperation between drivers. [Shen *et al.*, 2016] propose an online posted-price mechanism but they only consider autonomous vehicles (vehicles can move without any riders allocated) and their approach is not scalable. Furthermore, [Cheng *et al.*, 2014] propose a mechanism that satisfies IC, BB and IR, but they assume all riders depart from the same origin and so their method does not coordinate the pickup of riders at different locations. [Zhao *et al.*, 2014] propose a Vickrey–Clarke–Groves (VCG) mechanism, but their approach is not scalable and BB is only satisfied when there is no detour. More recent work on mechanism design, e.g., [Rheingans-yoo *et al.*, 2019; Lu *et al.*, 2018; Ma *et al.*, 2019], only focuses on drivers and does not consider rider incentives. Also, none of the above consider settings where riders can switch vehicles which can enable more efficient solutions, especially in settings with limited vehicles as in suburban areas.

In short, to date there exist no scalable mechanisms for general ridesharing settings that satisfy IC, IR and BB. To address this shortcoming, we make the following novel contributions:

- We formulate a more general ridesharing problem compared to existing approaches. In particular, this is the first model that considers both autonomous and non-autonomous vehicles and where riders can switch between vehicles if it is beneficial to do so.

- We prove that, unsurprisingly, computing the optimal allocation is NP-hard. Nevertheless, by introducing some weak assumptions, we show that it can be to reduced to a polynomial-time linear program.

- We show that approaches based on VCG mechanisms or naive greedy (where riders are allocated in order of their marginal cost) do not satisfy the necessary properties. Specifically, the standard VCG approach does not satisfy BB and a modified, budgeted balanced, version can-

---

An extended version including technical appendices is available at https://arxiv.org/abs/2105.11292.

not be computed efficiently using our novel polynomial-time reduction. Also, naive greedy violates a property called monotonicity, meaning it fails to satisfy IC.

- To address this, we propose a general methodology to design monotone greedy allocation mechanisms. We then apply this theory to produce a novel two-stage mechanism with carefully designed payments to ensure both IR and BB, as well as IC and scalability.

- We empirically evaluate the performance of different mechanisms and show the advantages of our methods. Our monotone greedy mechanism satisfies all desirable properties with typically less than a 9% social cost increase compared to the optimal allocation.

In what follows, Section 2 introduces the ridesharing model and its mechanism design formulation. Section 3 considers the VCG mechanism and its budget balanced version. Section 4 studies the greedy mechanisms. Section 5 provides an empirical evaluation of the mechanisms and Section 6 concludes. Due to space limitations, most proofs for the theoretical results are provided in the extended version of the paper.

2 The Model

In this section, we first provide a formal model of the ridesharing problem followed by an example. We then formulate this as a mechanism design problem.

2.1 The Ridesharing (RS) Problem

We assume finite time steps $\mathcal{T} = \{0, 1, \ldots, T\}$, set of riders $\mathcal{N} = \{1, 2, \ldots, N\}$ and vehicles $\mathcal{K} = \{1, 2, \ldots, K\}$. Let $w \in \mathbb{N}_+$ denote the vehicle capacity (i.e., the maximum number of riders in a vehicle). The road network is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where edges $\mathcal{E}$ denote road segments, vertices $\mathcal{V}$ denote locations, and loops $(v, v) \in \mathcal{E}$ are used to model stationary vehicles. The length of an edge $e \in \mathcal{E}$ is given by l_e and denotes the travel distance. For simplicity we assume $l_{(v,v')} = 1$ if $v \neq v'$, and $l_{(v,v)} = 0, \forall (v, v') \in \mathcal{E}$.[1] Let $O_i^p, D_i^p \in \mathcal{V}$ denote the origin and the destination of rider $i \in \mathcal{N}$. Also, let O_k^c denote the initial location of vehicle $k \in \mathcal{K}$, which does not have to be where riders are (in case of autonomous vehicles). We denote by $\mathcal{P}$ the set of all paths in $\mathcal{G}$. A *path* is a vector of size S, denoted by $\langle v \rangle_S \equiv \langle v_1, \ldots, v_S \rangle$, where an element $v_s \in \mathcal{V}$ is the s-th location in the path and $(v_s, v_{s+1}) \in \mathcal{E}, \forall s < S$. Since paths do not include information of time, we define a *route* as a tuple of vectors $\langle \langle v \rangle_S, \langle t \rangle_S \rangle$ where the path departs from location v_s at time t_s. Route $\langle \langle v \rangle_S, \langle t \rangle_S \rangle$ satisfies $t_{s+1} = t_s + l_{(v_s, v_{s+1})}$ if $v_s \neq v_{s+1}$. If $v_s = v_{s+1}$, it means the rider or vehicle stays at v_s for an amount of time given by $t_{s+1} - t_s$ (e.g., to wait for a vehicle or because they already arrived at the destination). The set of routes is denoted by $\mathcal{R}$. The route of rider i and the route of vehicle k are denoted by $r_i^p, r_k^c \in \mathcal{R}$, respectively. We let $\text{edge}(r, t) \in \mathcal{E}$ be

a function that returns the edge in route r that the rider or vehicle is traversing at t. E.g., $\text{edge}(\langle \langle A, B, C \rangle, \langle 0, 1, 2 \rangle \rangle, 1) = (B, C)$. Similarly, $\text{loc}(r, t) \in \mathcal{V}$ returns the location at t. E.g., $\text{loc}(\langle \langle A, B, C \rangle, \langle 0, 1, 2 \rangle \rangle, 1) = B$. Riders cannot move without vehicles but vehicles can move without riders if they are autonomous. If they are not autonomous, then they need at least one rider.

We denote the *assignment* of vehicles to riders with tensor $B \in \mathcal{B} = \{0, 1\}^{T \times N \times K}$. $B[t, i, k] = 1$ is possible only if the path of rider i is the same as vehicle k at time t. The assignment may change over time, which means a rider can switch vehicles on the way[2]. We do not specifically distinguish between drivers and riders. Any rider in a vehicle can be a driver if the vehicle is not autonomous. Furthermore, riders can use a taxi as the outside option ($B[t, i, k] = 0, \forall t \in \mathcal{T}, k \in \mathcal{K}$) if it is more beneficial than using ridesharing. We assume that the waiting time for a taxi is zero and the taxi takes the shortest path between the origin and destination of the rider. However, a rider cannot combine a taxi with ridesharing.

An overall *allocation* is denoted by $\pi = \langle \boldsymbol{r}^p, \boldsymbol{r}^c, B \rangle \in \mathcal{R}^N \times \mathcal{R}^K \times \mathcal{B}$, where $\boldsymbol{r}^p = \langle r_i^p \rangle_{i \in \mathcal{N}}, \boldsymbol{r}^c = \langle r_k^c \rangle_{k \in \mathcal{K}}$ are the routes for riders and vehicles, respectively. We will also use $r_i^p(\pi), r_k^c(\pi)$ to denote a route for rider i and vehicle k given allocation π. An allocation is feasible if, for each rider i, the route r_i^p consist of a path between O_i^p and D_i^p, and the vehicle capacity is not exceeded. Formally, the set of *feasible allocations* is defined by:

$$
\begin{aligned}
\Pi = \{ &\langle \boldsymbol{r}^p, \boldsymbol{r}^c, B \rangle \in \mathcal{R}^N \times \mathcal{R}^K \times \mathcal{B} \mid \\
&\forall t \in \mathcal{T}, i \in \mathcal{N}, k \in \mathcal{K}, r_i^p, r_k^c \in \mathcal{R}: \\
&\text{loc}(r_i^p, 0) = O_i^p, \ \text{loc}(r_i^p, T) = D_i^p, \ \text{loc}(r_k^c, 0) = O_k^c, \\
&B[t, i, k] = 1 \Rightarrow \text{edge}(r_k^c, t) = \text{edge}(r_i^p, t), \\
&\exists t' \in \mathcal{T}, \forall k' \in \mathcal{K} : (B[t', i, k'] = 0, l_{\text{edge}(r_i^p, t')} > 0) \\
&\quad \Rightarrow B[t, i, k] = 0, \\
&\sum_{i' \in \mathcal{N}} B[t, i, k] \leq w \}.
\end{aligned}
\tag{1}
$$

The first line of constraints specifies origins and destinations of the routes. The second constraint is the definition of B. The third constraint means that a taxi cannot be combined with ridesharing (if riders traverse an edge using a taxi, they always use a taxi). The fourth is the capacity constraint.

If vehicles are not autonomous or have no dedicated driver, Π needs an additional condition that any moving vehicle needs at least one rider: $l_{\text{edge}(r_k^c, t)} > 0 \Rightarrow \sum_{j \in \mathcal{N}} B[t, j, k] > 0$.

Now that we have defined the allocation, we proceed with defining the costs which the system aims to minimise. These costs include travel and waiting time, fuel costs and taxi labour costs. Formally, the travel time of rider i including waiting time is denoted by $T_i(\pi) = \min\{t_s | r_i^p = \langle \langle v \rangle_S, \langle t \rangle_S \rangle, v_s = D_i^p\}$, which is the earliest time to reach the destination. Furthermore, let T_i^0 denote the travel time

---

[1]The former ensures that a rider is always at a vertex at each time step. Furthermore, since l_e is used to compute costs (discussed below), the latter ensures that stationary vehicles do not incur fuel costs. Longer roads can be represented by joining multiple edges.

[2]In practice, this can take time, but we ignore this for simplicity. The model can easily be extended by introducing waiting times for vehicles when riders switch at some designated safe areas.

for rider i using the shortest path (e.g., taxi). The travel time of vehicle k is denoted by $T_k^c(\pi) = \sum_{t \in \mathcal{T}} l_{\text{edge}(r_k^c(\pi),t)}$. Furthermore, let $\alpha, \beta \in \mathbb{R}$ denote the (additional) labour cost per time step for a taxi driver and fuel cost per distance for a vehicle, respectively. Let $\gamma_i \in \Gamma = [0, \gamma_{max}]$ represent the value of time, i.e., a subjective cost rider i is willing to pay to avoid a unit of travel time. Then the cost of rider i is: $c_i(\pi) = c_i^0 = (\alpha + \beta + \gamma_i)T_i^0$ when using a taxi, and $c_i(\pi) = \gamma_i T_i(\pi)$ when using ridesharing. In addition, the fuel cost of the ridesharing system is calculated separately and is given by $c_F(\pi) = \sum_{k \in \mathcal{K}} \beta T_k^c(\pi)$. Given this, the *ridesharing (RS) problem* is to find a feasible allocation $\pi \in \Pi$ that minimizes the social cost including fuel cost $C_F(\mathcal{N}, \pi) = \sum_{i \in \mathcal{N}} c_i(\pi) + c_F(\pi)$.

To illustrate the model, Figure 1 shows an example with $N = 2, K = 2$ and $T = 5$. The two figures show two possible allocations: in left figure, for example, $r_1^p = \langle\langle A, A, B\rangle, \langle 0, 3, 4\rangle\rangle$ and $r_2^p = \langle\langle B, B, C\rangle, \langle 0, 1, 2\rangle\rangle$. Rider 2 is assigned to vehicle 1 at $t = 1$, i.e., $B[1, 2, 1] = 1$ while rider 1 is not assigned to vehicle 2 until $t = 3$, i.e., $B[3, 1, 2] = 1$. If $\gamma_1 = 2$ and $\gamma_2 = 1$, then the optimal allocation is the one on the right, with $T_i = T_k^c = 3$ and social cost $C_F = 15$.

2.2 Mechanism Design Formulation

A mechanism takes reports from riders on their privately-known types as input, and computes the allocation and payment as output. In our model, we assume O_i^p and D_i^p are easy to track with GPS and so γ_i is the only private information, which is referred to as rider i's *type*. Let $\boldsymbol{\gamma} \in \Gamma^N$ be the type profile of all riders, $-i$ the set of riders except for i, and $\boldsymbol{\gamma}_{-i}$ the types of all riders except i. For simplicity, we assume that all riders and vehicles are ready to depart at $t = 0$, but this can be easily extended. A *mechanism* is then defined as a function $\mathcal{M} : \Gamma^N \to \Pi \times \mathbb{R}^N$, that computes a feasible allocation $\pi \in \Pi$ from a given type profile and payment vector $\boldsymbol{x}(\pi) \in \mathbb{R}^N$, where $x_i(\pi)$ is the payment of rider i to the system. We sometimes use $\pi(\boldsymbol{\gamma})$ to emphasize the dependence on $\boldsymbol{\gamma}$. If i chooses a taxi, $x_i(\pi) = 0$. Sometimes we use shorthand notation $\pi(\gamma_i) = \pi(\gamma_i, \boldsymbol{\gamma}_{-i})$. Given this, the utility of rider i is defined as $u_i(\pi) = -c_i(\pi) - x_i(\pi)$.

In practice, the ridesharing mechanism may work as in the following example. First, rider i reports their type γ_i using a slider in a suitable smartphone app, along with O_i^p and D_i^p. Note that γ_i is not necessarily truthful. Once the report profile $\boldsymbol{\gamma}$ is obtained, the mechanism computes $\pi(\boldsymbol{\gamma})$ and $\boldsymbol{x}(\pi)$, and reveals them to riders before travel. Finally, x_i is charged after the travel. To this end, the mechanism needs to achieve the following properties:

Budget balance (BB), which is formulated as:

$$\sum_{i \in \mathcal{N}} x_i(\pi) \geq c_F(\pi). \tag{2}$$

Individual rationality (IR), which is formulated as:

$$u_i(\pi) \geq -c_i^0, \ \forall i \in \mathcal{N}. \tag{3}$$

Dominant-strategy incentive compatibility (DSIC), or truthfulness, which is formulated as:

$$\begin{aligned} u_i(\pi(\gamma_i, \boldsymbol{\gamma}_{-i})) &\geq u_i(\pi(\gamma_i', \boldsymbol{\gamma}_{-i})), \\ \forall \gamma_i' \in \Gamma, \ \forall i \in \mathcal{N}, &\forall \gamma_{-i} \in \Gamma^{N-1}. \end{aligned} \tag{4}$$

3 The VCG Mechanisms

Next, we will show that the well-known VCG mechanism [Shoham and Leyton-Brown, 2008] does not achieve BB and polynomial-time computation at the same time when applied to the RS problem. This mechanism first computes an optimal allocation $\pi(\boldsymbol{\gamma})$ given (reported) type profile $\boldsymbol{\gamma}$ and then it computes the payment $\boldsymbol{x}(\pi)$ as the externality that each rider imposes on other riders. Specifically, for our RS problem, the objective is to minimize $C_F(\mathcal{N}, \pi)$, and so the VCG payment is computed as $x_i(\pi) = C_F(-i, \pi(\boldsymbol{\gamma})) - C_F(-i, \pi(\boldsymbol{\gamma}_{-i}))$. As a result, the optimal allocation π and the payment $\boldsymbol{x}$ guarantee DSIC and IR. In the extended version, we show that the RS problem is NP hard by reducing the travelling salesman problem. However, with a mild assumption, an algorithm based on integer linear program (ILP) can solve it in polynomial time. The algorithm relaxes the ILP into a linear program (LP), that outputs an integer solution of the original ILP. The detail of the algorithm and proofs are also in the extended version. However, VCG has the following issue.

Proposition 1 VCG does not satisfy BB in the RS problem.

Proof. Suppose both riders 1 and 2 have the same origin O and destination D, which are directly connected with an edge. VCG allocates the vehicle to both riders for sharing. Since the absence of either rider does not change the allocation and cost of the other rider, the VCG payment is zero for both riders. This means the fuel cost of the vehicle is not recovered. □

Despite this negative result, we show it is possible to modify VCG to satisfy BB by sacrificing polynomial-time computation, as follows. Let $\mathcal{K}_i(\pi) = \{k \in \mathcal{K} | B[t, i, k] = 1, \exists t \in \mathcal{T}\}$ denote the set of vehicles that rider i uses. Then, the total fuel costs of all rides that a rider i is involved in is $c_F^i(\pi) = \sum_{k \in \mathcal{K}_i} \beta T_k^c(\pi)$. We then define an *imaginary* system cost by redundantly counting the fuel costs of vehicles as follows: $c_I(\pi) = \sum_{i \in \mathcal{N}} c_F^i(\pi)$. The total social cost including this imaginary cost is then given by $C_I(\mathcal{N}, \pi) = \sum_{i \in \mathcal{N}} c_i(\pi) + c_F(\pi) + c_I(\pi)$. Given this, we have a budget balanced VCG (BVCG) by replacing the objective function of VCG with C_I.

Proposition 2 Assuming autonomous vehicles, BVCG satisfies DSIC, IR and BB.

Briefly, this is because the payment is computed based on the imaginary cost c_I, which is redundant and larger than c_F. Autonomous vehicles guarantee the feasibility of vehicle routing even when i is absent in the payment computation. However, scalability is an issue. Since the new objective function with C_I is a non-linear function, we cannot use the ILP based polynomial-time algorithm.

4 Greedy Mechanisms

To address the issues with VCG mechanisms, we consider greedy mechanisms in this section. To this end, we first show that a naive greedy approach does not satisfy IC and we propose a general theory to design greedy DSIC mechanisms based on the concept of monotonicity. Then, we apply our theory to design a greedy DSIC ridesharing mechanism.
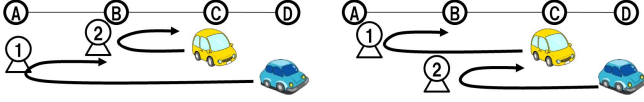
Figure 1: Example showing how monotonicity is achieved with our mechanism and violated with a naive greedy mechanism: $D_1^p = B, D_2^p = C, O_1^c = C, O_2^c = D$. Assume $l_e = 1, \forall e \in \mathcal{E}$ and expensive taxi $\alpha = 10$ and $\beta = 1$.

4.1 Monotonicty and Violation with Naive Greedy

[Myerson, 1981; Archer and Tardos, 2001] showed that, for single-valued domains, monotonicity of the allocation function $\pi(\boldsymbol{\gamma})$ is a necessary and sufficient condition for the existence of payments $\boldsymbol{x}(\pi)$ satisfying DSIC. Using notation from our model, monotonicity is defined as follows:

$$\gamma_i \leq \gamma_i' \Rightarrow V_i[\pi(\gamma_i)] \leq V_i[\pi(\gamma_i')], \ \forall \gamma_i, \gamma_i' \in \Gamma, \quad (5)$$

where, in general, $V_i : \Pi \to \mathbb{R}$ is the amount allocated, and referred to as *work* in [Archer and Tardos, 2001] and, for our setting, $V_i = -T_i(\pi)$ corresponds to the negative travel time.

Although monotonicity plays a key role in designing DSIC mechanism, finding appropriate monotone allocation functions π is non-trivial. Here, we show that a naive approach fails to implement a DSIC mechanism. We define a naive greedy allocation as follows. First, all riders are assigned to a taxi by default. Then, for each iteration it chooses rider i who minimizes the marginal social cost, given by $c_i - c_i^0 + c_F$, without changing the allocation in riders allocated so far, and computes the best allocation for the current rider.

Proposition 3 Mechanisms based on the naive greedy allocation do not satisfy monotonicity or DSIC.

Proof. Figure 1 also shows a counterexample whereby a naive greedy allocation fails to satisfy monotonicity. Let $\gamma_2 = 4$. If $\gamma_1 = 3$, the naive greedy allocates vehicle 1 to rider 2 in the first iteration because the rider brings the lowest cost $\gamma_2 T_2 - (\alpha + \beta + \gamma_2)T_2^0 + c_F = -5$ (left figure). However, if $\gamma_1 = 1$, the cost of the rider decreases as $\gamma_1 T_1 - (\alpha + \beta + \gamma_1)T_1^0 + c_F = -6$ and rider 1 is allocated first (right figure). Then, the lower γ_1 results in the earlier allocation and larger $V_1 = -T_1$, which is the violation of the monotonicity (5). This violation of monotonicity means mechanisms with the naive greedy allocation also violate DSIC, since this is a necessary condition for DSIC. $\square$

4.2 Design of Monotone Greedy Allocation

To address the issue above, we propose a characterisation that is useful for designing monotone allocation functions for greedy mechanisms. As we saw, the naive greedy violates monotonicity by giving priority to the riders with lower γ_i. Instead, we allocate agents in descending order of their (reported) type γ_i. In what follows, we use the index to denote the order in which agents are allocated. I.e., $j < i$ implies $\gamma_j \geq \gamma_i$ and that j is allocated before i. Let $\pi_{\leq i}$ denote the allocation up to and including i (with remaining agents allocated arbitrarily or, e.g., in the context of ride sharing, allocated to a taxi). Also, let $\Pi_{\leq i}(\pi_{\leq i-1})$ or simply $\Pi_{\leq i}$ denote the set of possible allocations given $\pi_{\leq i-1}$, keeping agents

$j < i$ fixed. Given this, we define a class of *greedy allocations with decreasing choices (GADC)* as follows:

$$\pi_{\leq i} = \underset{\pi \in \Pi_{\leq i}}{\operatorname{argmin}} \ J(\pi) \quad (6)$$

subject to $\Pi_{\leq i} \subseteq \Pi_{\leq j}$ if $j < i$. Here, $J : \Pi \to \mathbb{R}$ is an objective function. In words, GADC captures allocation functions which compute the optimal allocation for each i individually given a set of possible allocations, *and this set reduces as i reports a smaller γ_i* (and so appears later in the sequence).

Theorem 1 Let $\pi_{\leq i}(\gamma_a), \pi_{\leq i}(\gamma_b)$ denote i's allocation when reporting γ_a, γ_b respectively. A GADC satisfies monotonicity if and only if, $\forall \gamma_a, \gamma_b, i \in \mathcal{N}$:

$$J(\pi_{\leq i}(\gamma_a)) \leq J(\pi_{\leq i}(\gamma_b)) \Rightarrow V_i[\pi_{\leq i}(\gamma_a)] \geq V_i[\pi_{\leq i}(\gamma_b)]. \quad (7)$$

Intuitively, this means that, the more an agent contributes to decreasing the global objective (in our case, minimising the social cost), the better their allocation should be (in our case, the lower the travel time).

4.3 Monotonic Greedy Ridesharing Mechanism

We now apply Theorem 1 to our setting. First, we customize GADC from section 4.2 for our ridesharing problem. In particular, $\Pi_{\leq i}(\pi)$ is redefined as follows:

$$\Pi_{\leq i}(\langle \boldsymbol{r}^p, \boldsymbol{r}^c, B \rangle) = \{\langle \boldsymbol{r'}^p, \boldsymbol{r'}^c, B' \rangle \in \Pi |$$
$$\forall t \in \mathcal{T}, j < i, j' > i, k \in \mathcal{K} :$$
$$B'[t, j, k] = B[t, j, k], r'^p_j = r^p_j, \quad (8)$$
$$B[t, j', k] = 0\}. \quad (9)$$

This means that, when computing i's allocation, the allocation of the prior riders $j < i$ are not changed. With this definition, the assumption $\Pi_{\leq i} \subseteq \Pi_{\leq j}$ holds, because constraint (8) shrinks the set. Then, a greedy allocation for ridesharing is obtained by applying equation (6), with the objective function $J(\pi) = \sum_{i \in \mathcal{N}} c_i(\pi)$. In addition, to obtain monotonicity, (7) must be satisfied. To this end, we will show that we require an additional constraint whereby riders cannot be allocated to a taxi (i.e., their outside option). Although this constraint seems restrictive, we will relax this again later so that riders can use taxis if this is in their best interest (to satisfy IR). In the following, we refer to this setting as greedy allocation for ridesharing with no-taxi constraint or GARS-N. Formally:

$$\pi_{\leq i} = \underset{\pi \in \Pi^i}{\operatorname{argmin}} \sum_{i' \in \mathcal{N}} c_{i'}(\pi)$$
$$s.t. \ l_{\operatorname{edge}(r_i^p, t)} > 0 \Rightarrow \sum_{k \in \mathcal{K}} B[t, i, k] = 1, \forall t \in \mathcal{T}. \quad (10)$$

Lemma 1 GARS-N satisfies monotonicity.

Briefly, this is because minimizing J coincides with minimizing c_i for each rider i, since the constraints (8) and (9) do not allow i to change the allocations and costs of other riders. If riders do not use a taxi, $V_i = -T_i$ is proportional to c_i and the equivalent condition of monotonicity in Theorem 1 is satisfied. With the case in Figure 1, GARS-N allocates rider 1

first (right figure) when $\gamma_1 = 5$ and $\gamma_2 = 4$, and then $T_1 = 3$. Meanwhile, if $\gamma_1 = 1$, rider 2 is allocated first (left figure) and then $T_1 = 4$. Contrary to the naive greedy, the higher γ_1 results in the earlier allocation and larger V_1, and this satisfies monotonicity. Note that, without the no-taxi constraint, monotonicity can be violated as follows. If a taxi is cheap ($\alpha = 1$), then rider 1 uses a taxi when $\gamma_1 = 1$, resulting in $T_1 = 1$, which violates monotonicity. Also, note that the no-taxi constraint does not help naive greedy because nobody uses a taxi in the proof of Proposition 3.

Next, we consider the computation of the payment. Let $\hat{T}_i(\pi) = T_i(\pi) - T_i^0$ denote the normalized travel time. Following [Myerson, 1981], the unique payment satisfying DSIC is given by:

$$x_i = x_i^0 + \int_0^{\gamma_i} \gamma \frac{d}{d\gamma} \hat{T}_i(\pi(\gamma)) d\gamma \qquad (11)$$

where x_i^0 is a value which is independent of i's report. Solving the integral is challenging in general since it is necessary to recompute the allocations given all possible types of the rider whose payment we are computing. However, in our case, because $\hat{T}_i(\pi)$ only depends on the order of the allocation, it is sufficient to vary the position in the ordering. We refer to the GARS-N allocation with this payment as the GARS-N mechanism. Also, this mechanism can be computed in polynomial-time using the same technique in section 3.

Lemma 2 The GARS-N mechanism is DSIC and can be computed in polynomial-time.

To ensure BB, we compute the base payment x_i^0 based on report independent parameters $\boldsymbol{T}_0$, $\hat{\boldsymbol{T}}_{min}$ and c_{Fub}, where $\boldsymbol{T}_0$ is a vector of T_i^0. Furthermore, $\hat{\boldsymbol{T}}_{min}$ is a vector of normalized minimal travel times, which is computed by solving the following allocation problem:

$$\pi = \operatorname*{argmin}_{\pi' \in \Pi} \sum_{i \in \mathcal{N}} \hat{T}_i(\pi')$$
$$s.t.\ l_{\text{edge}(r_i^p, t)} > 0 \Rightarrow \sum_{k \in \mathcal{K}} B[t, i, k] = 1, \forall t \in \mathcal{T}, \forall i \in \mathcal{N}.$$

Finally, c_{Fub} is an upper bound on the fuel cost c_F of GARS-N. We assume that this parameter is given (in the experimental section, we explain how this is computed in our experimental setting). The base payment is computed by splitting c_{Fub} among riders, proportional to each agent i's relative shortest travel time, $T_i^0 - \hat{T}_{min,i} - \min_j(T_j^0 - \hat{T}_{min,j})$. Since this base payment covers c_{Fub}, GARS-N is BB.

Lemma 3 If c_{Fub} is given, the GARS-N mechanism is budget balanced.

We now discuss IR. Since the no-taxi constraint restricts outside options, it can cause the violation of IR in some settings. To address this issue, we use the following lemma.

Lemma 4 The GARS-N mechanism satisfies IR if

$$\hat{T}_i(\pi(0)) \leq [(\alpha + \beta) T_i^0 - x_i^0] / \gamma_{max}, \ \forall i \in \mathcal{N}. \qquad (12)$$

Briefly, (12) shows the lower bound of the cost that satisfies IR. We can now define the GARS-N mechanism with individual rationality (GARS-NIR) as follows. Before executing

---

Algorithm 1 GARS-NIR

1: **procedure** $\pi =$ GARS-NIR$(\gamma, \mathcal{N})$
2:     $\mathcal{N}_{IR} = []$
3:     **for** $i \in \mathcal{N}$
4:        **if** i satisfies (12) **then**
5:           $\mathcal{N}_{IR}.$append(i)
6:        **else**
7:           $B[t, i, k] = 0, \ \forall t \in \mathcal{T}, \forall k \in \mathcal{K}$
8:     Return GARS-N$(\gamma, \mathcal{N}_{IR})$

Algorithm 2 GARS-N

1: **procedure** $(\pi, \boldsymbol{x}) =$ GARS-N$(\gamma, \mathcal{N})$
2:     $\hat{\mathcal{N}} =$ Sorted $\mathcal{N}$ in descending order of γ_i
3:     $\pi =$ GREEDYALLOC$(\gamma, \hat{\mathcal{N}})$
4:     $t_i = \hat{T}_i(\pi), \Delta x_i = 0, \forall i \in \mathcal{N}$
5:     **for** $i \in \mathcal{N}$
6:        **for** $order_i \in [\hat{\mathcal{N}}.index(i), \ldots, N]$
7:           $\mathcal{N}' = \hat{\mathcal{N}}, \gamma' = \gamma$
8:           $j = \hat{\mathcal{N}}[order_i]$
9:           $\mathcal{N}'.$pop$(\hat{\mathcal{N}}.index(i))$
10:          $\mathcal{N}'.$insert$(order_i, i)$
11:          $\gamma_i' = \gamma_j$
12:          $\pi' =$ GREEDYALLOC$(\gamma', \mathcal{N}')$
13:          $\Delta x_i = \Delta x_i + (\hat{T}_i(\pi') - t_i)\gamma_j$
14:          $t_i = \hat{T}_i(\pi')$
15:        $x_i = \Delta x_i + x_i^0$

GARS-N, riders that violate condition (12) are excluded from the mechanism, and are allocated to a taxi. This process is repeated until there is no violation. Then GARS-N is executed with the remaining riders. Since no riders violate (12), this mechanism satisfies IR.

Algorithm 1 shows the pseudocode of GARS-NIR (using Python-like list operations). It filters out riders who violate IR condition (12) in line 4 and allocates them to a taxi (line 7). Then, GARS-N in Algorithm 2 computes the allocation for riders who satisfy (12). First, it sorts riders in descending order of reported γ_i (line 2). Then it computes an allocation by GREEDYALLOC (line 3). From line 4 to 15, it computes the second term of x_i in (11), by changing the order of riders. In line 4, variables are initialized. For each rider i (line 5), it changes the order of the rider (line 6 to 10) and the report with a consistent value γ_i' (line 11), and computes an allocation with the new order $\mathcal{N}'$ (line 12). Thus, the integral term in (11) is computed (line 13), and the payment x_i is given by adding the base payment x_i^0 (line 15). The GREEDYALLOC function (Algorithm 3) computes the allocation for each rider one by one by solving (10) in line 3.

Now we have the following result.

Theorem 2 The GARS-NIR mechanism satisfies DSIC, IR, BB and can be computed in polynomial-time.

Although GARS-NIR is not optimal, also note that no mechanism can simultaneously achieve optimality, DSIC, IR and BB [Myerson and Satterthwaite, 1983]. Hence obtain-

Algorithm 3 Greedy Allocation

1: **procedure** $\pi =$ GREEDYALLOC($\gamma, \mathcal{N}$)
2:      **for** $i \in \mathcal{N}$
3:         Compute $\pi_{\leq i}$ by (10)
4:         Return $\pi_{\leq i}$

Mechanism	OPT	DSIC	IR	BB	POLY
VCG	Y	Y	Y	N	Y
BVCG	N	Y	Y	Y	N
Hungarian	N	Y	Y	N	Y
GARS-NIR	N	Y	Y	Y	Y

Table 1: Characteristics of mechanisms.

ing three of these properties is the best one can achieve. It is also difficult to bound the efficiency of GARS-NIR, because we cannot use the standard bounding technique with naive greedy allocations [Nemhauser *et al.*, 1978], for the reason we discussed in Proposition 3. Determining theoretical efficiency bounds remains an open problem for future work.

5 Experimental Results

Given that we do not have formal efficiency bounds, we evaluate the mechanisms empirically using simulations using both synthetic and real data. For the former, we generate random road networks of 4 degrees maximum and average results over 32 simulations for each setting. For the latter, we use the New York taxi data set [NYC-OpenData, 2018]. We assume autonomous vehicles for all our experiments. We vary the taxi cost α and choose these loosely based on prices from London taxis. If we normalize the fuel cost to be $\beta = 1$, then $\alpha = 5$ is considered a moderate value and $\alpha = 1$ is cheap. As for the value of time, we set $\gamma_i = \gamma_{max} * i/N$ to simulate heterogeneous riders. c_{Fub} is estimated by sampling solutions of GARS-N with different allocation orders, and multiplying the maximum fuel cost among the solutions with a constant coefficient. Further details are provided in the extended version.

In what follows we first compare the performance of all mechanisms on a small setting to enable comparison with non-scalable mechanisms such as BVCG. We then evaluate the scalability of the greedy mechanism to larger settings. Since GARS-NIR computes all possible ways for a rider to switch between vehicles, this turns out to be time consuming computationally. Also, in practice, such switches are often not desirable. Hence, for large settings, we modify the feasible set Π of GARS-NIR to allow a rider to use only a single vehicle (sGARS-NIR). This version still satisfies the same properties as GARS-NIR.

5.1 Small Setting

We compare VCG, BVCG, GARS-NIR, the Hungarian mechanism and the outside option (Table 1). The Hungarian mechanism computes an optimal 1-to-1 matching between riders and vehicles (including taxis) using the Hungarian algorithm, and then applies payments similar to VCG. The mechanism
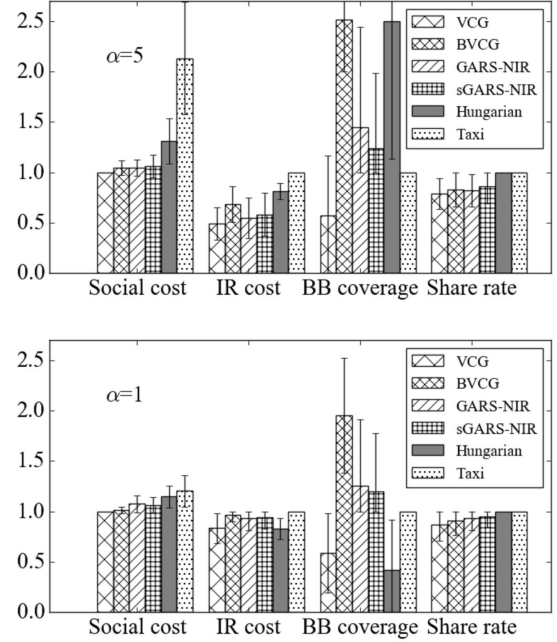
Figure 2: Comparison of mechanisms with different taxi costs: $\alpha = 5$ (top), $\alpha = 1$ (bottom).

is also DSIC, BB and IR, but does not support multiple riders per vehicle. For BVCG, a brute force search is used to compute vehicle allocations. Figure 2 (top) shows the results for $N = 3, K = 2, T = 4, V = 4$ and $\alpha = 5$ and the error bars show the 95% confidence intervals. Here, *social cost* is the ratio of $C_F(\mathcal{N}, \pi)$ over the cost of the optimal allocation. Furthermore, *IR cost* refers to $-u_i(\pi)/c_i^0$. If this value is above 1, it means that IR is violated. *BB coverage* shows $(\sum_{i \in \mathcal{N}} x_i(\pi))/c_F(\pi)$. A value less than 1 means BB is violated (however, a value close to 1 is preferred, as long as it is over 1, meaning that riders are not over paying). Share rate shows $\sum_{k \in \mathcal{K}} T_k / \sum_{i \in \mathcal{N}} T_i$. A smaller value means ridesharing needs fewers vehicles to cover the trips of riders.

Here, BVCG is budget balanced without a large increase in social cost (a 6.4% increase). Meanwhile, VCG recovers only 52% of the fuel costs on average in payments. However, a disadvantage of BVCG is its computational complexity when using the brute force search. Instead, GARS-NIR (and sGARS-NIR) can be computed efficiently while satisfying DSIC, IR and BB, and still keeping social cost near optimal (an 8.6% increase compared to optimal). Also, it shows that the performance of sGARS-NIR is close to GARS-NIR. Although the vehicle-switching behaviors of GARS-NIR are practical in suburban areas, it may not be so practical in urban areas. However, the result shows only $\frac{1}{6}$ of riders switch vehicles. Besides, it is possible to extend the model to introduce a waiting time for switching vehicles, and this can reduce the number of the cases. Also, it is easy to extend the model by mixing GARS-NIR with the version without switching (sGARS-NIR), and letting riders choose if they want to switch or not. In comparison, Figure 2 (bottom) shows the result when the
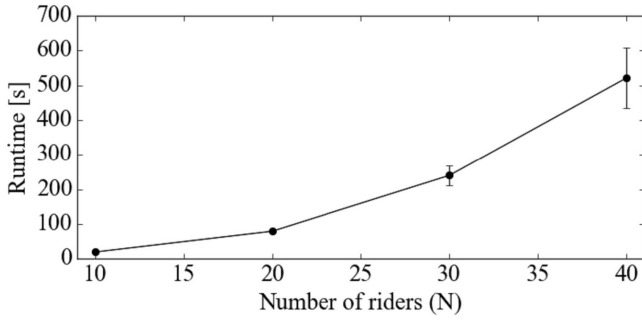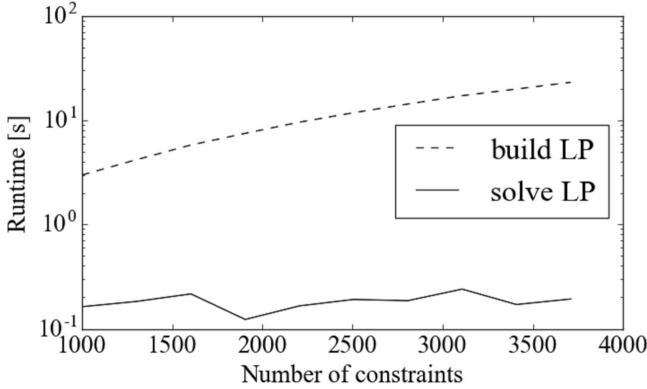
Figure 3: Runtime of sGARS-NIR.

Figure 4: Figure showing a breakdown of the runtime. As can be seen, most of the runtime is spent constructing LP object, instead of solving it.

outside option is cheap ($\alpha = 1$). In this case, the IR cost comes closer to 1 for all mechanisms because of the incentive to opt out. BB coverage also decreases for some mechanisms because as more riders opt out, there is less opportunity to split the fuel cost. Meanwhile, the increase in social cost is not large in GARS-NIR (4.5%) because riders properly opt out if taxi is better. While 1.07 passengers ride on a vehicle on average in case of $\alpha = 1$, it is 1.23 passengers in case of $\alpha = 5$. This demonstrates that more expensive taxis lead to a higher usage of ridesharing, enabled by GARS-NIR satisfying IR of riders.

5.2 Scalability

Next, we evaluate the scalability of sGARS-NIR. Since other mechanisms such as BVCG are not scalable, we do not include them here. Figure 3 shows the runtime of the mechanism with different N using parallel computation with 16 cores. Since the payment computation consists of independent computations of allocations with different orders of riders, it can be easily parallelized. For other parameters we use $K = N/2, T = 15, V = 10, \alpha = 5, \beta = 1$. sGARS-NIR takes less than 14 minutes when $N = 40$. Figure 4 shows that more than 99% of the computation time is spent constructing CPLEX LP objects. Hence, much of this can be precomputed and addressed with specalised implementation.
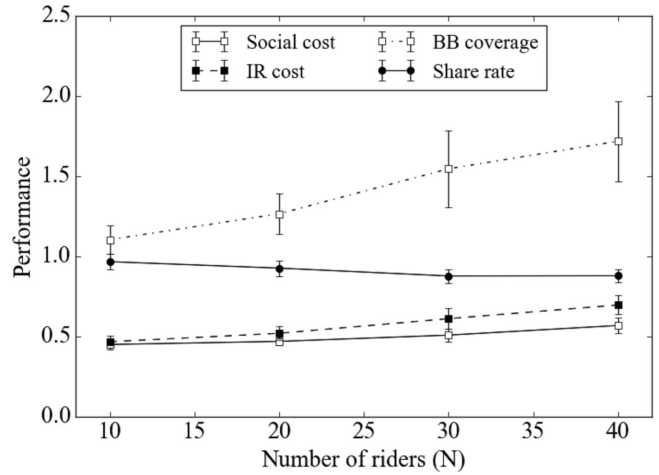
Figure 5: Performance as the number of riders (N) increases.

5.3 Experiments with Real-world Data

In addition to using synthetic data, we also conduct experiments using the New York taxi data set [NYC-OpenData, 2018]. We analyse the traffic demands in the central Manhattan area, which consists of 19 zones. A single trip data point includes the origin zone, the destination zone, departure and arrival time. We randomly generate trip demands with the trip distribution from February 2019. All riders are assumed to be picked up and dropped off at the center of their zones. We set $K = 20, T = 15, V = 19, \alpha = 5, \beta = 1$ and vary N. Figure 5 shows the ratio of social cost compared to the taxi cost, IR cost, BB coverage and share rate of sGARS-NIR. The share rate is improved as N increases since this increases the chance of sharing vehicles. However, with K fixed, increasing N causes a lack of vehicles and thus an increase in the IR cost and the social cost. This can be solved by providing an appropriate number of vehicles. The BB coverage shows over payment of riders with large N, but they still save compared to taking a taxi.

6 Conclusions

We propose a flexible ridesharing model that includes autonomous vehicles. The goal is to find allocation and payment mechanisms which are dominant-strategy incentive compatible, budget balanced, individually rational, can be computed in polynomial time and achieve high social welfare. To this end, we show that both the VCG and naive greedy mechanisms do not meet these requirements. We then present a characterisation for monotone greedy allocations and use this to develop a novel greedy mechanism which satisfies all of the desirable properties. In addition, through numerical simulations we show that our monotone greedy mechanism achieves close to optimal allocations.

Future directions include envy-free mechanisms and improving the implementation of the algorithm so it scales to even larger settings, as well as providing theoretical performance bounds for our greedy approach.

Ethical Impact

Our work has positive ethical impacts on society. First, our mechanism provides a fair ridesharing service in the sense that people get compensated for their detour. Second, the incentive compatibility of our mechanism maximizes the utility of individuals. Third, our ridesharing mechanism can benefit society and the environment by meeting more traffic demand with fewer vehicles than existing taxi services or privately-owned vehicles. A possible negative impact is that people might perceive the mechanism as unfair because the payments for a given service may differ depending on the other participants. We could address this by looking into envy-free mechanisms in future work.

References

[Archer and Tardos, 2001] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 482–491. IEEE, 2001.

[Chen *et al.*, 2019] Mengjing Chen, Weiran Shen, Pingzhong Tang, and Song Zuo. Dispatching through pricing: Modeling ride-sharing and designing dynamic prices. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 165–171, 2019.

[Cheng *et al.*, 2014] Shih-Fen Cheng, Duc Thien Nguyen, and Hoong Chuin Lau. Mechanisms for arranging ride sharing and fare splitting for last-mile travel demands. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1505–1506, 2014.

[Cordeau and Laporte, 2007] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29–46, 2007.

[Hasan *et al.*, 2018] Mohd Hafiz Hasan, Pascal Van Hentenryck, Ceren Budak, Jiayu Chen, and Chhavi Chaudhry. Community-based trip sharing for urban commuting. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 6589–6597, 2018.

[Kleiner *et al.*, 2011] Alexander Kleiner, Bernhard Nebel, and Vittorio Amos Ziparo. A mechanism for dynamic ride sharing based on parallel auctions. In *Twenty-Second International Joint Conference on Artificial Intelligence*, pages 266–272, 2011.

[Lu *et al.*, 2018] Alice Lu, Peter I. Frazier, and Oren Kislev. Surge pricing moves uber's driver-partners. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, 2018.

[Ma *et al.*, 2019] Hongyao Ma, Fei Fang, and David C Parkes. Spatio-temporal pricing for ridesharing platforms. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 583–583, 2019.

[Myerson and Satterthwaite, 1983] Roger B Myerson and Mark A Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of economic theory*, 29(2):265–281, 1983.

[Myerson, 1981] Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.

[Nemhauser *et al.*, 1978] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[NYC-OpenData, 2018] NYC-OpenData. 2018 yellow taxi trip data. https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq, 2018. Accessed: 2020-04-01.

[Rheingans-yoo *et al.*, 2019] Duncan Rheingans-yoo, Scott Duke Kominers, Hongyao Ma, and David C Parkes. Ridesharing with driver location preferences. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 557–564, 2019.

[Shen *et al.*, 2016] Wen Shen, Cristina V. Lopes, and Jacob W. Crandall. An online mechanism for ridesharing in autonomous mobility-on-demand systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 475–481, 2016.

[Shoham and Leyton-Brown, 2008] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, Game-Theoretic, and logical foundations*. Cambridge University Press, 2008.

[Spieser *et al.*, 2014] Kevin Spieser, Kyle Treleaven, Rick Zhang, Emilio Frazzoli, Daniel Morton, and Marco Pavone. Toward a systematic approach to the design and evaluation of automated mobility-on-demand systems : A case study in singapore. In *Road vehicle automation*, pages 229–245. Springer, 2014.

[Zhao *et al.*, 2014] Dengji Zhao, Dongmo Zhang, Enrico H Gerding, Yuko Sakurai, and Makoto Yokoo. Incentives in ridesharing with deficit control. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pages 1021–1028, 2014.