Fairness and Stability in P2P Ride-sharing without Payment

Paper #6052

Abstract

Unlike commercial ride-sharing, peer-to-peer (P2P) ride-sharing has been subject to limited research—although it can promote viable solutions in non-urban communities. This paper addresses the core problem in P2P ride-sharing: the matching of riders and drivers. We elevate users' preferences as a first-order concern and introduce novel notions of fairness and stability in P2P ride-sharing. We propose algorithms for efficient matching while considering these user-centric factors. Results suggest that individually rational, fair and stable solutions can be obtained in reasonable computational times, and can improve baseline outcomes based on system-wide efficiency exclusively. The proposed approach can be further enhanced via external incentives.

1 Introduction

By providing more flexible commuting options than public transportation and private vehicles, on-demand ride-hailing and ride-sharing platforms have become increasingly popular in urban areas. However, the availability and affordability of on-demand transportation remain much more limited in sub-urban and rural areas. In practice, an overwhelming majority of commuting trips rely on self-driving with private vehicles. An increasingly popular option to promote alternative forms of mobility in non-urban areas lies in peer-to-peer (P2P) ride-sharing: by bringing together commuters traveling along similar routes at similar times, P2P ride-sharing can enhance mobility while reducing the costs of transportation, traffic congestion, and greenhouse gas emissions. Moreover, P2P ride-sharing can also improve access to basic needs for disadvantaged populations with limited car ownership.

To be successful, P2P ride-sharing platforms require effective algorithms to match rider requests with available drivers. The topic of matching in ride-sharing platforms has attracted considerable research interest in recent years [Ozkan and Ward, 2017; Bertsimas *et al.*, 2019; Santi *et al.*, 2014; Bei and Zhang, 2018; Alonso-Mora *et al.*, 2017], building upon related problems such as the dial-a-ride problem (DARP) [Cordeau, 2006; Parragh *et al.*, 2010] and the vehicle routing problem with time windows (VRPTW) [Cordeau *et al.*, 2007], matching in spatial-temporal networks, and is sometimes studied jointly with the topic of pricing [Bimpikis *et al.*, 2016]. However, there is only limited research for P2P ride-sharing without direct payments from riders to drivers [Masoud and Jayakrishnan, 2017], which can provide viable solutions for a community, e.g., the community of residents from close-by regions and employees of the same company, featuring drivers who have their own travel plans and are willing to share part of their trips with riders. Moreover, most existing work in matching in P2P ride-sharing only considers the flexibility windows of the drivers and riders ("users", henceforth) as constraints and ignores users' preferences and incentives for participation. In addition, the predominant objective used in this setting is to minimize total costs (e.g., travel costs and inconvenience costs). However, such approaches do not capture the impact of matching decisions on individual users, including the fairness among users and whether or not the users will accept the matching outcome. In this paper, we address these limitations.

In contrast to previous work, we focus on matching riders and drivers in P2P ride-sharing without payment and study the problem from a user-centric perspective, with the objective of balancing system-wide efficiency and user satisfaction. We elevate users' preferences as a first-order concern by considering the users' preferred travel times and modeling drivers' altruism—motivated by the community-based P2P context. Furthermore, we explicitly consider the fairness and stability of the matching outcome. Fairness is formalized by guaranteeing that every rider is matched with at least a threshold probability. Stability is formalized by guaranteeing that users have no incentive to reject the current matching and take an alternative transportation option on his own or engage in independently shared rides outside of the platform. We develop a multi-objective framework that quantifies the trade-offs between efficiency, fairness, and stability.

This paper makes the following contributions. 1) We extend the RTV framework from [Alonso-Mora *et al.*, 2017] to incorporate users' preferred times in P2P ride-sharing; the resulting model is computationally complex, and we propose a pruning algorithm to enhance its performance. 2) We formalize the notions of fairness and stability in P2P ride-sharing building upon a user utility model that incorporates altruism; we further propose an enhanced notion of B-stability that allows for the use of external incentives (e.g., coupons) to stabilize the matching. 3) We analyze the price of fairness and stability, showing that the price of fairness and stability can be arbitrarily large, and moreover, fairness can result in a very unstable market without external incentives. 4) We design algorithms for computing efficient solution given fairness and stability constraints and evaluate them through extensive experiments. Results show that solutions with enhanced fairness and stability can be found in shorter, or equivalent, computational times as solutions based on efficiency objectives exclusively. Furthermore, suitable external incentives can result in fair and stable outcomes.

2 Related Work

As ride-sharing has become a prominent aspect of urban transportation, it has also attracted extensive research attention [Furuhata *et al.*, 2013; Berbeglia *et al.*, 2007]. This paper focuses on matching users in P2P ride-sharing and is different from most existing work due to the characteristics of P2P ride-sharing—including users' preferred travel times, maximum detour times, and altruism. In addition, we add the notions of fairness and stability to the model.

Well-studied notions of fairness include max-min fairness [Bertsimas *et al.*, 2011] and envy-freeness [Bogomolnaia and Moulin, 2001], in both deterministic and randomized resource allocation and assignment problems. In a probabilistic setting, [Liu and Knightly, 2003] limits the probability that the difference in individual user's utility larger than a threshold. Recent work on randomized online matching analyzes the marginal probability of a pair of items being matched to demonstrate the fairness of the matching [Cohen and Wajc, 2018]. In this paper, we propose a novel notion of fairness based on the max-min marginal probability of an individual user being matched that combines these ideas.

Stability is well studied in two-sided matching [Manlove *et al.*, 2002; Iwama and Miyazaki, 2008]. In ride-sharing, monetary mechanisms have been proposed to achieve stability [Bistaffa *et al.*, 2015]. However, the notion of stable matching in ride-sharing without payments is not studied until recently [Wang *et al.*, 2017] and only a simple setting with unit capacity for drivers is analyzed in detail—which contrasts with our own setting where each driver can dynamically pick up and drop off multiple riders.

Finally, our multi-objective framework relates to the notions of the price of fairness and the price of stability in resource allocation [Bertsimas *et al.*, 2011; Anshelevich *et al.*, 2008]. To our knowledge, the joint relationships between efficiency, fairness, and stability have not been studied. We address this question in the context of P2P ride-sharing.

3 Efficient Matching with User Preferences

In this section, we first define the P2P ride-sharing problem, and present our matching algorithm that incorporates user' preferred time when maximizing system-wide efficiency.

3.1 Model

Let $\mathcal{R}$ be the set of riders. A rider r is characterized by his origin o_r, destination q_r, a time window $W_r = [\tau_r^e, \tau_r^l]$ describing the earliest possible departure time and latest possible arrival time, and value of the trip ν_r. In addition, we consider the rider's preferred departure time $\tau_r^\star \in W_r$ and his maximum acceptable detour time Δ_r.

Let $\mathcal{D}$ be the set of drivers. A driver $d \in \mathcal{D}$ is characterized by his origin o_d, destination q_d, time window $W_d = [\tau_d^e, \tau_d^l]$, value ν_d, preferred departure time $\tau_d^\star$, and maximum acceptable detour time Δ_d —all defined as for the riders. In addition, we denote by k_d the capacity of driver $d \in \mathcal{D}$ (i.e., number of seats in his vehicle).

Let $V := \{o_i \cup q_i : i \in \mathcal{R} \cup \mathcal{D}\}$ be the set of locations containing all users' origins and destinations. We consider a finite, discretized time horizon $[T] := \{1, 2, 3, ..., T\}$. Let $\texttt{time}(u, v) \in \mathbb{Z}^{\geq 0}$ be the shortest travel time from $u \in V$ to $v \in V$. If any two users' origins and/or destinations are co-located, we construct two different nodes in V with $\texttt{time}(\cdot) = 0$. Let $\texttt{time}_i := \texttt{time}(o_i, q_i)$ be the user i's default travel time.

If a rider r is not matched, he can complete the trip with a cost λ_r ($\leq \nu_r$), which can be seen as the cost of an alternative transportation mode. If rider r is matched, he incurs a cost $C_{tt'}^r$ when he gets picked up at time $t \in W_r$ and dropped off at time $t' \in [t + \texttt{time}_r, \tau_r^l]$. In this paper we consider the following cost model, which is inspired by [Alonso-Mora *et al.*, 2017; Wang *et al.*, 2017]

$$C_{tt'}^r := c_{dev}^r \cdot |t - \tau_r^\star| + c_{trl}^r \cdot (t' - t),$$
$$C_{tt'}^d := c_{dev}^d \cdot |t - \tau_d^\star| + c_{trl}^d \cdot (t' - t),$$

where c_{dev}^r (resp. c_{dev}^d) is the cost per unit of deviation and c_{trl}^r (resp. c_{trl}^d) is cost per unit of traveling time experienced by rider r (resp. driver d). It follows the natural assumptions that the more the user deviates from their preferred time and the more they detour, the higher the cost.

A *driver-schedule* is an ordered sequence of node-time pairs describing how a single driver travels. For example, $((o_d, 0), (o_r, 3), (q_r, 5), (q_d, 10))$ is a driver-schedule that starts at time 0, picks up a rider r at time 3, drops r off at time 5, then arrives at the driver's destination at time 10. A *stop* is a node in a driver-schedule. A driver-schedule is feasible if it starts with a driver's origin, ends with his destination, traversing a subset of riders' origins and destinations and satisfying constraints associated with travel times and time window. Let $\mathcal{S} = 2^{\mathcal{R}}$ be the set of all possible subsets of riders. We say a (d, S) pair where $d \in \mathcal{D}$ and $S \in \mathcal{S}$ is *compatible* if there exists a feasible driver-schedule for driver d to pick up and drop off all riders in S.

A *schedule* is a collection of feasible driver-schedules, one for each driver in $\mathcal{D}$, with each rider shown up in at most one driver-schedule. Let Π be the set of all possible schedules. Given a schedule $\pi \in \Pi$, denote by S_d^π the subset of riders that driver d is matched to. Let $S_{\mathcal{D}}^\pi = \cup_{d \in \mathcal{D}} S_d^\pi$ be the set of riders that are matched under π. Let d_r^π be the driver that rider r is matched to (which may be *null* if r is not matched). Let S_r^π be a subset of riders that are matched to the same driver with r (including r) in π . We denote by $\overline{t_r^\pi}$ and $\underline{t_r^\pi}$ the time when r is picked up and is dropped off respectively. $\overline{t_d^\pi}$ and $\underline{t_d^\pi}$ are defined similarly for a driver d.

A *matching* is an assignment of drivers $\mathcal{D}$ to subsets of riders $\mathcal{S}$ such that each driver is assigned to exactly one subset, and each rider is assigned to at most one driver. Let $\mathcal{M}$ be set of all possible matchings. Each schedule $\pi \in \Pi$ defines exactly one matching, and a matching $M \in \mathcal{M}$ may correspond to multiple schedules. Thus, we sometimes use $\pi \in \Pi$ to refer to a matching and a schedule simultaneously. S_d^M, d_r^M, S_r^M are defined in a similar way as S_d^π, d_r^π, S_r^π.

The objective of the matching problem is to find an optimal schedule π^* that minimizes the total cost in the system, i.e.,

$$\pi^* = \arg\min_{\pi\in\Pi} \sum_{d\in\mathcal{D}} \sum_{r\in S_d^\pi} C^r_{\overline{t}^\pi_r \underline{t}^\pi_r} + \sum_{d\in\mathcal{D}} C^d_{\overline{t}^\pi_d \underline{t}^\pi_d} + \sum_{r\in\mathcal{R}\setminus S^\pi_{\mathcal{D}}} \lambda_r$$

Let c_{dS} be the minimum cost of driver d satisfying riders in S, i.e. $c_{dS} = \min\limits_{\pi:\pi\in\Pi, S_d^\pi = S} C^d_{\overline{t}^\pi_d \underline{t}^\pi_d} + \sum_{r\in S} C^r_{\overline{t}^\pi_r \underline{t}^\pi_r}$ if (d, S) is compatible; otherwise $c_{dS} = \infty$. Then the system's objective is equivalent to

$$\min\nolimits_{\pi\in\Pi} \sum\nolimits_{d\in\mathcal{D}} c_{dS_d^\pi} + \sum\nolimits_{r\in\mathcal{R}\setminus S^\pi_{\mathcal{D}}} \lambda_r \tag{1}$$

3.2 Algorithm for Efficient Matching

We describe our algorithm that finds the optimal schedule π^* at a high level. We defer the details of our algorithm to the Appendix[1]. We follow the Request-trip-vehicle (RTV) framework proposed in [Alonso-Mora *et al.*, 2017], a state-of-art framework for large-scale matching in ride-sharing and instantiate it for our problem setting.

The RTV framework has three main parts: (a) construction of all compatible (d, S) pairs; (b) computation of the minimum cost (c_{dS}) for each compatible pair (d, S); and (c) computation of the cost-minimizing matching. The main challenge in applying the framework to our problem lies in (b). Since we incorporate the novel aspect of users' preferred times, existing approaches cannot be directly applied and we propose a novel algorithm TripCost for computing c_{dS}.

More specifically, in part (a), for each driver $d \in \mathcal{D}$, we incrementally construct a list of subsets of riders that are compatible with d by gradually increasing the size of the subset. In each step, we add one rider to an existing compatible subset S of size $h-1$, getting a subset S' of size h. If the subsets of S' with $h-1$ riders are all compatible with d, i.e., already in the list, algorithm TripCost of part (b) is called to further verify the compatibility of S'. When no more subsets with size h can be constructed, we will construct subsets with size $h+1$. Let $\mathcal{E} \subseteq \mathcal{D} \times \mathcal{R}$ be the set of compatible (d, S) pairs.

For part (b), we develop TripCost(d, S), a novel tree-search based algorithm to compute c_{dS}. Each node of the tree represents a partial route, i.e., an ordered list of stops that the driver needs to visit, with the root representing the origin of the driver. In each step in the tree-search, we compute a lower bound of the cost of a full route that extends the current partial route by checking the must-visit stops remaining to be visited, and prune the branch if it is worse than the optimal solution found so far. Each leaf node represents a full route, and we need to determine the optimal timing to visit each stop. For this purpose, we design a dedicated MILP based on a constructed time-location graph with duplicated edges.

[1]https://www.dropbox.com/s/3lodwdqkiy081j6/p2p.pdf?dl=0

We also use two pruning methods to improve TripCost. First, we develop dynamic programming (DP)-based approach to compute a tighter lower bound of the route at the leaf node before computing TripMILP, which enables us to avoid unnecessary MILP computations. Intuitively, each DP states contain a location and a time, and it computes the minimum cost to reach the location at that time based on previous states. Second, for each (d, S) pair, we learn promising routes from the optimal schedule found for (d, S') where $S' \subset S$ and (d', S) where $d' \neq d$. By getting these routes before the tree search, the pruning efficiency increases significantly.

In part (c), we use a standard MILP for matching problem (same as in [Alonso-Mora *et al.*, 2017]) to find the cost-minimizing matching given the c_{dS} for all the compatible (d, S) pairs. The variables used in the MILP include x_{dS} and y_r. $x_{dS} = 1$ if (d, S) is matched and 0 otherwise; and $y_r = 1$ if rider r is unsatisfied and 0 otherwise.

4 Modeling Fairness and Stability

We now introduce notions of fairness and stability, beyond total cost. We define user utility and develop an algorithm for a fair and stable solution based on user preferences.

Utility model

Each rider's utility is defined to be (*the value of the trip*) $-$ (*the cost they suffer*). If rider r is picked up at t and dropped off at t', his/her utility is $U_r = \nu_r - C^r_{tt'}$. If the rider is not matched, then $U_r = \nu_r - \lambda_r$. We assume that drivers are altruistic, i.e. a driver gains additional utility proportional to the utility gained by his passengers. Thus the driver's utility is defined to be (*the value of the trip*) $+$ (*altruistic utility*) $-$ (*the cost they suffer*). We define two utility functions for a driver. Let $U_d := \nu_d - C^d_{tt'}$ be a driver's *base utility*. Let $\tilde{U}_d := U_d + \rho_d \sum_{r\in S_d} U_r$ be a driver's *altruistic utility* where ρ_d is the altruistic factor of the driver d and S_d is the subset of riders matched to d. If driver d is not matched, then they incur a cost of $c^d_{trl} \cdot (\mathtt{time}_d)$ and their (base or altruistic) utility is $U_d = \nu_d - c^d_{trl} \cdot \mathtt{time}_d$.

Proposition 1. *The cost minimization problem is equivalent to maximizing the sum of base utilities.*

The idea is that $\sum_{r\in\mathcal{R}} \nu_r + \sum_{d\in\mathcal{D}} \nu_d$ is independent of π, so utility maximization problem becomes equivalent to cost minimization. However, if we consider altruistic utility the problem is no longer equivalent. Therefore, we only focus on the cost minimization and social welfare maximization problems, where *social welfare* is defined to be $\sum_{r\in\mathcal{R}} U_r + \sum_{d\in\mathcal{D}} \tilde{U}_d$. Throughout the paper, we focus more on the cost-minimization problem; however, utilities are used to define stability and individual rationality and all algorithms are can be easily extended for social welfare maximization.

Fairness Model

Many P2P systems are long-term systems; therefore, we consider probabilistic matching where the probability of choosing a matching can be seen as a frequency of the matching over different days. Let p^ℓ be the probability that we choose matching $M^\ell \in \mathcal{M}$. Let φ_r^M be the probability of rider r being matched in the matching M. Let $\varphi_r := \sum_{M^\ell\in\mathcal{M}} p^\ell \cdot \varphi_r^{M^\ell}$

be the probability that rider r is matched in our probabilistic matching, acorss all possible matchings. Let $Cost(M^\ell)$ be the total cost of matching M^ℓ, i.e. $Cost(M^\ell) = \sum_{(d,S)\in M^\ell} c_{dS} + \sum_{r\notin M^\ell} \lambda_r$. Then the expected cost of a probabilistic matching is $\sum_{M^\ell \in \mathcal{M}} p^\ell \cdot Cost(M^\ell)$,

We formalize fairness by maximizing the lowest probability of matching, across all riders in the system. In other words we want to maximize $\min_{r\in\mathcal{R}} \varphi_r$. Note that, if there exists a rider r that cannot be feasibly matchec, then the value is always 0. Thus we only focus on riders that can be matched to a driver. Let now assume that $\mathcal{R}$ be the set of feasible riders.

Definition 1. *A probabilistic matching $\mathcal{M}$ is θ-*fair *if $\varphi_r \geq \theta$ for all riders $r \in \mathcal{R}$.*

We look at two fairness problems, namely computing cost minimizing solution subject to θ-fairness, and computing the fairest solution, i.e. maximum possible θ.

We also quantify the price of fairness, denoted by PoF. PoF is defined as the loss in the system's cost when fairness considerations are included. If the cost-minimizing solution already maximizes fairness, then PoF $= 1$, otherwise PoF increases as the loss increases.

Definition 2. PoF $= \max_{\mathcal{M}^F} \frac{Cost(\mathcal{M}^F)}{Cost(\mathcal{M}^*)}$, *where $\mathcal{M}^*$ are the cost-minimizing (probabilistic) matchings, and $\mathcal{M}^F$ is the fairest (probabilistic) matching.*

Stability Model

We now turn to stability. Note that our fairness definition focuses at the rider level. But forcing a driver to satisfy riders—for fairness considerations—may result in unstable outcomes. We first define stability at the individual-level.

Definition 3. *A user is* individually rational *(IR) if he/she does not get worse utility by participating in the matching system. In other words, the rider is IR if $U_r \geq \nu_r - \lambda_r$, and the driver is IR if $\tilde{U}_d \geq \nu_d - c^d_{trl} \cdot \mathtt{time}_d$.*

We extend the idea of IR to the group level by considering ex-post stable matching. We want to ensure there's no group of users that can benefit by deviating from participating. The traditional stable matching definition is defined by preference [Gale and Shapley, 1962]. We can implicitly define preference using the utility function.

Definition 4. *We say $(d,S) \succ_i (d',S')$ if $u^i_{dS} > u^i_{d'S'}$ for $i \in (\{d\}\cup S)\cap(\{d'\}\cup S')$, where u^i_{dS} is i's utility in matching (d,S).*

Definition 5. *We say (d,S) is a* blocking pair *of M if (d,S) is currently not matched in M but $(d,S) \succ_d (d, S^M_d)$ and $(d,S) \succ_r (d^M_r, S^M_r)$ for all $r \in S$.*

Definition 6. *We say a matching M is* stable *if there is no blocking pair (d,S).*

Note that the IR constraint is a special case of stability when the blocking pair is $(d,\emptyset)$ and $(null, r)$ for all drivers $d \in \mathcal{D}$ and all riders $r \in \mathcal{R}$.

Definition 7. *We say a probabilistic matching is* ex-post stable *if for any possible outcome of our probabilistic matching, the matching is stable.*

The price of stability (PoS) is defined similar to PoF.

Definition 8. PoS $= \max_{\mathcal{M}^S} \frac{Cost(\mathcal{M}^S)}{Cost(\mathcal{M}^*)}$, *where $\mathcal{M}^*$ are the cost-minimizing (probabilistic) matchings, and $\mathcal{M}^S$ is a stable (probabilistic) matching.*

A (probabilistic) matching that is fair and stable simultaneously may not exist. To resolve this concern, one may provide *external incentives* with a total budget of B to enhance stability. Practically, these external incentives may take the form of coupons provided by the platform to the users. We, therefore, introduce the notion of *B-stable matching*.

Definition 9. *A matching M is B-*stable *if $\exists \beta \in \mathbb{R}^{|\mathcal{D}\cup\mathcal{R}|}_{\geq 0}$ where $\sum_{i\in\mathcal{D}\cup\mathcal{R}} \beta_i \leq B$ and for all (d,S) pairs that are currently not matched in M, there exists a user $i \in S \cup \{d\}$ such that*

$$\beta_i + u^i_{d^M_i S^M_i} + \varepsilon \geq u^i_{dS}$$

(Recall u^i_{dS} is i's utility in the matching (d,S)).

5 Efficiency-Fairness-Stability Trade-offs

We now extend the algorithm from section 3.2, to incorporate fairness and stability. We also provide theoretical results on the trade-offs between efficiency, fairness, and stability.

5.1 Fairness

We provide an LP to compute a θ-fair probabilistic matching. Let $M^1, ..., M^\eta$ be all possible feasible matchings. Let m be the indicator function such that $m^\ell_i = 1$ if user $i \in M^\ell$. Let variable p^ℓ be the probability of choosing the matching M^ℓ. Let θ be the threshold probability. Recall $Cost(M^\ell)$ is the cost of the matching defined by $\sum_{(d,S)\in M^\ell} c_{dS} + \sum_{r\notin M^\ell} \lambda_r$. Then the following LP computes min-cost probabilistic matching subject θ-fairness.

$$\min_p \quad \sum_{\ell\in[\eta]} Cost(M^\ell) p^\ell \tag{2}$$

$$\text{s.t.} \quad \sum_{\ell\in[\eta]} m^\ell_i p^\ell \geq \theta \qquad \forall i \in \mathcal{R} \tag{3}$$

$$\sum_{\ell\in[\eta]} p^\ell = 1 \tag{4}$$

$$p^\ell \geq 0 \qquad \forall \ell \in [\eta] \tag{5}$$

Note η can be exponentially large. Thus instead of enumerating all M^1,...,M^η from the beginning, we follow the standard column generation method and incrementally add matchings one by one. In each iteration, we solve the primal LP with a subset of matchings and obtain dual variables w of constraint (3) and α of constraint (4). Then we solve the slave problem to decide the matching to be added to the primal. We aim to find a matching that maximize M^ℓ, s.t. $\sum_i m^\ell_i w_i + \alpha - Cost(M^\ell)$, which is equivalent to minimizing $Cost(M^\ell) - \sum_{i\in M^\ell} w_i$. It can be found by replacing c_{dS} with $c_{dS} - \sum_{i\in S\cup\{d\}} w_i$ in the matching MILP.

One missing piece is to find an initial feasible matching to bootstrap the column generation, which can be done solving the following LP, again through column generation. Note that $\theta = 0$ is a feasible solution for the LP (6), thus there is a trivial feasible solution to bootstrap the column generation for it. Denote the optimal solution for the LP (6) by p^0 and θ^0. Then θ^0 is the maximum level of fairness that any probabilistic matching can achieve. Therefore p^0 provides a feasible matching to the LP (2) or LP (2) is infeasible.

$$\max_{p,\theta} \quad \theta \tag{6}$$

$$\text{s.t.} \quad (3) - (5) \tag{7}$$

Proposition 2. *The Pareto frontiers characterizing the trade-off between fairness (θ) and efficiency ($Cost$) are piecewise linear in P2P ride-sharing problem.*

The proof follows from global sensitivity analysis from [Bertsimas and Tsitsiklis, 1997]. However, the Pareto frontier may contain an exponential number of points. We also present an algorithm that finds a δ-approximate Pareto frontier by computing $\frac{1}{\delta} + 1$ points ($\delta = 0$ achieves the exact Pareto frontier). We defer the algorithm and the definitions to the Appendix.

Proposition 3. *There exist problem instances where* $\mathsf{PoF} = C$ *for any* $C \geq 1$.

Consider the following system with two riders r_1, r_2 with $\lambda_{r_1} = \lambda_{r_2} = \varepsilon$, and one driver d. The driver suffers cost of $1 - \varepsilon$ when only satisfying r_1, and suffers cost of C when he satisfies r_1 and r_2. Cost minimizing solution is to only satisfy r_1 and obtain total cost of 1. However, fairest solution of satisfying all riders result in cost of C for any $C \geq 1$.

5.2 Stability

Additional pruning methods are added to ensure IR. We add the IR constraints to the TripMILP to prune more infeasible (d, S) pairs. Details are deferred to the Appendix.

For stability, we define the following constraints for all (d, S) to ensure that the matching is stable. In the algorithm, we pre-compute $\succ_i$ for all users $i \in \mathcal{D} \cup \mathcal{R}$, then add following constraints to the matching MILP.

$$\sum_{\substack{(d,S') \\ \succ_d (d,S)}} x_{dS'} + \sum_{r \in S} \sum_{\substack{(d',S') \\ \succ_r (d,S)}} x_{d'S'} \quad + \sum_{\substack{r \in S: \\ \emptyset \succ_r (d,S)}} y_r + x_{dS} \geq 1$$

Proposition 4. *There exist problem instances where* $\mathsf{PoS} = \Omega(\lambda \cdot |\mathcal{R}|)$*, where λ is the unsatisfied cost of a user.*

The high-level idea is that one highly-demanded driver might prefer to satisfy only one rider, which may incur a huge unsatisfactory cost. But the cost-minimizing solution always matches the driver with more riders to avoid cost surge. (This also shows that the social welfare maximizing solution may not be stable.) Furthermore, θ-fair solution is not ex-post stable, $\forall\ \theta > 0$. The fairness constraint matches the driver to a different rider with positive probability, causing instability. The detailed proof is deferred to the Appendix.

For achieving a B-stable matching, we add more constraints to the matching MILP. We introduce new variables β_i for all $i \in \mathcal{D} \cup \mathcal{R}$. β_i indicates how much external incentive is allocated to user i. Let $S^+ := S \cup \{d\}$ be the set of users in (d, S). We also introduce variables μ^i_{dS} for all $i \in S^+$ and $(d, S) \in \mathcal{E}$. $\mu^i_{dS} = 1$ if user i β_i-prefers his current matching over (d, S). The matching is stable with our external incentive if for all feasible pairs (d, S), there exists user $i \in S^+$ such that i β_i-prefers his/her current matching over (d, S). Let $\mathcal{E}_i$ be set of feasible pairs $(d, S) \in \mathcal{E}$ such that $i \in S^+$. Then, adding the following constraints to the matching MILP to give a B-stable matching.

$$\beta_i + \textstyle\sum_{(d',S') \in \mathcal{E}_i} u^i_{d'S'} x_{d'S'} + u^i_{\emptyset} y_i \geq u^i_{dS} \cdot \mu^i_{dS}$$

$$\forall (d, S) \in \mathcal{E} \quad \forall i \in S^+$$

$$\textstyle\sum_{i \in d \cup S} \mu^i_{dS} \geq 1 \quad \forall (d, S) \in \mathcal{E}$$

$$\textstyle\sum_{i \in \mathcal{D} \cup \mathcal{R}} \beta_i \leq B$$

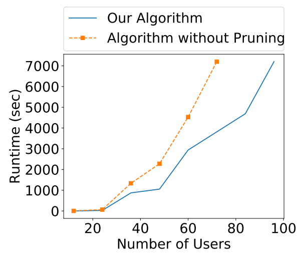
6 Experiments

In this section, we experimentally test our model and algorithms. All MILPs and LPs are solved by Gurobi Optimizer on a machine with 3.1 GHz Xeon E5 desktop with 16 GB RAM. All graphs are averaged over 10 instances except for special case experiments.

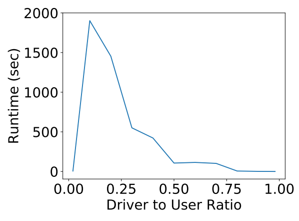
We used problem instances that simulate a typical neighborhood in a morning rush hour. We used a 50×50 grid graph. `time` is defined to be the Euclidean distance rounded up to an integral value. Users are drawn based on 5 different types. Different types of users are described in Figure 3. Travel costs (c^i_{trl}) are set to 3 per minute, and deviation costs (c^i_{dev}) are set to 1 per minute. All drivers have a capacity of $k_d = 4$, and altruistic factor of $\rho_d = 1.2$. The value of the user i is set to $\nu_i = c^i_{trl} \cdot \mathtt{time}_i \cdot U[1, 2.5]$, where $U[a, b]$ denotes uniform distribution over interval $[a, b]$. Most of the experiments are done in this setting with varying changes as we describe below.

Figure 1 compare the performance of the algorithms in different settings. Figure 1(a) show the effects of pruning. The experiments were run without IR constraint, and a driver to user ratio of 1:4. The results show that our scheduling algorithm can scale up to 80 users within 2 hours. Also, our pruning method decreases runtime by nearly an hour with 60 users. Figure 1(b) shows how runtime changes as the ratio changes. The experiments were done with 50 agents as we vary the driver to user ratio. Results show that the runtime is largest with a ratio of 1:5. The ratio corresponds to the time when drivers can potentially pick all riders up with full capacity. Figure 1(c) shows the runtime when we add the IR constraint and stability constraint. The results show that adding an IR constraint and stability constraint reduces the runtime significantly. Adding such constraints reduces the number of feasible matchings; therefore, the algorithm can scale up to a significantly larger number of users (110 users within 30 minutes). This suggests that considering IR and stability enhances both the practical benefits of the solution and the scalability of the algorithm. Figure 1(d) shows the runtime for our fairness algorithm. Even though the fairness algorithm uses our baseline algorithm, the runtime did not increase significantly. The fairness algorithm can still scale up to 140 users within 2 hours. Results show that the column generation method does not increase the runtime significantly.

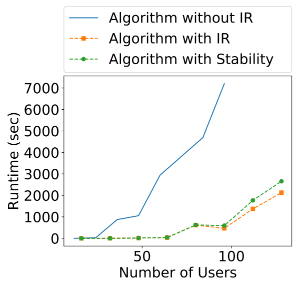
In Figure 2, we look at the trade-offs between altruism, fairness, stability, and efficiency. Figure 2(a) shows the effects of the altruistic factor (ρ) on total cost and satisfied rider utilities. As drivers become more altruistic, drivers are willing to satisfy more socially beneficial riders. Therefore, as the altruistic factor increases, the total cost decreases and satisfied rider utilities increase. Furthermore, even small altruistic factors of 0.5 can achieve a solution with a very low cost.



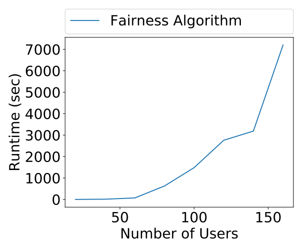
(a) Runtime Pruning



(b) Runtime Ratio

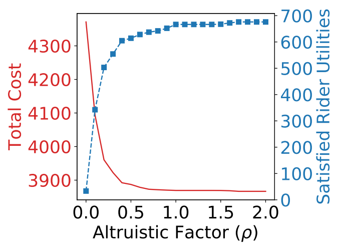


(c) Runtime IR and Stability

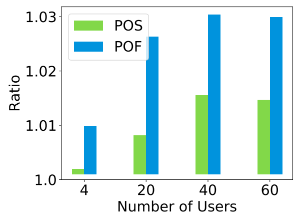


(d) Runtime Fairness

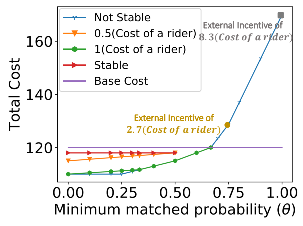
Figure 1: Scalability experiments of different algorithms and setting.



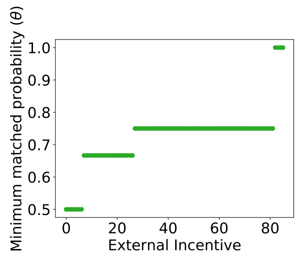
(a) Affect of Altruism



(b) PoS and PoF



(c) Pareto-Frontier with different external incentive



(d) Benefit of external incentive

Figure 2: Affects on efficiency, fairness, and stability

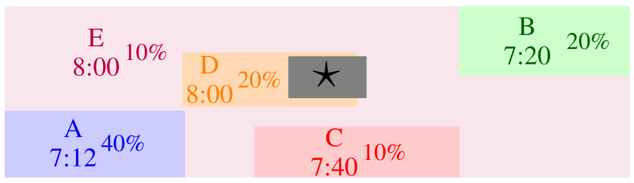


Figure 3: The figure describes our experimental setting. The gray ⋆ region correspond to the neighborhood, i.e. every user's origin. Each region correspond to different type of user's destination, percentage, and latest departure time. For example, There are 40% of type A users. Each type A user i's destination is drawn from left bottom corner, τ_i^e drawn from [7:00,7:12], $\tau_i^\star$ drawn from $[\tau_i^e$,7:12], $\tau_i^l =$ 7:12+$\texttt{time}_i$. Where 7:00 is the earliest possible departure time of all users, and $\max_{u,v\in V} \texttt{time}(u, v)$ is 1 hour.

Figure 2(b) shows PoS and PoF with different numbers of users (4, 20, 40, 60 with driver to user ratio of 1:4). Even though we showed theoretically that PoS and PoF can be arbitrarily large, in practice incorporating fairness and stability considerations results in added costs of only 3% and 2%, respectively.

Figure 2(c) and 2(d) show the trade-off between fairness, stability and cost efficiency. We constructed a special instance with big PoF to better understand the trade-off. The instance contains 1 very flexible driver and 4 riders. The cost of satisfying a rider is set to 10 (both rider and driver would suffer a cost of 10). Furthermore, the driver may suffer extra travel costs between satisfying riders. The base cost corresponds to the costs of the driver and riders not participating in the matching system. Figure 2(c) shows that enforcing higher fairness probabilities could result in a total cost in excess of the base cost. The example shows that enforcing fairness probability of at least 2/3 is more costly than leaving all users unmatched. Furthermore, higher external incentives decrease the price of stability and enable the stable matching to achieve higher fairness probability. More specifically, Figure 2(c) shows the stable solution can only achieve fairness probability of 0.5. However, providing external incentives of 10, 27, 83 can achieve fairness levels of 2/3, 3/4, 1, respectively. These results show that even limited external incentives (equal to the cost of satisfying one rider) can achieve a reasonable fairness guarantee (2/3).

7 Conclusion

Unlike prior work, this paper focuses on user-centric matching in P2P ride-sharing. (a) We propose a model that incorporates users' preferred times; (b) we incorporate fairness and stability in our P2P ride-sharing model; (c) we provide theoretical and experimental results showing trade-offs between fairness, stability, and cost efficiency. While, in theory, the price of fairness and the price of stability can be arbitrarily large, experimental results show that fair and stable solutions can be obtained through moderate efficiency losses and in reasonable computational times.

Finally, we point out two potential future research directions. The first one is to design mechanisms to ensure incentive-compatibility, i.e., ensuring that there are no misreported values or misreported preferred times associated with uncoordinated incentives. Another direction is to study the trade-offs between fairness and stability in a broader area. Although the notion of fairness has been studied in many different contexts, there is little work investigating both fairness and stability in complex systems. Ultimately, incorporating both fairness and stability in analytic models can achieve more socially beneficial solutions and practical results.

References

[Alonso-Mora *et al.*, 2017] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3):462–467, 2017.

[Anshelevich *et al.*, 2008] Elliot Anshelevich, Anirban Dasgupta, Jon Kleinberg, Eva Tardos, Tom Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

[Bei and Zhang, 2018] Xiaohui Bei and Shengyu Zhang. Algorithms for trip-vehicle assignment in ride-sharing. In *Proc. AAAI*, 2018.

[Berbeglia *et al.*, 2007] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, 2007.

[Bertsimas and Tsitsiklis, 1997] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[Bertsimas *et al.*, 2011] Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. The price of fairness. *Operations research*, 59(1):17–31, 2011.

[Bertsimas *et al.*, 2019] Dimitris Bertsimas, Patrick Jaillet, and Sébastien Martin. Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research*, 2019.

[Bimpikis *et al.*, 2016] Kostas Bimpikis, Ozan Candogan, and Daniela Saban. Spatial pricing in ride-sharing networks. 2016.

[Bistaffa *et al.*, 2015] Filippo Bistaffa, Alessandro Farinelli, Georgios Chalkiadakis, and Sarvapali D Ramchurn. Recommending fair payments for large-scale social ridesharing. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 139–146. ACM, 2015.

[Bogomolnaia and Moulin, 2001] Anna Bogomolnaia and Hervé Moulin. A new solution to the random assignment problem. *Journal of Economic theory*, 100(2):295–328, 2001.

[Cohen and Wajc, 2018] Ilan Reuven Cohen and David Wajc. Randomized online matching in regular graphs. Working paper, 2018.

[Cordeau *et al.*, 2007] Jean-François Cordeau, Gilbert Laporte, Martin WP Savelsbergh, and Daniele Vigo. Vehicle routing. *Handbooks in operations research and management science*, 14:367–428, 2007.

[Cordeau, 2006] Jean-François Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006.

[Furuhata *et al.*, 2013] Masabumi Furuhata, Maged Dessouky, Fernando Ordóñez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.

[Gale and Shapley, 1962] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[Iwama and Miyazaki, 2008] Kazuo Iwama and Shuichi Miyazaki. A survey of the stable marriage problem and its variants. In *International Conference on Informatics Education and Research for Knowledge-Circulating Society*, pages 131–136. IEEE, 2008.

[Liu and Knightly, 2003] Yonghe Liu and Edward Knightly. Opportunistic fair scheduling over multiple wireless channels. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1106–1115. IEEE, 2003.

[Manlove *et al.*, 2002] David F Manlove, Robert W Irving, Kazuo Iwama, Shuichi Miyazaki, and Yasufumi Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276(1-2):261–279, 2002.

[Masoud and Jayakrishnan, 2017] Neda Masoud and R Jayakrishnan. A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. *Transportation Research Part B: Methodological*, 106:218–236, 2017.

[Ozkan and Ward, 2017] Erhun Ozkan and Amy Ward. Dynamic matching for real-time ridesharing. 2017.

[Parragh *et al.*, 2010] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 37(6):1129–1138, 2010.

[Santi *et al.*, 2014] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, and Carlo Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37):13290–13294, 2014.

[Wang *et al.*, 2017] Xing Wang, Niels Agatz, and Alan Erera. Stable matching for dynamic ride-sharing systems. *Transportation Science*, 2017.

A Notation Table

Notation	Description
T	Time horizon
$\mathcal{R}$	Set of riders
$\mathcal{D}$	Set of drivers
$o_{r/d}$, $q_{r/d}$	Origin, Destination of a rider/driver.
$W_{r/d}$	Window of a rider/driver.
$\tau^e_{r/d}$	Earliest departure time for a rider/driver.
$\tau^l_{r/d}$	Latest arrival time for a rider/driver.
$\tau^\star_{r/d}$	Preferred departure time for a rider/driver.
$\Delta_{r/d}$	Maximum detour time for a rider/driver.
$\nu_{r/d}$	Value gain by complete trip.
k_d	Capacity of the driver d
ρ_d	Altruistic factor for the driver d
$\mathcal{S} = 2^{\mathcal{R}}$	Set of all possible subset of riders.
$\mathcal{E}$	Set of all feasible (d, S) paths
c_{dS}	Cost of driver d satisfying $S \in \mathcal{S}$
λ_r	Unsatisfied cost of rider r
$\mathtt{time}(u, v)$	Distance from node u to node v.
$\mathtt{time}_r$	$\mathtt{time}(o_r, q_r)$; Distance from o_r to q_r.
$C^r_{tt'}$	Total cost for the rider r with picked-up time t and dropped-off time t'.
$c^{r/d}_{dev}$, $c^{r/d}_{trl}$	Deviation/travel cost per unit time.
$\Pi/\mathcal{M}$	set of all schedules/matchings.
π	schedule; Ordered (stop,time) pairs
M	matching; assignment of $\mathcal{D}$ to $\mathcal{S}$
ω	route; ordered stops
$S^{\pi/M}_d$	Set of riders that driver d is matched in π/M.
$d^{\pi/M}_r$	The driver that rider r is matched in π/M.
$S^{\pi/M}_r$	The set rider r is matched in π/M.
$N(\omega)$	Possible next stop of the route
ω_{end}	The last node in the route
$\omega + u$	Add node u at the end of ω
$r(v)$	rider that the node v corresponds to
$\mathtt{nxt}^\omega_v$	The node that's next to v in ω.
$\mathtt{Pas}_\omega(v)$	The set of riders in the car at node $v \in \omega$

B Omitted Algorithms

In this section we will describe omitted algorithms.

B.1 Construction of RTV-graph

RTV-graph construction requires construction of RV-graph, decomposition then the construction of RTV-graph.

RV-graph:

We first compute RV-graph as following: For all pairs of (d, r) we check whether the driver d can pick up the rider r and drop off within both of their time windows. If they are feasible we add an edge between d and r. Furthermore, for each pair (r_1, r_2), we check whether any virtual driver can pick up both riders and drop them off within their time windows. If they are feasible we add an edge between r_1 and r_2.

Decomposition:

Then we run the decomposition algorithm, we decompose the instance into several smaller sub-problems. In other words, we try to find a subset of drivers $D \subseteq \mathcal{D}$, such that all feasible riders of D is not feasible with any other driver $d' \in \mathcal{D} \setminus D$. If the problem is decomposable, we solve each sub-problem separately/independently. The decomposition allows our algorithm to run in parallel.

RTV-graph

RTV graphs has 3 types of nodes, we have driver nodes for all driver $d \in \mathcal{D}$, trip nodes for all (feasible) subset of riders $S \subseteq \mathcal{R}$, and rider nodes for all rider $r \in \mathcal{R}$.

We first add an empty trip. We have edge between d and empty trip for all $d \in \mathcal{D}$. Then we add trip S of size 1. We have edge between r and S if $r \in S$. And we have an edge between S and d if (d, S) is feasible. Let $\mathcal{S}_1$ be set of the feasible trip of size 1. Similarly, we will maintain a feasible trip of size i as $\mathcal{S}_i$.

For trip size $i \in \{2, ..., |\mathcal{R}|\}$, for each trip $S' \in \mathcal{S}_{i-1}$, add a rider r from feasible rider $\mathcal{S}_1$. Let $S = S' \cup \{r\}$. If $|S| < i$ and any subset of S with size $i - 1$ is not in $\mathcal{S}_{i-1}$ then continue to next rider in $\mathcal{S}_1$. Otherwise compute $\mathsf{TripCost}(d, S)$

Algorithm 1 Construction of RV-graph

```
1: $v_{RV} \leftarrow \mathcal{D} \cup \mathcal{R}$
2: $E_{RV} \leftarrow \emptyset$
3: for each rider $r \in \mathcal{R}$ do
4:     for each driver $d \in \mathcal{D}$ do
5:         if $\mathsf{TripCost}(d, \{r\})$ is feasible then
6:             $E_{RV} \leftarrow E_{RV} \cup (d, r)$
7:             $w((d, r)) \leftarrow \mathsf{TripCost}(d, \{r\})$
8:     for each rider $r' \in \mathcal{R}$ do
9:         $d \leftarrow (o = o_r, q = q_r, W = W_r, k = 1)$
10:        if $\mathsf{TripCost}(d, \{r'\})$ is feasible then
11:            $E_{RV} \leftarrow E_{RV} \cup (r, r')$
12:        else
13:            $d \leftarrow (o = o_{r'}, q = q_{r'}, W = W_{r'}, k = 1)$
14:            if $\mathsf{TripCost}(d, \{r'\})$ is feasible then
15:                $E_{RV} \leftarrow E_{RV} \cup (r, r')$
16: return RV-graph $= (v_{RV}, E_{RV})$.
```

B.2 $\mathsf{TripCost}(d, S)$

Now we will describe $\mathsf{TripCost}(d, S)$. It contains Tree-Search part the MILP part.

Let $N(\omega) := \{o_r : o_r \notin \omega \cap r \in S\} \cup \{q_r : o_r \in \omega \cap q_r \notin \omega \cap r \in S\}$ be the possible next stop of the route. Let ω_{end} be the last node in the route. Let $\omega + v$ be the function that adds v at the end of ω.

Tree Search

TripMILP

Let $z_{vt} = 1$ if the driver goes to vth node in ω at time t. Let $g_{rtt'} = 1$ if the user r is picked up at time t and dropped off at time t'. Let $c_{rtt'} := c^r_{trl}(t' - t)$ be the corresponding travel cost, where ω_r is the partial route start from o_r and end at q_r. Let $c_{vt} := c^{r(v)}_{dev}|t - \tau^\star_{r(v)}|$ be the corresponding deviation cost.

Dynamic Program for $\mathsf{TripCost}(d, S)$

$$\mathcal{Q}(v,t) = \quad c_{dev}^{r(v)}\left(\left|t-\tau_{r(v)}^{\star}\right|\right) + \min_{t'\in\left[t_{\omega}^{+},\tau_{\mathtt{nxt}_v^{\omega}}^{l}\right]}\left\{c_{trl}^{d}\cdot(t'-t_{\omega}^{+}) + \sum_{r'\in\mathtt{Pas}_{\omega}(v)}(1+\rho_d)c_{trl}^{r'}\cdot(t'-t_{\omega}^{+}) + \mathcal{Q}(\mathtt{nxt}_v^{\omega},t')\right\} \tag{8}$$

(where $t_{\omega}^{+} := t + \mathtt{time}(v, \mathtt{nxt}_v^{\omega})$)

Algorithm 2 Construction of RTV-graph

```
1:  $E \leftarrow \emptyset, V \leftarrow \mathcal{D} \cup \mathcal{R} \cup \mathcal{S}$
2:  **for** each driver $d \in \mathcal{D}$ **do**
3:      $\mathcal{S}_i = \emptyset\ \forall i \in \{1, ..., |\mathcal{R}|\}$
4:      [Add trips of size one]
5:      **for** $e = (d, r) \in E_{RV}$ **do**
6:          $\mathcal{S}_1 \leftarrow S = \{r\}$.
7:          $E \leftarrow E \cup \{(r, S), (S, d)\}$.
8:      [Add trips of size two]
9:      **for** all $\{r_1\}$,$\{r_2\} \in \mathcal{S}_1$ and $(r_1, r_2) \in E_{RV}$ **do**
10:         **if** $\mathsf{TripCost}(d, \{r_1, r_2\})$ is feasible **then**
11:             $\mathcal{S}_2 \leftarrow S = \{r_1, r_2\}$.
12:             $E \leftarrow E \cup \{(r, S), (S, d)\}$
13:             $w((S, d)) \leftarrow \mathsf{TripCost}(d, \{r_1, r_2\})$
14:     [Add trips of size $i$]
15:     **for** $i \in \{3, ..., |\mathcal{R}|\}$ **do**
16:         **for** all $S_1 \in \mathcal{S}_{i-1}$ and $\{r\} \in \mathcal{S}_1$ **do**
17:             $S \leftarrow S_1 \cup \{r\}$
18:             **if** $|S| < i$ or $\exists S_{i-1} \subseteq S$ of size $i$-1 s.t. $S_{i-1} \notin \mathcal{S}_{i-1}$ **then**
19:                 **continue**
20:             **if** $\mathsf{TripCost}(d, S)$ is feasible **then**
21:                 $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup S$
22:                 $E \leftarrow E \cup \{(r_i, S)\}\ \forall r_i \in S$.
23:                 $E \leftarrow E \cup \{(S, d)\}$
24:                 $w((S, d)) \leftarrow \mathsf{TripCost}(d, S)$
25:     $\mathcal{S}_d \leftarrow \cup_{i\in[|\mathcal{R}|]}\mathcal{T}_i$
26: **return** $G = (V, E)$.
```

Algorithm 3 $\mathsf{Tree\text{-}Search}(d, S, \omega)$

```
1:  **if** $w(\omega) > c_{dS}^{*}$ **then**
2:      **return** False
3:  [$\sigma$ is at the leaf node]
4:  **if** $|\omega| = 2 \cdot |S| + 1$ **then**
5:      $\omega' \leftarrow \omega + q_d$
6:      $\tau_{\omega'} \leftarrow \tau_{\omega} + \mathtt{time}(\omega_{end}, v)$
7:      **if** $\tau_{\omega'} > \tau_{r(\omega_{end})}^{l}$ **then**
8:          **return** False
9:      $w(\omega') = \mathsf{TripMILP}(\omega')$
10:     **if** $c_{dS}^{*} > w(\omega')$ **then**
11:         $c_{dS}^{*} \leftarrow w(\omega')$
12:         **return** $c_{dS}^{*}$
13: [$\sigma$ is at the intermediate node]
14: **for** $v \in N(\omega)$ **do**
15:     $\omega' \leftarrow \omega + v$
16:     **if** $\mathtt{Pas}_{\omega'}(v) > k_d$ **then**
17:         **continue**
18:     $\tau_{\omega'} \leftarrow \tau_{\omega} + \mathtt{time}(\omega_{end}, v)$
19:     **if** $\tau_{\omega'} > \tau_{r(\omega_{end})}^{l}$ **then**
20:         **continue**
21:     $w(\omega') = w(\omega) + c_{trl}^{d} \cdot \mathtt{time}(\omega_{end}, v) + \sum_{r\in\mathtt{Pas}_{\omega}(v)}(1 + \rho_d') \cdot c_{trl}^{r} \cdot \mathtt{time}(\omega_{end}, v)$
22:     $w(\omega') = \mathsf{Tree\text{-}Search}(d, S, \omega')$
23:     **if** $c_{dS}^{*} > w(\omega')$ **then**
24:         $c_{dS}^{*} \leftarrow w(\omega')$
25:         **return** $c_{dS}^{*}$
26: **return** $c_{dS}^{*}$
```

Let $T(v)$ be the set of feasible time for the node v. Let $v(r)$ be the mapping of an user r to the corresponding origin node, i.e. o_r. Let $v'(r)$ be the mapping of an user r to correspond destination node, i.e. q_r. Let $t^+_\omega := t + \texttt{time}(v, \texttt{nxt}^\omega_v)$. We define same variables for driver nodes as well.

- Let $S^+ = S \cup \{d\}$ be the set of all users in the route.
- Let $C^d(z, g)$ be the driver d's cost with variables z and g, i.e. $\sum_{v\in\{o_d,q_d\}} \sum_{t\in T(v)} c_{vt}z_{vt} + \sum_{t\in T(o_d)} \sum_{t'\in T(q_d)} c_{dtt'}g_{dtt'}$.
- Let $C^r(z, g)$ be the rider r's cost, i.e. $\sum_{v\in\omega\setminus d} \sum_{t\in T(v)} c_{vt}z_{vt} + \sum_{r\in S} \sum_{t\in T(o_r)} \sum_{t'\in T(q_r)} c_{rtt'}g_{rtt'}$.
- Let $u^d(z, g)$ be the driver d's utility, i.e. $\rho_d \cdot \sum_{r\in S}(\nu_r - \sum_{t\in T(o_r)} c_{o_r t}z_{o_r t} - \sum_{t\in T(o_r)} \sum_{t'\in T(q_r)} c_{rtt'}g_{rtt'}) + \nu_d - \sum_{t\in T(o_d)} c_{o_d t}z_{o_d t} - \sum_{t\in T(o_d)} \sum_{t'\in T(q_d)} c_{dtt'}g_{dtt'}$.
- Let $u^r(z, g)$ be the rider r's utility, i.e. $\nu_r - \sum_{t\in T(o_r)} c_{o_r t}z_{o_r t} - \sum_{t\in T(o_r)} \sum_{t'\in T(q_r)} c_{rtt'}g_{rtt'}$
- Let Υ be set of tuples (r, t, t') that are not feasible due to maximum detour constraint, i.e. $(r, t, t') \in \Upsilon$ for all $r \in S^+$, $t \in T(o_r)$, and $t' \in [t + \Delta_r + \texttt{time}_r, \tau^l_r]$

If we finish one tree branch, we get the minimum cost of the branch. If this computed value is less than cost of any node in the tree, we can safely prune that node, because lower bound of the branch is bigger than the value we computed already. In fact, we can stop the algorithm when min weight node is bigger than minimum cost we computed. We can continue the process until we solve or prune all tree branches. c_{dS} is set to minimum cost among all tree branches.

$$\min_{z,g} \quad C^d(z, g) + (1 + \rho'_d) \cdot \sum_{r\in S} C^r(z, g) \tag{9}$$

$$\text{s.t.} \sum_{t\in T(v)} z_{vt} = 1 \qquad \forall v \in \omega \tag{10}$$

$$g_{rtt'} \le z_{o_r t} \qquad \forall r \in S^+, t \in T(o_r), t' \in T(q_r) \tag{11}$$

$$g_{rtt'} \le z_{q_r t} \qquad \forall r \in S^+, t \in T(o_r), t' \in T(q_r) \tag{12}$$

$$\sum_{t\in T(o_r)} \sum_{t'\in T(q_r)} g_{rtt'} = 1 \qquad \forall r \in S^+ \tag{13}$$

$$z_{vt} \le \sum_{q\in[t^+_\omega, \tau^l_{\texttt{nxt}^\omega_v}]} z_{\texttt{nxt}^\omega_v q} \qquad \forall v \in \omega, t \in T(v) \tag{14}$$

$$u^r(z, g) \ge \nu_r - \lambda_r \qquad \forall r \in S \tag{15}$$

$$u^d(z, g) \ge \nu_d - c^d_{trl} \cdot \texttt{time}_d \tag{16}$$

$$g_{rtt'} = 0 \qquad \forall (r, t, t') \in \Upsilon \tag{17}$$

$$z_{vt}, g_{rtt'} \in \{0, 1\} \tag{18}$$

B.3 Dynamic Programming for $\mathsf{TripCost}(d, S)$

Dynamic programming table consist of $\mathcal{Q}(v, t)$ for all $v \in \omega$ and $t \in W_d$, observe ω consist of $2 \cdot (d + |S|)$ nodes, one for origin and one for destination. The value of $\mathcal{Q}(v, t)$ is the minimum cost we can get by going to v (correspond to pickup or drop-off) at time t. Let $r(v)$ be the rider the node v correspond to. Let $\texttt{nxt}^\omega_v$ be the node that's next to v in ω. We say $v <_\omega v'$ if v appears before v' in ω. Let $\texttt{Pas}_\omega(v) := \{r : o_r <_\omega v \text{ and } v <_\omega q_r\}$ be the set of riders that are currently in the car when we arrive at node v. This set can be easily obtained by going through the route. Let τ^e_v be the earliest time for node v, i.e. if $v = o_r$ then $\tau^e_v = \tau^e_r$; if $v = q_r$ then $\tau^e_v = \tau^e_r + \texttt{time}_r$. τ^l_v is similarly defined as the latest time for node v. The value of $\mathcal{Q}(v, t)$ can be computed by following equation.

After solving all $\mathcal{Q}(v, t)$, the value $\mathcal{Q}(o_d, \tau^e_d)$ is the minimum deviation cost and travel cost following ω. Then the total cost would be *distance cost of the branch* + $\mathcal{Q}(o_d, \tau^e_d)$

Algorithm 4 $\mathsf{TripDP}(\omega)$

```
1: $v \leftarrow \omega_{end}$ (i.e. $q_d$)
2: for $t \in [\tau^e_d + \texttt{time}_d, \tau^l_d + \texttt{time}_d]$ do
3:     $\mathcal{Q}(v, t) = 0$
4: for $v \in \omega$ in inverse order do
5:     for $t \in [\tau^e_v, \tau^l_v]$ do
6:         $\mathcal{Q}(v, t) \leftarrow (8)$
7: return $\mathcal{Q}(o_d, \tau^e_d)$
```

We use dynamic programming to obtain lower-bound solution of the leaf node. Solving dynamic programming before the MILP can reduce number of times we compute MILP significantly.

B.4 Pruning Methods

Our solution approach might solve exponential number of MILPs and size of $\mathcal{S}$ can be exponentially large. To overcome such inefficiency we add pruning methods to speed-up our algorithm. Our main bottleneck is that our tree-search is exploring exponentially many leafs that are not optimal. Such phenomena happens when the first leaf node we found was far away fro optimal. Our pruning method try to predict a path that are promising to be an optimal solution. Therefore, the promising gives better upper-bound to reduce number of leaf nodes we search.

When we compute (d, S_k) for some set S_k of size k, we already computed (d, S_{k-1}) for all set S_{k-1} of size $k - 1$. Our first pruning method try to learn from (d, S_{k-1}) to predict promising path for S_k. We also predict a promising path from (d', S_k) for all d' that are similar to d. We say d and d' are similar if their L_∞-norm of origin and destination is within ε for small enough ε. In many P2P platform, there are many drivers with similar origins and destination; therefore, our pruning method can be very powerful in many real-world setting.

B.5 Matching MILP

Let $\Gamma_r \subseteq \mathcal{S}$ to be set of all subsets that r belongs to, i.e. $\Gamma_r = \{S \in \mathcal{S} | r \in S\}$.

$$\min_{x,y} \quad \sum_{d\in\mathcal{D}}\sum_{S\in\mathcal{S}} c_{dS}x_{dS} + \sum_{r\in\mathcal{R}} \lambda_r y_r \tag{19}$$

$$s.t. \quad \sum_{d\in\mathcal{D}}\sum_{S\in\Gamma_r} x_{dS} + y_r = 1, \qquad \forall r\in\mathcal{R} \tag{20}$$

$$\sum_{S\in\mathcal{S}} x_{dS} = 1 \qquad \forall d\in\mathcal{D} \tag{21}$$

$$x_{dS}\in\{0,1\} \qquad \forall d\in\mathcal{D}, S\in\mathcal{S} \tag{22}$$

$$y_r\in\{0,1\} \qquad \forall r\in\mathcal{R} \tag{23}$$

B.6 δ-approximate Pareto Frontier

We say a set of points $\mathcal{F}$ is a δ-approximate if for all points (c^*,θ^*) in actual Pareto frontier $\mathcal{F}^*$, we have a point $(c,\theta)\in\mathcal{F}$ such that 1) $\sqrt{(c-c^*)^2+(\theta-\theta^*)^2}\le\delta$, i.e. ℓ_2 norm is less than δ.

We can first compute the maximum θ using the LP (6). Let $\theta_{\max}$ be the maximum θ we can achieve. Then run the LP (2) for all $\theta=\delta\cdot i$ for all $\delta\cdot i\in[0,\theta_{\max}]$ and $i\in\mathbb{N}$.

Theorem 1. *The algorithm gives δ-approximate Pareto Frontier.*

This shows that any point between (c_1,θ_1) and $(c_2\theta_2)$ is also feasible. Since $\theta_1>\theta^*-\delta$ and $c^*\ge c_1$, the segment $(c^*,\theta^*-\delta)$-(c^*,θ^*) and (c_1,θ_1)-$(c_2\theta_2)$ must intersect at one point p. and the point p is feasible and within δ-ball around the point (c^*,θ^*); therefore, our algorithm δ-approximate any Pareto Frontier point.

C Omitted Proof

In this section we will provide omitted claims and proofs in the main paper.

C.1 Riders are automatically IR

Claim 1. *Riders are individually rational in cost-minimization solution.*

Proof. Suppose for a contradiction that a rider r is getting utility lower than $\nu_r-\lambda_r$ in the efficiency optimal schedule π^*. Then consider a schedule π' where r is not matched and everyone else is matched the same as π^*. In other words, we are picking up and dropping off all riders at the same time, but we are just not picking up the rider r (This schedule may contain unnecessary waiting). It is easy to see that π' is a feasible schedule. All stop, time pairs stay the same, so time window constraint satisfies. Drivers are picking up same or less riders, thus the capacity constraint satisfies. For each rider $r'\neq r$, he is experiencing exactly the same deviation and distance traveled, so his utility is the same. Utility for each driver that did not pick up r stays the same, because his schedule and the riders that he picks up stays the same. These holds for both welfare maximization and efficiency maximization.

We only need to consider the utility for the driver d who picked up r in π^*. For efficiency maximization, $U_d(\pi^*)=\nu_d-C^d_{tt'}$. recall we are considering the same schedule as π^*, i.e. the driver is going to exactly same stops at exactly same times. Therefore, he suffers exactly same deviation cost and travel cost, thus we have $U_d(\pi')=U_d(\pi^*)$, i.e. the driver is getting the same utility. Everyone except r is getting same utilities and r is getting better utility in π', thus π^* is not an efficiency optimal schedule. □

The claim is not true for social welfare maximization. Consider the following example with 1 driver d and 2 riders (r_1,r_2). All of their origin is at location a and all of their destination is at location b. Suppose $\texttt{time}(a,b)=5$, $\nu_d=\nu_{r_1}=\nu_{r_2}=10$, $\rho_d=1/2$, $W_d=[0,5]$, $\tau^\star_d=0$, $W_{r_1}=W_{r_2}=[0,6]$, $\tau^\star_{r_1}=\tau^\star_{r_2}=1$. All user's deviation cost and travel cost is 1 (i.e. $c_{dev}=c_{trl}=1$).

Then only feasible schedule for the driver is leaving origin at time 0, and arriving at time 5. If he chooses to pick both riders up, then the riders suffer the deviation cost of 1. Therefore, each rider's utility is $\nu_r-C^r_{tt'}=10-6=4$. Note $\nu_r-\lambda_r=5$, thus the schedule is not individually rational for the riders. The utility for the driver is $\nu_d-C^d_{tt'}+\rho_d\sum_{r\in S_d}U_r=10-5+\frac{1}{2}\sum_r 4=9$. Thus we get social welfare of $4+4+9=17$. Whereas, only individually rational solution is not picking up any rider which gives social welfare of $5+5+5=15$. This shows the riders may not be individually rational in social welfare maximizing solution.

If $\nu_r=\lambda_r\forall r\in\mathcal{R}$, then the claim holds for the SW maximization.

C.2 Proof of Proposition 5.1

Proof. Consider the following example with 1 driver and m riders. The driver d has very flexible window and capacity of 2. We have $m-2$ riders that has strict and narrow window so that satisfying them give cost of Q, and cannot be satisfied 2 of them simultaneously. More formally we have rider r_i with window $[i,i+m]$ and the $\texttt{time}_r=m$ for all $i\in[3,m]$. The driver can satisfy any of them with his flexible window , but cannot satisfy two of them simultaneously due to their strict window. We also have 2 more riders r_1 and r_2 that can be satisfied together, but if the driver satisfy r_1 only, then he gets total cost of ϵ. Whereas satisfying (r_1,r_2) together gives cost of Q.

First we will look at cost minimizing matching. The driver can just satisfy the rider r_1, and gets the cost of ϵ. However, in this matching all other riders are not satisfied even though they are feasible. Therefore, this matching is not fair for the riders.

The most fair solution is when the driver satisfies all riders with probability $\frac{1}{m-1}$. This matching can be obtained by satisfying $m-2$ riders with probability $\frac{1}{m-1}$ and satisfy (r_1,r_2) pair with probability $\frac{1}{m-1}$. This results in cost of Q; therefore, gives the PoF $=Q/\epsilon$. □

The example also holds for arbitrarily big PoF when we define the price based on utility model. Recall there exist M^* such that it's a deterministic matching. If can select M^* as one of our matching with some probability p^*, then we get PoF within p^* factor when PoF is defined by utility (maximizing problem). However, above example shows most fair solution does not contain M^*, this implies even in utility model, PoF can be arbitrarily big.

C.3 Proof of Proposition 4

Let $S^\pi_{\mathcal{D}} := \cup_{i\in\mathcal{D}} S^\pi_i$ be the set of riders that are matched to any driver in $\mathcal{D}$ in the schedule π.

$$\begin{aligned}\max_{\pi\in\Pi}\left(\sum_{r\in\mathcal{R}} U_r + \sum_{d\in\mathcal{D}} U_d\right) \quad &= \quad \sum_{r\in\mathcal{R}} \nu_r + \sum_{d\in\mathcal{D}} \nu_d \\ &- \sum_{r\in S^\pi_{\mathcal{D}}} C^r_{t^\pi_r \underline{t^\pi_r}} - \sum_{r\in\mathcal{R}:r\notin S^\pi_{\mathcal{D}}} \lambda_r - \sum_{d\in\mathcal{D}} C^d_{t^\pi_d \underline{t^\pi_d}}\end{aligned}$$

Note $\sum_{r\in\mathcal{R}} \nu_r$ and $\sum_{d\in\mathcal{D}} \nu_d$ are independent of π. Therefore the above optimization is equivalent to the cost minimization problem.

$$\min_{\pi\in\Pi} \sum_{d\in\mathcal{D}} \sum_{r\in S^\pi_d} C^r_{t^\pi_r \underline{t^\pi_r}} + \sum_{d\in\mathcal{D}} C^d_{t^\pi_d \underline{t^\pi_d}} + \sum_{j\in\mathcal{R}:j\notin S^\pi_{\mathcal{D}}} \lambda_j$$

On the side note, the social welfare maximization problem can be described as following minimization problem.

$$\begin{aligned}\min_{\pi\in\Pi} \qquad & \sum_{d\in\mathcal{D}} \sum_{r\in S^\pi_d} (1+\rho_d) C^r_{t^\pi_r \underline{t^\pi_r}} + \sum_{d\in\mathcal{D}} C^d_{t^\pi_d \underline{t^\pi_d}} \\ & + \sum_{j\in\mathcal{R}:j\notin S^\pi_{\mathcal{D}}} \lambda_j - \sum_{d\in\mathcal{D}} \sum_{r\in S^\pi_d} \rho_d \nu_r\end{aligned}$$

C.4 Proof of Proposition 5.2

Proof. Consider an example where d and r prefer each other the most. But there's r' who live far away. and d' who live close to r but farther from r' than d, and all matching result in positive utilities for all of them. Suppose (d', r') cannot be matched. Then SW maximizing solution would be (d, r') and (d', r) but then (d, r) form a blocking pair. Same is true for efficiency maximizing if $\nu_d > c^d_{trl} \cdot \mathtt{time}(d, r')$. This can lead to arbitrary big PoS. Suppose (d, r) gives each of them utility of 2ϵ, and other matching gives them utility of 0. Whereas if d' or r' is matched they get utility of Q ($\rho_{d'} = Q/\epsilon$). Then SW maximizing matching matched (d, r') and (d', r) and get total utility of $2Q$, whereas only stable matching gives utility of 2ϵ. Therefore we get PoS $= 2Q/2\epsilon = Q/\epsilon$. □

C.5 Existence of B-stable Solution

Proposition 5. *There exists B such that fairest solution is B-stable.*

Let U^*_i be the utility that user i is gaining in the stable matching. Let U_i be the utility that user i is gaining in the fairest matching. We give each user i the incentive $\beta_i = U^*_i - U_i$. Then all users are getting the same level of utility as in the stable matching; therefore, the matching is B-stable for $B = \sum_i \beta_i$.

C.6 Omitted Stable Matching Proposition

Proposition 6. *A stable matching exists if we assume rider's utility is non-increasing as number of riders in the same trip increases, i.e. riders are not complementary.*

Proof. We will follow standard GS algorithm, where riders propose and drivers only accept at most one rider. By the property in GS algorithm, all the matched riders are matched with optimal driver among all 1-to-1 stable matching. Furthermore, none of matched rider prefer match with 2 or more riders, thus there's no blocking pair containing 2 or more riders. Therefore, this form a stable matching.

□

D Different θ_r

We can also run the algorithm with heterogeneous riders. The P2P platform may want to provide different probability guarantee for different riders. The system may want to benefit users who provided benefit previously. Then we will have different threshold θ_r for each rider $r \in \mathcal{R}$. Then the following algorithm computes a feasible solution that satisfy threshold constraints.

$$\begin{aligned}\max_{p,\delta z} \quad & z \\ \text{s.t.} \quad & \delta_i \le \sum_i m^\ell_i p^\ell & \forall i \in \mathcal{R} \\ & z \le \delta_i - \theta_i & \forall i \in \mathcal{R} \\ & \sum_\ell p^\ell = 1 \\ & p^\ell \ge 0\end{aligned}$$

E NP-hardness

The P2P ride-sharing problem without deviation cost, capacity, and IR constraint is still NP-hard. The problem can be reduce from TSP problem.

For each node v in TSP problem, we have a rider r where his origin and destination is v, and he has time window of $[0, \infty]$ with $\nu_r = \infty$.

The driver has origin and destination at some vertex v. The driver has $\rho = \infty$, with time window $= [0, \infty]$. It is easy to see that his valuation is maximized when he satisfy all riders. Thus the problem becomes satisfying all rides subject to minimizing distance travlled. Note this solution is equivalent to TSP problem.

F Discussion

We will discuss assumptions we made throughout the main paper. We assumed $\lambda_r \le \nu_r$, if $\lambda_r > \nu_r$, then the rider would not take the trip and experience utility of 0 when he is not matched, because $0 > \nu_r - \lambda_r$. Then the cost of unsatisfied riders become $\sum_{r\in\mathcal{R}:r\notin\S^\pi_{\mathcal{D}}} \min\{\lambda_r, \nu_r\}$. The algorithm can extend to this case by replacing λ_r with λ'_r, where $\lambda'_r := \min\{\lambda_r, \nu_r\}$. We first replace (1) to $\min_{\pi\in\Pi} \sum_{d\in\mathcal{D}} c_{dS^\pi_d} + \sum_{r\in R\setminus S^\pi_{\mathcal{D}}} \lambda'_r$ and utility of rider when he is not matched to $U_r = \nu_r - \lambda'_r$. Note this also changes the definition of $Cost(M^\ell)$ and individual rationality. Note new definition of λ' makes intuitive sense, if a rider's alternative option is not take the trip, then he suffer opportunity cost of $\lambda_r = \nu_r$.

We assumed linear decomposible cost function. However, users cost function may be more complicated than just a linear function. Since our optimization is based on MILP and LP, our algorithm can be extended to any convex cost function with suitable convex solver.

Note our current model is vulnerable to misreported value. Our algorithm relies on all users to report truthfully. If a rider r reports a value $\nu_r = \infty$, the algorithm will try to find a matching that matches r with cost less than λ_r. One simple approach is to redefine IR constraint to be $U_r \geq 0$, However, this does not align with our stable matching definition. We leave the incentive-compatibility to the future direction.

Notion of envy-free allocation is very well suitable in P2P ridesharing model. However, it is well known that envy-free allocation may not exist in any discrete deterministic allocation; therefore, it does not exist in our model either. However, EF1 allocation exists, and α-envy-free allocation exist and randomized envy-free allocation exist. There are many envy-free allocation that can be related to our setting and closely related to stability of the market in driver's side. We leave the envy-free fairness as another potential future direction.