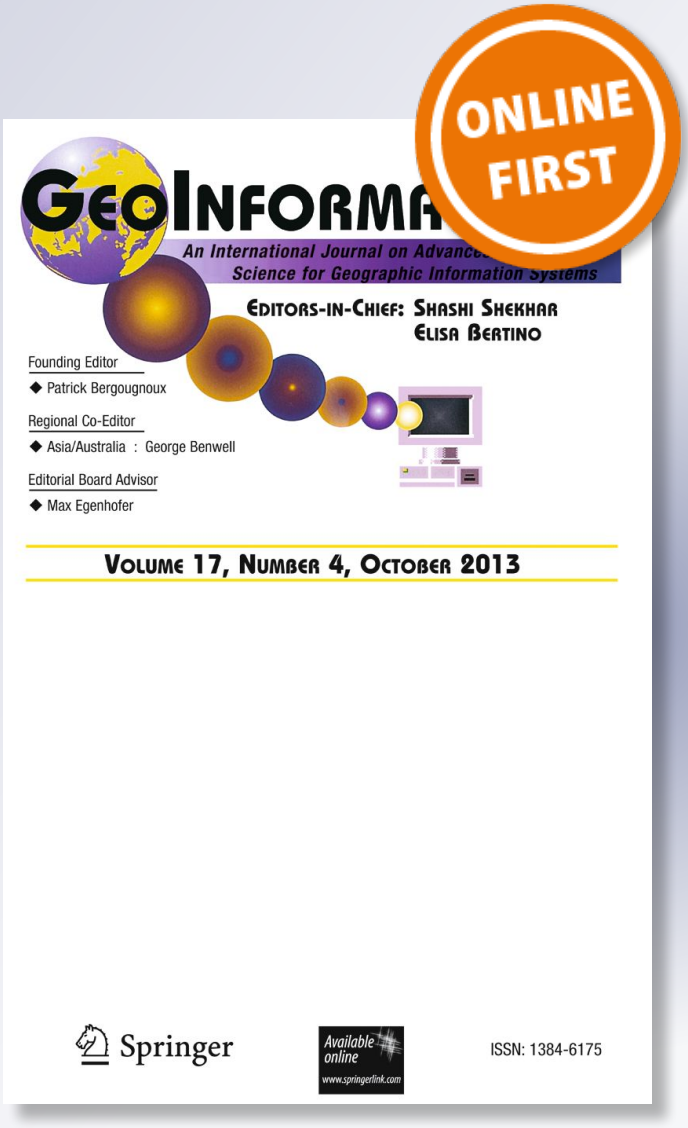
# *Optimum versus Nash-equilibrium in taxi ridesharing*

**Luca Foti, Jane Lin & Ouri Wolfson**

**GeoInformatica**
An International Journal on Advances of Computer Science for Geographic Information Systems

ISSN 1384-6175

Geoinformatica
DOI 10.1007/s10707-019-00379-6




Springer

GeoInformatica
https://doi.org/10.1007/s10707-019-00379-6

# Optimum versus Nash-equilibrium in taxi ridesharing

Luca Foti[1] · Jane Lin[2] · Ouri Wolfson[3]

Received: 10 February 2019 / Revised: 5 June 2019
Accepted: 15 August 2019
Published online: 24 August 2019


**Abstract**
In recent years, Transportation Network Companies (TNC) such as Uber and Lyft have embraced ridesharing: a passenger who requests a ride may decide to save money in exchange for the inconvenience of sharing the ride with someone else and incurring a delay. When matching passengers, these services attempt to optimize cost savings. But a possible scenario is that while passenger A is matched to passenger B, if matched to passenger C then both A and C would have saved more money. This leads to the concept of "fairness" in ridesharing, which consists of finding the Nash equilibrium in a ridesharing plan. In this paper we compare the optimum plan (i.e., benefit maximized at a global level) and the fair plan in both static and dynamic contexts. We show that in contrast to the theoretical indications, the fair plan is almost optimum. Furthermore, the fairness concept may help attract more passengers to rideshare and thus further reduce vehicle miles traveled. If social preferences are included in the total benefit, we demonstrate that the optimum ridesharing plan may be unboundedly and predominantly unfair in a sense that will be formalized in this paper.

**Keywords** Geospatial analysis · Optimal/fair matching · Ridesharing graph · Road transportation

## 1 Introduction

Mobility-on-demand (MOD) ridesharing services have become increasingly popular, especially in big cities, because they represent a valid solution to issues such as air pollution, fuel consumption, traffic congestion. However, the way these MOD ridesharing services are probably implemented by the Transportation Network Companies (TNC) such as Uber and

✉ Ouri Wolfson
owolfson@gmail.com

[1] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy

[2] Department of Civil and Materials Engineering, University of Illinois at Chicago, Chicago, IL 60607, USA

[3] Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA

Lyft have important drawbacks from a passenger's point of view. Specifically TNCs probably group passengers according to system optimum to maximize its gain or minimize its cost. However, system optimum results in some passengers incurring a higher cost (or lower saving), which is "unfair" to those passengers. In other words, if passengers were allowed to self-organize, they would do so differently than TNC's, namely fair ridesharing.

We show in this paper that fairness and optimum in ridesharing are equivalent to user equilibrium and system optimum in traffic and parking [1]. In other words, the gap between user equilibrium and system optimum that exists in these other transportation domains, carries over to ridesharing through the mechanisms outlined in this paper. This means that fair ridesharing may be suboptimal to the entire system, and optimum ridesharing may be unfair to individual passengers. Moreover, we show in this paper that optimum ridesharing may be detrimental to most passengers, and in fact unboundedly so. Such situations may arise particularly if a passenger's social preferences (e.g., preference for a foodie ridesharing partner) are considered in calculating the benefits of ridesharing. Overall, giving passengers a sense of "fairness" may attract more customers, thus have an overall positive impact on ridesharing demand.

In this paper we compare the fair and the optimal ridesharing plans (i.e. trip matchings). More specifically, we develop algorithms to compute fair ridesharing plans for two variants of fairness: an even-split benefit function where the benefit of ridesharing is split evenly between the ridesharing partners, and an uneven-split benefit function. We show that both cases are well grounded in practice. We then quantify the gap between fair and optimum ridesharing both theoretically and experimentally. Theoretically, we show that for the uneven-split variants of fairness, the Price of Anarchy (PoA)[1] is unbounded, whereas for the even split it is tightly bounded by two. Furthermore, we show that a fair ridesharing plan always exists for even-split benefit, it may not exist for uneven-split benefit, and that optimum plans may be very unfair in a sense that is precisely defined.

Experimentally we examine the gap from a practical perspective, using the NYC taxi database of over 700 million trips. We measure the gap in terms of mileage saving (or dollar saving assuming a linear relationship between the two). We determine that the gap is small, i.e. 2% on average.

### 1.1 Static and dynamic models in ridesharing

The comparison between fair and optimum ridesharing is carried out between two different model settings, i.e., static and dynamic. Both static (e.g., [2–4]) and dynamic ridesharing models (e.g., [5–8]) have been well studied in the literature.

Within a static model, requests which specify a pick-up time within a given time interval are pooled together and assigned to empty vehicles available at the common pickup location. Hence, occupied vehicles are excluded from the assignment because their current routes are already assigned. A practical scenario for this static model is a taxi station, e.g., at an airport. Consider a scenario in which a person's flight has landed at 10 am, but she needs a ride to go from the airport to her destination: the passenger registers to a specific pool (e.g., "10:30–10:35" pool) so that there is time to pick up her checked baggage. Rides in a pool are combined, and assigned to empty taxis waiting in a queue at the terminal. Assuming empty taxis are always available at the taxi station, a request or a group of requests is never assigned to a vehicle that is in progress of serving other requests.

[1] Price of Anarchy or PoA is the maximum ratio between the benefit of the optimum plan and the fair one

In a dynamic model, new requests can be combined and assigned to a taxi anytime and anywhere even if that taxi may already be occupied, as long as it has enough empty seats to accommodate the requests. A new route has to be computed in order to satisfy all the requests either newly assigned to or already onboard the vehicle. Hence, the vehicle's route changes dynamically.

### 1.2 Relevant work

The recent scientific literature has paid close attention to mobility-on-demand ridesharing in general [9–15], and taxi ridesharing in particular [2, 3, 6, 8, 16–20]; in taxi ridesharing the drivers do not have their own destinations, and this in turn expands the number of possible ridesharing routes. In these references, the discussed issues relate to how to match passengers and taxis efficiently (i.e. quickly), effectively (i.e. maximizing the benefits), taking advantage of multimodality (specifically walking), and what savings (e.g., in terms of mileage, number of trips, or dollar values) can be obtained by this matching in various practical situations. Agatz et al. [21] provide a review of optimum ridesharing.

Santi et al. [2] introduce the unweighted ridesharing graph used in this study, and provide experimental results in the Manhattan area. In the dynamic model, Alonso-Mora et al. [5] propose an any-time optimal approach, also used here, which finds the optimum solution to the problem of minimizing travel delays; but due to the complexity of the problem and the fact that a solution has to be found in real-time, a suboptimal solution is found there. In contrast, in this study we find the optimum solutions; the reason is that our study's purpose is to compare the gap between optimum and fair plans, and we are not bound by real-time considerations. And we use an adaptation of the any-time algorithm which maximizes benefit rather than minimizing delay.

Ridesharing is a cooperative game, specifically a coalition formation (CF) game in which a set of trip requests is partitioned into a coalition-structure i.e. disjoint subsets of requests; each subset is a coalition whose members will rideshare ([22]).[2] The coalition structure is formed subject to constraints dictated by bounds on delay due to ridesharing, passengers' preferences for other passengers, and objective functions such as cost or mileage savings. The approach to ridesharing taken in the CF literature has been to optimize the objective function, and subsequently redistribute the benefit in order to make the optimum as stable as possible (see [10]); where stability means that passengers' preferences provide no incentive to switch coalitions, and is equivalent to our fairness concept. There are three problems with this approach. First, the approach is incompatible with quantitative social ridesharing (the approach used here) since preferences cannot be naturally distributed among passengers, i.e., they are a nontransferable utility in CF terminology. Second, even if the benefit can be distributed, passengers cannot be given an intuitive and understandable formula for the way ridesharing benefit is split. For example, consider two trips $A$ and $B$, both of which originate at the same time from an airport, and are combined to rideshare such that $A$ is dropped off first, and then $B$ is dropped off. The CF approach may still distribute to $A$ more of the savings that ridesharing produces, even though ridesharing inconveniences $B$ more than $A$ (due to the fact that the combined trip follows the shortest path to $A$, but not to $B$). Passenger $B$ may understandably consider this distribution unfair, and consequently refuse to rideshare with $A$, or refuse to use the ridesharing system in general. Third, the opportunities for redistribution of the benefit for

[2] A coalition-structure is a ridesharing-plan in our terminology.

the purpose of fairness are limited. More specifically, it is possible that the optimum coalition structure is not amenable to fair redistribution while keeping the total benefit of each coalition within its members (a natural requirement for ridesharing) (see [10, 23]).

In this paper we take a Hedonic Game approach ([24]), which means that we treat benefits as nontransferable (even though sometimes the benefits represent $-savings). The reason is that in our setting the individual benefit of each passenger in a ridesharing group (a coalition) depends only on the ridesharing partners and is independent of other coalitions and their members. Furthermore, this individual benefit is given by a predefined formula that is simple and easily understandable. For example, the $-savings of a combined trip is distributed evenly among the passengers; or, it is distributed proportionally to the increase (compared to the solo trip) in the distance traveled.

Social preferences in ridesharing have been discussed in the literature [10, 25, 26]. Our social-ridesharing model is compatible with [25], but in contrast to [10, 26] we do not impose the qualitative requirement that a ridesharing group form a connected social network subgraph. The passengers specify social preferences that are weighted in conjunction with spatio-temporal compatibility of trips. Another commonality between the present paper and [10] is that they also discuss fairness/stability. However, [10] does not compare between the fair and optimal ridesharing plans, but introduces an algorithm that incorporates fairness into an optimal solution using the CF approach discussed above. Also, the [10] notion of fairness differs from the one in this paper.

Thaithatkul et al. [14] investigate day-to-day dynamics of a ridesharing matching problem using a modified Stable Roommates Problem (SRP) approach in a static setting, allowing within-day and day-to-day variation in ridesharing decisions. They use a specific pair-wise utility function, and find that multiple day-to-day equilibria exist under certain conditions. No distinction between SRP and optimum ridesharing is drawn and consequently no comparison to optimal ridesharing is discussed.

Finally, in our prior investigation [27], we have introduced the concept of fair ridesharing, devised algorithms for static fair ridesharing plans, and analyzed their complexities. However, [27] does not quantify the difference between fair and optimum ridesharing plans, which is the focus of this paper. Nor does it discuss and quantify the difference between the static and the dynamic models, or between the uneven and even-split of the benefits. In our most recent investigation [18], we focus on the comparison of the fair and optimum ridesharing plans in dynamic models. This paper builds on [18, 27], adding the following: 1. we integrate the two papers under a single model; 2. we conduct further experiments, quantifying the difference between fair and optimum ridesharing plans in both static and dynamic models, with even and uneven split; 3. We show that for a set of ride requests the fair ridesharing plan is unique, and thus the PoA and a related concept, the Price of Stability, are identical; 4. We show that for the even-split fair plan the bound of 2 on the PoA is tight, and that this bound holds even for the geometric variant of the problem where the benefit is given in terms of mileage savings on a concrete road network; 5. we introduce the concepts of predominant and unbounded unfairness in ridesharing, and demonstrate the optimum plan may be unboundedly and predominantly unfair; and this is more likely to occur when considering social-preferences for the purpose of devising the ridesharing plan.

The framework of our ridesharing platform is the following. The passenger's request expresses willingness to rideshare by providing a bound on the delay; and the system returns with a discount (compared to riding alone). When installing the app, the user can also specify criteria and preferences for social ridesharing. This framework is slightly more flexible than Uber's and Lyft's, where the system proposes the delay and the discount and there is no social

consideration. And our framework is not based on auctions, thus a large body of work related to truthfulness in ridesharing (e.g. [28–30]) is not directly applicable.

### 1.3 Contributions

In summary, the main contributions of this paper are as follows:

- We introduce the concept of fair ridesharing for an arbitrary benefit function, and contrast it with optimum ridesharing; and we show how to incorporate quantitative social ridesharing into the frameowork.
- We introduce the concepts of predominant and unbounded unfairness in ridesharing, and demonstrate the optimum plan may be unboundedly and predominantly unfair.
- We distinguish between even and uneven split of the benefit and prove that the PoA is tightly bounded by two in the former, even in the geometric setting of ridesharing (see sec. 5.1), and unbounded in the latter.
- We show experimentally that for ridesharing worst case analysis is significantly different than the average case in the sense that in practice the difference between fair and optimum ridesharing is small (around 2%); this is regardless whether the benefit is evenly or unevenly split.

The rest of the paper is organized as follows. Section 2 defines fair and optimum ridesharing schemes. Section 3 presents the algorithms used to compute ridesharing plans in both the static and dynamic models. Section 4 describes the settings of the numerical experiments, followed by the results in sec. 5. Finally, conclusions and future work are discussed in sec. 7.

## 2 The ridesharing model

In this section, we first define the concepts of a ridesharing pool (sec. 2.1). Then we define the pairwise ridesharing graph (sec. 2.2). In sec. 2.3 we define the concepts of a fair and an optimal ridesharing plan, and prove several related properties. In sec. 2.4 we discuss social ridesharing, and in sec. 2.5 we discuss the extension of the concepts to ridesharing that combines more than two trips.

### 2.1 Pooling requests

A *road network* is a weighted directed graph where a vertex is a road intersection, and an edge is the road segment between two intersections. The weight of an edge represents time to traverse the road segment or the length of the segment. A *trip request*, or a *request* for short, is a tuple which consists of: pick-up and drop-off locations on the road network, pick-up time availability, passengers count in a traveling party, and a fraction representing the bound on the *delay* (compared to solo travel along the fastest path) that the trip can tolerate due to ridesharing. For example, if the bound on the delay is 20%, then an hour-long trip along the shortest path can be lengthened to at most 1.2 h to facilitate ridesharing.

Typically, requests are considered with queuing based formulations [31], thus vehicles are matched to requests as the latter arrive in real time. In this paper we group requests issued within a specific time interval into a “pool” to facilitate ride matching.

**Ridesharing Pool** *(definition 1):* a ridesharing *pool* is a set of requests characterized by a time interval which ranges from "start pool time" to "end pool time" (e.g., if pool size is 5 min, then a possible pool starts at "10:00:00" and ends at "10:04:59"): all the requests which have been issued for a pick up within this interval (e.g., pick up time is at "10:02:00") are added to this set.

### 2.2 Ridesharing graph and ridesharing plan

We represent the pairwise ridesharing opportunities of a pool of requests as a weighted graph, called ridesharing graph (RSG) [5]. In an RSG = (**V**, **E**), each node $v_i \in \mathbf{V}$ is a single request, and each edge $e_{ij} \in \mathbf{E}$ connects two ridesharing requests $v_i$ and $v_j$. Intuitively, if $e_{ij} \in \mathbf{E}$ it means that requests $v_i$ and $v_j$ can be combined and serviced by a single vehicle. Formally, this means that:

1. There is a path $P$ in the road network from one of the two origins to one of the two destinations, such that
   a. $P$ goes through the remaining origin and destination, and
   b. in $P$ each origin comes before its corresponding destination; and
2. $P$ satisfies the delay constraints of requests $v_i$ and $v_j$.

The weight of edge $e_{ij}$ , denoted $w_{ij}$, is rational positive number representing the *total benefit* gained by combining requests $v_i$ and $v_j$ in a vehicle traveling along path P over the execution of the two single requests separately (i.e., riding alone). The total benefit of a ridesharing partnership may include dollar saving, mileage saving, pollution saving, social benefit (see sec. 2.4) or a combination of such factors.

The weight of an edge $w_{ij}$ must be positive since if $w_{ij} \leq 0$ then $v_i$ and $v_j$ should not be combined, i.e. $e_{ij} \notin \mathbf{E}$. For simplicity we assume that the edge weights are distinct, i.e. for every pair of different edges $e_{ij}$ and $e_{kl}$, $w_{ij}$ and $w_{kl}$ are different. If this condition is not satisfied initially, the RSG can be easily modified to satisfy it by adding to some edges a fraction that is arbitrarily close to 0.

The total benefit $w_{ij}$ is split between the *individual benefit* of $v_i$, denoted $w_{ij}(i)$, and that of $v_j$, denoted $w_{ij}(j)$, such that $w_{ij}(i) + w_{ij}(j) = w_{ij}$. Intuitively, this means that the individual benefits of the two partners sum up to the total benefit of the partnership. The individual benefits are calculated according to a predefined formula.

Figure 1 is an example of RSG. In Fig. 1, the individual benefits are calculated as follows. The total benefit of an edge is the dollars saved compared to the two requests being serviced by different vehicles, and the total benefit is split evenly between the two requests. Other formulas for calculation of the individual benefits are discussed in sec. 2.4 and 4.3. In terms of notation, on each edge $e_{ij}$ between requests $v_i$ and $v_j$, a label which encloses two values in square brackets is depicted: the value closest to request $v_i$ ($v_j$) indicates the individual benefit of request $v_i$ ($v_j$) in this shared ride.

**Ridesharing Plan** (*definition 2*): Given an RSG $G$, a ridesharing plan (rsp) on $G$, or a plan for short, is a set **S** of edges in the $G$ such that no request appears in different edges of **S**.

Intuitively, each edge $e_{ij}$, if exists, represents a shared ride combining requests $v_i$ and $v_j$, and thus the edges are selected such that no request belongs to more than one edge. In graph theory, an rsp is also known as a matching of the RSG.

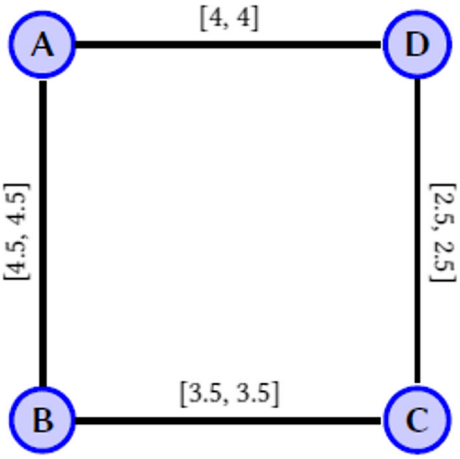



**Fig. 1** An example even-split Ridesharing Graph (RSG)

### 2.3 Fairness vs. optimality in ridesharing

**Fair ridesharing** (*definition 3*): Given an RSG $G$, an rsp **F** on $G$ is fair if there is no pair of requests A and B that are unmatched in **F**, but the individual benefit of A when matched with B is higher than A's individual benefit when matched with its partner in **F**; and the same for B.

In other words, an rsp is fair if there is no pair of unmatched requests that would both benefit more if they were matched. Observe that fairness requires reciprocity. That is, an rsp may be fair even if for a pair of unmatched requests A and B, A would benefit more when matched with B, but B would not. This is analogous to A making a partnership offer that is turned down by B, and it is fair for B to turn down the offer. Our notion of rsp fairness is the application of the stable matching concept, used in economics and mathematics, to ridesharing.

For example, in Fig. 1, the rsp that combines (A,B) and (C,D) is fair, yielding an overall dollar saving of \$14, but the rsp = {(A,D), (B,C)} is unfair because A and B can both save more by ridesharing with each other. Even though D may incur higher saving if pairing up with A over C, the pairing of A and D is not reciprocal because A prefers B and vice versa. Hence, in the rsp = {(A,B), (C,D)}, no one can benefit more by changing his/her partner unilaterally, and thus it is a fair one.

This concept of fairness results from the application of Nash Equilibrium to ridesharing. Specifically, in a game setting of ridesharing, passengers are the players; the strategies to each passenger are the feasible matches with other passengers; the payoff for each passenger is his/her net benefit in ridesharing relative to riding alone. By definition, the Nash equilibrium describes a desirable strategy in the game in which no player can improve his/her payoff by changing strategy unilaterally [32]. Fair ridesharing as defined above is a matching problem, and it can be formulated as a Nash Equilibrium [33].

For reasons of complexity, existence of a solution, and practicality we further distinguish an even-split and an uneven-split RSG in this study. An *even-split* RSG is one in which, for each edge $e_{ij} \in \mathbf{E}$, there exists $w_{ij}(i) = w_{ij}(j)$, as in Fig. 1; an *uneven-split* RSG is one where there exists at least one $e_{ij} \in \mathbf{E}$ such that $w_{ij}(i) \neq w_{ij}(j)$.

Even and uneven split cases may arise in practice. For example, the taxi shared ride service at the O'Hare airport in Chicago (flychicago.com) pairs passengers going to downtown Chicago, and if two passengers are paired, then each one of them pays \$25, regardless of

who gets dropped off first. This results in an even-split RSG. On the other hand, it may make more sense for the passenger who gets dropped off first to save less. This results in an uneven-split RSG.

**Theorem 1** A fair rsp may not exist for an uneven-split RSG.

**Proof** A fair rsp is a variant of the Stable-Roommate Problem (SRP), and it is established that a solution to an SRP may not exist. For example, consider the RSG of Fig. 2 and the rsp **R** = {(A,D), (B,C)}. This rsp is unfair to A, since both A and C save more when pairing with each other than with their partners in **R** (namely D and B respectively).

Now consider any other rsp say **R'.** The benefit of partnering with D is positive for each other request, thus **R'** will partner some request, i.e. B or C, with D. Using the same argument as used above for A, it can be shown that **R'** will be unfair to that request. [].

If a fair rsp does not exist, then at least one pair of requests is treated unfairly (i.e. the requests prefer each other over the partners assigned to them by the rsp); however, this is unavoidable.

**Optimum ridesharing** (*definition 4*): Given an RSG *G*, an *optimum* rsp is one that yields the maximum total benefit among all possible rsp's on *G*.

For example, in Fig. 1, the optimum rsp combines (A,D) and (B,C), providing the largest total dollar saving of $15; the fair rsp = {(A,B), (C,D)} gives an overall dollar saving of $14. Observe that, even though A and B could rideshare, and A and D could rideshare as well, A, B, and D cannot rideshare at the same time since B and D are not sharable (i.e., no edge between B and D).

Observe further that changing the split of a single edge may produce a different fair plan, even if the total dollar saving of the edge does not change. For example, in Fig. 1, if edge (A,B) has a split of {3,6}, instead of {4.5,4.5}, which means that A incurs a saving of $3 in ridesharing with B and B incurs a saving of $6 in ridesharing with A, then the optimum plan is also a fair one, i.e., $rsp_o = rsp_f = \{(A,D), (B,C)\}$.

In contrast to Theorem 1, it is easy to see that an optimum rsp always exists.

Given an RSG *G*, the *Price of Anarchy* (PoA) is the ratio between the optimum rsp on G, and the worst fair rsp, i.e.,

$$PoA = \frac{max_{p \in P} Benefit(p)}{min_{p \in E} Benefit(p)} \tag{1}$$

where $P$ is the set of all rsp's on $G$, $E \subseteq P$ is the set of fair rsp's, and $Benefit(p)$ is the benefit of rsp $p$.

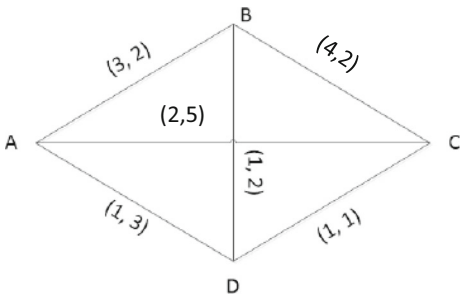



**Fig. 2** A Ridesharing Graph for which a fair rsp does not exist

Similarly, the *Price of Stability* (PoS) is the ratio between the optimum rsp and the best fair rsp on $G$, i.e., in the formula (1) above the *min* is replaced by *max*.

**Theorem 2** In an even-split RSG, the PoA and the PoS are both bounded by 2, and the bound is tight.

**Proof** The greedy algorithm used to compute the fair rsp is described in sec. 3.1.1(c) and is called Even-Split-Nash-Equilibrium (ESNE). In essence, it selects edges of the RSG in descending order of benefit and finds a collection of disjoint sets of edges of maximum total benefit. This algorithm can be easily shown to be a 2-approximation of the optimum (see [34]). Theorem 5 in sec. 3.1.1 indicates that for a given even-split RSG there is a unique fair rsp, thus PoA and PoS are bounded by 2.

To see that the bound is tight consider again the RSG of Fig. 1, but modify the total benefit of the edges as follows: (A,B) = x + $\varepsilon_1$, (A,D) = x + $\varepsilon_2$, (D,C) = $\varepsilon_1$, (C,B) = x where $\varepsilon_1 > \varepsilon_2$. The PoA (and PoS) in this case is arbitrarily close to 2. [].

**Theorem 3** In an uneven-split RSG, the PoA and the PoS are unbounded.

**Proof** Assume that in Fig. 1 request D's individual benefit for ridesharing with A is not 4, but an arbitrarily large value X (e.g., D's social score, defined below, for A is very high). In this case the optimum rsp is {A,D} and {B,C}, whereas the fair rsp is {A,B} and {C,D}. The ratio between the two is a constant times X, i.e. an arbitrarily large value. [].

### 2.4 Incorporation of social preference scores

We have defined in sec. 2.2 a RSG as a weighted graph. The weight $w_{ij}$ represents the total benefit gained by combining requests $v_i$ and $v_j$ over the execution of $v_i$ and $v_j$ separately (i.e., riding alone). Previously the weight was measured in terms of dollar or mileage saving. In this section, we generalize the definition of a weight to include a passenger's social preference scores and show that, when incorporating social scores, the optimum rsp may no longer be sensible in contrast to the fair rsp.

From sec. 2.2, we have

$$w_{ij} = w_{ij}(i) + w_{ij}(j) \tag{2}$$

where $w_{ij}$ is the total benefit of edge $e_{ij}$, $w_{ij}(i)$ and $w_{ij}(j)$ are the individual benefits of $v_i$ and $v_j$, respectively. For each individual benefit, we define it as a weighted sum of the out-of-pocket monetary saving, delay, as well as social preferences such as personal preferences on what type of person to rideshare with. For example, a female passenger may prefer to rideshare with another female passenger; singles may prefer other singles; a wine lover may want to be able to strike a conversation with another wine lover. That is,

$$w_{ij}(i) = \sum_k \alpha_k(i) x_k(i) \tag{3}$$

where $x_k(i)$'s are the individual benefit scores for out-of-pocket monetary saving, delay, social preferences, etc., and $\alpha_k(i)$'s are the weights reflecting the relative importance of those benefit scores. For example, if someone strongly prefers a partner who is a classical music lover, the

person may place a very high weight on that score. We assume that the weights and individual scores for social features (e.g. classical-music lover) are provided truthfully.

We now demonstrate that when social scores are incorporated in the ridesharing benefit calculation, optimum ridesharing may be unboundedly unfair. Consider the scenario in Fig. 3, where D feels strongly about partnering with A only. For example, suppose D places a great deal of emphasis on his/her ridesharing partner being single. Between D's potential partners, i.e., A and C, A is single but C is not. Hence, D has a very high overall benefit in ridesharing with A (e.g. X = 1000) over C (e.g. 100). In this case, even though A and B prefer each other, and C prefers D, the optimum plan is $rsp_o = \{(A,D), (B,C)\}$. As a result, the optimum plan benefits only D and inconveniences all others, thus it is clearly unfair. Furthermore, the inconvenience may be arbitrarily large. The following discussion formalizes these concepts.

Let $G = (V,E)$ be an RSG, and assume that a fair plan on it exists. Let $P$ be an rsp on $G$, and $S$ a subset of the set of requests $V$. We say that $P$ *unfair on* $S$ if for each request $r$ in $S$, the individual-benefit of $r$ in $P$ is lower than the individual benefit of $r$ in every fair plan defined on G. We say that $P$ is *predominantly unfair*, if $S$ contains a majority of the requests in $V$.

Furthermore, $P$ is *unboundedly unfair on* $S$, if for any number $N$, the individual benefits on $G$ can be modified[3] to create RSG $G'$ which has a fair plan, such that for $P$ on $G'$ the following condition ***C1*** is satisfied:

> ***C1***: for each request $r$ in $S$, the individual-benefit of $r$ in $P$ is lower than the individual benefit of $r$ in every fair plan defined on $G'$ by a factor of at least N, i.e. for every $r \in V$ and for every fair plan $F$, $b(F,r)/b(P,r) > N$, where $b(Q,r)$ is the individual benefit of $r$ in rsp $Q$.

$P$ is *predominantly unboundedly unfair*, if it is unboundedly unfair on a set of requests $S$ containing a majority of the requests in $V$.

**Proposition 1** Consider the RSG in Fig. 3. The optimum plan $P = \{(A,D), (B,C)\}$ on it is predominantly unboundedly unfair.

**Proof** For each $X > 10$ and $\varepsilon < 1$ there is a single optimum plan and a single fair plan, and each member of $S = \{A, B, C\}$, has a higher individual benefit in the fair plan $\{(A,B), (C, D)\}$ than in $P$. Thus, $P$ is unfair on $S$. Now let N be a number, and let $X = N$. For $r = A$, $B$, and $C$, and $F = \{(A,B), (C, D)\}$, $b(F,r)/b(P,r) > N$. [].

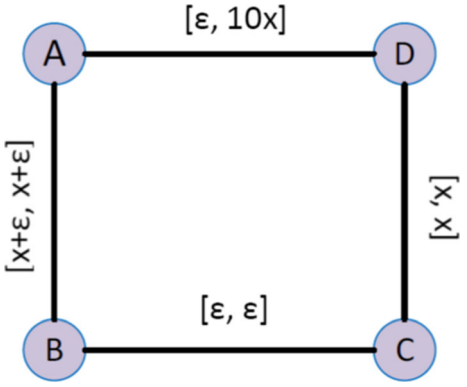



**Fig. 3** An example of an rsp that is optimal but predominantly and unboundedly unfair

We postulate that often a Transportation Network Company may choose to make the majority of its customers happy, rather making a single customer very happy. In other words, the TNC may choose the fair plan over the optimum one if the latter is predominantly and unboundedly unfair.

### 2.5 Ridesharing graph with more than two requests in a shared ride

In this case the ridesharing problem is represented as a weighted undirected Ridesharing Hyper-graph (RSH) [2]. In an RSH *(V, H)* each node in $\boldsymbol{V}$ is a request, and each hyper-edge $E \in \boldsymbol{H}$ is a set of nodes; the cardinality of $E$ is $c$ or lower, where $c$ is the maximum capacity of a vehicle. The *weight* of $E$ is the total benefit obtained if the corresponding set of requests is serviced by a single vehicle. The individual benefit in an RSH is defined analogously to the RSG.

An (*un*)*even-split* RSH is one in which the total benefit represented by the weight of each hyper-edge is (un)evenly split among its requests.

A *ridesharing plan* is a set of node-disjoint hyper-edges in the RSH. An *optimum* rsp is one of maximum total weight. An rsp **F** is *fair* if there is no hyper-edge $E \in \boldsymbol{H}$ that is not in **F**, but each request $r$ in $E$ has a higher individual benefit than $r$'s individual benefit in **F**.

## 3 Ridesharing algorithms

In this section we present the algorithms used to produce the optimum and the fair rsp within the static and dynamic models. They are summarized in Table 1. The algorithms for the static model are presented in sec. 3.1 and the ones for the dynamic model in sec. 3.2.

This section has two objectives: first to discuss possible algorithms to solve each one of the problems in Table 1, and second to discuss the algorithms implemented in the experiments presented in sec. 4.

### 3.1 Static models

The static models assume that: (a) all ridesharing trip requests within a pool are known in advance, (b) all trips originate at a single location, namely a traffic hub $H$ (e.g., an airport),[4] and (c) there are sufficient empty taxis available at the pickup location. Assumption (c) is strong, but in practice it may often be the case as taxi drivers often have very good knowledge about where the traffic demand hotspots are. For example, at an airport taxis often wait in line for customers.

We consider two types of RSGs: pairwise (at most two requests) RSG (sec. 3.1.1); and a general RSH in which any number $k$ of requests can be merged, where $k$ is constrained by the taxi capacity (sec. 3.1.2). In each type, we compute three rps's: the optimum rsp, the even-split

[3] The modification adjusts only the weights of the edges of $G$. The nodes and edges remain the same, thus P is an rsp on the resulting graph $G'$ as well.

[4] Due to this assumption, which implies a queue of taxis waiting for passengers, we are able to consider in a realistic setting ridesharing that is independent of vehicles locations and availability; these are being considered in the dynamic models.

**Table 1** Set of rsp's considered in this study

| | Optimum | Fair |
|---|---|---|
| Static model | Combining at most two requests (pairwise) | Even-split<br>Uneven-split |
| | Combining more than two requests (k > 2) | Even-split<br>Uneven-split[a] |
| Dynamic model | Adding any feasible number of new requests to the onboard ones | Even-split<br>Uneven-split[b] |

[a] the algorithm is discussed but not implemented in the experiment described in sec. 4

[b] only the case of combining at most two requests (new and onboard together) is implemented in the experiment described in sec. 4

fair rsp, and the uneven-split fair rsp. The algorithms for finding the rsp's are of different complexity.

#### 3.1.1 Combining at most two requests

Consider a pool of $n$ requests.

(a) Construct the RSG

Each request in the pool is a node in the RSG. Two nodes are connected by an edge if the benefit of combining them is positive. For example, assume that the benefit, i.e. weight, of an edge $e_{ij}$ is the distance saved by combining (ridesharing) the requests $v_i$ and $v_j$. Then the weight $w_{ij}$ is $(S-L)$, where $L$ is the shortest path in the road network that starts at the hub $H$, ends at the destination of $v_i$ (or $v_j$) and goes through the destination of $v_j$ (or $v_i$); and $S$ is the sum of the two respective shortest paths from the origin to the destination. The individual benefits are either $(S-L)/2$ in the even-split case, or are computed according to the split equation (see sec. 4.1 for an example of split equation) in the uneven-split case. The weight of the edge needs to be computed for each pair of requests in the pool. If the weight of an edge (i.e. the individual benefit of each request in the edge) can be computed in constant time,[5] then the RSG-construction takes $O(n^2)$. If the weight of an edge cannot be computed in constant time, e.g. in the above example where the weight is the vehicle-miles saved, then the RSG-construction takes $O(n^2)$ shortest path computations; each shortest path is computed in the graph $G(N, A)$ representing the road network. Overall, the time complexity in this case is $O(n^2(|A| + |N| \log |N|))$.

(b) Compute the optimum rsp

The optimum rsp can be computed with any maximum weight matching on the RSG. The Edmonds maximum matching is employed for the weighted case [35]. It yields a $O(n^{2.5})$ time complexity.

[5] For example, the weight of an edge can be computed in constant time if the Euclidean distance saved is used as the weight. Specifically, the weight of $e_{ij}$ is $(S-L)$, where $L$ = min {[(Euclidean distance from $H$ to the destination of $v_i$) + (Euclidean distance from the destination of $v_i$ to the destination of $v_j$)], [(Euclidean distance from $H$ to the destination of $v_j$] + (Euclidean distance from the destination of $v_j$ to the destination of $v_i$)]; and $S$ is the sum of the two Euclidean distances from the hub to the two destinations.

(c) Compute an even-split fair rsp

The even-split fair rsp is computed iteratively by combining two requests at a time; the requests combined are the ones connected by the edge with the highest individual benefit (which is the heaviest edge in this case) in the remaining RSG, and removing them from RSG (along with their outgoing edges). This is done while there are still edges available in RSG. More specifically, the algorithm is as follows.

More specifically, the fair rsp **F** is computed by the following algorithm called **Even-Split-Nash-Equilibrium (ESNE):**

(a) Let **F** consist of the empty set.
(b) While there are edges in the remaining RSG do:

> (b.1) find the edge $e_{XY}$ with the highest individual benefit in the remaining RSG,
> (b.2) put the edge $e_{XY}$ in **F** (i.e. combine trips X and Y), and remove X,Y and their adjacent edges from the remaining RSG.[]

In terms of time complexity of ESNE, observe that there are at most $O(n^2)$ edges in the RSG, and sorting them takes $O(n^2 \log(n))$. A data structure in which each request points to its adjacent edges in the sorted list, and vice versa, can be prepared in $O(n^2 \log(n))$. Then the algorithm can be completed by a linear scan of the sorted list, where the processing of each entry can be done in constant time. Thus, the time complexity of the algorithm ESNE is $O(n^2 \log(n))$.

**Theorem 4** For an even-split RSG, the rsp **F** computed by ESNE is fair.

**Proof** Assume by contradiction that there exist two trips A and B that are not paired with each other in **F**, but both A and B have a higher individual benefit if they rideshare with each other, rather than with their assigned partners in **F**. Then the edge $e_{AB}$ must have been removed at step b.2 in some iteration $k$ of ESNE. Furthermore, at iteration $k$ another edge, say $e_{AC}$, must have been put in **F**. The fact that ESNE selected $e_{AC}$ rather than say $e_{AB}$ means that the weight of $e_{AC}$ is not lower than that of $e_{AB}$. Therefore, since the RSG is evenly split, A's individual benefit in ridesharing with C is not lower than A's individual benefit in ridesharing with B. This is a contradiction to the assumption that A's individual benefit is higher when ridesharing with B. [].

Theorem 4 indicates that for an even-split RSG a fair rsp always exists. This is in contrast to an uneven-split RSG, for which Theorem 1 indicated that a fair rsp may not exist.

**Theorem 5** For an even-split RSG, there is a unique fair rsp.

**Proof** Assume by way of contradiction that there exists another fair rsp, **E**, that is different than **F** computed by ESNE. Then sort the edges of **F** and **E** in decreasing order by weight, and consider the first position $k$ where the edges of the two rsp's are different; i.e. $f_{vw}$ in **F** is different than the $k$'th edge of **E**. Due to the fact that the edge-weights of the rsp are distinct (see sec. 2.2), $v$ appears in the sorting of **E** after position $k$, and so does $w$. Due to the way **F** was constructed, $v$ has a lower individual benefit in **E** than in **F**; and so does $w$. Thus **E** is unfair. []

(d) **Compute an uneven-split fair rsp**

The uneven-split fair rsp algorithm **USNE** works as follows.

1) each request in the RSG sorts its neighbors (possible partners in the ridesharing plan) by individual benefit in descending order;
2) the sorted lists are solved with the algorithm that solves the "Stable Roommates Problem with Incomplete Lists" (SRP) [36] to find a fair rsp.

Overall, the time complexity of USNE is $O(n^2 \log(n))$.

#### 3.1.2 Combining more than two requests (k > 2)

We assume that a vehicle can hold at most $c$ passengers, and thus the RSH consists of hyper-edges of cardinality of at most $c$. For the experiments described in sec. 4 we set $c = 4$.

(a) **Construct the RSH**

Each request in the pool is a node in the RSH. The construction is similar to that of the RSG discussed above in sec. 3.1.1. If the weight of a hyper-edge can be computed in constant time, then the RSG construction takes $O(n^c)$.

(b) **Compute the optimum rsp**

Finding a maximum matching in a hyper-graph is a NP-complete problem. By a straightforward reduction it can be shown that, consequently, finding the optimum rsp is NP-complete.

For the experiments described in sec. 4 we find the optimal rsp by a computation that is exponential in the worst case. However, for speed up we use a Branch-and-Bound technique inspired by the solution to the "Graph-Constrained Coalition Formation" (GCCF) problem ([26, 37]).

(c) **Compute an even-split fair rsp**

The even-split fair rsp is computed with an equivalent of Algorithm ESNE; i.e. it selects hyper-edges in decreasing order of their individual benefits. Observe that here, in contrast to ESNE, it is possible that at some iteration of step b.1 of ESNE, a selected hyper-edge $E$ may have a lower weight than that of another hyper-edge $K$ in the remaining RSH, even though $E$'s benefit is higher (due to the fact that $E$ has a lower cardinality). The worst-case complexity of the algorithm is $O(n^c \log(n))$.

(d) **Compute an uneven-split fair rsp**

The problem of finding an uneven-split fair rsp or establishing that there is none becomes NP-complete; this can be demonstrated by a straightforward reduction from the "Stable Roommates Problem with triple rooms" (3D-SR) discussed in Iwama et al. [38].

As shown in sec. 4 in the context of combining only two requests at a time, there is no significant difference between the even- and the uneven-split fair plans; thus, for ridesharing of more than 2 requests, sec. 4 compares the optimum plan to the even-split fair plan only. In

other words, an uneven-split fair rsp for combining more than 2 requests is not considered in the experiments presented in sec. 4.

### 3.2 Dynamic models

In the dynamic models, new requests may appear dynamically in space and time, and they are grouped into pools as in the static case. The algorithm for finding an rsp also involves assigning a taxi to a group of newly matched requests in a pool. The total number of taxis is fixed and limited, which means that an assigned taxi may need to travel to the pickup location and it may already be occupied at the time of assignment. The implications are as follows: 1) we consider the general k-request rsp's (where $k > 2$) for both optimal and fair rsp's; and 2) in contrast to the static model, a taxi may dynamically change its route to service a newly arrived request, and the on-board requests need to be considered when computing the revised route. Therefore each pool consists of *new requests,* or NRs, i.e., requests whose pick-up time availability falls between start-pool-time and end-pool-time; and *onboard requests,* or ORs, which are requests that are on board taxis.

(a) **Construct an extended RSH (e-RSH)**

In the dynamic model requests are only meaningful when assigned to a taxi. Furthermore, different taxi assignments can provide different benefits. To implement this idea, we first construct the RSH ($\mathbf{V, E_H}$) with respect to NRs only, following the approach discussed in sec. 3.1.2. So the nodes in the RSH are NRs and the hyper-edges are disjoint sets of the NRs sharing rides. Next, we extend the RSH to include ORs and create an *extended*-RSH (e-RSH).

This is how the e-RSH ($\mathbf{V_e}$, $\Xi$) is created. The nodes of the e-RSH are the NRs in the pool and the taxis. A taxi $T$ is *feasible* for a hyper-edge $E$ of the corresponding RSH if: 1) currently $T$ can accommodate the NRs in $E$; 2) the delay constraints of the ORs in $T$ and of the NRs in $E$ can be satisfied; and 3) the benefit of combining the ORs in $T$ with the NRs in $E$ is positive. For each hyper-edge $E$ of the RSH and taxi $T$, if $T$ is feasible for $E$ then a new hyper-edge $\Xi_{ET}$ is created in the e-RSH; $\Xi_{ET}$ consists of the requests in $E$ and taxi $T$.[6] Otherwise, no hyper-edge is created for the pair in the e-RSH.

If a hyper-edge $E$ of the RSH is not connected to a taxi in the corresponding e-RSH, meaning that the matched new requests in the hyper-edge cannot be allocated to a taxi in the current pool, these new requests are pushed to the next pool.

In the instantiation of the e-RSH construction used in sec. 4, the weight of an e-RSH hyper-edge is the distance saved by combining the NRs and ORs. If the taxi is empty at the time of being assigned to the NRs, then the weight is computed among the NRs the same way as in sec. 3.1.1(a).[7] If the taxi is already occupied, then the weight is the difference $(S - L)$, where:

- $S$ is the sum of
- the sum of the shortest paths to satisfy the NRs, and

[6] Observe that all the subsets of a hyper-edge are also hyper-edges in the RSH, since if requests *A, B, C* can be combined, then clearly they can be pairwise combined. For the purpose of constructing the e-RSH, singleton subsets of an RSH hyper-edge are also hyper-edges with weight 0 (since there is no ridesharing benefit). It means that $\{A\}$, $\{B\}$, and $\{C\}$ in the example are also hyper-edges that are matched with taxis.

[7] Observe that this formulation does not consider the distance that the empty taxi travels to the pickup location.

- the remaining taxi path to satisfy the ORs at the time the algorithm is run;
- $L$ is the shortest path to satisfy all the NRs and ORs at the time the algorithm is run.

The overall time complexity for constructing the e-RSH is $O(n^4m)$ shortest path computations, i.e., $O(n^4m(|A| + |N|\ log\ |N|))$, where $m$ is the number of taxis, $n$ is the number of total requests in the pool, and $G(N, A)$ is the graph representing the road network.

(b) **Compute the optimum rsp**

To solve for the optimum rsp for a given e-RSH, we apply an adaptation of the Integer Linear Programming (ILP) problem presented in [5]. In contrast to [5], we maximize the overall benefit, rather than minimize the cost; also our ILP uses hyper-graphs rather than graphs. Otherwise the ILP's are similar, and we present ours here simply for completeness.

In order to speed up the process of finding the optimum rsp, we start from an initial solution found by the following greedy algorithm:

1) sort the hyper-edges of the e-RSH by weight in the descending order;
2) add the remaining hyper-edge with the highest weight, $\Xi$ ($\in\boldsymbol{\Xi}$), to the solution;
3) remove the nodes (i.e., new requests and taxi) and the connected hyper-edges that contain nodes in $\Xi$ from the remaining e-RSH;
4) repeat steps 2) and 3) until no hyper-edges remain in the e-RSH.

Then, the ILP presented in Table 3 improves this initial solution to find the optimal one.

The ILP *assigns* taxis ($v_j$'s) to groups of matched new requests ($g_k$'s), i.e. edges of the RSH, to maximize the total savings.

In addition to the notations listed in Table 2, there are two sets of binary variables used in the ILP:

$$\in_{gkvj} = \begin{cases} 1, & \text{if } \left(g_k \cup \{v_j\}\right) \in \mathrm{E_{RV}}, \text{and } \left(g_k \cup \{v_j\}\right) \text{is in the solution} \\ 0, & \text{otherwise} \end{cases}$$

and, (4)

$$\mathcal{X}_{ri} = \begin{cases} 1, & \forall v_j \in \mathrm{V} \text{ and } \forall g_k \in \mathrm{G}_{r_i}, \text{hyperedge } \left(g_k \cup \{v_j\}\right) \text{is not in the solution} \\ 0, & \text{otherwise} \end{cases}$$

$\epsilon_{g_k v_j}$ is 1 if hyper-edge $(g_k \cup \{v_j\})$ of the e-RSH is in the solution, and is 0 otherwise. $\mathcal{X}_{r_i}$ defines the assignment status of a new request to a taxi: if $\mathcal{X}_{r_i} = 1$, then in the solution, new request $r_i$ is not assigned to any taxi $v_j$. The union of the two sets is denoted by $\mathcal{X} = \{\epsilon_{g_k v_j}\} \cup \{\mathcal{X}_{r_i}\}$. The saving $s_{g_k v_j}$ represents the weight of the e-RSH hyper-edge that connects $g_k$ and $v_j$.

In the objective function shown in Table 3 line (2), each $\mathcal{X}_{r_i}$ is multiplied by $p$ and subtracted in order to penalize new requests that are unassigned to a taxi. The constraints of this ILP problem are: (3) each taxi can be assigned to at most one new request set $g_k \in \mathrm{G}_{v_j}$; and (4) each new request is either a member of some new request set $g_k \in \mathrm{G}_{r_i}$ that is assigned to a taxi, or is not assigned at all.

**Table 2** Notations in ILP

| | |
|---|---|
| R | set of all new requests in the pool, i.e., $R = \{r_i, i = 1, 2, \ldots, n\}$, where $n$ is the number of requests in a given pool. |
| V | set of all taxis, i.e., $V = \{v_j,\ j = 1, 2, \ldots, m\}$, where $m$ is the total number of taxis serving a given area. |
| $E_{RV}$ | the set of hyper-edges in the e-RSH, |
| $G_{v_j}$ | set of $g_k$'s such that taxi $v_j \in V$ is feasible for all the $g_k$'s in the set, i.e., $G_{v_j} = \{g_k \mid (g_k \cup \{v_j\}) \in E_{RV}$ and $v_j \in V\}$ |
| $G_{r_i}$ | set of $g_k$'s such that each $g_k$ contains new request $r_i$, i.e., $G_{r_i} = \{g_k \mid r_i \in g_k, g_k$ is a hyperedge of RSH$\}$ |
| $V_{g_k}$ | set of $v_j$'s such that each $v_j$ is feasible for a given $g_k$ ($\subset$R), i.e., $V_{g_k} = \{v_j \mid (g_k \cup \{v_j\}) \in E_{RV}\}$ |
| p | The positive real number representing the penalty for a request to be unsatisfied in the current pool, thus transferred to the next one. |

We employ the Mosek solver for the ILP optimality; the algorithm is an anytime algorithm that improves an initial solution and is inherently parallel, which improves the computation time.

### (c) Compute an even-split fair rsp

We solve the even-split fair rsp for an e-RSH by an algorithm that is similar to the adaptation of ESNE to hypergraphs given in 3.1.2(c) for the static model. Observe that each hyper-edge $\Xi$ of the e-RSH consists of a single taxi $T$ and a set of NRs and ORs aboard $T$, and the benefit of $\Xi$ is evenly split among the NRs and ORs. Specifically, the fair rsp **F** is found as follows:

1) examine hyper-edges of the e-RSH in decreasing order of individual benefit.
2) for each hyper-edge $\Xi$ examined, insert $\Xi$ in the solution **F** and remove those hyper-edges that contain either an NR or a taxi in $\Xi$.
3) repeat step 2) until all hyper-edges have been removed.

### (d) Compute an uneven-split fair rsp

Similar to the static model, the problem of finding a fair rsp in an uneven-split e-RSH is NP-complete. For simplicity, we only analyze the case with at most two requests. Specifically, if a taxi is empty at the time of assignment, then at most two new requests can be combined and assigned to the taxi; if the taxi is already occupied with ongoing requests, at most one new request can be assigned. We employ the same algorithm, i.e., USNE, devised for the static model. We find that there is no significant difference between the even- and uneven-split fair plans in the dynamic models (see in Section 5.2).

**Table 3** Dynamic model: ILP formulation of maximum matching

(1) Initial solution by greedy algorithm

$$\sum_{optimum} = argmax \quad C(\mathcal{X}),$$

(2) *where* $C(\mathcal{X}) = \sum_{(g_k, \cup, \{v_j\}) \in E_{RV}} s_{g_k v_j} \in_{g_k v_j} - \sum_{r_i \in R} \mathcal{X}_{r_i} p$

(3) s.t. $\sum_{g_k \in G_{v_j}} \in_{g_k v_j} \leq 1, v_j \in V, \qquad j = 1, 2, \ldots, m$

(4) $\sum_{g_k \in G_{r_i}, v_j \in V_{g_k}} \in_{g_k v_j} + \mathcal{X}_{r_i} = 1, r_i \in R, \qquad i = 1, 2, \ldots, n$

## 4 Experimental design

### 4.1 Network setting and data

We use the New York City road network extracted from the Open Street Map. The experiments are conducted in a subarea of NYC that contains LaGuardia and Manhattan; the subarea road network is a directed graph which consists of 59,792 intersections and 72,557 road segments.

We use the New York City taxi data (see [39] for details) to generate the trip and ridesharing requests. The NYC taxi data are stored in CSV format with over 700 million trips over four years of taxi operations in NYC, organized by year and month. Each taxi trip provides, among others, pick-up and drop-off locations (longitude/latitude coordinates); pick-up and drop-off timestamps; number of passengers on board; actual travel time; and distance traveled. We treat each taxi trip record as a single trip request. For simplicity, each request's pick-up location and drop-off location are mapped to the closest road network intersection [2]. We also assume the pick-up timestamp is identical to the pick-up time availability.

Pools are created based on the pick-up timestamp. For example, if the pool size is set to be 5 min, then the 10:00–10:05 pool contains all the requests whose pick up timestamp is within this time interval. We assume that each taxi can have at most four passengers at any given time.

### 4.2 Performance measure

In the experiments we measure the potential benefit of a rsp in terms of the percentage of vehicle miles traveled (VMT) saved by ridesharing, which is given by:

$$\%VMT\_saved = \frac{\left(\sum_{i=\{1,\ldots,n\}} miles_{sp,i}\right) - \left(\sum_{j=\{1,\ldots,m\}} miles_{traveled_j}\right)}{\sum_{i=\{1,\ldots,n\}} miles_{sp,i}} \times 100\% \qquad (5)$$

where $miles_{sp,i}$ is the number of miles on the shortest path for satisfying request $i$ alone (i.e., no ridesharing); $miles\_traveled_j$ is the miles traveled by taxi $j$ with some passengers on board; $n$ is the number of satisfied requests[8]; and $m$ is the total number of taxi trips. In other words, the distance saved is the difference between the total miles traveled without and with ridesharing.

In order to calculate the delay due to ridesharing, which is bounded by a parameter provided by the user, we compute the time of each taxi route. To do so we assume that the travel speed on each link is the product of the maximum link speed (given by the road network) and a congestion fraction (cf) representing the slowdown due to traffic. In the experiments we set the congestion fraction value to be 0.7, which means that the link travel speed is 70% of the maximum link speed for the entire road network.[9] The same fraction applies when computing both the optimum and the fair rsp's.

[8] Unsatisfied requests are not taken into account to compute the total distance saved in the experiments.

[9] In fact, the cf. value matters only for the dynamic model. The reason for this is that it can be easily shown that in the static model, since there is no waiting after the pool-end time, the delay bound is satisfied for a value $v1$ of cf. if and only if it is satisfied for any value. However, this is not the case for the dynamic model, where the delay of a request is the sum of the travel delay and the waiting delay (which in turn may arise due to a request being transferred from one pool to the next).

### 4.3 Uneven-split mechanism of benefit

In the experiments, when analyzing an uneven-split RSG, the total benefit is distributed proportionally to the increase (compared to the corresponding shortest path) in the distance traveled. Specifically, for any two requests $v_i$ and $v_j$ that are connected by an edge, suppose that the destination of $v_i$ is $SP_i$ miles away along its shortest path, and $D_i$ miles away in the joint path; and that the destination of $v_j$ is $SP_j$ miles away along its shortest path, and $D_j$ miles away in the joint path.

Let:

$$inc_i = \frac{D_i}{SP_i}$$
$$inc_j = \frac{D_j}{SP_j}$$

Therefore, $inc_i$ ($inc_j$) represents the increase in miles traveled for request $v_i$ ($v_j$) relative to its shortest path. Then, for every edge *(h,k)* of the RSG the individual benefits (mileage savings) of $h$ and $k$ are calculated according to the following equations:

$$w_{hk}(h) = \frac{inc_h}{inc_h + inc_k} \times w_{hk} \tag{6}$$

$$w_{hk}(k) = \frac{inc_k}{inc_h + inc_k} \times w_{hk} \tag{7}$$

For example, consider the situation in which passengers A and B rideshare starting from origin H; passenger A is dropped off first, and then the taxi continues to the destination of passenger B. Then $inc_A$=1 since the joint path follows the shortest path to dest(A). In contrast, $inc_B$ is higher, say 1.5, since the joint path from H to dest(B) goes through dest(A) rather than directly. Now, assume that the benefit of ridesharing is 5 units. Then passenger A gets 2 units and passenger B gets 3 units of this benefit.

### 4.4 Sensitivity analysis

We vary the following parameters and present their effects on the rsp's:

- *Willingness to rideshare*: probability that a passenger will participate in ridesharing. For example, if in a pool there are 50 requests and willingness to rideshare is 90%, then the pool analyzed consists of 45 random requests;
- *Maximum delay tolerated*: maximum percentage of the shortest drive time from pickup location to drop-off location that a passenger is willing to tolerate in order to rideshare. For example, if a request requires 30 min along the shortest path and maximum delay tolerated is 10%, then the passenger who makes the request accepts ridesharing with someone else if he/she can be dropped off within at most 33 min;
- *Pool size*: the time interval during which requests are grouped into a pool.

# 5 Experimental results

## 5.1 Static model settings and results

In the static model we consider only requests that originate at the LaGuardia airport and terminate at an intersection in the considered NYC road network. The total distance saved by ridesharing a pool of $n$ requests is the weight of the respective (hyper-)edge, as described in sec. 4.2.

In the pairwise rsp (combination of at most two requests) discussed in sec. 5.1.1, the demand consists of all the requests issued in the year 2013, each day from 7 am to midnight. Overall, there are about two millions such requests. In the rsp involving more than two requests in each shared ride (sec. 5.2.2), since the computation is exponential for the reasons explained in sec. 3.1.2, we randomly choose and analyze 100 pools of requests issued in 2013.

We assume that a sufficient number of empty taxis are available at the LaGuardia airport to serve the demand. Consequently, no request is propagated from one pool to another because, if a request cannot find a rideshare partner, it rides alone in a taxi. Thus the set of requests in each pool is fixed and identical in the fair and optimum simulations.

Table 4 shows the settings for each analyzed parameter considered in the analysis. Values in bold indicate the *default* for the parameters. For example, when measuring the mileage saved as a function of the willingness to rideshare, maximum delay tolerated and pool size are fixed to 10% and 5 min, respectively.

Results are obtained by averaging all the pools for the entire year of 2013. So, for example, if the pool size is 10 min, the number of pools analyzed is 6×24×365.

### 5.1.1 Combining at most two requests

In this subsection we compare the optimum rsp with the even- and uneven-split fair rsp's, and evaluate the number of pools for which an uneven-split fair plan does not exist. First, we compare the optimum and even-split fair plans (see Fig. 4). As expected, the %VMT_saved increases with the willingness to rideshare, delay tolerance, and pool sizes, in both the optimum and the fair plan. This is because a greater value in any one of these parameters is positively correlated with greater ridesharing opportunities for each pool. For the set of parameters tested, the maximum %VMT_saved peaks at a value between 30 and 38%. Santi et al. [2] demonstrated 40% reduction in vehicle trips with a maximum delay of 150 s (or a journey time of 25 min if the maximum delay is 10% of the journey time).

The difference between the optimum and the even-split fair plans is calculated as:

$$((\text{mileage savings of optimum rsp})-(\text{mileage savings of fair rsp}))/(\text{mileage savings of optimum rsp})$$

**Table 4** Static model: parameters settings

| Parameter | Settings |
|---|---|
| Willingness to rideshare (%) | 10, 20, 30, 50, 70, **90** |
| Maximum delay tolerated (%) | 5, 7.5, **10**, 12.5, 15, 20 |
| Pool size (min) | **5**, 6, 7, 8, 9, 10 |

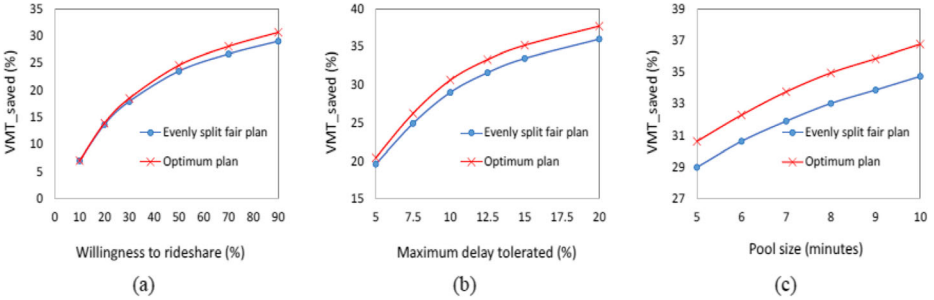



**Fig. 4** Static model combining at most two requests: %VMT_saved by the optimum and even-split fair plans

This difference is no greater than 2% of Vehicle-Miles-Traveled in all the parameter values tested. Furthermore, by comparing the fair and optimum plans for each pool we determined that the difference between the two is less than 15% in at least 90% of the pools. This is surprising given the tight bound of 2 on the PoS (see theorem 2). One may speculate that this is due to the geometric nature of the problem, and in fact, in ridesharing the bound of 2 on the PoS is not tight. In Fig. 5 we demonstrate by an example that this is not the case. The example shows a road network with a Hub (LaGuardia), four destinations A, B, C, and D, and the distances between them (Fig. 5a). Figure 5b shows the corresponding RSG. The fair rsp is (B,C) which has a benefit of $x\text{-}y + \varepsilon$, whereas the optimum plan is {(A,B), (C,D)} which has a benefit of $2(x\text{-}y)$. The ratio between the benefits of the two plans can be arbitrarily close to 2. In other words, the experimental analysis indicates that the worst case rarely occurs in practice.

We also compare the performance between the even-split fair plan and the uneven-split fair plan. For the purpose of this comparison we ignore pools for which an uneven-split fair plan does not exist. The difference in performance is measured by the % difference in VMT, denoted $\%\Delta_{VMT}$, between the uneven- and the even-split fair plan. It is defined as:

$$\%\Delta_{VMT} = \frac{VMT_u - VMT_e}{VMT_n} \times 100\% \tag{8}$$

where $VMT_u$, $VMT_e$, and $VMT_n$ denote, respectively, the total VMT in the uneven-split fair plan, the even-split fair plan, and the plan without ridesharing (i.e. each request is serviced by a different taxi). The results are summarized in Fig. 6. The average $\%\Delta_{VMT}$ is negative for all the parameter values considered. This indicates a slightly greater VMT reduction in an uneven-split fair plan than an even-split one (although the difference is statistically insignificant). This is surprising since in the worst case the PoS is unbounded in the uneven case, and bounded by

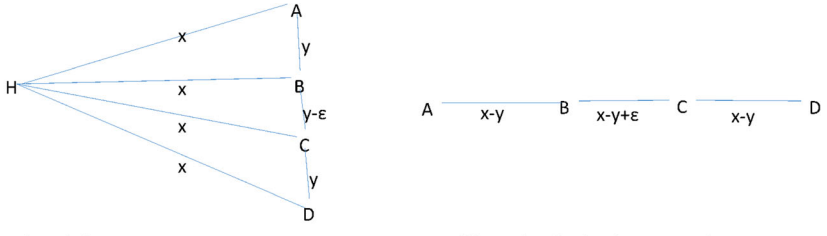



**Fig. 5** Requests A, B, C, D on a road network, and their corresponding Ride-Sharing Graph

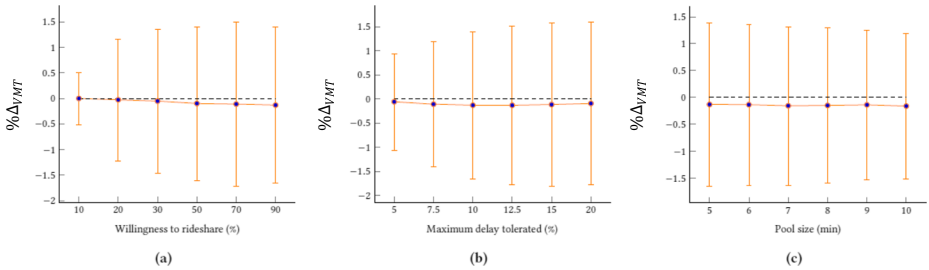



**Fig. 6** Static model combining at most two requests: $\%\Delta_{VMT}$ between uneven- and even-split fair plans. The figure gives the average and range of the standard deviation

2 in the even case. Again, this indicates that worst case analysis in ridesharing does not provide a reliable comparison in practice.

Figure 7 presents the percentage of pools in which there does not exist an uneven-split fair plan as a function of the willingness to rideshare (**a**), the maximum delay tolerated (**b**), and the pool size (**c**). All three plots suggest that the nonexistence probability increases as a function of the ridesharing opportunity.

### 5.1.2 Combining more than two requests

Figure 8 presents the case in which $k$ requests ($k > 2$) can rideshare in the same taxi, keeping all other input parameters at the default values as listed in Table 4. As expected, the %VMT_saved yields a higher value when more requests can be combined into a single shared ride. However, the gap between the optimum and the fair plan is still no more than 2%, and it shrinks as the number of passengers allowed to rideshare in a taxi increases. This suggests that the system wide performance between the optimum and the fair ridesharing plan is quite consistent.

## 5.2 The dynamic model settings and results

As discussed in sec. 3.2, the dynamic models are much more computationally expensive than the static ones. Thus, we consider the much smaller Manhattan road network, which consists of 3933 road intersections and 8400 road segments. Furthermore, we only consider the requests originating and terminating in Manhattan from 10 am to noon on January 25th,

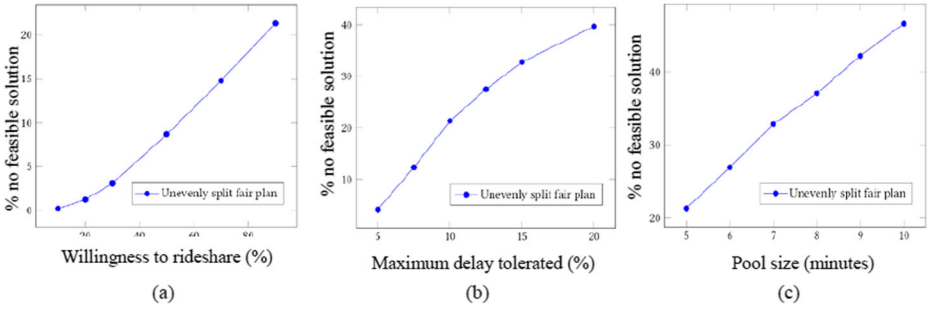



**Fig. 7** Static model combining at most two requests: % of pools for which no fair rsp exists with the uneven-split RSG

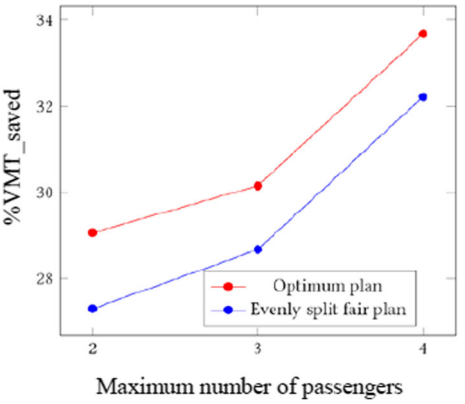



**Fig. 8** Static model combining more than two requests: %VMT_saved with the optimum and fair plans

2013. In this time span, about 40,000 requests were issued. We group the requests into pools of 30 s. The represents on average about 167 new requests issued each 30 s in that two hour span. In contrast, in the static model a 10-min pool has on average less than 50 requests. The reason is that the requests in the dynamic case do not originate at a single location, as LaGuardia in the static case, but anywhere in Manhattan.

We discuss two scenarios. In sec. 5.2.1 we conduct two simulations, one producing a fair plan, and the other an optimum plan. Then we compared the mileage-savings. A similar comparison is reported in sec. 5.2.2 between the even- and uneven-split RSGs. In the uneven-split fair simulation, if an uneven-split fair plan does not exist for a pool, then an even-split fair plan is used. For simplicity we assume that each request involves a single passenger.

If a request cannot be satisfied in one pool due to an insufficient number of feasible taxis, then it is propagated to the next pool. Consequently, in the fair and optimum simulations a pool in a time interval, say 10:21:00–10:21:30, may consist of different sets of requests. The reason is that the sets of requests propagated from the 10:20:30–10:21:00 pool are different in the two simulations. In turn, the latter two sets differ because the fair and optimum plans combine and assign requests to taxis differently. In other words, a request $r$ in the 10:20:30–10:21:00 pool may be satisfied by the optimum rsp, but not by the fair one. Then the 10:21:00–10:21:30 time-interval pool will contain $r$ in the fair case, but not the optimum.

Observe that in the static model, since empty taxis are always available, no request is propagated from pool to pool; if a request cannot find a rideshare partner, it rides alone in a taxi. Thus the set of requests in each time-interval pool is identical for the fair and optimum simulations. In the dynamic model, if the delay tolerated is exceeded then the request is labeled *unsatisfied* and propagates to the next pool. And we use the %satisfied_requests as an additional measure of performance in comparing the fair and optimum plans.

**Table 5** Dynamic model: parameters settings

| Parameter | Settings |
|---|---|
| Willingness to rideshare (%) | 90 |
| Maximum delay tolerated (%) | 5, 7.5, 10, 12.5, 15, 20 |
| Pool size (sec) | 30 |
| Penalty to delay a request (p) | 0.1 |

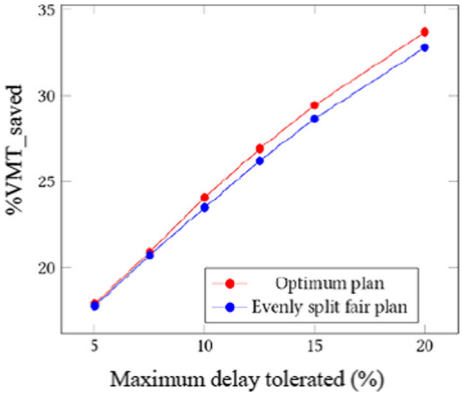



**Fig. 9** Dynamic model: %VMT_saved with the optimum and even-split fair plans

Furthermore, in the ILP algorithm we consider a penalty for delaying a request ($p = 0.1$). This should increase the %unsatisfied-requests in exchange for an increased VMT_saved. Even so, as discussed in sec. 5.2.2, the gap in %VMT_saved between the fair and optimum plans is miniscule.

At the beginning of a simulation, taxis are initialized as empty at random road intersections. During the simulation, if a taxi is not assigned to any request and does not have any passenger on board at current time step then it follows a shortest path from its current location to a random road intersection. In practice, unless they are in a high-demand area, it is not uncommon that taxis drive around to look for customers.

In the experiments, we vary the value of the maximum delay tolerated while keeping constant the willingness to rideshare and the pool size. The parameters settings are summarized in Table 5.

### 5.2.1 Optimum plan vs. even-split fair plan

In this comparison, 5000 taxis were randomly selected from the NYC taxi data set. As shown in Fig. 9, the *%VMT_saved* in the optimum plan increases from 17.9% to 33.7% as the maximum delay tolerated increases from 5% to 20%. A very similar trend is also observed for the even-split fair plan. Again, the gap between the optimum and the even-split fair plan is within two percentage points.

Figure 10 shows the trend in %satisfied_requests. Recall that in the dynamic model if the delay tolerated is exceeded then the request is labeled *unsatisfied*. Thus the %

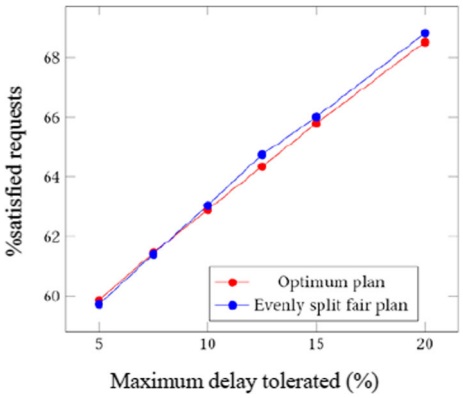



**Fig. 10** Dynamic model: % satisfied requests

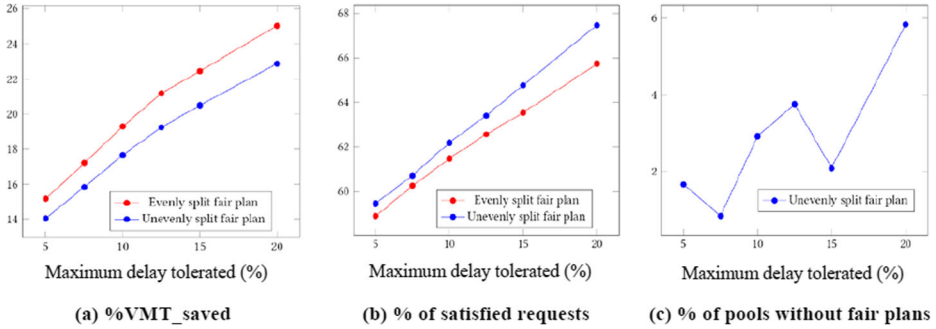



**Fig. 11** Dynamic model: comparison between even- and uneven-split fair plans

satisfied_requests improves as the maximum delay tolerated relaxes. When the delay tolerance is at 5%, less than 60% of the requests are satisfied in both the optimum and even-split fair plans. At 20% of the tolerance level, less than 70% of the requests are satisfied. Again the performances of the optimum and the fair plans are almost identical with respect to the %satisfied_requests, with the fair plans even having a slight advantage.

### 5.2.2 Even-split vs. uneven-split fair plan

In this comparison, we consider only the pairwise fair plans. In order to maintain an acceptable %satisfied_requests, we increased the taxi fleet to 7500 vehicles in the simulations.

Observe that in Fig. 11 both the even- and uneven-split fair plans have very similar %VMT_saved and %satisfied_requests, with no more than two percentage point difference between the two. This is a desirable feature because it suggests that the even-split fair plans have similar benefits, but is much more efficient computationally. Lastly, it is interesting to observe that in the dynamic models the % of pools with a nonexistent solution (Fig. 11(c)) is far lower than in the static settings (Fig. 9). This suggests that in the dynamic model most ridesharing requests are accommodated satisfactorily, another desirable feature.

## 6 Conclusion

In this paper we have defined fair ridesharing plans, analyzed algorithms for their evaluation, and compared them with the optimal ridesharing plans that are most likely used by Transportation Network Companies such as Uber. The comparison was done both theoretically and experimentally. A fair ridesharing plan is one in which the grouping of passengers that ride together does not violate individual self-interests of the passengers. We have shown that when social considerations are taken into account in grouping the passengers, the fair plan should be used to avoid serious anomalies and inconsistencies in grouping. For reasons of fair-plan existence we distinguished between fair plans in which the ridesharing benefit is evenly split among the passengers, and the ones in which it is unevenly split. Specifically, an uneven-split fair plan does not always exist, whereas an even-split always exists.

Theoretically, the gap between the fair and optimum ridesharing plans is evaluated by the traditional Price of Anarchy and Price of Stability measures. Since we have shown that the fair plan is unique, there is no difference between the two. We have shown that the PoA may be

unbounded in the uneven-split benefit case, but it is tightly bounded by 2 in the even-split case. This means that the ratio between the benefit of the optimum plan and the fair one is unbounded in the first case, but is at most 2, and can be 2, in the second case. The tight bound of 2 holds even in the purely geometric sense, i.e. when the benefit is measured in terms of the mileage saved on a realistic road network, and realistic trip requests.

The experimental comparison between the plans used the NYC 2013 taxi demand data. We have found that fair ridesharing plans produce very close VMT savings compared to the optimum, in both the static and dynamic models. Specifically, we have found that the overall difference in VMT saving between the optimum and fair plans is no more than 2% in both the static and the dynamic ridesharing models. This means that the fair ridesharing concept, which can attract more ridesharing passengers, can do so without a significant loss in efficiency. Furthermore, fairness can increase ridesharing participation, thus further reduce VMT. For example, our results indicate that if ridesharing requests increases from 10% to 15%, then the %VMT saved in the static fair ridesharing plans increases by over 40%.

The gap between the even- and uneven-split fair plans is also found to be small in terms of VMT saving. However, the uneven-split fair plan is computationally more expensive and sometimes intractable, and may not even exist. These findings suggest that an even-split fair plan may be practically the most reasonable approach. These results also indicate that for ridesharing worst case analysis is probably not a good predictor of practical performance.

In this paper, we considered two fixed payoff splitting strategies, the even and uneven benefit split. Observe that the split itself is a subject of mechanism design to incentivize or disincentivize a ridesharing partnership. An interesting theorectical future research direction is how to minimally modify an optimum ridesharing plan to make it fair.

Experimentally, the most important future work direction is to implement the fair plan in real ridesharing in order to determine whether indeed it will increase ridesharing acceptance, and how it should be fine-tuned.

**Acknowledgements** This study was supported in part by the NSF Grants IIS-1213013 and IIP-1534138.

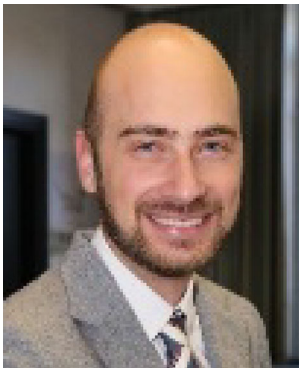
## References

1. Ayala D, Wolfson O, Xu B, Dasgupta B, Lin J (2011) Parking slot assignment games. In: Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems, pp 299–308
2. Santi P, Resta G, Szell M, Sobolevsky S, Strogatz SH, Ratti C (2014) Quantifying the benefits of vehicle pooling with shareability networks. Proceedings of the National Academy of Sciences (PNAS) 111(37): 13290–13294
3. Lin, J., Sasidharan, S., Ma, S., & Wolfson, O. (2016). A model of multimodal ridesharing and its analysis. In 2016 17th IEEE International Conference on Mobile Data Management (MDM), vol. 1 IEEE, p 164–173
4. Qian X, Zhang W, Ukkusuri SV, Yang C (2017) Optimal assignment and incentive design in the taxi group ride problem. Transp Res B Methodol 103:208–226
5. Alonso-Mora J, Samaranayake S, Wallar A, Frazzoli E, Rus D (2017) On-demand high-capacity ride-sharing via dynamic trip- vehicle assignment. Proceedings of the National Academy of Sciences (PNAS) 114(3):462–467
6. Huang Y, Bastani F, Jin R, Wang X (2014) Large scale real-time ridesharing with service guarantee on road networks. Proceedings of the VLDB Endowment 7(14):2017–2028
7. Ma S, Zheng Y, Wolfson O (2013) T-share: a large-scale dynamic taxi ridesharing service. 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 410–421

8. Ma S, Zheng Y, Wolfson O (2015) Real-time city-scale taxi ridesharing. IEEE Trans Knowl Data Eng 27(7): 1782–1795
9. Masoud N, Jayakrishnan R (2017) A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. Transp. Res. B Methodol 106:218–236
10. Bistaffa F, Farinelli A, Chalkiadakis G, Ramchurn SD (2017) A cooperative game-theoretic approach to the social ridesharing problem. Artif Intell 246:86–117
11. Ben Cheikh S, Tahon C, Hammadi S (2017) An evolutionary approach to solve the dynamic multihop ridematching problem. Simulation-transactions of the Society for Modeling and Simulation International 93(1):3–19
12. Nourinejad M, Roorda M (2016) Agent based model for dynamic ridesharing. Transportation Research Part C – Emerging Technologies 64:117–132
13. Thaithatkul P, Seo T, Kusakabe T, Asakura Y (2015) A passengers matching problem in ridesharing systems by considering user preference. J East Asia Soc Transp Stud 11:1416–1432
14. Thaithatkul P, Seo T, Kusakabe T, Asakura Y (2017) Simulation approach for investigating dynamics of passenger matching problem in smart ridesharing system. Transportation Research Procedia 21:29–41
15. Czioska P, Mattfeld D, Sester M (2016) GIS-based identification and assessment of suitable meeting point locations for ride-sharing, 19th euro work. Gr Transp Meet EWGT 2016
16. Barann B, Beverungen D, Muller O (2017) An open-data approach for quantifying the potential of taxi ridesharing. Decision Intelligence 246:86–117
17. Ma S, Wolfson O (2013) Analysis and evaluation of the slugging form of ridesharing. In Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems, pp. 64–73
18. Foti L, Lin J, Wolfson O, Rishe N (2017) The Nash equilibrium among taxi ridesharing partners. Proceedings of the 25th ACM SIGSPATIAL international conference on advances in geographic information systems, https://doi.org/10.1145/3139958.3140028
19. Pelzer D, Xiao J, Zehe D, Lees MH, Knoll AC, Aydt H (2015) A partition-based match making algorithm for dynamic ridesharing. IEEE Trans. Intell. Transp. Syst 16(5):2587–2598
20. Tian C, Huang Y, Liu Z, Bastani F, Jin R (2013) Noah: in proceedings of the 2013 ACM SIGMOD international conference on management of data. ACM, pp. 985–988
21. Agatz N, Erera A, Savelsbergh M, Wang X (2012) Optimization for dynamic ride-sharing: a review. Eur J Oper Res 223(2):295–303
22. Cerquides J, Farinelli A, Meseguer P, Ramchurn SD (2013) A tutorial on optimization for multi-agent systems. Comput J 57(6)
23. Chalkiadakis G, Elkind E, Wooldridge M (2011) Computational aspects of cooperative game theory. Synthesis Lectures on Artificial Intelligence and Machine Learning 5:1–168
24. Bogomolnaia A, Jackson M (2002) The stability of hedonic coalition structures. Games and Economic Behavior 38(2):201–230
25. Fu X, Huang J, Lu H, Xu J, Li Y (2017) Top-k taxi recommendation in Realtime social-aware ridesharing services. In Int. conf. on SSTD. Springer
26. Bistaffa F, Farinelli A, Ramchurn SD (2014) Sharing rides with friends: a coalition formation algorithm for ridesharing, Proceedings of the twenty-ninth AAAI conference on artificial intelligence, pp. 608–614
27. Wolfson O, Lin J (2017) Fairness versus optimality in ridesharing a dynamic ridesharing system, Proceedings of the 18th IEEE international conference on Mobile data management, pp. 118–123, https://doi.org/10.1109/MDM.2017.25
28. Kamar E, Horvitz E (2009) Collaboration and shared plans in the open world: studies of ridesharing, Proc. of the 21st IJCAI. Morgan Kaufmann Publishers Inc, San Francisco
29. Asghari M, Deng D, Shahabi C, Demiryurek U, Li Y (2016) Price-aware real-time ride-sharing at scale: an auction-based approach. Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems, ISBN: 978-1-4503-4589-7, https://doi.org/10.1145/2996913.2996974
30. Asghari M, Shahabi C (2017) An on-line truthful and individually rational pricing mechanism for ride-sharing. Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL'17, pages 7:1–7:10
31. Zhang R, Pavone M (2016) Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. The International Journal of Robotics Research 35(1–3):186–203
32. Nash J (1950) Equilibrium points in n-person games. Proc Natl Acad Sci 36(1):48–49
33. Ayala, D., Wolfson, O., Dasgupta, B., Lin, J., & Xu, B. (2018) Spatio-temporal matching for urban transportation applications. ACM Transactions on Spatial Algorithms and Systems (TSAS) 3(4):11
34. Chandra B, Halldorsson MM (2000) Greedy local improvement and weighted set packing approximation. Journal of Algorithms 39:223–240

35. Galil Z (1986) Efficient algorithms for finding maximum matching in graphs. ACM Comput Surv (CSUR) 18(1):23–38
36. Gusfield D, Irving RW (1989) The stable marriage problem: structure and algorithms. MIT press
37. Bistaffa F, Farinelli A, Cerquides J, Rodríguez-Aguilar JA, Ramchurn SD (2016) Algorithms for graph-constrained coalition formation in the real world. ACM Trans Intell Syst Technol pp:1–23
38. Iwama K, Miyazaki S, Okamoto K (2007) Stable roommates problem with triple rooms. Proc. 10th KOREA-JAPAN joint workshop on algorithms and computation (WAAC 2007), pp. 105–112
39. Swoboda AJT (2015) New York City taxicab transportation demand modeling for the analysis of ridesharing and autonomous taxi systems. **B.S. thesis**, Department of Operations Research and Financial Engineering, Princeton University





**Luca Foti** received his joint MS degree in Computer Science from University of Illinos at Chicago and Politecnico di Milano, Italy, in 2017. He is currently a Data Scientist and Project Manager at Laife Reply Milano, Lombardy, Italy. Focused on Artificial Intelligence in Healthcare area, he is involved in the development of a Deep Learning platform to detect and classify tumors in medical images (X-RAIS); moreover, he is the Activity Leader of EU-funded projects in Occupational Medicine area collaborating with international partners.

**Jane Lin** Ph.D., is Professor in transportation engineering and systems in Department of Civil and Materials Engineering at University of Illinois at Chicago. She has published over 75 refereed papers. She is Editor of Transport Policy, and Associate Editor of Transportation Research Part D. She received her MS and PhD from University of California, Davis, and BS from Tsinghua University, Beijing, China. She was a post-doctoral fellow at Harvard University before joining UIC.

**Ouri Wolfson** Ph.D., is the Richard and Loan Hill Professor of Computer Science at the University of Illinois at Chicago, and an Affiliate Professor in the Department of Computer Science at the University of Illinois at Urbana Champaign. He is the founder of Mobitrac, a venture-funded high-tech startup that was acquired in 2006, and of Pirouette Software. Wolfson authored over 220 publications, and holds seven patents. He is a Fellow of the ACM, AAAS, and IEEE, a University of Illinois Scholar, and a past ACM distinguished lecturer. He co-authored six award winning papers. Wolfson's research is in big/mobile data.