The Nash Equilibrium Among Taxi Ridesharing Partners

Luca Foti
Politecnico di Milano
Milan, Italy
luca.foti@mail.polimi.it

Jane Lin, Ouri Wolfson
University of Illinois
Chicago, IL, USA
{janelin,wolfson}@uic.edu

Naphtali D. Rishe
Florida International University
Miami, FL, USA
rishen@cs.fiu.edu

ABSTRACT

Ride sourcing services such as Uber and Lyft have become widespread in large cities for everyday mobility. When matching passengers, these services attempt to optimize cost savings at a global level. However, a possible scenario is that a passenger A is matched to passenger B even though if A were matched to passenger C, then both A and C would have saved more money. This introduces the concept of "fairness" in ride sharing, which consists of finding the Nash equilibrium in ridesharing. In this paper we compare optimum and fair ridesharing theoretically and experimentally. We show that although theoretically the gap between fair and optimum is large, in practice it is very small.

CCS CONCEPTS

• **Applied computing → Transportation**; • **Information systems → Spatial-temporal systems**;

KEYWORDS

Optimum, Fairness, Static, Dynamic, Ridesharing

1 INTRODUCTION

The way ridesharing is currently implemented by Mobility on Demand (MOD) companies such as Uber and Lyft have significant drawbacks. Specifically, passengers' preferences, benefits, and constraints are not considered in selecting their ridesharing partners; for example, everything else being equal, a passenger may prefer a ridesharing partner in their social network over a stranger. As we demonstrate in Sec. 2 there may be a conflict between the way MOD companies would group passengers, namely optimal ridesharing, and the way passengers would be grouped if their preferences of ridesharing partners were considered, namely fair ridesharing. This means that fair ridesharing may be suboptimal and optimum ridesharing may be unfair[1]. The concept of fairness results from the application of Nash Equilibrium to ridesharing. Fairness and optimum in ridesharing are equivalent to user equilibrium and system optimum in traffic and parking ([2]).

[1]Of course, in fair ride-sharing, the preferences will not be specified for each ride. Instead, when installing the ridesharing app (e.g. Uber) the user will specify preferences that are taken into consideration in all matchings. The concept of fair ridesharing was introduced in [12].

SIGSPATIAL'17, November 7–10, 2017, Los Angeles Area, CA, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5490-5/17/11.
https://doi.org/10.1145/3139958.3140028

In this paper we discuss two variants of fairness, the evenly split and the unevenly split, and quantify the gap between fair and optimum ridesharing. We do so theoretically and experimentally. On the theoretical side we show that for the unevenly-split variant of fairness, the price of anarchy (poa is the maximum ratio between the benefit of the optimal plan and the fair one) is unbounded, whereas for the evenly split one it is bounded by a constant. Then we examine the gap from a practical perspective, using the NYC taxi database of over 700 million trips (see [11]). We do so in terms of mileage (or \$-savings assuming a linear relationship between the two). We determine that the gap is at most 2% on average.

The comparison between fair and optimum ridesharing is carried out in two different models, static (see e.g. [6], [9], [10]), and dynamic (see e.g. [1], [5], [7], [8]). To our knowledge, these models are compared here for the first time.

The rest of the paper is organized as follows. In Sec. 2 we introduce the model and prove the poa results. In Sec. 3 we describe the algorithms used to compute groups for ridesharing, and the experimental results.

2 THE MODEL AND PRICE OF ANARCHY

A (trip) *request* consists of: 1. *pick-up* and *drop-off* locations, 2. *pick-up time availability*, and 3. *passenger-count.* In the experiments, for each request we also consider a bound on the delay due to ridesharing, but this parameter is not necessary in the model.

Sometimes a vehicle is matched to a request as soon as the latter arrives, and before the next request is serviced. Another option, which is the one followed in this paper, and better facilitates ridesharing, is to group requests issued for a specific time interval into a "pool".

Pools and Ridesharing graphs. Given a time interval starting at "s = start pool time" and ending at "e = end pool time", a *pool(s,e)* is the set of requests whose pick-up time availabilities are between s and e. For example, if a pool starts at "10:00:00" and ends at "10:04:59" then it consists of all the requests which have a pick-up time availability within this interval. The pairwise ridesharing opportunities of a pool are represented by a weighted graph (see [1], [12]), called the *ridesharing graph (RSG)* (see e.g. Figure 1). In an *RSG(V,E)*, each node $v_i \in V$ is a single request, and each edge, $e_{i,j} \in E$ connects two potential ridesharing partner requests v_i and v_j. The weight of edge $e_{i,j}$, denoted $w_{i,j}$, is the *benefit* obtained by combining requests v_i and v_j, compared to the execution of the two single requests separately. The benefit $w_{i,j}$ is split between the *individual-benefit* of v_i, denoted $w_{i,j}(i)$ and the *individual-benefit* of v_j, denoted $w_{i,j}(j)$, such that $w_{i,j}(i) + w_{i,j}(j) = w_{i,j}$. Intuitively, this means that the individual benefits of the two partners sum up to the benefit of the partnership. Furthermore, each individual benefit must be positive in order for $e_{i,j}$ to exist; i.e., if one of these is negative, then v_i and v_j should not rideshare[2]. In Figure 1, on each edge $e_{i,j}$ between requests v_i and v_j, a label which encloses two values in square brackets is noted: the value closest to request v_i (resp. request v_j) indicates the individual benefit of request v_i (resp. request v_j) in this shared trip.

A *ridesharing plan (rsp)*, is a set S of edges in the ridesharing graph, such that no request appears in different edges of S. Intuitively, each edge e represents a trip combining the endpoints of e, thus the edges are selected such that no request belongs to more than one edge. The *benefit of the rsp* is the total benefit of the benefit of the edges in S.

If more than two requests can be combined, then we have a ridesharing hyper-graph (RSH): nodes still represent requests and hyper-edges represent a potential shared trip among the connected (two or more) requests. Benefits and rsp are defined analogously to the pairwise case.

Static and dynamic models in ridesharing. There are two possible approaches to deal with group requests: the static and the dynamic models. Within the *static* model, requests which specify a pick-up time within a given interval are pooled together and then assigned to empty vehicles. Hence, vehicles that are already processing some other requests are not taken into account for this assignment, i.e., their initial route is static and cannot be modified. A practical scenario for this model is a taxi station, e.g. at an airport. A person's flight has landed at 10am, but she needs a ride to go from the airport (hub) 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 [6]. Rides in a pool are combined, and assigned to empty taxis waiting in a queue at the terminal. Since empty vehicles are always available at zero-distance, a request or a group of requests is never assigned to a vehicle that is in progress of serving other requests.

Within the *dynamic* model, new requests can be combined and then assigned to a vehicle; the vehicle may already be occupied, i.e. serving some other requests. Thus, a new route has to be computed in order to satisfy all the requests assigned to the vehicle. Hence, the occupied vehicle's route changes dynamically, i.e. while being executed. In other words, in contrast to the static model, some of the requests that are grouped may already be executing, i.e. ongoing.

The fair rsp. We employ the definition of fairness as introduced in [12]. In particular, in a *fair* ridesharing plan, there is no pair of unmatched requests which would both benefit more if they were matched. In Figure 1 the rsp that combines A-B and C-D is fair, providing an overall benefit of 14, but the rsp A-D and B-C is unfair because A and B can both benefit more by ridesharing with each other. Observe that fairness requires the pairing of A and B only in the case of reciprocity; if A prefers B (i.e. A saves most when partnered with B), but B prefers another partner, then it may be fair to pair A with D. But since both A and B prefer each other over other partners, it is unfair to pair them with other partners.

[2]The individual-benefit may consist simply of the dollars saved by combining the two requests; or it may include other preference-criteria such as Facebook-friendship and gender. Specifically, the benefit may be a weighted sum of dollar-saved and social-score (as in [4]). And in turn, the social-score can be a weighted sum of values assigned to preference-criteria such as gender and Facebook-friendship. Social-score can be calculated as a matching score by Match.com.

For reasons of complexity and existence of a solution, we distinguish between an *evenly split* RSG, i.e. an RSG in which for each edge $e_{i,j}$, $w_{i,j}(i) = w_{i,j}(j)$, as in Figure 1; and an *unevenly-split* RSG, i.e. an RSG which is not evenly split. Both cases make practical sense. For example, the taxi attendant at O'Hare airport in Chicago 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 evenly-split RSG. On the other hand, it may make sense for the passenger who gets dropped off first to save less. This results in an unevenly-split RSG.

The optimum rsp. Given an RSG, the *optimum* rsp is the rsp that has the maximum-benefits among all rsp's.

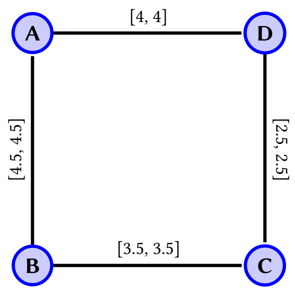


Figure 1. Evenly-split Ridesharing graph

In Figure 1, the optimum rsp combines A-D and B-C, providing the largest total benefit of 15; on the other hand, A and B benefit more if matched together, so the fair plan combines A-B and C-D, providing an overall benefit of 14.

Observe that changing the split of a single edge may produce different fair rsp's, even if the total benefit of the edge does not change. For example, if in Figure 1 the edge (A,B) is labeled [3,6], instead of [4.5,4.5], i.e. the benefit of A in ridesharing with B is 3, then the optimum plan is also fair. Moreover, it is noteworthy that in an unevenly split RSG, a fair rsp may not exist, even though it always exists in an evenly split RSG (see [12]). In contrast, an optimum rsp always exists.

The Price of Anarchy (PoA) is the ratio between the optimal solution and the worst equilibrium, i.e. $PoA = max_{p \in P}\ Benefit(p)$ / $min_{p \in Equil}\ Benefit(p)$, where P is the set of all rsp's, $Equil \subseteq P$ is the set of fair rsp's, and $Benefit(p)$ is the benefit of rsp p.

Theorem 1. *In an evenly split RSG, the PoA is bounded by 2.*

(Proof:) The algorithm used to compute the fair rsp is described in [12]. It selects edges of the RSG in decreasing order of benefit. This algorithm is discussed in [3] in a different context than fairness, and is shown to be a *2*-approximation of the optimum.

Theorem 2. *In an unevenly split RSG, the PoA is unbounded.*

(Proof:) Assume that in fig. 1 passenger D's individual benefit for ridesharing with A is not 4, but an arbitrarily large value X (e.g., D's social score 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. arbitrarily large.

3 ALGORITHMS AND EXPERIMENTS

In this section we present the algorithms used to produce the optimum and the fair ridesharing plans for our experiments conducted on the NYC taxi database. In the experiments we assume that a taxi can hold at most 4 passengers. The focus is on the dynamic model, since the algorithms for the static model are discussed in [12] with the following main results: the ridesharing hypergraph (RSH) can be computed in $O(n^4)$ road-network shortest path computations; computing the optimum rsp is NP-complete in general, but since the number of nodes in a hyperedge is limited to 4, the computation is manageable; the evenly split RSH fair rsp is computed in $O(n^4 \; log \; n)$, where n is the number of requests (nodes) in the RSH; the unevenly split RSH fair rsp is NP-complete; thus in the experiments we compute it only for the pairwise ridesharing.

In the dynamic model, in addition to grouping, the algorithm for finding the rsp also needs to assign a taxi to a group of new requests in a pool. The total number of taxis is fixed and limited, thus the assigned taxi may already be occupied.

Computing the extended RSH: In this context, a combination of new requests is meaningful when assigned to a taxi and, furthermore, different taxi assignments can provide different benefits. In order to express this idea, we first compute the RSH with respect to new requests only, as in the static case. Then we extend it as follows to create the *extended-RSH (e-RSH)*. The nodes of e-RSH are new requests and taxis. For each pair of (taxi, RSH-hyperedge), such that the taxi is *feasible* for the hyperedge (i.e. the taxi can satisfy the delay bounds of all the requests involved (i.e. onboard and new)), we create a new hyperedge in e-RSH. The weight of the hyperedge is the distance saved by combining the new and the onboard requests. Obviously, if the taxi is empty, then the weight is computed on the new requests only as in the static case. If the taxi is occupied, then the weight is difference $S - L$ where S is the sum of the shortest paths of the new requests and the remaining taxi path to satisfy the onboard requests; and L is the shortest path to satisfy all the new and onboard requests, from the current taxi location. If the weight of the hyperedge is negative, then it is eliminated. If there exists no taxi that is feasible for an RSH hyperedge, then no new hyperedge is added to e-RSH. Thus, if some request is disconnected in e-RSH (i.e., there is no new hyperedge involving the request), it cannot be allocated a taxi in current pool, and is moved to the next one.

The optimum rsp: Starting from the e-RSH, we formulate an adaptation of the Integer Linear Programming (ILP) problem provided in [1]: in contrast to [1], we aim to maximize overall benefits. In order to speed up the process of finding the optimum rsp, we start from an initial greedy solution: we sort the hyperedges of the e-RSH by total benefit in decreasing order (i.e., from the heaviest to the lightest weight); we add to the solution the hyperedges following this order and, as they are inserted, we remove nodes (i.e., new requests and taxi) and hyperedges they are related to from the remaining e-RSH. Then, the ILP presented in Table 1 improves this initial solution to find the optimal one.

In order to explain the objective function and the constraints of the ILP, it is useful to refer, for each hyper-edge h of the e-RSH, to: 1. the set of new requests, or *nr-set* for short, in h; and, 2. the *taxi* in h. Observe that the set of nr-sets, which the description below indexes and uses, is actually the set of hyper-edges in the RSH (not the e-RSH). The ILP *assigns* taxis to nr-sets optimally, i.e. in a way that maximizes the total benefit.

Table 1. Dynamic model: ILP for maximum matching

(1) Initial guess: $\sum_{greedy}$

(2) $\sum_{optim} := \underset{\mathcal{X}}{\arg\max}\, C(\mathcal{X})$,

where $C(\mathcal{X}) = \sum\limits_{i,j \in \mathcal{E}_{TV}} b_{i,j}\epsilon_{i,j} - \sum\limits_{k=\{0,\ldots,n\}} \chi_k$

(3) s.t. $\sum\limits_{i \in \mathcal{I}_j^V} \epsilon_{i,j} \leq 1, \forall v_j \in \mathcal{V}$

(4) $\sum\limits_{i \in \mathcal{I}_k^R, j \in \mathcal{I}_i^T} \epsilon_{i,j} + \chi_k = 1, \forall r_k \in \mathcal{R}$

The constraints of this ILP problem are: (3) each taxi is assigned to at most one nr-set; and (4) each new request is either a member of some nr-set that is assigned to a taxi, or is not assigned at all.

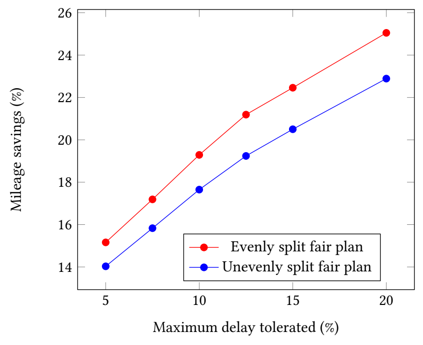
The evenly split extended RSH fair rsp: We employ an algorithm that is analogous to the one used in the static model. Specifically, we sort the hyperedges by individual benefit in decreasing order and select the hyperedges according to this order, iteratively: we remove those hyperedges that connect either a new request or a taxi that has just been inserted in the solution. The solution consists of the selected hyper-edges, in the sense that for each hyper-edge in the solution, its nr-set is assigned to its single taxi.

The unevenly split extended RSH fair rsp: The problem of finding a fair rsp in the unevenly split RSH is NP-complete. Thus in the experiments only two requests can be processed at the same time; i.e., if a taxi is empty, then at most two new requests can be combined and assigned to the taxi, and if it is already occupied with an ongoing request, at most one new request can be assigned. We employ the same algorithm devised for the static model.

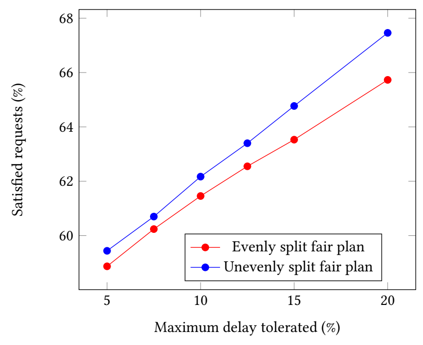
Dynamic model experimental settings. We have considered the Manhattan road network, which is a directed graph consisting of 3,933 road intersections and 8,400 road segments. Only requests originating and terminating in Manhattan are considered. In particular, we have simulated two hours on January 25th 2013, from 10am to noon. In this time span, about 40,000 requests have been issued. We have analyzed pools of 30 seconds. On average about 167 new requests are issued each 30 seconds in the two hours simulated.

We considered three scenarios in which all the pools are processed by: a fair evenly split rsp, or a fair unevenly split rsp, or an optimum plan. In the simulations, for simplicity we assume that each request involves a single passenger. For the unevenly split RSG, we distribute the total benefit proportionally to the increase (compared to the shortest path) in the distance traveled by each passenger in the combined path. If an unevenly-split rsp does not exist, an evenly split one is used instead. 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, 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.

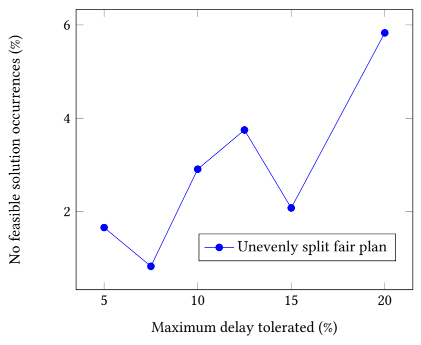
In contrast, in the static model, since empty taxis are always available, no request is propagated from pool to pool; if a request cannot find a rideshare partner, it rides alone in a taxi. Thus the set



(a) % mileage savings



(b) % satisfied requests



(c) % no fair solutions (unevenly split)

Figure 2. Dynamic model: comparison between evenly and unevenly split fair plans

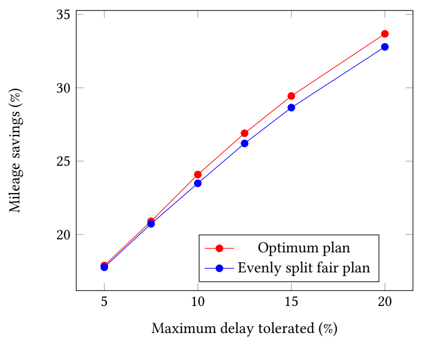


Figure 3. Dynamic model: % mileage savings

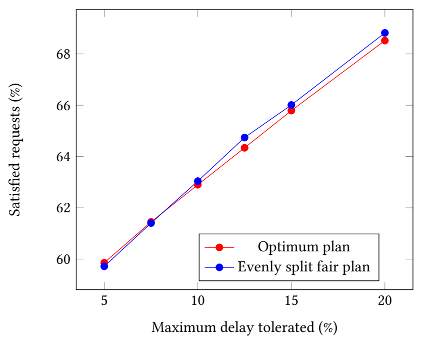


Figure 4. Dynamic model: % satisfied requests

of requests in each time-interval pool is fixed, and identical in the fair and optimum simulations.

Another implication of the propagation from pool to pool in the dynamic model is that, if the delay tolerated is exceeded, then the request is *unsatisfied*. Again, due to ample supply of taxis, this cannot happen in the static model.

In the simulations, 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, then it follows a shortest path from its current location to a random road intersection.

The total distance saved, computed after the end of the 2-hour simulation time, is given by:

$$\left(\sum_{i=\{1,\dots,n\}} miles_{sp,i}\right) - \left(\sum_{j=\{1,\dots,m\}} miles\_traveled_j\right), \qquad (1)$$

where $miles_{sp,i}$ indicates the shortest path in miles for satisfying request i alone, and $miles\_traveled_j$ indicates the miles traveled by taxi j with some passengers on board. n and m are the numbers of satisfied requests and taxis, respectively.

The results are summarized in figures 2, 3 and 4. Mileage savings gives the decrease compared to the total distances of individual requests; delay tolerated is the time increase (compared to the time of each individual request) allowed to facilitate ridesharing (time to traverse a link is taken as 70% of the time at maximum speed). Indeed, the performances of the fair rsp's (both even and uneven) are almost equal to that of the optimum rsp's.

Acknowledgement. Supported by NSF IIS-1213013, IIP-1534138

REFERENCES

[1] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences* (2017).

[2] D. Ayala, O. Wolfson, B. Xu, B. Dasgupta, and J. Lin. 2011. Parking slot assignment games. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM.

[3] B. Chandra and M. M. Halldórsson. 2001. Greedy local improvement and weighted set packing approximation. *Journal of Algorithms* 39, 2 (2001).

[4] X. Fu, J. Huang, H. Lu, J. Xu, and Y. Li. 2017. Top-k Taxi Recommendation in Realtime Social-Aware Ridesharing Services. In *Int. SSTD*. Springer.

[5] Y. Huang, F. Bastani, R. Jin, and X. S. Wang. 2014. Large scale real-time ridesharing with service guarantee on road networks. *Proc. VLDB* (2014).

[6] J. Lin, S. Sasidharan, S. Ma, and O. Wolfson. A Model of Multimodal Ridesharing and its Analysis. In *IEEE MDM 2016*.

[7] S. Ma, Y. Zheng, and O. Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *IEEE ICDE (2013)*.

[8] S. Ma, Y. Zheng, and O Wolfson. 2015. Real-time city-scale taxi ridesharing. *IEEE Transactions on Knowledge and Data Engineering* 27, 7 (2015).

[9] X. Qian, W. Zhang, S. Ukkusuri, and C. Yang. 2017. Optimal assignment and incentive design in the taxi group ride problem. *Transportation Research Part B* (2017).

[10] P. Santi, G. Resta, M.l Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti. 2014. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences* 111, 37 (2014).

[11] A. J. T. Swoboda. 2015. New York City Taxicab Transportation Demand Modeling for the Analysis of Ridesharing and Autonomous Taxi Systems. (2015).

[12] O. Wolfson and J. Lin. 2017. Fairness versus Optimality in Ridesharing. *Proc. MDM* (2017).