Stiftung Universität Hildesheim

Student: Josué Daniel Rodríguez Quintana

Matriculation Number: 304121

Seminar Data Analytics III

Analyzed paper: Balancing Efficiency and Fairness in On-Demand Ridesourcing

Instructor: M. Eng. Hadi Samer Jomaa

August 2020

Index

Introduction

Platforms that distribute resources such as vehicles to transport passengers or goods have increased popularity over the last years, not only from the side of consumers but also from the drivers perspective who for diverse reasons take part of the chain, mostly to generate an extra income. Some popular platforms have more than one hundred thousand active workers [2] whose final income depends heavily on the current market demand and the decisions taken by the assignment algorithms of the platform they work for. Naturally, the main goal of those companies and their systems is to maximize their revenue, thus assigning the nearest available vehicle to a new user request. Although this is logical from the efficiency perspective, it brings fairness issues, as an example, some drivers might be assigned more or better rides just for being nearer to the passengers opportunistically, whereas some new unlucky drivers might be given less rides or unwanted destinies. As a result, some drivers would benefit over others and produce an inequality situation among drivers. In the end, this can result on drivers deregistering from the platforms. The paper discussed here, written by Lesmana, Zhang and Bei, delve into a concrete solution to trade between efficiency and fairness.

Terminology

Ridesourcing: Term that describes a mode of transportation where users and drivers are connected through a platform.
Assignment: In the context of ridesourcing, defines the decision made by the platform to connect specific users to specific drivers.

Summary of the paper

Related work

The resource allocation algorithms for vehicles have been widely studied and embedded in several systems, good examples of them are the greedy match (Lee, Wang, Cheu and Teo, 2002) which make use of real traffic conditions to match vehicles and requests, to avoid assignment errors derived from the assumption that the shortest distance is the only important criteria (Auto drivers must follow the traffic regulations and very often this regulations prohibit them to take the shortest geographical path). Further research works enable taxis to behave as agents which can collaborate with each others (in a multi-agent environment) to negotiate the best decision to take or leave a request. (Seow, Dang and Lee, 2010). Other research publications concentrate in centralized systems that take the decisions based on a learning (using historical data) and a planning approach. This last example was deployed as the dispatch algorithm of a very popular taxi platform in China (Xu, Li, Gua, Zhang, et al. 2018).

There have also been developments on offline systems which take decisions in batches, meaning that the assignment algorithm runs only after the batch is complete in specific

times. This problem is better known as the "Dial-a-Ride Problem", because it is mostly used in centralized taxi drivers societies which take the requests via telephone with an anticipated time (for example, one day before the ride).

Also the concept of fairness has been applied for several allocation algorithms, from multiportfolio optimization, which tries to balance conflicting objectives and distributing them across the participants in an equitable way (Iancu and Trichakis, 2014), to the envy-free cake cutting problem, where each element of the system (called agent) has its own preferences and the efficient allocation needs to be balanced with the preferences of the agents (Cohler, Lai, Parkes, and Procaccia, 2011). There was alsoresearch on the load balancing job scheduling prolem, where the fairness is expressed as the max-min sense, which means that the equilibrium between the maximum and the minimum workload is the goal (Kleinberg, Rabani, and Tardos, 1999). Nevertheless, the authors claim that the concept of fairness in ridesourcing systems has not been addressed in the literature, because the waiting time and pick-up time are constraints only relevant for the ridesourcing problem [1].

Methodology

Firstly, the authors introduce the notation used among the paper to describe the main three components of a ridesourcing system: Vehicles, Requests and their connection

- A set of n vehicles $V = \{v_1, v_2, ..., v_n\}$
- A set of m requests $R = \{r_1, r_2, ..., r_m\}$
- A set of edges which assign vehicles to requests: $E \subseteq \{\{v, r\} : v \in V, r \in R\}$
- A bipartite graph which describes the relation between vehicles and requests through Edges:
  - $G = (V, R, E)$

The goal of the systems described in the related work, makes clear that the efficiency is the primary goal from those systems, and to measure the efficiency is necessary to have a weight for each edge. This weight is based on the revenue the vehicle drivers would get by serving a specific trip. In an efficient scenario, there is no constraint that handles a disparity of revenue, where some vehicles would be more benefit than others, resulting in an unfair situation for some drivers. Therefore, the authors select a fairness concept to derive their contributions: The Rawlsian egalitarian justice, also known as distributive justice, is the criterion derived from Strict Egalitarianism, which "calls for the allocation of equal material goods to all members of society"[3], in the opposite, the principle from Rawlsian allows some relaxation on the strict equality when the introduced inequality's goal is to help the least advantaged. Under this principle, there is more to be discussed about which criterion to select to measure the welfare, and this is linked to our efficiency criterion. To our specific problem setting, the selected efficiency criterion is based on the

utilitarianism, an ethical theory that "holds that the most ethical choice is the one that will produce the greatest good for the greatest number"[4], which is translated to the ridesourcing problem as the sum of all the requests that a vehicle serves.

Secondly, the authors introduce a key concept, this is the **trade-off**, term that defines "a situation in which two opposing situations or qualities are balanced"[5], in the efficiency and the fairness in most cases the optimum cannot be found simultaneously, and a trade-off between both is what one could aim for. Not necessarily this trade-off has to be maintained at all times, and this responds to the necessity of switching the fairness off when the demand has a peak. This phenomenon is not exclusive from ridesourcing problem settings, but in several others where resources are distributed and a peak requires to concentrate resources in a section to overcome the peak with the maximum availability (as discussed in the load balancing job scheduling research [12]).

Therefore, the authors come with the question: "Given a fairness threshold, how to find an assignment to connect vehicles and requests which fulfills the fairness threshold and maintains an acceptable system efficiency."[1]

Preliminaries

As described before, the ridesourcing problem requires matching vehicles to requests in a bipartite graph. Graphically, that would be represented as in illustration 1.

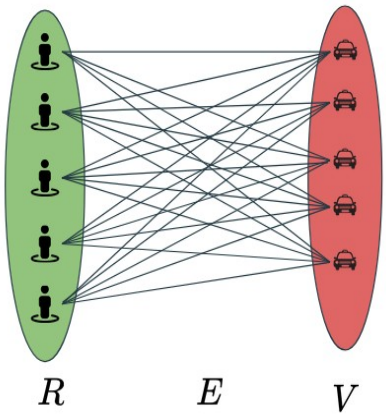


Illustration 1: Bipartite graph G

Each vehicle can be assigned to one request and this bipartite graph is said to be weighted because each edge contains a weight to join a vehicle with a request. This weight is formally represented as:

$$\{v, r\} \in E$$

This weight is also known as the *utility* u_{vr}, and this utility is computed with the record of trips served by the vehicle in previous trips, plus the current request to serve. For example, a vehicle could be now serving its third trip, and have a record of two previous trips. The current trip contribution is called *trip utility* and represented as w_{vr}. The previous trips contribution is defined as the *historical utility*, and represented as h_v .

Finally, the utility for this vehicle-request is defined as:

$$u_{vr} = w_{vr} + h_v \quad (1)$$

Now, to compute the trip utility the authors propose to penalize the time taken by the vehicle to displace until the request's departure point, this is called the *trip cost*, represented as: ι_{vr} . Additionally, the time needed to travel from the pick up location until the destiny of the request is defined as the *trip value* τ_r . To "balance the value-cost effect", the authors propose a constant c as a multiplier of the *trip value.*
Illustration 2 shows the outcome of the concepts and the final formula for w_{vr}:

$$w_{vr} = c\tau_r - \iota_{vr} \quad (2)$$

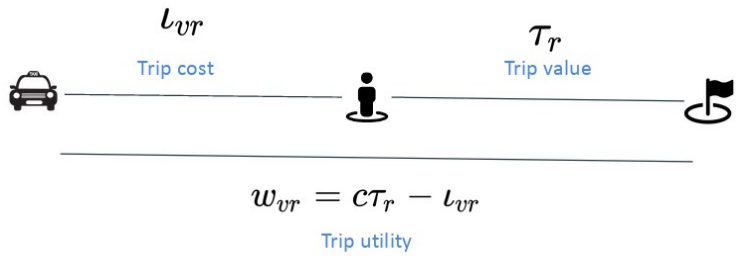


Illustration 2: Trip utility.

To analyse the tradeoff between efficiency and fairness, the parameter Δ is introduced, this is formally defined as:

$$\Delta := \max_{r \in R} \max_{\{v,r\},\{v',r\} \in E} |w_{vr} - w_{v'r}|$$

which means that each request is compared to different vehicles, and this absolute difference is maximized. Once found, the maximum difference from all requests is selected. Now it is possible to substitute w_{vr} and reduce the expression as follows:

$$\Delta := \max_{r \in R} \max_{\{v,r\},\{v',r\} \in E} |(c\tau_r - \iota_{vr}) - (c\tau_r - \iota_{v'r})|$$

$$\Delta := \max_{r \in R} \max_{\{v,r\},\{v',r\} \in E} |\iota_{v'r} - \iota_{vr}| \quad (3)$$

Resulting in the difference between trip cost. This parameter is useful to measure the disparity and formulate the theorems which support this contribution.

After this, to name the graph, trip utilities and historical utilities, the authors refer to this set of elements as an instance *I*.
As it was mentioned in the Terminology section, an *assignment* refers to the specific combination that connects vehicles with requests, this decision is a combination of edges where at most one vehicle is assigned to one request. As in illustration 3, this decision is called an assignment and denoted by the letter *M*:

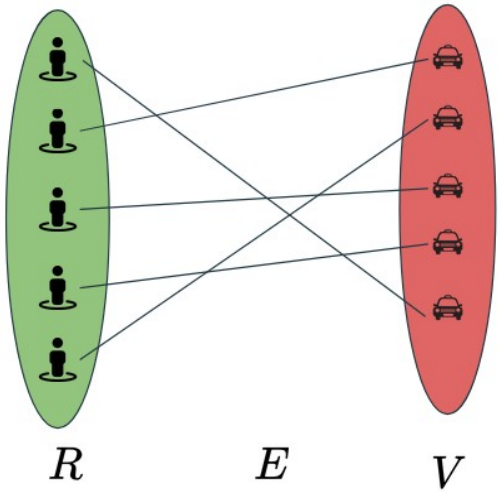


Illustration 3: Assignment M

This matching is the purpose of the whole algorithm, this has to be done under specific constraints. To measure the efficiency of this matching, the following metric is proposed:

$$\mathcal{E}(M) := \sum_{v \in V} u_{v,M(v)} = \sum_{v \in V} h_v + w_{v,M(v)} \qquad (4)$$

Meaning that the efficiency of the assignment M is the sum of the utility of all the vehicles.
Likewise, the metric for fairness is defined as the minimum utility among all the vehicles:

$$\mathcal{F}(M) := \min_{v \in V} \{u_{v,M(v)}\} = \min_{v \in V} \{h_v + w_{v,M(v)}\} \qquad (5)$$

Finally, finding the optimum efficiency-wise and fairy-wise is an important step, this can be done by comparing all the assignments, this is denoted by $\mathcal{M}$, containing all the assignments M to compare. From this, it is possible to find the:

- Optimal efficiency:
  - $\mathcal{E}_{opt} := \max\{\mathcal{E}(M) | M \in \mathcal{M}\}$ (6)
  - The assignment which matches the optimal efficiency is known as M_{eff}
- Optimal fairness (Eq. 6):
  - $\mathcal{F}_{opt} := \max\{\mathcal{F}(M) | M \in \mathcal{M}\}$ (7)
  - The assignment which matches the optimal fairness is known as M_{fair}

Ridesharing

The concept ridesharing is a compound term derived from “ride” and “share”, where multiple users can share the same ride, whenever the paths converge for them. This term is also known as “carpooling”. The main advantage for the users sharing rides is that they can split the costs proportionally to their use, and the trip cost ι_{vr} definition has to be

redefined to follow that concept. Therefore, the authors define a passenger as p to make the difference between the users sharing a ride. A set of passengers is defined as S_v, and a new physical constraint is added, which is the capacity of the vehicle. The set of passengers cannot exceed the capacity of the vehicle. In practice, each vehicle can differ on the number of seats available, but for simplification purposes, the authors define a constant for capacity as χ for all the vehicles of the system. (In practice, χ_v would be more useful). Also the delay time is introduced as constraint, meaning that users already in a ride can tolerate a delay in their trip derived from picking up the next passengers, but this delay should be within a specific threshold, this maximum waiting time is denoted as Ω.

Finally, the *trip utility* should include all the passengers, which is translated in finding the positive time difference between pickup time *pu(r)* and the maximum between the assignment time and the drop off time of the passenger *do(p)*.

$$\iota_{vr} := \max\{0, pu(r) - \max\{t, do(p)\}\} \quad (8)$$

Multi-period assignment

So far the problem setting has been defined to consider a one time assignment from all the requests (As described in the *Dial-a-ride problem*). In practice, a real time system should be capable of assigning vehicles among the day. Concretely, the authors suggest splitting the time into T intervals or periods (See Illustration 4).

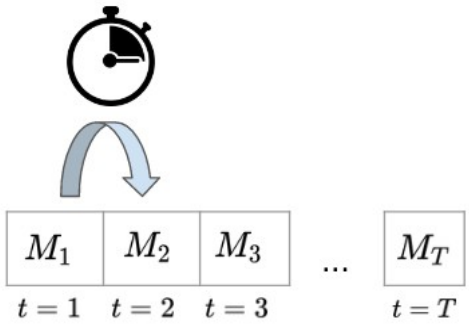


Illustration 4: Multi-Period Assignment

Each period is divided exactly in *n* seconds. The purpose of this division is to make assignments in batches, and to accomplish this, a waiting period between assignment and collecting new requests is needed.

The only change from the single period setting is that the historical utility is updated in each batch, meaning that at a time t, the historical utility for the next period is the current historical utility plus the trip utility of the next assignment period.

Formally, this is:

$$h_v^{t+1} = h_v^t + w_{vr}^{t+1} \quad (9)$$

The efficiency and fairness definitions stay unchanged.

Efficiency-Fairness Tradeoff

This is the main contribution from the authors, they begin by providing a mechanism to evaluate the fairness and the efficiency boundaries that can be achieved in this trade-off. To make a simple analogy, this trade-off can be interpreted the way one would select "hot" vs. " cold" and all the possible values in between in this scale. Instead of hot and cold, we would have "Efficient" vs. "Fair" and all the values in between. This could be visually seen as a slider commonly used in software:

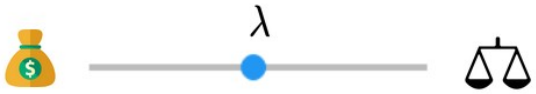


Illustration 5: Trade-off analogy.

This slider is formally defined as λ, and it is a continues variable ranging between zero and one, where one is the fairest possible value: $0 \leq \lambda \leq 1$.

Theorem 1

With this small concept in mind, the authors derive the first theorem, which states that for any assignment *M*, its fairness boundary is defined with the inequality:

$$\mathcal{F}(M) \geq \lambda \mathcal{F}_{opt} \quad (10)$$

Likewise, for efficiency tho boundaries are defined by:

$$\mathcal{E}(M) \geq \frac{2}{2+\lambda}(\mathcal{E}_{opt} - n\Delta) \quad (11)$$

Where *n* is the number of vehicles. Both definitions must be satisfied simultaneously.
To have a better understanding of this, these are the resulting formulations with the extreme values of λ.
For $\lambda = 0$ (No fairness required)

$$\mathcal{E}(M) \geq \frac{2}{2+0}(\mathcal{E}_{opt} - n\Delta)$$

$$\mathcal{E}(M) \geq \mathcal{E}_{opt} - n\Delta$$

The efficiency of any assignment is greater or equal to the optimum efficiency minus n times Δ.
On the contrary, with $\lambda = 1$ (Fairest)

$$\mathcal{E}(M) \geq \frac{2}{2+1}(\mathcal{E}_{opt} - n\Delta)$$

$$\mathcal{E}(M) \geq \frac{2}{3}(\mathcal{E}_{opt} - n\Delta)$$

The theoretical lowest efficiency that any assignment can reach is two thirds of the optimum, in other words, the fairest solution sacrifices one third of the efficiency under some circumstances.

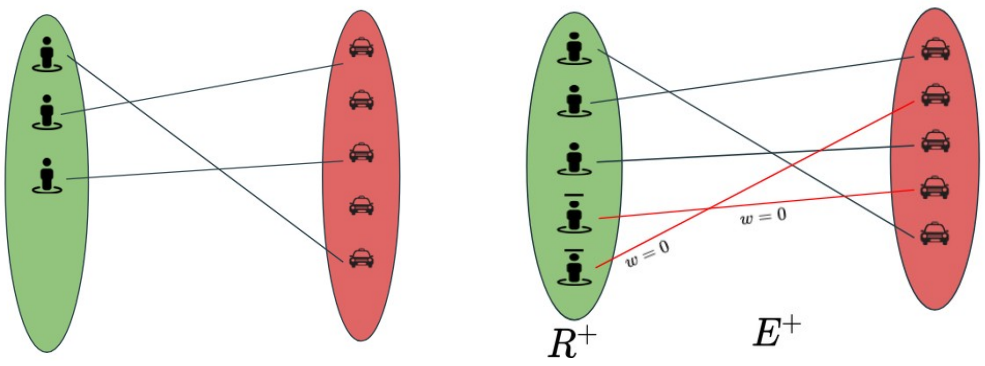
Fair Assignment Algorithm

Following is the developed algorithm to compute an assignment under the fairness constraint. To begin, the input parameters are the problem instance I (consisting of the bipartite graph G(V,R,E), the set of trip utilities and historical utilities per vehicle):

$$\mathcal{I} = \{G(V, R, E), \{w_{vr}\}_{\{v,r\}\in E}, \{h_v\}_{v\in V}\} \quad (12)$$

The fairness threshold λ must be also given and a pre-assignment M_{old}. In the paper the authors do not make clear enough that this pre-assignment is the efficient assignment, and this needs an algorithm to assign efficiently, namely the Hungarian Algorithm[13]. From this M_{eff}, there is a pre-algorithm to compute the M_{fair}, which is vaguely explained by the authors. Nevertheless, their implementation is clear enough to deduce it :

1. The bipartite graph must have by definition all their elements from both sides matching one element from each other. In practice, there can be more requests than vehicles (or vice versa), in the paper the authors assume there will be more vehicles available than requests, and propose to deal with this situation by creating "phantom" requests (or no-serve-requests), which would contribute with a utility of 0 and therefore their utility is equals to their historical utility. Left illustration shows an example of imbalance between requests and vehicles, whereas the right illustration shows the creation of the no-serve-requests and their trip utility set to zero.



This new set of requests is identified as R^+ and the set of edges as E^+.

2. Assign the efficient assignment as the initial fair assignment:
   - $M_{fair} \leftarrow M_{efficient}$
3. Initialize the fairness threshold as a negative infinite value:
   - $f \leftarrow -\infty$
4. Start to iterate while the minimum utility among all the vehicles is greater than the threshold *f:*
   - *While* $\min_{v \in V}\{h_v + w_{v,M_{fair}(v)}\} > f$
   1. Update the threshold f with the minimum utility from the fair assignment.
      - $f \leftarrow \min_{v \in V}\{h_v + w_{v,M_{fair}(v)}\}$
   2. For all the edges which utility is under the fairness threshold, delete them from the graph and assign their trip utility as 0.
      - For all $E \in E^+ | h_v + w_{v,r} < f$
        - $E \leftarrow \emptyset$
        - $w_{v,r} \leftarrow 0$
      - Compute a new assignment with the efficient assignment algorithm. Now the edges under the fair constraint will not be selected, therefore, this is assignment is our new M_{fair} for this iteration.
        - $M_{fair} \leftarrow M_{efficient}$
5. The optimum fairness is the minimum vehicle's utility from our fairest assignment.
   - $F_{opt} \leftarrow f$
6. The fairness threshold that the trade-off aims to reach is lambda times the optimum fairness threshold:
   - $f \leftarrow \lambda F_{opt}$

The authors compacted this whole algorithm in one line:

$$G_f := (V, R^+, E_f = \{\{v, r\} \in E^+ | h_v + w_{v,r} \geq f\}) \quad (13)$$

Reassign Algorithm

Up to this point, all the input elements from the **reassign algorithm** are computed, this algorithm needs the two opposite assignments M_{eff} and M_{fair}, to output a *new assignment* M_{new} following the fairness constraint. The steps to do this are described as follows (Remark: $M(v)$ means "the request r assigned to the vehicle v for this assignment M"):

1. Compute a fair assignment M_{fair} (As described bellow). And an efficient assignment M_{eff}.
2. Set the new assignment as the efficient one: $M_{new} = M_{eff}$.

3. Iterate while there is a vehicle which *utility* from the current assignment is under the fairness constraint f.
   - While $v \in \mathcal{V} \mid h_v + w_{v,M_{new}(v)} < f$:
   1. Obtain the request from the new assignment for the selected vehicle: $r \leftarrow M_{new}(v)$
   2. Delete the edge between the vehicle and that request:
      - $M_{new}(v) \leftarrow \emptyset$
   3. While there is another vehicle v'which request from the *new assignment* matches the fair assignment:
      - While $v \in \mathcal{V} \mid h_v + w_{v,M_{new}(v)} < f$
      1. Delete the edge between the vehicle v' from the *new assignment*: $M_{new}(v') \leftarrow \emptyset$
      2. Assign the fair request from vehicle v to the *new assignment*: $M_{new}(v) \leftarrow M_{fair}(v)$
      3. Now that v was assigned, v' needs a new request, so it is transferred to be the new v that the "while condition" will validate: $v \leftarrow v'$.
   4. To have a perfect matching from the graph, we need to be sure that v was assigned to a request, in case this v was swapped in the previous loop, it still remains unassigned, therefore the last step is to assign the fair request from the last vehicle v to the *new assignment*:
      - $M_{new}(v) \leftarrow M_{fair}(v)$
4. Return the *new assignment* M_{new}.

Lemma 2

To support the authors algorithm **Reassign**, they output a lemma based on their previous theorem: The new assignment's fairness is guaranteed to be above the f as stated in (10):

$$\mathcal{F}(M_{new}) \geq f$$

And the efficiency for the new assignment substituting in (11) the λ value obtained in the first algorithm step 6:

$$f = \lambda \mathcal{F}_{opt}$$

$$\lambda = \frac{f}{\mathcal{F}_{opt}}$$

Results in:

$$\mathcal{E}(M_{new}) \geq \frac{2\mathcal{F}_{opt}}{2\mathcal{F}_{opt} + f}(\mathcal{E}(M_{old}) - n\Delta)$$

(Where M_{old} is the first assignment computed with the efficient assignment algorithm).

Theorem 3 (Lower Bound)

To conclude their demonstration, the authors also analyze the lower bound, where at any λ value and an α strictly larger than $\frac{2}{2+\lambda}$, the fairness and efficiency cannot be found simultaneously:

$$\mathcal{F}(M) \geq \lambda\mathcal{F}_{opt}$$

$$\mathcal{E}(M) \geq \alpha(\mathcal{E}_{opt} - n\Delta)$$

i.e. with λ=1 and $\alpha = 1 \geq \frac{2}{2+1}$, the algorithm should find an assignment with fairness equals to the optimum fairness and at the same time an efficiency of nearly the optimum, which is (theoretically) not possible.

Experiments - Reproducing the paper

The authors conduct a series of experiments to demonstrate the capabilities of their algorithm and specially to demonstrate that the theoretical lower bound in efficiency loss is not necessarily reached when using this algorithm (otherwise many companies would not use this approach of fairness if their revenue is in danger).

To support all their theoretical contributions, they also make publicly available their experiments under a repository in the platform GitHub: https://github.com/zxok365/On-Demand-Ridesourcing-Project . They share their simulation codes, project documentation as well as their data used (in external storage services). To validate their experiments in this Seminar review, I tested the author's repository and generated a Jupyter Notebook with some of the artifacts used for this validation; this code is available under: https://github.com/jdanrq/Balancing-Efficiency-and-Fairness-in-On-Demand-Ridesourcing.

The authors divide the experiments in three types, the first is the *single batch* scenario, meaning all the requests come within a short period of time and they are dispatched once by the algorithm. The purpose of single batch experiment is to analyze the worse-case efficiency loss.
The second experiment is the *multi-period* setting, in this experiment, the single batch experiment is repeated T times as described already in the Multi-period assignment section.
The third experiment is an extension of the multi-period setting, but as a Ridesharing problem.

Dataset

The data used for the experiments comes from the New York City Taxi Trip Dataset, a compound of 697622444 trip recordings happening between 2010 and 2013. It contains dates, pickup times, drop-off times, coordinates, taximeter distance, fare amounts and tip amounts.[6] From this dataset, the authors restrict the experiments to only 15.2 million trips happening in May 2013, they also exlude trips lasting less than 2.5 minutes or more than 1 hour. Furthermore, only a small region of New York is used: the Manhattan Island. This is the base to build a roadmap to represent the points where requests and vehicles can be located. The roadmap of the island contains 3671 nodes and 7674 edges joining the edges. Illustration 6 contains a snapshot of this roadmap, using folium [7] (A python wrapper of the open source map library leaflet.js).

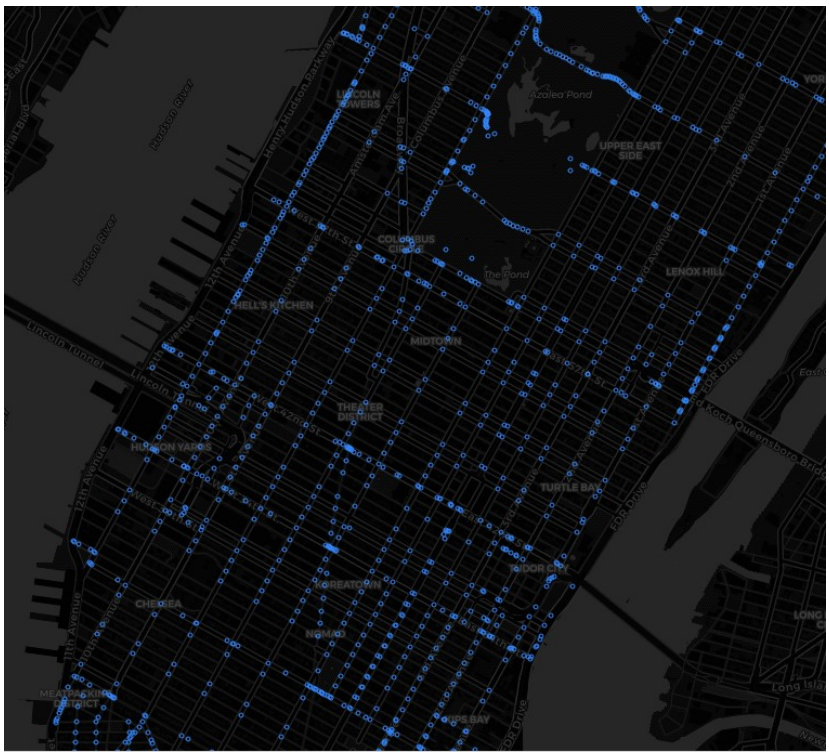


Illustration 6: Manhattan Roadmap (Nodes)

As a remark, the nodes are represented with the blue circles, while the edges with the orange lines:

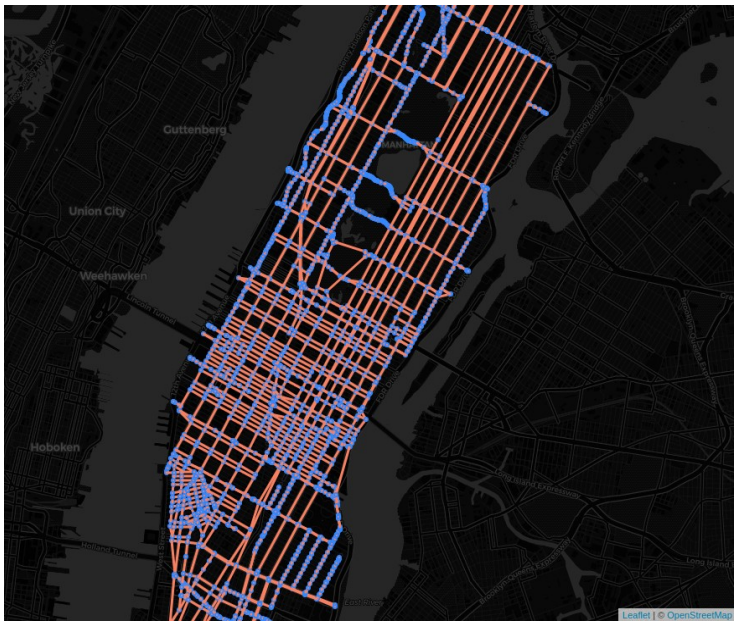


Illustration 7: Manhattan Roadmap (Nodes & Edges)

The weight in each edge is an average from the time that vehicles take to translate from node A to node B. This weight is a realistic approach because in practice, some streets might be more occupied than others, even though their distance is the same. (Nevertheless, no online traffic monitoring is considered, which would be beneficial in terms of accuracy rather than using statical historical data).
To fit the constraints of the region, all the requests coordinates need to be rounded to the nearest node in the roadmap, in the following illustration, requests (green) are originally located outside the island (many of them coming from the John F. Kennedy International Airport:

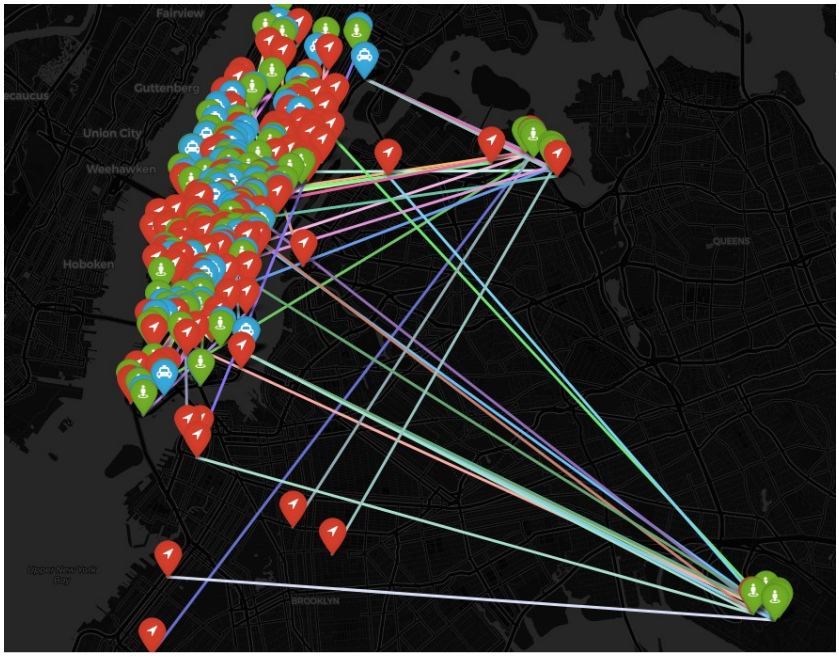


Illustration 8: Requests (green), destination (red) and vehicles (blue).

Single-Batch

For this experiment, the trips coming within a period of 30 seconds are considered a batch. From the filtered trips selected for the experiment, each batch contains from 105 to 142 trips, this can be expressed as:

$$m = |R| \in [105, 142]$$

From the New York city dataset only the requests are extracted, the vehicle position is randomly generated. For each single batch, an amount of 1.2 times the number of requests is the number of vehicles randomly created within the Manhattan Region:

$$n = 1.2m$$

As mentioned before, the purpose of this experiment is to try to level up the income of the disadvantaged drivers, therefore, from the 1.2m vehicles, m are initialized with a historical utility between 200 and 400, whereas 0.2m vehicles are initialized with low values between 50 and 100:

$$|V_H| = m, h_v \sim U(200, 400)$$
$$|V_L| = m, h_v \sim U(50, 100)$$

Finally, the maximum waiting time is set to 210 seconds, the vehicle capacity for all vehicles is set to 4 and the constant c for equation (2) is set to 1:

$$\Omega = 210s$$
$$\chi = 4$$
$$c = 1$$

The results of this experiments are materialized in two different plots, the first one represents the ratio of fairness versus the ratio of efficiency. In the first, the ratio has as denominator the optimum:

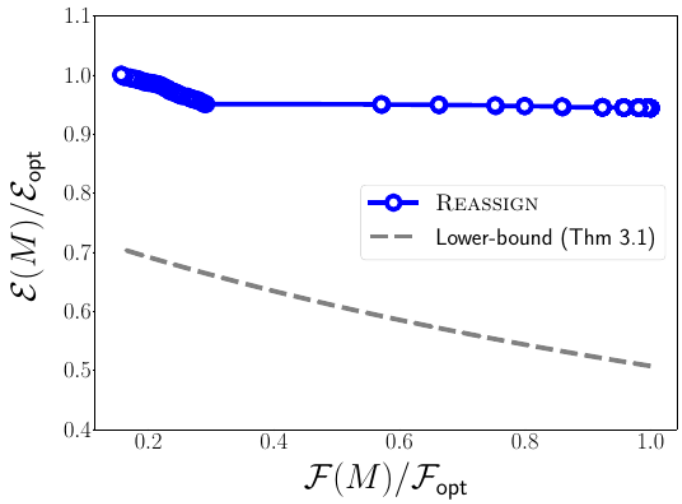


Illustration 9: Efficiency-Fairness Tradeoff [1]

The plot shows that for high levels of efficiency (assignment efficiency equals optimum efficiency), the theoretical lower bound discussed in Theorem 3 starts at 70% and goes

down when the fairness is incremented. Experimentally, with the conditions described above, the efficiency starts by 1 and decreases roughly to 0.9 starting at the fairness ratio 0.3. This is the behavior that the authors were expecting to demonstrate: that there is no such big impact of efficiency when the fairness is prioritized. However, the authors only tested for this specific time and with this specific hyper parameters, which might not be enough to generalize if the REASSIGN algorithm will keep this behavior with different settings.
The second plot aims to show how the vehicle fleet would be ranked according to their income, where the objective is to have a more evenly distributed plot (or at least no such difference between the smallest and greatest utility among the vehicles). For this experiment, I recreated the experiment for three fairness values: $\lambda = 0$, $\lambda = 0.5$ and $\lambda = 1$. The experiments show that for small λ values, the inequality for the disadvantaged vehicles is maintained, whereas higher values of λ demonstrate that the utility can be distributed more fairly among the vehicles (λ=0.5 and λ=1 output the same assignment).

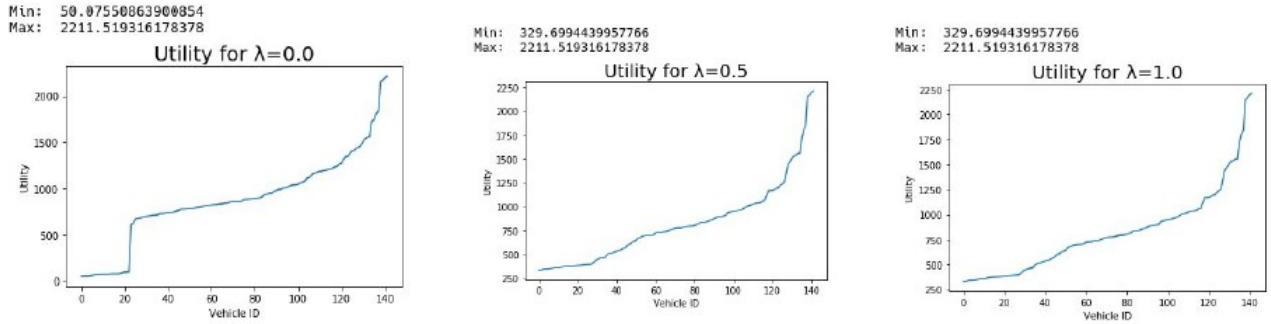


Illustration 10: Vehicle utilities for different λ values.

Multi-period (Single-ride and Ridesharing)

For this experiment, an amount of 240 periods was simulated: $t \in \{1, 2, 3, ..., 240\}$, each period lasting 30 seconds, with a total time of 7200 seconds (2 hours). This experiment was repeated over 10 days to study the stability and variance of the results.
Opposite to the single-period, for this experiment all the vehicles were initialized with the historical utility as zero and an amount of 2000 vehicles was randomly generated. The maximum waiting time was set lower than in the single-period settings: $\Omega = 150s$ and the maximum delaying time set to: $\Gamma = 300s$.
For the Ridesharing setting, the trip utility for each vehicle is the contribution of all passengers.

Results

Firstly it is described the efficiency-fairness tradeoff, in both variants, it is observed that in opposite to the single-period setting, even with higher fairness ratio, the loss of efficiency is minimal. This can be explained because in the single-period experiment, the assignment was carried out only once, which is not showing the long effect for multiple reassignments. Nevertheless, as it was described before, all the vehicles are initialized with the same historical utility, which directly has an effect on the equality of requests distribution.

Because this is an artificial experiment, it doesn't show the results that could be reached in an initialization with unequal utility. The same pattern is observed for the ridesharing scenario. In both cases, the theoretical lower bound does not comply with the Theorem 3, because of the "Dependence of future instances and uncertainty inherent to future demand distributions"[1], in words of the authors.

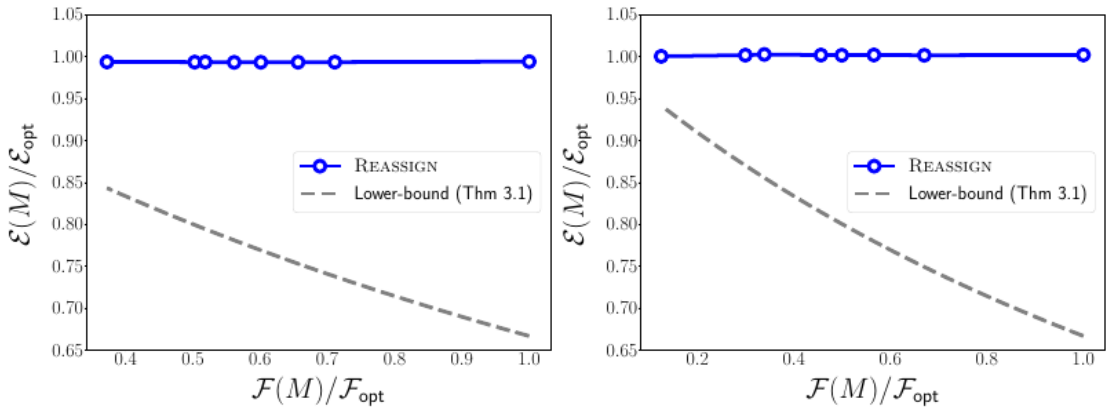


Illustration 11: Efficiency-fairness tradeoff for the multi-period experiment. [1]

Finally, with different levels of fairness thresholds, the authors tested in 10 different days and measured the fairness ratio. The resulting plots in Illustration 12 show the high variance of fairness in the single-ride setting, meaning that the results vary widely in those ten days, as expressed in the confidence interval, whereas in the ridesharing setting the CI is smaller for the majority of λ values, but also the fairness ratio is significantly lower, which means that this extra constraint of sharing a ride has a direct impact on the fairness that the REASSIGN algorithm can reach:

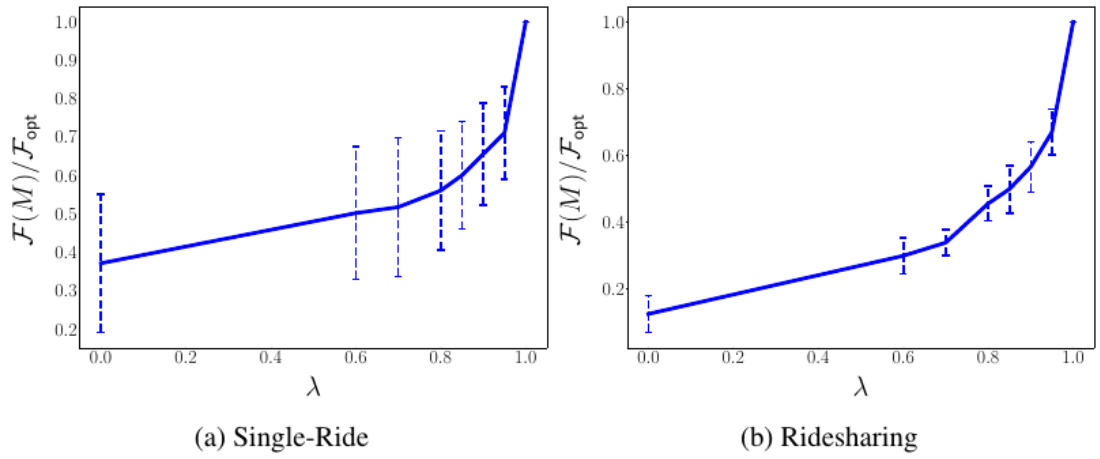


Illustration 12: Fairness ratio for different λ values. [1]

This results suggest that the REASSIGN algorithm can provide fair assignment decisions without a big impact in efficiency.

Discussion and Conclusions

The authors demonstrate in their paper how to introduce fairness in a specific kind of resource allocation algorithm and derive a procedure to reassign matches between vehicles and requests to get a minimum fairness. They propose a way to quantify the fairness but also omit to mention some implications from their approach:

- The REASSIGN algorithm requires to compute several times an assignment algorithm (in their experiments is used the Hungarian Algorithm): Once as an input for the efficient assignment, and several times inside the algorithm to find the optimum fairness. As they described it before, this is an iterative procedure where each edge under the fairness threshold is eliminated and the assignment recomputed. The authors omit to formally define the computational complexity of this procedure, and furthermore, they do not mention the complexity of the REASSIGN procedure. In practice, computing several times the assignment requires much more computational power to achieve the same task as the efficient version already does, and it can greatly reduce performance, which is crucial for a real-time system.
- The authors do not clarify the case of having more requests than vehicles. In that scenario, some vehicles need to be left out and some should be integrated to the bipartite graph, but they only explain how to handle the situation where there are more vehicles than requests (by adding non-serve requests). This is specially relevant because those edges request-vehicle can have an impact in the final assignment algorithm and consequently in the overall fairness.

As positive points from their research, these are the highlights:

- Their algorithm allows to dynamically decide which fairness threshold the system will use, the λ value can be treated as a hyperparameter and attend the needs of the system, when there is a peak in demand from vehicles, this fairness can be relaxed and when the system stabilizes, the fairness could be augmented to leverage utility among vehicles. A further experiment in this setting is needed to prove the benefits of varying λ versus leaving it as a static variable.
- They are able to introduce one kind of fairness (egalitarian justice) in a ridesourcing allocation algorithm. There are several well proven allocation algorithms that hardly ever were thought to need to be fair, because they attended allocation of material/computational resources or they need to be fairly balanced to reduce stress in some resources. As soon as the society starts to use those algorithms to allocate human resources, there is a need to involve humanitarian concepts (here fairness) in their design and take into account the satisfaction of the users and the providers. Further fairness concepts could be applied for the same problem setting (i.e. proportional fairness).

References

1. Nixie S. Lesmana, Xuan Zhang, Xiaohui Bei, Balancing Efficiency and Fairness in On-Demand Ridesourcing (in Proc. NeurIPS 2019)
2. Betancourt, S. (2019, March 28). Chile: As mobility apps takeover, more people use them as income. Retrieved August 06, 2020, from https://www.onlinemarketplaces.com/articles/24650-Chile-As-mobility-apps-takeover-more-people-use-them-as-income
3. Lamont, J., & Favor, C. (2017, September 26). Distributive Justice. Retrieved August 06, 2020, from https://plato.stanford.edu/entries/justice-distributive/
4. The University of Texas at Austin (2020, August 05). Utilitarianism. Retrieved August 06, 2020, from https://ethicsunwrapped.utexas.edu/glossary/utilitarianism
5. Cambridge Dictionary. (n.d.). TRADE-OFF: Meaning in the Cambridge English Dictionary. Retrieved August 06, 2020, from https://dictionary.cambridge.org/dictionary/english/trade-off
6. Dan Donovan, Brian; Work. New york city taxi trip data (2010-2013), 2016.
7. Folium. (n.d.). Retrieved August 06, 2020, from https://python-visualization.github.io/folium/
8. Der-Horng Lee, Hao Wang, Ruey Cheu, and Siew Teo. Taxi dispatch system based on current demands and real-time traffic conditions. Transportation Research Record: Journal of the Transportation Research Board, (1882):193–200, 2004.
9. Kiam Tian Seow, Nam Hai Dang, and Der-Horng Lee. A collaborative multiagent taxi-dispatch system. IEEE Transactions on Automation Science and Engineering, 7(3):607–616, 2010.
10. Dan A Iancu and Nikolaos Trichakis. Fairness and efficiency in multiportfolio optimization. Operations Research, 62(6):1285–1301, 2014.
11. Yuga J Cohler, John K Lai, David C Parkes, and Ariel D Procaccia. Optimal envy-free cake cutting. In Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.
12. Jon Kleinberg, Yuval Rabani, and Éva Tardos. Fairness in routing and load balancing. In 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039), pages 568–578. IEEE, 1999.
13. Mordecai J. Golin, Bipartite Matching and the Hungarian Method (bigraph formalism), Course Notes, Hong Kong University of Science and Technology.
14. Lesmana, Zhang, Bei. (n.d.). Trip-Vehicle Assignment Algorithm (REASSIGN). Retrieved August 06, 2020, from https://github.com/zxok365/On-Demand-Ridesourcing-Project
15. Rodriguez Quintana, J. (2020). Balancing-Efficiency-and-Fairness-in-On-Demand-Ridesourcing. Retrieved August 06, 2020, from https://github.com/jdanrq/Balancing-Efficiency-and-Fairness-in-On-Demand-Ridesourcing