# Towards a geographically even level of service in on-demand ridepooling

Pieter Schuller[1], Andres Fielbaum[1] and Javier Alonso-Mora[1]

***Abstract*— On-demand ridepooling systems usually need to decide which requests to serve, when the number of vehicles is not enough to transport them all with waiting times that are acceptable by the users. When doing so, they tend to provide uneven service rates, concentrating rejections in some zones within the operation area. In this paper, we propose two techniques that modify the objective function governing the assignment of users to vehicles, to prioritize requests originated at zones that present a relatively large rejection rate. The goal is to diminish the Gini Index of the rejections' rate, which is a well established way to measure inequality in economics. We test these techniques over an artificial small network and a real-life case in Manhattan, and we show that they are able to reduce the Gini Index of the rejection rates. Moreover, the overall rejection rate can be simultaneously reduced, thanks to utilizing the vehicles more efficiently.**

## I. INTRODUCTION

On-demand mobility systems are changing how people transport worldwide. While expanding rapidly thanks to several virtues offered to the users, recent studies have shown that they are increasing congestion because users are attracted mostly from public transport ([1]). To face this problem, *pooled mobility on-demand* (PMoD) has been suggested, where different passengers can use a vehicle simultaneously, thus reducing the number of required vehicles.

Several centrally-controlled methods have been proposed to determine how to assign users to vehicles in PMoD (such as [2], [3], [4], [5]). All of them aim at providing efficient assignments, accounting for factors such as users' total traveling times and/or vehicles-hour-traveled. However, they do not consider equity aspects in their methods. This is not a minor issue, as PMoD can naturally evolve towards uneven situations: if the origins are concentrated in a certain area (for instance, in the afternoon peak), and users are then delivered somewhere else, trying to keep a somewhat proportional rate of vehicles is a difficult task.

This situation is exemplified in Figure 1, where we exhibit the percentage of requests that were rejected by the system, due to insufficient available vehicles, at each corner in Manhattan when simulating the operation of a PMoD system using the assignment method by [2]. The highest rejection rates occur at the center of the system, coinciding with the high-demand zones, because too many vehicles are kept serving in the peripheries.

In this paper, we propose techniques to modify assignment methods in order to provide similar rejection rates every-

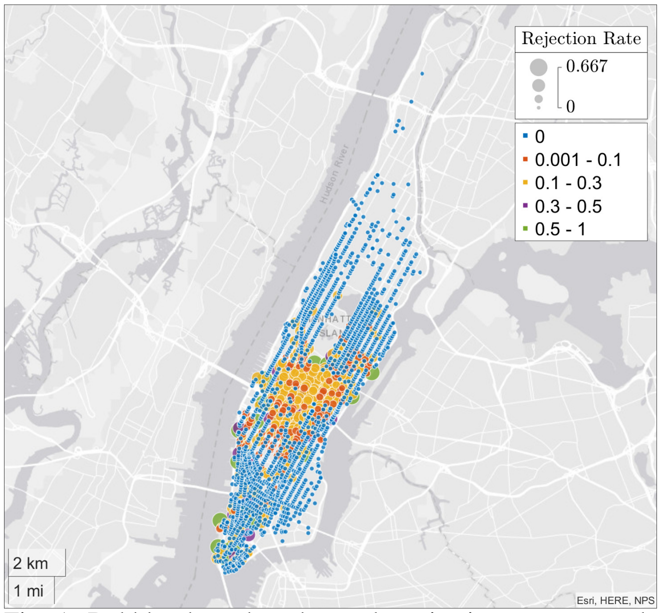


Fig. 1: Bubble chart that shows the rejection rate per node simulating a PMoD service in Manhattan. The size of each bubble indicates the height of the rejection rate, and is given a color based on its size to give a clearer image. The central area that exhibits the largest rejection rates coincides with the area where most requests are originated.

where, i.e., our purpose is to provide an equal level of service from a geographical point of view.

Equity has been recognized as a relevant issue in transportation systems in the last years. In the context of mobility on demand, [6] proposes a deep-learning approach to operate a system that considers its equity impacts, while [7] points to similar profit for every driver and also similar rejection rates at every zone (as we do), but using a method that deals only with hundreds of users and tens of zones. Age-related equity impacts of driverless vehicles are studied by [8]. Public transport, transport infrastructure and road pricing policies have also been analyzed from an equity perspective ([9], [10], [11]).

## II. METHODOLOGY

### A. The Gini Index

Consider a PMoD system that needs to serve a large number of requests, which are not known beforehand, so the decisions regarding which requests to serve and with which vehicles are taken as the requests appear. To measure how uneven the service rate is within a certain area of operation, we consider the set of possible requests' origins $N$ (that might be represented as nodes, as sectors of an arc, as points within a two-dimensional area, or any other option), and we

[1]Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: ptschuller@hotmail.com; a.s.fielbaumschnitzler@tudelft.nl; j.alonsomora@tudelft.nl).

This research was partially funded by Didi Udian Technology (Shenzhen) Co. Ltd.

divide it into a finite number of disjoint zones $z_1, \dots, z_k$ such that $N = \cup_{i=1}^{k} z_i$. Consider now the end of the period of operation (e.g., a day). We denote by $K_i$ the number of requests that emerged from zone $z_i$ and by $F_i$ the number of those requests that were rejected. The rejection rate per zone is defined as:

$$R_i = \frac{F_i}{K_i} \tag{1}$$

A perfectly even system would yield $R_1 = \dots = R_k$, whereas an extremely uneven system would concentrate all the rejections in a single zone. These characteristics are captured by the well-known *Gini Index* $GI$, that has been traditionally utilized to measure wealth inequality within a country, but that has also been considered for transport analysis ([11], [12]). A detailed explanation of how to compute $GI$, including its explicit mathematical expression, can be found in [12]; for our purpose, it suffices to explain that $GI$ is a function that takes a vector of numbers $a_1, \dots, a_m$, returning $GI(a_1, \dots, a_m) \in [0, 1]$, such that the higher $GI$, the more unevenly distributed the vector.

There could be a trivial way to achieve $GI = 0$: consider a large rejection rate $\rho$, such that it is easy for the system to present $R_i = \rho$ for every zone $z_i$ (in the worst case, one could take $\rho = 1$, i.e., to serve nobody). Of course, this is not a desirable solution. Therefore, we want our method to **decrease $GI$ without increasing the overall rejection rate**, or ensuring that the rejection rate does not increase more than some pre-defined percentage (such as 1% or 2%).

## B. Methods that assign batches of users

The problem of how to assign groups of users to vehicles is complex, as the possible ways to group the users can be enormous, and two well-known NP-Hard problems are involved: VRP and Dial-A-Ride. This complexity has led to different approaches. Most of them can be categorized either as:

- Event-based: Each time a new request emerges, it is assigned to a vehicle or rejected.
- Batch-based: The method waits to accumulate some requests, and assigns all of them together. When the system waits for a fixed lapse of time, the approach is denominated *receding horizon*, a quite common approach.

Each of these two alternatives needs to chain consecutive solutions. That is to say, after deciding the assignment (for the single or the batch of users), the vehicles are instructed to follow some routes, and these instructions are updated when a new assignment occurs. The aim of the methods is that such partial assignments and vehicles' instructions yield good results at the end of the period of operation, i.e., that they are able to chain efficiently with the subsequent assignments.

Given the existence of these different approaches, we do not expect to propose techniques that can work with any assignment methods. Nevertheless, the techniques we propose are quite general. They can work with any batch-based method, that assigns each batch of requests through the following steps:

1) Determine which *trips* might be served by the PMoD system. A trip is defined as a group of requests together with a vehicle, so that serving such a trip means that the vehicle's route will be updated to transport the mentioned requests.
2) For each trip, determine a cost.
3) Decide which trips to serve, through an objective function that considers the cost of each served trip and a penalty for each request that is rejected.

Note that such a procedure admits many different methods. In the first step, we do not require anything specific regarding how to define which are the feasible trips, i.e., our techniques can work with exhaustive methods and also with heuristics; certain specific rules (such as first-in-first-out, or limiting the number of changes faced by a user) can also be included. In the second step, the cost functions can also be general, considering different combinations of users' and operators' costs. Moreover, if different users imply different fares and the system is for-profit, this can also be included in these cost functions or in the rejection penalties, which can be user-dependant. The only true requirement is that the total costs of the system can be expressed as a sum of the individual cost of each served trip. Finally, our techniques can also be applied when there is a hierarchical optimization, in which the first objective is to serve as many requests as possible, just by utilizing a very high rejection penalty. Below we test our techniques modifying the method developed by [2]. Other methods that could make use of our techniques are the ones by [4], [5] (that are based on [2]) and by [13], [14].

It is implicit in the description above that our techniques pursuing more equity shall affect the system at each assignment of a batch. Again, the expectation is that applying these techniques consecutively throughout the period of operation will yield more even results at the end.

We now introduce the respective notation. Recall that a trip $t$ is formed by a set of requests, that we denote $req(t)$, and a vehicle denoted $veh(t)$. The set of trips is denoted[1] $T$. The cost of each trip $t \in T$ is denoted $c(t)$. The rejection penalty of a request $r$ is denoted $p_Q(r)$. Costs and the penalty are assumed to be monetized in the same currency. The set of requests is $Q$ and the set of vehicles is $V$. The problem can be stated as an integer-programming problem:

$$\min_{x,y \in \{0,1\}} \sum_{t \in T} x_t c(t) + \sum_{r \in R} y_r p_Q(r) \tag{2}$$

$$\text{s.t.} \quad y_r + \sum_{t: r \in req(t)} x_t = 1 \quad \forall r \in Q \tag{3}$$

$$\sum_{t: v = veh(t)} x_t \le 1 \quad \forall v \in V \tag{4}$$

[1]Computing the set of trips can be algorithmically heavy, but current assigning techniques are able to handle it.

Binary variables $x_t$ are equal to 1 if and only if the trip $t$ is served, while $y_r = 1$ if and only if the request $r$ is rejected. Eq. 2 represents the objective function. Eq. 3 ensures that each request belongs exactly to one served trip or is rejected, while Eq. 4 ensures that no vehicle can be assigned to more than one trip. Note that this is just a formal way to describe the problem, but we do not require that the problem is actually solved through the ILP, i.e., we admit other combinatorial optimization techniques.

### *C. Altering the objective function*

Consider an assignment decision that is taking place at time $\tau$. We shall modify the objective function displayed in Eq. 2 in order to make the assignment method point to an even distribution of the rejection rate. The idea we want to incorporate is that the system is less willing to reject a request originating from a zone with an already above-average rejection rate. For this, we denote by $R_i$ the rejection rate in zone $z_i$ up to time $\tau$, and by

$$\Delta R_i(\tau) = R_i(\tau) - \overline{R(\tau)} \tag{5}$$

That is, the difference between $R_i(\tau)$ and the overall rejection rate of the system at the same time. With this notation, we might modify any of the two terms in Eq. 2, i.e. the cost of the groups that are going to be executed, or the rejection penalties for those requests that are not going to be served. As these changes do not depend directly on the requests, but on which zones they are originated, we will use $i(r)$ to say that request $r$ is departing from the zone $z_{i(r)}$.

We define the *Technique* R as the one that modifies the rejection penalty, taking as its objective function:

$$\sum_{t\in T} x_t c(t) + \sum_{r\in R} y_r p_Q(r, \Delta R_{i(r)}(\tau)) \tag{6}$$

I.e., the rejection penalty now also depends on $\Delta R_{i(r)}(\tau)$. We want to define $p_Q(r, \Delta R_{i(r)}(\tau))$ in such a way that it increases with $\Delta R_{i(r)}(\tau)$, but also ensuring that the rejection penalty is always larger than the most expensive trip (otherwise, the system might artificially increase the overall rejection rate). With this in mind, we define $p_Q(r, \Delta R_{i(r)}(\tau))$ as

$$p_Q(r, \Delta R_{i(r)}(\tau)) = \max\{p_Q(r) + \delta \Delta R_{i(r)}(\tau), \max_{t\in T} c(t)\} \tag{7}$$

Where $\delta$ is a parameter to be tuned[2]. Similarly, the *Technique* T modifies the costs of trips $t$, depending on $\Delta R_t(\tau)$, which is defined as Eq. 5 but replacing $R_i(\tau)$ by the average among the rejection rates of the zones where the requests in $t$ are located, as shown in Eq. 8:

$$\Delta R_t(\tau) = \sum_{r\in req(t)} \frac{R_{i(r)}}{|req(t)|} - \overline{R(\tau)} \tag{8}$$

[2]Other functional forms could also be used. Our numerical simulations suggest that this is the most effective one.

In this case, the costs should decrease with $\Delta R_t(\tau)$, so that trips that come from high-rejection areas are more likely to be served. The modified objective function now is:

$$\sum_{t\in T} x_t c(t, \Delta R_t(\tau)) + \sum_{r\in R} y_r p_R(r) \tag{9}$$

To obtain a function that decreases with $\Delta R_t(\tau)$ and that is also tuneable, we use

$$c(t, \Delta R_t(\tau)) = \max\{c(t) - \lambda \Delta R_t(\tau), \frac{c(t)}{P}\} \tag{10}$$

Where $P$ and $\lambda$ are tuning parameters. We will use $P = 2, 4$, and refer to the technique as $T_2$ and $T_4$, respectively.

## III. NUMERICAL EXPERIMENTS

We now test the techniques R, $T_2$ and $T_4$, applied upon the method in [2], simulating the operation of PMoD systems in a small artificial example and in a real-life case in Manhattan. Before doing so, we explain which is the correct benchmark of the Gini Index to compare with, and we provide a brief description of the base method from [2].

### *A. Posterior Gini Index*

We first explain how to evaluate the results of our techniques, and in particular how to compute a proper benchmark. At the end of the period of operation we obtain the resulting overall rejection rate and Gini Index. When we run the simulations using the basic method (without using the techniques proposed here, or equivalently, taking $\lambda$ or $\delta = 0$), we denote these results by $R_{end}$ and $GI_{end}$, respectively. When the method is run using our techniques, we denote the modified results by $R'_{end}$ and $GI'_{end}$.

If $R'_{end} \leq R_{end}$ and $GI'_{end} \leq GI_{end}$, our techniques are improving both objectives (equity and serving as many requests as possible) and therefore the obtained solution is better than the original one.

However, if $R'_{end} = R_{end} + \epsilon\%$ for a certain threshold $\epsilon$, there is a trade-off. Moreover, in such a case having $GI'_{end} \leq GI_{end}$ might not be good enough, as there is a trivial way to modify *a posteriori* the results of the original method, reducing $GI_{end}$ while keeping $R_{end} \leq R'_{end}$: taking the results from the original method, and including some artificial rejections in the zones with the lowest rejection rates, until reaching $R'_{end} = R_{end} + \epsilon\%$; a technical detail is that some zones might have few total requests, so even one extra rejection can be too significant, so we have to avoid them. Such an idea is detailed in Algorithm 1, where $X$ denotes the absolute number of requests that can be added (i.e., the difference between the total rejections with and without using our techniques), and that outputs the *posterior Gini Index* when increasing the rejection rates in $\epsilon\%$.

**Algorithm 1** Computing the Posterior Gini Index.

```
1: Input: $F_{i,end}, K_{i,end}$ for every zone $z_i$, and the number of rejections to be added $X$.
2: for $\ell = 1 : X$ do
3:     Define $\varphi = \{i : \frac{F_{i,end}+1}{K_{i,end}} \leq \frac{1}{k}\sum_{j=1}^{k}\frac{F_{j,end}}{K_{j,end}}\}$; % We can only include artificial rejections in zones where we would not exceed the average rejection rate.
4:     $i^* = \text{argmin}_{i\in\varphi}\{\frac{F_{i,end}}{K_{i,end}}\}$;
5:     $F_{i^*,end} \leftarrow F_{i^*,end} + 1$; % We add one artificial rejection to the zone in $\varphi$ that presents the lowest rejection rate.
6: end for
7: Output: $R^P_{i,end} = \frac{F_{i,end}}{K_{i,end}}$ for every zone $z_i$.
```

## B. Brief description of the assignment method

Our techniques modify the assignment method developed by [2]. Such a method performs an exhaustive search of the feasible trips, considers only users' costs, and imposes the same penalty (prior to applying our techniques) to every rejected request. Let us explain the method in some detail:

- A trip $t$ is feasible if all the requests in $req(t)$ can be served without violating some predefined upper bounds on the waiting times and total delays. These constraints also apply to the requests that were previously being served by $veh(t)$ (whose traveling times might increase when the vehicle's route is updated).
- The cost of a trip $c(t)$ depends only on the total delay that would be faced by all users in $req(t)$, and the extra delay that would be imposed to the users that were being served by $veh(t)$.
- To execute an exhaustive search of the feasible trips, the method takes advantage of the fact that for $t$ to be feasible, any subtrip formed by $veh(t)$ and a subset of $req(t)$ must be feasible as well.
- The assignment is decided by the ILP defined by Eqs. 2-4, where $p_Q(r)$ is a constant function. Once a request is rejected, it is erased from the system.
- After deciding the assignment, a rebalancing step is performed: idle vehicles (i.e., that had no passengers before the assignment and received none) are sent towards the origins of the rejected requests. This step does not interact with our techniques and remains exactly the same.

## C. Results in an artificial small graph

We first test the techniques R, $T_2$ and $T_4$ over a small $20 \times 20$ grid, in which all arcs are bidirectional. Origins and destinations are placed on the nodes of the network, which is served by 65 vehicles and 5 requests emerge per minute, that follow: i) A fully random pattern (RAND), ii) 10% of the requests go from left to right (10L2R) while the rest are random, and iii) 20% of the requests go from the center to the surroundings (20C2S) while the rest are random. In this graph, each node is considered as a different zone, therefore there are 400 zones.

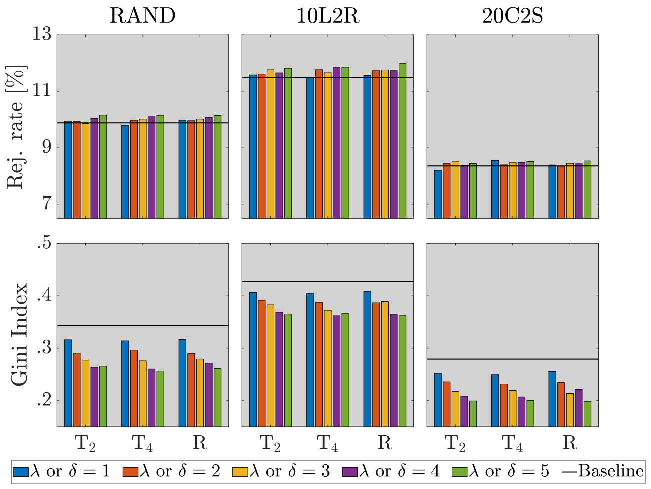


Fig. 2: Resulting Gini Indices and overall rejection rates when the techniques $T_2$, $T_4$ and R are applied with different tuning parameters. The results without any technique are marked with a horizontal black line.

Results are displayed in Figure 2, where we show the resulting overall rejection rates and Gini Indices for the different techniques and varying the tuning parameters. Some conclusions follow:

- All the techniques effectively achieve a more even distribution of the rejection rates. The more aggressive the method (i.e., the higher $\delta$ or $\lambda$), the larger the impact.
- There are cases in which the overall rejection rate also diminishes, meaning that the techniques might be able to induce a more efficient positioning of the vehicles. This is probably related with the consecutive chains between the different assignments throughout the period of operation: in order to achieve a lower Gini Index, some vehicles are pushed towards the high-rejection areas, which makes them available for some other trips that will emerge nearby in the near future[3].
- When the overall rejection rate increases, it does in mild numbers.
- Results depend heavily on the tuning parameter, and which parameter to choose depends as well on the scenario.

The comparison against the posterior Gini Index is shown in Figure 3. Each column represents the lowest Gini Index achieved by the respective technique from Figure 2, but excluding the cases in which the increase in the rejection rate exceeds the $\epsilon\%$ threshold of the respective row.

Thus, the first row shows the cases in which using the technique actually reduces the rejection rate. No technique is able to achieve such a result in every scenario, but if the correct method is chosen, **it is always possible to reduce both the Gini Index and the number of rejections**. When some increase in the rejection rate is allowed, the Gini Index is almost always lower than the posterior one (obtained with algorithm 1), and the difference can be quite significant in

[3]In a similar note, some previous studies use predictive techniques in order to reduce the number of rejections in the system ([15], [16]).

some cases. It is worth remarking that the benchmark we are comparing with (red lines in Figure 3) is calculated after the full period of operation, whereas our techniques run online, and yet they are able to select assignments that are more even than the benchmark. Figure 3 also suggests that $T_2$ is the best technique, as it yields the lowest Gini Indices in most cases. All the techniques work worse in the scenario 10L2R, which might be caused by a lower flexibility of the system to decide how to assign.

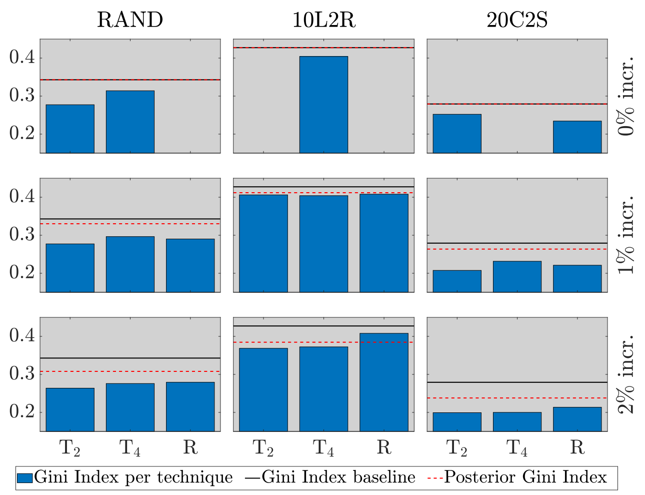


Fig. 3: Comparison of the best Gini Indices achieved by each method (columns) when a certain increase in the rejection rate is accepted (rows). The black lines represent the Gini Index without applying our techniques, and the red lines represent the posterior Gini Index. Missing columns mean that the respective technique is not able to find a solution within the accepted rejection threshold.

Figure 4 shows sensitivity analyses, in which only the "0% increase" row is displayed. Figure 4 left shows that the results of our techniques are better when the system is less demanded. For instance, our techniques achieve a lower Gini index 7/9 times if three requests per minute are generated, whereas this is only achieved 3/9 times in the scenario with six requests per minute. On the other hand, such results do not change significantly with the scale of the problem (Figure 4 right, where the number of requests and vehicles change in the same proportion).

### D. *Results in Manhattan*

We now test our techniques in a real-life case. As done by [2], [5], we simulate the operation of the PMoD system over Manhattan, using the publicly available dataset of taxi rides (the precise data we use is from 07/03/2014). To this end, we utilize a graph representing the respective road network, and we cluster the nodes using the method described in [15], that selects "centers" such that each node in the graph is no further than a certain threshold $\varepsilon$ from its closest center. We use $\varepsilon = 2.5$[min], which divides Manhattan in 146 zones, which will be taken as the basis for the computation of the Gini Indices.

Utilizing the same assignment method as in the previous subsection, and varying again the values of the tuning parameters, we can assess if the techniques $T_2$ and R are able to improve the spatial equity of the system. As previous section suggested that $T_2$ outperforms $T_4$, and due to the large computational times required for these simulations, we are not testing $T_4$ in this scenario.

The overall results are depicted in Figure 5, where we randomly select different-sized subsets of the requests, scaling the fleet accordingly. The most relevant conclusions are:

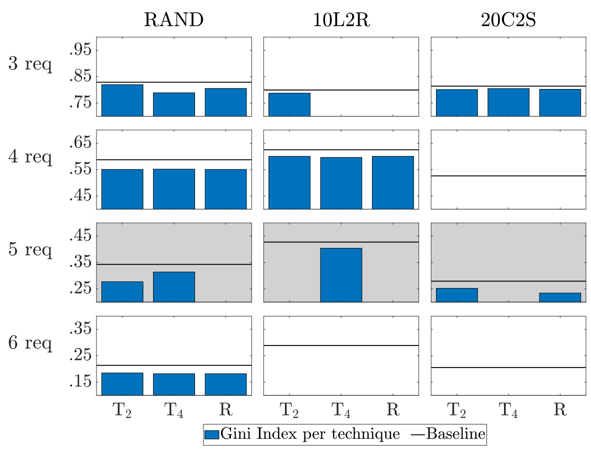
- The Gini Index is indeed reduced in every column in the bottom row, i.e., regardless of the scenario, technique and value of $\lambda$ or $\delta$.
- **In every scenario, at least one technique is able to reduce both the Gini Index and the number of rejections,** if using the correct value for the parameter. However, the best technique and parameter to use are scenario-dependant.
- It is no longer true that the larger the parameters, the larger the impact. This suggests that a more intricate demand pattern relates to these techniques in complex ways that are hard to control, meaning that finding the correct value for the parameter is far from being simple.

## IV. CONCLUSIONS

In this paper, we have proposed two techniques to modify batch-based assignment methods for on-demand ridepooling, so that the probability of being served by the system is similar for every request regardless of its origin. Both techniques are based on changing the objective function when deciding the assignments, either increasing the rejection penalty for requests that come from zones that already accumulate a high number of rejections, or reducing the cost of accepting requests from the same regions.

To test our ideas, we have run simulations in an artificial small network and using real-life data from Manhattan. In both cases, our techniques have been able not only to diminish the Gini Index, as expected, but also to increase the number of served requests if the correct tuning parameters are chosen, due to the more efficient use of the vehicles that they induce. Therefore, our results show that there is room for a synergy between the two objectives discussed here (equity and the total number of served users). Moreover, if a small increase in the rejection rate is accepted, the Gini Index can be decreased even more.

Which technique to use, and which is the optimal value for the parameters $\lambda$ and $\delta$, depend on the scenario where the PMoD system operates. This is troublesome, as the operator cannot always know the scenario in advance. This might be solved by combining both techniques, and defining the values for $\delta$ and $\lambda$ online. In other words, such parameters can evolve with the system depending on the changing conditions (number and location of the requests), which could be done using learning techniques or simulating a large number of scenarios to select the most robust values for the parameters. These are the most relevant questions posed as future research by this paper.



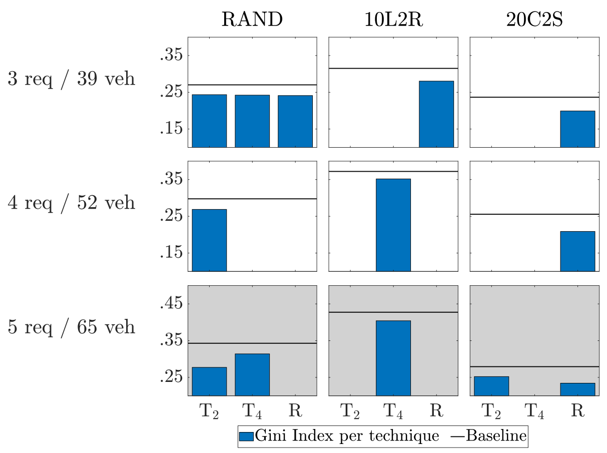


Fig. 4: Sensitivity analyses over a small artificial network. In the left, we vary the number of requests emerging per minute, while in the right we also adjust the number of vehicles to keep the same proportion. The rows corresponding to the original experiment are marked with a grey background. Missing columns mean that the respective technique is not able to find a solution within the accepted rejection threshold.

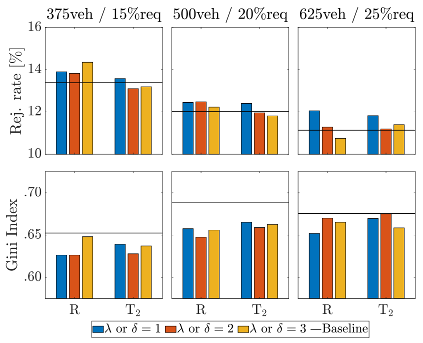


Fig. 5: Overall rejection rates and Gini Indices achieved by the R and $T_2$ techniques, with different tuning parameters when simulating the operation of a PMoD system in Manhattan with real requests. The results without any technique are marked with a horizontal black line.

## References

[1] M. Diao, H. Kong, and J. Zhao, "Impacts of transportation network companies on urban mobility," *Nature Sustainability*, pp. 1–7, Feb. 2021.

[2] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, Jan. 2017.

[3] D. J. Fagnant and K. M. Kockelman, "Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in Austin, Texas," *Transportation*, vol. 45, no. 1, pp. 143–158, Jan. 2018.

[4] A. Simonetto, J. Monteil, and C. Gambella, "Real-time city-scale ridesharing via linear assignment problems," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 208–232, Apr. 2019.

[5] A. Fielbaum, X. Bai, and J. Alonso-Mora, "On-demand ridesharing with optimized pick-up and drop-off walking locations," *Transportation Research Part C: Emerging Technologies*, vol. 126, p. 103061, May 2021.

[6] D. Zhao, X. Guo, X. Li, and S. Samaranayake, "Demand-driven operational design for shared mobility with ride-pooling options," Mar. 2020. [Online]. Available: ecommons.cornell.edu/handle/1813/70213

[7] N. Raman, S. Shah, and J. P. Dickerson, "Data-driven methods for balancing fairness and efficiency in ride-pooling," 2020. [Online]. Available: www.mlforeconomicpolicy.com/papers/MLEconPolicy20paper10.pdf

[8] K. Faber and D. van Lierop, "How will older adults use automated vehicles? assessing the role of AVs in overcoming perceived mobility barriers," *Transportation Research Part A: Policy and Practice*, vol. 133, pp. 353–363, Mar. 2020.

[9] A. Fielbaum and S. Jara-Diaz, "Assessment of the socio-spatial effects of urban transport investment using Google Maps API," *Journal of Transport Geography*, vol. 91, p. 102993, Feb. 2021.

[10] F. Di Ciommo and K. Lucas, "Evaluating the equity effects of road-pricing in the European urban context – the Madrid metropolitan area," *Applied Geography*, vol. 54, pp. 74–82, Oct. 2014.

[11] A. M. Ricciardi, J. C. Xia, and G. Currie, "Exploring public transport equity between separate disadvantaged cohorts: a case study in Perth, Australia," *Journal of Transport Geography*, vol. 43, pp. 111–122, Feb. 2015.

[12] D. Hörcher and D. J. Graham, "The Gini index of demand imbalances in public transport," *Transportation*, 2020. [Online]. Available: doi.org/10.1007/s11116-020-10138-4

[13] S. Lotfi, K. Abdelghany, and H. Hashemi, "Modeling framework and decomposition scheme for on-demand mobility services with ridesharing and transfer," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 1, pp. 21–37, Jan. 2019.

[14] C. Riley, A. Legrain, and P. Van Hentenryck, "Column generation for real-time ride-sharing operations," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, Jun. 2019, pp. 472–487.

[15] A. Wallar, M. Van Der Zee, J. Alonso-Mora, and D. Rus, "Vehicle rebalancing for mobility-on-demand systems with ride-sharing," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2018, pp. 4539–4546.

[16] A. Fielbaum, M. Kronmuller, and J. Alonso-Mora, "Anticipatory routing methods for an on-demand ridepooling mobility system," *arXiv preprint arXiv:2106.14685*, 2021.