2017 International Conference on Data Science and Advanced Analytics

# The Data and Science Behind GrabShare Carpooling

Muchen Tang, Serene Ow, Wenqing Chen, Yang Cao, Kong-wei Lye, Yaozhang Pan
Email: [muchen.tang, serene.ow, wenqing.chen, yang.cao, kongwei.lye]@grab.com, yaozhang.pan@gmail.com

***Abstract*—As internet-based ride hailing platforms such as Grab, Lyft, Uber become more and more popular, the demand for pooling services have increased sharply in recent years. In December 2016, Grab first rolled out its own pooling (ride-sharing) service "GrabShare" in Singapore. By providing considerable discount to passengers along with good matching performance to drivers, GrabShare enjoyed a rapid growth in Singapore to reach two million accumulative rides within the first two months. The success in Singapore motivates its expansion to other cities such as Manila, Jakarta, Kuala Lumpur, Ho Chi Minh City and so on [1].**

**In this article we explicitly present how the GrabShare algorithm was developed from a data perspective and how various ways of formulating the problem can have different impact to Grab, passengers, and drivers. Specifically, we start from a data perspective on how the GrabShare matching problem can be tackled by formulating an optimization model and how the optimization challenges are solved. Subsequently, we also present our critical data insights on improving user experience in the design of GrabShare and its continuous improvement for diverse markets.**

## I. A Little History

By matching different travelers with similar itineraries in both time and their geographic locations, ride-sharing can improve driver utilization and reduce traffic congestion. This concept of pooling (or called ride-sharing), has been a popular concept for decades due to its significant societal and environmental benefits. Tremendous interest in the real-time or dynamic pooling system has grown in recent years, either from a pooling matching algorithm (e.g., [2], [3]) or a system efficiency perspective [4]. We refer interested readers to [5]–[8] as a comprehensive overview on how optimization and operations research models in academic literature can support the development of real-time pooling systems and innovative thinking on possible future ride-sharing modes.

Leveraging on an internet-based platform that integrates passengers' smart-phone data in real-time, we are able to provide a ride-sharing service that allows passengers to spend less while enabling drivers to earn more. Companies such as Didi, Grab, Lyft and Uber have managed to transform the concept of a real-time pooling service from imagination into reality. Even though the problem of how to match drivers and riders in real-time has been extensively studied by various optimization technologies in literature (e.g., Avego's ride-sharing system [5] and Lyft match making [9]), there has been renewed interest in the problem and how we solved it in practice.

Let us turn the clock back to late 2015. It was when Grab's Data Science (Optimization) team was born. The team decided to eschew the literature and current state of the art, and challenged ourselves to designing the GrabShare matching algorithm from the ground up, from basic principles. Indeed, its main task was to make ride matching decisions (which is combinatorial) in order to maximize the overall system efficiency, while satisfying specific constraints to guarantee good user experience (such as detour, overlap, trip angle, and efficiency). A general optimization problem comprises of three main parts: 1. Objective function, 2. Constraints, and 3. Decision Space. The constrained optimization problem takes the usual form:

$$\begin{aligned} \min \quad & f(X) \\ \text{s.t.} \quad & g(X) \leq 0 \\ & h(X) = 0. \end{aligned}$$

Here $X$ denotes a set of decision variables that correspond to real-world decisions we can adjust or control. The objective function $f(X)$ is either a cost function that we want to minimize, or a value function that we want to maximize. The constraints are mathematical expressions of physical restrictions to decision variables on the possible solutions, which could have either inequality form: $g(X)$ or equality form: $h(X)$ or both.

In this article, we discuss how the GrabShare matching algorithm is tackled as an optimization problem and how its various formulations can have different impact to Grab, passengers, and drivers. Differing from previous studies in literature, which mainly focus on improving overall system efficiency using conventional operations research methods, we approach the problem from a more data-driven perspective. Our key focus was on extracting critical insights from data to improve GrabShare user experience, from the point of design and development of the matching algorithm and throughout subsequent continual efforts of product improvement. In Section II, we start by presenting some data-based results on single ride service GrabCar that motivated the concept of GrabShare. The subsequent Section III outlines our main GrabShare ride matching algorithm along with user experience specifications. Section IV talks about how we can rely on data for the design of GrabShare and its continuous improvement. Concluding remarks and future extensions are summarized in Section V.

978-1-5090-5004-8/17 $31.00 © 2017 IEEE
DOI 10.1109/DSAA.2017.68

## II. From GrabCar to GrabShare

From 2012 onwards, Grab has a mature product named "GrabCar" that serves millions of individual traveling requests by an integrated dispatching system. The drivers' locations and other states are maintained in the system such that we can simultaneously find drivers and make assignments for thousands of traveling requests. With a GPS-enabled mobile-phone, the users (known as passengers) can use Grab's passenger app to place transportation requests from specified origin to destination. In the article we use the term "booking" to denote a confirmed transportation request placed by a passenger, which contains explicit pickup and drop-off information. At the same time, drivers who have registered with their own or rented vehicles can login to Grab's system through a driver app to indicate their readiness to take nearby passengers. The GrabCar service is similar to a traditional taxi service in that a completed GrabCar ride consists of three steps: 1. A passenger makes a booking; 2. A GrabCar driver is assigned to the booking; 3. Assigned driver picks up the passenger and ferries him/her to the destination and the ride completes.

It is common for people to arrange for manual ride-sharing with our friends to save on travel cost as well as to socialize and connect during the trip, if they happen to be traveling in the same direction. By making use of real-time integrated ride information in the Grab system, we are able to automatically match strangers traveling in similar directions and assign the same vehicle to both their journeys, allowing them to effectively car-pool. Before promoting the concept of GrabShare, we verify its potential from the existing GrabCar bookings. For example, in the morning peak hour we map every single booking into a four-dimensional vector with the latitudes and longitudes of pickup and drop-off locations. In addition, the latitudes and longitudes are transformed into Universal Transverse Mercator (UTM) format to map the earth's surface to an 2-dimensional Cartesian Coordinate System for distance calculation. After applying DBSCAN cluster method [10] with parameter "eps=300", which means that only bookings with distance of less than 300 meters can be considered as neighborhoods, we observed eight clear clusters of booking with close pickup and drop-off locations in Figure 1. The booking requests within each cluster can be allocated and fulfilled with less vehicles, through pooling. Even though not all of them may be willing to share vehicles with others, at least those with unallocated bookings (around 8%) may benefit. After repeating this analysis for different time periods, we observe that a certain percentage of the bookings can be covered with good performing clusters in Table I. We can see that the coverage rate for different time periods, which fluctuates from 35% to 45% for most part of the day (coverage during mid-night and early morning hours is much smaller as the booking amount is much smaller). Because bookings in same cluster are "near perfect matches" with very close pickup and drop-off locations, the potential of GrabShare was found to be quite promising because we can expect even more opportunities for matching in the middle of a trip.

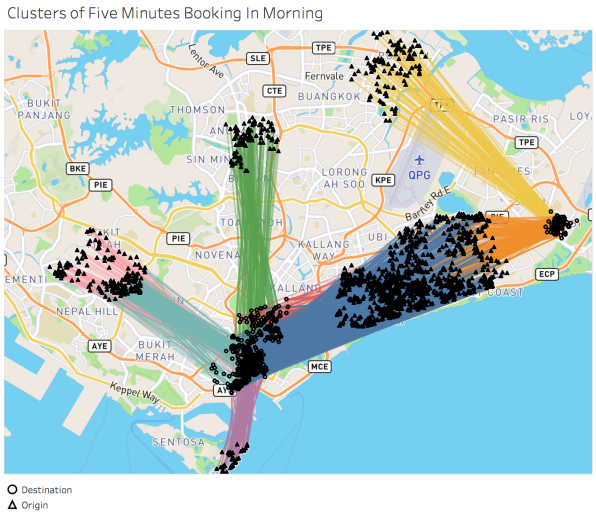


Fig. 1. Morning booking clusters with similar itineraries

TABLE I
CLUSTER COVERAGE RATE OF DIFFERENT TIME PERIODS.

| Hours | 8–10 | 10–13 | 14–16 | 16–18 | 18–22 | Others |
|---|---|---|---|---|---|---|
| Coverage | 46% | 39% | 35% | 38% | 43% | 22% |

The assignment flow of typical GC bookings is stated in Table II. For every newly arrived booking, we search for nearby drivers and check for their availability condition. If no driver is available, we recycle it to the next round of assignment. Otherwise we select the most suitable driver for them. Leveraging the current system structure, we extend the GrabCar service to GrabShare by maintaining more detailed booking and driver state information along with an additional check on seat reservation.

Specifically, Table III gives the assignment flow of GrabShare bookings. We can see that its overall structure is the same with GrabCar except for two differences. Firstly, the candidate driver set is different. For every new GrabShare booking, we search for in-transit GrabShare drivers who are currently serving at least one GrabShare booking. Therefore, we need to check seat availability condition to ensure that the vehicle has enough remaining seats to serve the new GrabShare booking. Mathematically, the following seat reservation constraint needs to be satisfied for a successful assignment

TABLE II
GRABCAR BOOKING ASSIGNMENT FLOW:

**for** Every generated GrabCar booking $bk_i$ **do**
  Find nearby available GrabCar drivers of $dr_j$
  **if** There are available GrabCar drivers **then**
    Assign the best driver $dr_{j*}$ to passengers.
  **else**
    Recycle the booking to next round broadcasting and assignment
  **end if**
**end for**

TABLE III
GRABSHARE BOOKING ASSIGNMENT FLOW:

**for** Every generated GrabShare booking $bk_i$ **do**
  Find nearby in-transit GrabShare driver of $dr_j$ and check seat availability condition.
  **if** There exists find suitable match pair $(bk_i, dr_j)$ **then**
    Update occupied passengers $o_p(dr_j) \leftarrow o_p(dr_j) + r_p(bk_i)$.
  **else**
    Recycle the booking to next round broadcasting and assignment.
  **end if**
**end for**

between booking $bk_i$ and driver $dr_j$:

$$s(dr_j) \geq o_p(dr_j) + r_p(bk_i), \tag{1}$$

where $s(dr_j)$ denotes the total capacity of the vehicle $dr_j$, $o_p(dr_j)$ is one of the maintained variable that denotes the current occupied capacity of the vehicle $dr_j$ and $r_p(bk_i)$ is the required capacity for booking $bk_i$. To make it consistent, we also need to update the vehicle occupied capacity variable $o_p(dr_j)$ by adding the booking required capacity $r_p(bk_i)$ after every successful assignment or removing it if cancellation occurs. Secondly, GrabShare's user experience is different from GrabCar due to the sharing concept. Here we defined some measures to evaluate the GrabShare matching, taking into consideration of trip angle, eta (short for Expected Time of Arrival), detour and efficiency. These measures are used to exclude unacceptable matches and to quantify how good the match is. The details are explained in the Optimization Model in the subsequent section.

## III. GRABSHARE MATCHING ALGORITHM

The matching algorithm is the core of the GrabShare product which could dramatically affect this product's success. In this Section we discuss how insights from data motivate our formulation of optimization models to support "booking/driver" assignment decisions. Specifically, the matching algorithm addresses the matching problem by two critical questions. First, we need to build a consistent way of determining the quality of the pair. Second, we need to guide the GrabShare driver by a step-by-step sequence of pickup and drop-off actions so that driving is the only task they need to do. In Subsection III-A we describe how we can formulate the matching problem as an optimization problem with constraints and objective function. Subsection III-B gives further route selection considerations in improving user experience.

Here we use a specific term "match" to describe the pairing between a new GrabShare booking and an in-transit GrabShare driver (The pairing of a GrabShare booking to an unoccupied vehicle can be treated the same as GrabCar). Without loss of generality, we denote $bk_0$ as the new booking and the $dr_0$ as the in-transit GrabShare driver who is currently serving bookings $bk_1, bk_2, \cdots, bk_n$, which are indexed in decreasing order by their booking generation time. Also, we introduce the concept "route" as a sequence of pickup and drop-off steps. Therefore, a matching route is a route consisting of all the relevant bookings' pickup and drop-off steps. For the match

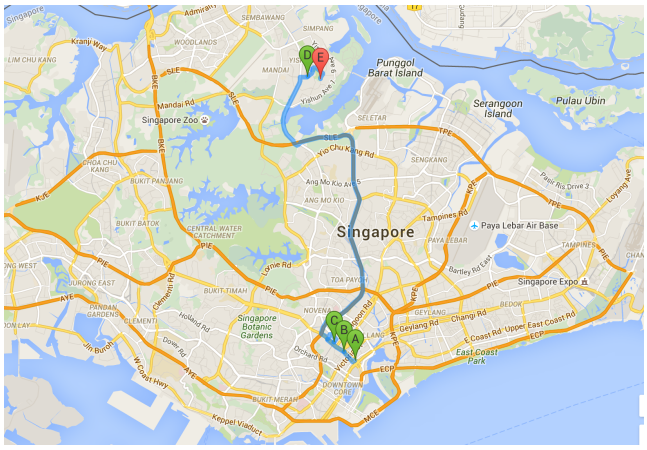


Fig. 2. GrabShare match case in Singapore

between new booking $bk_i$ and an in-transit driver $dr_j$, we denote $\{route_l^{i,j}\}_{l \in \mathcal{L}}$ as the set of potential feasible matching routes. In this article we may simplify $route_l^{i,j}$ into $route_l$ by removing booking and driver indices if there is no necessity to identify them. As there might be multiple feasible matching routes for every individual match, we decompose the match scoring problem into the matching route level by evaluating specific matching route $route_l$ of booking $bk_0$ and $dr_0$.

### A. Optimization Model

Let us start from a simple matching route scenario of two bookings ($n = 1$) as shown in Figure 2. At the first step the driver receives the first GrabShare booking from point $A$ to $D$ (25 minutes direct trip time). After the driver picks up the first passenger and reaches location $B$ on his way to $D$, he/she is assigned to pickup the second booking from $C$ to $E$ (21 minutes direct trip time). A GrabShare match happens and the final route sequence is generated as $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$. With pooling, it takes 29.5 minutes for the first passenger and 27 minutes for the second passenger to reach their destinations, respectively. Overall it is a good match as the passengers are only delayed a little bit by pooling with a promising driver utilization rate (the driver only needs to drive 23.72km in total to serve two bookings, instead of a total of 39.13km if they are served separately).

Let us start from some obvious metrics motivated from this example. First of all, we observe that both first and second parties in this example are travelling roughly in the same direction. In addition, only a small deviation to the driver's original driving direction is caused by the interruption step to pick up the second passenger. A small interruption angle is important in a GrabShare match because the act of making a U turn to pick up a new passenger would give the impression of a large detour, hence resulting in a negative experience for both the driver and passenger, which is consistent with Lyft's user research finding:

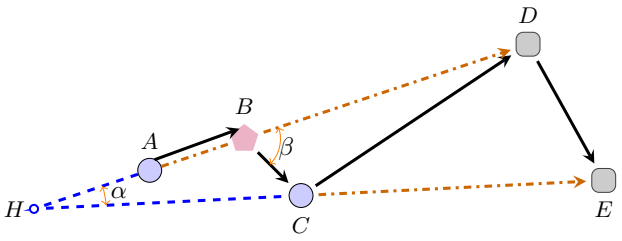


Fig. 3. Illustration of angle constraints

Users would often rather have a 10 minutes detour that had no backtracking than a 5 minute detour with backtracking [11].

For a specific route $route_l$ generated from a match between new booking $bk_0$ and in-transit GrabShare driver $dr_0$, we place an explicit constraint on the trip angle as follows:

$$\alpha_k^0 \leq \alpha^* \quad \forall k \in \{1, 2, \cdots, n\} \tag{2}$$

where $\alpha_k^0$ is the trip angle between booking $bk_0$ and booking $bk_k$, and a constraint on interruption angle $beta$:

$$\beta_l^0 \leq \beta^*, \tag{3}$$

where $\beta_l^0$ is the angle between the driver's current driving direction and the direction to the interrupted pickup step of booking $bk_0$ in route $l$. The explicit definition of $\alpha$ and $\beta$ for $n = 1$ case is given in Figure 3.

Second, similar to GrabCar service, the pickup distance (time) for the new booking is also important:

$$eta(bk_k, route_l) \leq eta^* \quad \forall k \in \{0, 1, 2, \cdots, n\} \tag{4}$$

Third, we observe that the match in Figure 2 has very little relative delay percentage: 4.5 minutes delay over a 25 minutes direct trip for the first booking and 6 minutes delay over a 21 minutes direct trip. Therefore, we can prevent poor quality matches by placing specific constraints on the passenger delay as follows:

$$delay(bk_k, trip_l) \leq delay^* \quad \forall k \in \{0, 1, 2, \cdots, n\}. \tag{5}$$

Finally, imagine the extreme case shown in Figure 4, in which the route consists of a series of bookings of trips going in the same direction and every booking's drop-off location coincides with its subsequent booking's pickup location. This should not be considered as a good matching route because it counters the most fundamental principle of ride-sharing: we encourage ride-sharing to improve vehicle utilization rate. Here we introduce the concept of route "efficiency" as the ratio of total driving distance when the bookings are served

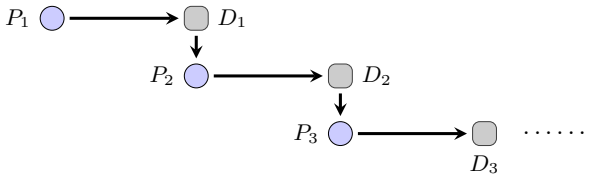


Fig. 4. Extreme case of zero overlap match routes

separately to the total route driving distance. Efficiency is a good measure of route performance: the matching route case in Figure 4 is not a considerably good match because its efficiency is 1, while the matching route case in Figure 2 is a better match because it has much higher efficiency 39.13km/23.72km=1.65. Hence, a constraint in route efficiency is given as:

$$efficiency(route_l) \geq efficiency^*. \tag{6}$$

With the above specified constraints and a proper defined objective function $\hat{f}(\cdot)$ that measures the benefit we gain from the matching route, we formulate the route planning of match between new booking $bk_0$ and in-transit GrabShare driver $dr_0$ as the following optimization model:

$$\max_{l \in \mathcal{L}} \left\{ \hat{f}(route_l) : (2), (3), (4), (5), (6) \right\}. \tag{7}$$

In general, $\hat{f}(\cdot)$ is a function of components like "eta index" that measure passenger's pickup experience, "delay index" that measures passenger's experience on delay and the matching route efficiency. Various form of the objective function $\hat{f}(\cdot)$ could be considered at different product stages.

### B. Route Selection Criteria

For every single match, there are multiple route options. For example, if a booking from $\overline{B}$ to $\underline{B}$ is matched with an in-transit driver who is serving an existing booking from $\overline{A}$ to $\underline{A}$, there are four potential route options: $(\overline{A}, \overline{B}, \underline{B}, \underline{A})$, $(\overline{A}, \overline{B}, \underline{A}, \underline{B})$, $(\overline{B}, \overline{A}, \underline{B}, \underline{A})$ or $(\overline{B}, \overline{A}, \underline{A}, \underline{B})$. The potential route set size is $\mathcal{O}(n!)$. This would significantly affect the complexity of the optimization problem (7) for large $n$.

We have to sacrifice some level of optimality by considering a narrow route set for quick computation. Fortunately some considerations around user experience shed some light on the solution. First, imagine a case that we assign the first booking to an unoccupied vehicle and the driver begins the trip to pick up the first passenger. If a second booking is subsequently matched and we re-schedule the driver to pick up the passenger from the second booking first, this may confuse and upset the driver. To ensure good passenger and driver experience, we prefer not to disorganize the original route steps in order. As a consequence, the potential route set size reduces to $\mathcal{O}(n^2)$ when a new booking is matched with an in-transit driver. In addition, if too many steps are inserted between passengers' pickups and drop-offs, the passenger experience would be bad because of excessive detour. To address this, we need an additional constraint on the matching route such that

passengers' drop-off step would be at most $K$ steps after its pickup step ($K$ is configured as two or three in production). Therefore, the potential route set is further reduced to $\mathcal{O}(n)$.

## IV. Data Driven Insights

Grab is powered by big data. We have access to one of the region's largest dataset that forms the foundation for us to apply machine learning, optimization, simulation, forecasting and other advanced techniques to solve bottlenecks and problems in transport and logistics to bring greater convenience to millions of passengers and drivers. In this Section we discuss how we make use of data at different product stages of GrabShare.

### A. Design Stage

In this subsection we describe our efforts in utilizing data to drive the GrabShare algorithm design. In particular, we discuss how historical data can help us more accurately estimate driver pickup and drop-off actions, and how to measure passengers' perceptions on "eta" and "delay".

In actual scenarios, the actions of picking up and dropping off usually require additional time, as the driver has to find a proper parking place or to communicate with the passenger. The typical amount of additional time differs from city to city due to traffic conditions and from driver to driver as their operations proficiency levels are different. By learning from drivers' historical recorded time of pressing the "pickup" and "drop-off" button and comparing their GPS pings, we recover the actual pickup and drop-off action time as the period from the moment the driver is within a 140 meter distance to target location to the time when we observe movements after "pickup" or "drop-off" action. Using these recovered data (around 1 million records), we trained a simple decision tree regression model with $depth = 4$, with $R^2 = 0.92$ on training set and mean absolute proportional error 7.6% on test set. This model is simple enough to give quick predictions because only around ten features are involved, including proficiency level (partitioned by number of completed rides by that driver) related features, booking time (e.g., in peak hours, Friday, weekends or others) and booking location related features.

Intuitively, we think a passenger's satisfaction level would be sufficiently high as long as his/her pickup eta is less than a certain threshold level (no significant difference if pickup eta is either half minute or one minute). This hypothesis is also reflected in data on booking cancellation rate against pickup eta values. From Figure 5 we can see that the cancellation rate is at the same level when eta is not greater than three minutes and increase linearly when eta grows from three to fifteen minutes. Specifically, we could model the cancellation probability as a piecewise linear function of eta value and use it as our "eta index" to denote passenger experience. The critical parameters of this piece-wise linear function can be estimated from booking cancellation data. In particular, their Maximum Likelihood Estimator (MLE) could be obtained by solving various bound-constrained optimization problems via L-BFGS-B algorithm [12]. In the planning stage where we don not have enough GrabShare data, GrabCar data can be used instead because their drivers' behavior on pickup and drop-off is similar. Similarly, a small level of travel delay should not negatively affect passenger experience because they are awarded with a discounted ride. We could also model passengers' perception on delay as piecewise linear functions and use the same method to get data-based estimations.

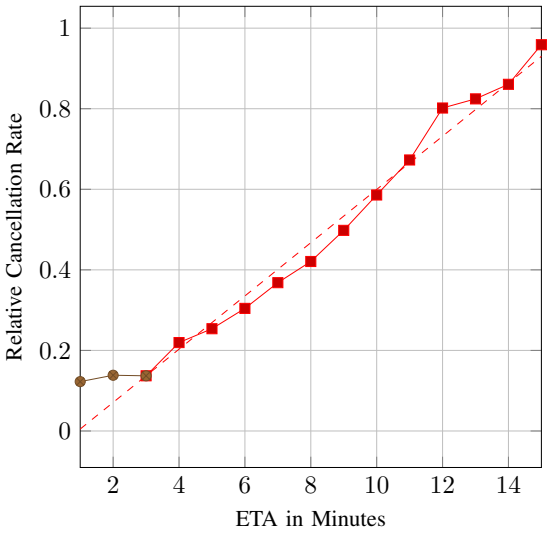


Fig. 5. Cancellation rate versus eta

We built a complete database to track historical matches for next step analysis. At the beginning stages of GrabShare, the main objective was to train and familiarize passengers and drivers with the product. The objective function concentrated more on the passenger level in terms of delay and eta experience. However, after analyzing the historical performance from data, we realized that the average trip efficiency in the first two weeks is not attractive enough. We thus put more emphasis on trip efficiency in the matching algorithm. Hearing the voice of our users is important to us, we collect users feedback on good and bad match examples for continuous improvement on the matching algorithm. For example, the angle constraints (2) and (3) were added after we observe a significant number of reported bad match cases with large angles $\alpha$ and $\beta$.

### B. Data based Simulation Tool for Continuous Improvement

We built our in-house GrabShare simulation tool to help us understand the markets and algorithm. In particular, we built the simulator as a black-box with booking data and initial driver location data as inputs, where the matching algorithm is run whenever this event is triggered. Proper randomness is introduced in passenger/driver's acceptance and cancellation behavior and driver movements. To improve its accuracy, we compared its output performance metrics such as allocation rate, match rate, average trip efficiency with real values, and took further actions to tune the simulator accordingly to close the gap.

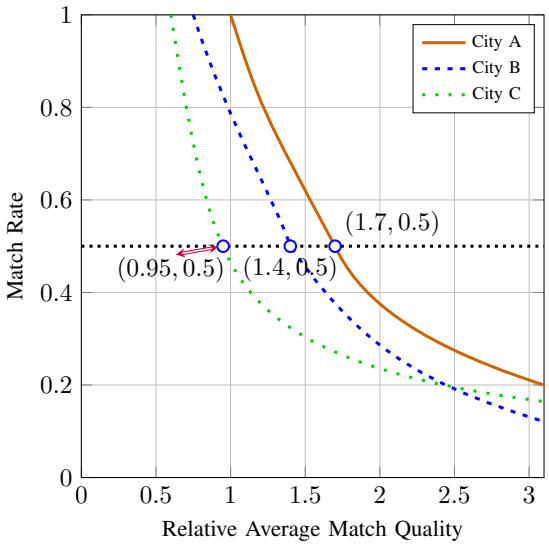


Fig. 6. Market evaluation

At the planning stage, we could use the simulator to investigate performances of different markets. With proper assumption on certain conversion rate (how many GrabCar bookings would turn to be GrabShare bookings), we proportionally sampled GrabCar booking data as the simulator input and draw a continuous curve of the system match rate versus average match quality as those in Figure 6. Here we define match quality as a combination of delay, overlap, and passenger pickup ETA measurements to reflect passenger and driver's overall experience of matches we provide. For the case given in this example, we think City A is best for launch because its average match quality is the highest when certain target match rate is met.

The GrabShare assignment-flow in Table III gives us a greedy-assignment strategy. The work in [13] finds that optimal batch processing of ten minutes window could improve match rate from 28% to 58%. In reality, our real-time GrabShare system could use at most one minute time window for batch processing to guarantee good passenger experience. Specifically, we use the $\mathcal{O}(n^3)$ implementation [14] of Hungarian algorithm [15] to solve for the optimal batch assignment problem under different simulation instances. We observe that average optimality benefits vary from 5% to 20% for batch time windows of 5 to 30 seconds. Indeed, faster algorithms are available in the literature ( [16] and [17]). However, the $\mathcal{O}(n^3)$ implementation is preferred because it is more stable than others and it is fast enough in practice (can solve instances of 1,000 bookings with less than 1 second).

After gathering enough GrabShare booking data, the simulator can further help us optimize our matching algorithm to achieve desired system performance. Specifically, we measure the overall system performance as a function of match rate and match quality. If we merge all the tunable variables into vector

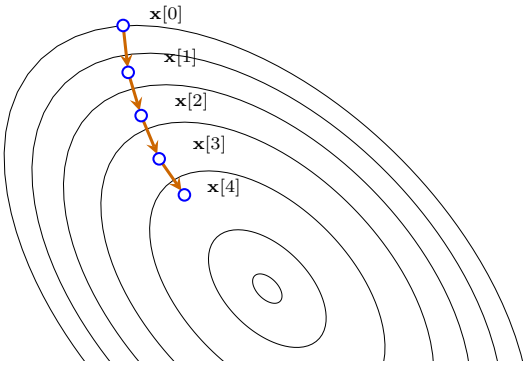


Fig. 7. Iterative method using gradient

$\mathbf{x}$, we could formulate this task as an optimization problem:

$$\max f(\mathbf{x}) = h(mq(\mathbf{x}), mr(\mathbf{x})). \tag{8}$$

Here $h(\cdot)$ is an increasing function of system match quality $mq(\cdot)$ and system match rate $mr(\cdot)$. For example, if we want a general target that match rate be higher than 0.3 and match quality higher than 0.25, a possible form of $h(\cdot)$ could be:

$$h(x,y) = (x-0.25)^+ - 2(0.25-x)^+ + (y-0.3)^+ - 2(0.3-y)^+.$$

As illustrated in Figure (7), general constrained optimization problems could be solved by gradient method if close form of $mq(\mathbf{x})$ and $mr(\mathbf{x})$ are available. However, the simulator could only provide us estimations of $mq(\mathbf{x})$ and $mr(\mathbf{x})$. To reduce computational effort, we use the sampling average estimation as a replacement of the gradient $\nabla f(\mathbf{x}_k)$ at the $k$th iteration at $\mathbf{x}_k$:

$$g(\mathbf{x}_k)_i = \frac{\hat{f}(\mathbf{x}_k + c_k * \mathbf{e}_i) - \hat{f}(\mathbf{x}_k - c_k * \mathbf{e}_i)}{2c},$$

where $\mathbf{e}_i$ is the ith unit vector, $c_k$ is some positive sampling step size at $k$th iteration and $\hat{f}(\cdot)$ denotes the estimation of $f(\cdot)$ by simulator.

## V. Conclusion and Future Extensions

This article describes how we developed GrabShare from concept to a real product. Its main contribution resides in illustrating the approach in practice rather than theory, by demonstrating how the effective use of data was beneficial in the design, deployment and subsequent improvement of the product. So far, GrabShare has been successful in generating awareness and increased openness of passengers towards ride-sharing in the bid to improve Southeast Asia's transportation conditions. Optimization is a continual, iterative process – a constant search for the best solution amidst changing preferences and behavior. Our efforts continue to seek to harness insights from data to improve GrabShare in the following directions:

- Training predictive models for quicker and more efficient classification of "good match" and "bad match", which

will help us improve user experience and reduce system computational load.

- Training accurate predictive models on passengers' matching probability and expected overlap when the booking is generated. Based on this our system could provide a higher ride discount to matched passengers with less detour, and less discount for unmatched passengers as they are actually served as GrabCar passengers.
- Building a reliable model to predict driver's acceptance probability for every passenger-driver pair, which can improve platform efficiency in batch processing. This approach could also be applied to GrabCar batch processing.

## REFERENCES


[1] Grab, "Grab extends grabshare regionally with malaysias first on-demand carpooling service," 2017. [Online]. Available: https://www.grab.com/my/press/business/grabsharemalaysia/

[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, Mar 2017.

[3] A. Conner-Simons, "Study: carpooling apps could reduce taxi traffic 75 precent," 2016. [Online]. Available: https://www.csail.mit.edu/ridesharing_reduces_traffic_300_percent

[4] D. Dimitrijevic, N. Nedic, and V. Dimitrieski, "Real-time carpooling and ride-sharing: Position paper on design concepts, distribution and cloud computing strategies," in *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*. IEEE, 2013, pp. 781–786.

[5] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.

[6] A. Amey, J. Attanucci, and R. Mishalani, "Real-time ridesharing: opportunities and challenges in using mobile phone technology to improve rideshare services," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2217, pp. 103–110, 2011.

[7] N. D. Chan and S. A. Shaheen, "Ridesharing in north america: Past, present, and future," *Transport Reviews*, vol. 32, no. 1, pp. 93–112, 2012.

[8] M. Furuhata, M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang, and S. Koenig, "Ridesharing: The state-of-the-art and future directions," *Transportation Research Part B: Methodological*, vol. 57, pp. 28–46, 2013.

[9] Lyft, "Matchmaking in lyft line — part 1," 2016. [Online]. Available: https://eng.lyft.com/matchmaking-in-lyft-line-9c2635fe62c4

[10] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[11] Lyft, "Matchmaking in lyft line — part 2," 2016. [Online]. Available: https://eng.lyft.com/matchmaking-in-lyft-line-691a1a32a008

[12] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[13] X. Wang, "Optimizing ride matches for dynamic ride-sharing systems," Ph.D. dissertation, Georgia Institute of Technology, 2013.

[14] I. Jurin, 2015. [Online]. Available: https://github.com/antifriz/hungarian-algorithm-n3

[15] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[16] R. Freling, A. P. Wagelmans, and J. M. P. Paixão, "Models and algorithms for single-depot vehicle scheduling," *Transportation Science*, vol. 35, no. 2, pp. 165–180, 2001.

[17] J. P. Paixão and I. Branco, "A quasi-assignment algorithm for bus scheduling," *Networks*, vol. 17, no. 3, pp. 249–269, 1987.