Globally-Optimized Realtime Supply-Demand Matching in On-Demand Ridesharing

Yifang Liu
Smule
ifnliu@gmail.com

Will Skinner
Engine ML
will@engineml.com

Chongyuan Xiang
Opendoor Labs
xiangcy93@gmail.com

ABSTRACT

In on-demand ridesharing, optimizing the quality of supply-demand matches and minimizing the number of unfulfilled trip requests is an important business problem. One approach is to match supply with demand based on greedy local heuristics, which is sub-optimal at the market level. Another approach is to find the globally optimal matching over the whole marketplace. However, because the computation has high latency, and usually requires aggregate supply and demand data over time, instead of only the supply and demand at the time of individual trip requests, it is unfeasible to perform the global optimization for every trip request in realtime. This paper proposes a solution that performs a global optimization offline periodically based on forecasted supply and demand data, and uses the offline results to guide realtime supply and demand matching. The solution is a novel two-stage robust optimization scheme for supply-demand matching in on-demand ridesharing. We implemented the proposed solution and evaluated it using simulated trips based on public industry data. It reduces the number of unfulfilled trips significantly compared to a state-of-the-art approach.

KEYWORDS

Ridesharing, Supply-Demand Matching, Robust Optimization

ACM Reference Format:
Yifang Liu, Will Skinner, and Chongyuan Xiang. 2019. Globally-Optimized Realtime Supply-Demand Matching in On-Demand Ridesharing. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313579

1 INTRODUCTION

A core value in the on-demand sharing economy is to effectively match supply and demand in realtime. Improving supply-demand matching can increase the utilization of suppliers and therefore reduce the service price. Through network effects, lower prices and higher utilization can help expand both supply and demand further in the market.

In this paper, we will focus on ridesharing, which has been used by hundreds of millions of people across the world. In this context, a supply unit stands for a driver, and a demand unit stands for a rider. How to match riders and drivers is at the core of a successful ridesharing platform's marketplace efficiency optimization.

The primary goal of supply-demand matching is to minimize the total unfulfilled demand. This calls for a non-trivial global optimization for trip-level supply-demand matching over all individual units of demand and supply in the whole geospace of a market (e.g. San Francisco Bay Area or Greater Los Angeles). However, in practice, a realtime trip request usually requires a matching decision to be made within only a few seconds. The global supply-demand matching optimization is often too complicated to finish in such a small amount of time.

As a result, one of the main limitations existing matching systems often have is the sacrifice of global optimality for quick suboptimal matching. They usually use local, greedy heuristics to perform supply-demand matching at individual trip request level, which often yield sub-optimal matching solutions, and cause high volumes of unfilled trip requests.

Another usual limitation of existing matching systems is the handling of future supply and demand. Existing matching methods often rely on point estimation, i.e., without considering the probability distribution of the predicted values. This is a main source of sub-optimal matching solutions.

To tackle the two challenges above, we propose a new approach, Globally-Optimized Realtime Matching (GORM). The algorithm has three main steps. First, we use historical time series data to predict the probabilistic distribution of supply and demand over the whole marketplace in the next 5 minutes. Second, we perform global optimization of supply-demand matching at an aggregated level over the whole market for the next 5 minutes using the forecasted data. To cope with the forecasting error, our global optimization will also find supplies to match the unpredicted extra true demands. Third, we use the pre-computed global optimization result to guide a local heuristic for realtime per-trip supply-demand matching.

In order to keep computational complexity low and maintain reasonable signal-to-noise ratios, we use hexagons (about 600 meters in diameter), as the smallest unit of geo area. For hexagonal geofences, we use Uber's open source library, H3 [3, 11].

The main contributions of this paper include:

(1) We present a new framework (GORM) for realtime efficient supply-demand matching for on-demand ridesharing. GORM combines periodical (offline) global optimization for market-wide aggregate supply-demand matching and realtime (online) local heuristics for per-trip supply-demand matching.
(2) We present a novel linear programming formulation for robust optimization that is resilient to the uncertainty in the underlying market data.
(3) We tested the proposed solution with simulated market dynamics based on real-world ridesharing data, and showed over 7-22% reduction on unfulfilled demand compared with a state-of-the-art supply-demand matching algorithm.

2 RELATED WORK

Because supply-demand matching is very important to the ridesharing business, there has been much related work on this topic.

One type of previous work focused on the realtime factor of the problem, and proposed to greedily find the nearest drivers for every incoming trip request [1, 4, 5, 8, 9]. Those approaches were likely to find suboptimal solution in terms of overall market efficiency.

Another type of previous work used more complex algorithms to find a globally more optimized matching of supplies and demands. The authors of the Didi Chuxing's Taxi order dispatch system [12] proposed a combinatorial optimization approach. They first predicted riders' destinations based on historical data, then estimated drivers' order acceptance rate based on the predictions, and finally searched for a globally optimized matching based on the above estimation. The main drawback of this approach was that it was hard to do such a large scale computation in realtime. Also, this approach did not put dispatch radius as a constraint, which could lead to long dispatch time and unsatisfactory rider experience. Another work [10] proposed an agent based model called NTuCab, where each agent processed N rider/driver pairs, and jointly minimized the rider waiting time. However, this approach did not try to minimize unfulfilled demands and could lead to suboptimmal market efficiency.

3 THE GORM FRAMEWORK

This section explains the globally-optimized realtime matching (GORM) framework for the global matching problem. The effectiveness of the GORM framework is related to the following aspects:

1) Forecasting the spatial and temporal distribution of demand and supply across the whole market's geo space over a short time period, e.g., 5 minutes in our case. The aggregate demand/supply forecast is always a probability distribution, not a point estimate. The GORM framework relies on the probabilistic estimation as the input to the global optimization problem.

2) Computing the globally-optimized matching solution into the future, based on the forecasts of demand and supply - their probability distribution. The global optimization needs to be resilient to the prediction error in the forecasts. GORM achieves this by optimizing the expected value of the cost function over the probabilistic demand/supply forecasts.

3) Performing the local trip-level matching, which is a heuristic guided by the forward-looking global-optimization result. There is a time constraint on the local matching decision. This way, each trip request can be matched to a supply in seconds, while the decision is aligned with the globally-optimized supply-demand matching at aggregate market level.

The GORM framework combines the forward-looking global matching optimization and the realtime local matching heuristic, by performing the 3 steps in Algo 1 for all the supply-demand matching decisions across the whole market over every 5-minute time window.

In the GORM framework, Step 1 can be performed with an existing time series forecasting algorithm, e.g., multivariate ARIMA. Since it is a relatively mature practice with off-the-shelf statistical tools, we will skip the explanation on it, and spend time on Steps 2 and 3 in the main body of the GORM framework.

ALGORITHM 1: Globally-Optimized Realtime Matching

```
input  : the 5-minute time window to perform
         supply-demand matching over; all the hexagons
         in the whole marketplace
output : at the hexagon aggregate level,globally-optimized
         inter-hexagon supply relocating flow
         $x_{i,j}, \forall i, j \in \{hexagons\}$; at individual request
         level, a driver (if feasible) being matching to a trip
         request.
1 Step 1) Perform time series forecasting (with confidence
    intervals) for demand and supply in the next 5 minutes in the
    whole market, i.e., obtaining their confidence intervals
    ($\alpha = 0.05$): $\forall i \in \{hexagons\}$, $[T_i^l, T_i^u]$, $[S_i^l, S_i^u]$ for the next 5
    minutes;
2 Step 2) Perform optimization on the global matching problem,
    taking as the input the demand/supply forecasting across the
    whole market over the next 5-minute window, i.e.,
    $\forall i \in \{hexagons\}$, $[T_i^l, T_i^u]$, $[S_i^l, S_i^u]$. As a result, we solve the
    problem for $\forall i, j \in \{hexagons\}$, $x_{i,j}$ optimizing the whole
    market's efficiency over the next 5 minutes;
3 Step 3) Perform realtime supply-demand matching for each
    individual trip request, based on the globally optimized
    inter-hexagon supply flows from Step 2. Basically a demand
    in hexagon $j$ is preferred to be matched to a supply from
    hexagon $I$ with the highest supply-to-demand ratio among
    all corresponding candidate hexagons $\forall i$ where $x_{i,j} > 0$.;
```

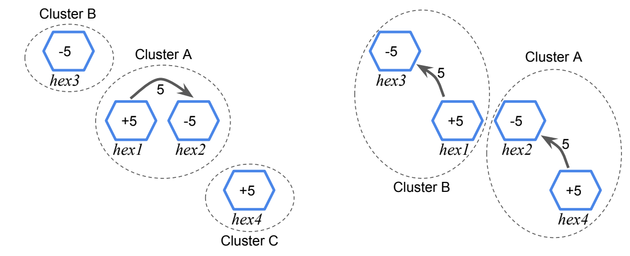

Figure 1: Supply-Demand Matching with Different Hexagon Clusters.

4 GLOBAL MATCHING

This section explains the global supply-demand matching, including its optimization goal, geospatial constraints, its implications for the ridesharing business, and our novel technique for handling the uncertainty in market forecasts, which are used as input to the global optimization problem.

An accurate yet compact representation of the spatial distribution of supply and demand requires dividing the whole market geospace into small basic geo units. Among all commonly-used shapes of the basic geo units, hexagons are our choice in solving this problem, because of their geometrical consistency, good coverage over the whole surface of the earth, and its roundness. Next we illustrate how global optimization can make a big difference in trying to achieve high market efficiency. In the following example, we consider two quite extreme ways of supply-demand matching:

1) matching individual demand to currently nearby supply by local greedy heuristics, without considering their interaction with other demand and supply further in the whole market; 2) matching all demand to all supply in the whole geo space of the marketplace across a certain time period, while minimizing the total unfulfilled demand globally.

In Fig. 1, we use a spatial cluster (i.e., Cluster A, B, or C in the example) to depict a group of one or more hexagons, between which supplies are matched to demands. There are a total of 4 hexagons in the whole marketplace, each of which is associated with the amount of supply units subtracting the amount of demand units. $hex2$ and $hex3$ both have 5 units of demand and no supply. $hex1$ and $hex4$ both have no demand and 5 units of supply. The spatial clusters on the left give an example of local greedy supply-demand matching. When demands in $hex2$ occur, we will match them with supplies from $hex1$ because the clusters are closest to each other. After that, when demand starts to surface in $hex3$, there are no supplies in Cluster B to match. Moreover, $hex1$ already run out of available supplies, and $hex4$ is too far away from $hex3$. Therefore, the system can only fulfill 5 out of 10 demand units. In contrast, the spatial clusters on the right side of the figure give an example of better globally optimized supply-demand matching in the same market scenario. In this case, we use $hex1$ to supply $hex3$, and $hex4$ to supply $hex2$. Therefore, all 10 demand units are fulfilled.

Now it is clear that the supply-demand matching problem needs a global optimization, we start to explain the problem and considerations around it. The input to our global matching optimization problem is a forecast of the demand and supply volumes across all individual hexagons in the whole market over a period of time (5 minutes in our case).

The primary goal of our matching problem is to minimize the total unfulfilled demand over all hexagons, i.e., minimize: $\sum_i \max(T_i - S_i, 0)$, where T_i denotes the volume of demand in hexagon i; and S_i denotes the total volume of supply in hexagon i, which is the summation of the supply originating from hexagon i and the supply relocated from other hexagons to hexagon i.

This goal is achieved by optimizing the supply movement flow between different pairs of hexagons. That is, the decision variable $x_{i,j}$ indicates the volume of supply that is relocated from hexagon i to hexagon j, in order to fulfill the demand in hexagon j. The supply movement from hexagon i to hexagon j effectively means dispatching the corresponding amount of supply in hexagon i to demand in hexagon j.

Because the forecast is subject to prediction error, our optimization problem formulation needs to take into account the uncertainty in the forecast. For the clarity of explanation, the rest of this section will describe our deterministic optimization formulation (without considering forecasting error) first, and then present the robust optimization formulation for coping with forecasting uncertainty.

4.1 The deterministic optimization

Before diving into the global optimization problem formulation, we explain a geospatial constraint on the supply movement. Supply can only move between two hexagons if they are within a dispatch radius to each other. This constraint is very important to a real-world on-demand ridesharing service: a unit of supply should be matched to a nearby unit of demand, to keep the estimated time to pickup (ETP) reasonably short for realtime trip requests.

The global matching optimization can be formulated into the following linear programming problem.

$$\begin{aligned} \text{Min:}\;\; & f(x) = \alpha \sum_i h_i \\ \text{S.t.:}\;\; & \forall i \in \mathbb{A}, h_i \geq T_i - s_i, h_i \geq 0 \\ & \forall i \in \mathbb{A}, s_i = S_i - \sum_j x_{i,j} + \sum_k x_{k,i} \\ & \forall i \in \mathbb{A}, \sum_j x_{i,j} \leq S_i \\ & \forall i, j \in \mathbb{A}, Dist_{i,j} \leq O, x_{i,j} \geq 0, x_{j,i} \geq 0 \end{aligned} \tag{1}$$

where $\mathbb{A}$ denotes the set of all hexagons in the whole market; $x_{i,j}$ is a decision variable representing the supply volume flowing from hexagon i to hexagon j; h_i is an auxiliary variable representing the unfulfilled demand units in hexagon i; s_i is an auxiliary variable representing the total volume of supply in hexagon i (including those originated in hexagon i and those moved in-and-out from/to other hexagons); O denotes the maximum distance between two hexagons $Dist_{i,j}$ that allows the inter-hexagon supply flows.

4.2 The robust optimization

Because forecasts are inherently uncertain, the optimization process must be robust to actual supply and demand being different from what was forecasted. Furthermore, we must deal with prediction errors in a manner that is consistent with the overall optimization. This is similar to a 2-stage stochastic programming problem in general [2], which can be formulated as follows in our case.

$$\begin{aligned} \text{Min}_x:\;\; & g(x) = f(x) + E_\zeta[Q(x,\zeta)] \\ \text{S.t.:}\;\; & Q(x,\zeta) = \min_y \Big(q(y,\zeta) = T(\zeta)x + W(\zeta)y\Big) \end{aligned} \tag{2}$$

where ζ represents the vector of one possible combination of true supply and demand volumes over all hexagons in the marketplace; $Q(x, \zeta)$ denotes the optimal value of the second stage problem on a specific value of ζ; y indicates the recourse decision (matching on individual trip request level in our case) in the face of ζ; $E_\zeta[Q(x,\zeta)]$ is meant to capture the expected value of cost by all possible true volume values of demand and supply over all hexagons (there is a probability distribution over these possible values).

In our case, we create an uncertainty-resilient linear programming formulation for our matching problem. Basically, the robust optimization formulation wants to minimize the expected unfulfilled demand over the probability distribution of possible volume values given by the confidence interval of the market forecasting. We choose to use 95% confidence interval, corresponding to $\alpha = 0.05$, in the forecast of the market in the next 5 minutes.

In its essence, the robust optimization part of the formulation tries to approximate different probabilities of an amount of supply occurring in a hexagon in reality with different unit costs (they are set as problem parameters) of using the corresponding supply, i.e., matching the supply with some demand. Similarly, the unfulfilled demand's probabilistic unit costs are formed this way.

This is done by setting the unit cost of supply proportional to $1 - CDF$ (the reverse cumulative distribution function) at the volume range corresponding to the supply being charged. Let's use a simplified made-up example to see how this works. Assume our forecast with confidence interval says that S_i, the value of the supply volume originates in hexagon i, has 5% chance to randomly fall at somewhere between 1 and 10, while the chance for the value to take a random place between 11 and 15 is 95%. Now if we use supply within the [1,10] range, on average 97.5% of the time, we will be able to find the supply in the true market reality, because when S_i takes a value in [11,15] by 95% chance, we are warranted with obtaining supply from the first 10 of them. Therefore, we assign the unit cost of supply in [1,10] with $\beta(1-97.5\%)$. By similar reasoning, we can assign the unit cost of supply in [11,15] with $\beta(1-47.5\%)$.

This way, the linear relation between the cost and the probability of forecasted demand/supply approximates the expected value of the uncertain cost term $E_\zeta[Q(x,\zeta)]$, and our robust optimization formulation approximates the stochastic programming formulation given in Equ. 2.

In practice, without loss of generality, we simplify the forecast probability distribution to being piece-wise uniform, just like in the simplified example above. The approximation of more sophisticated forecast probability distribution can be done in similar manner.

$$\begin{aligned}
\text{Min:}\ & g(x) = \alpha \sum_i h_i + \sum_i \beta_i \tilde{s_i} + \sum_i \gamma_i \tilde{h_i} \\
\text{S.t.:}\ & \forall i \in \mathbb{A}, h_i \geq T_i - s_i, h_i \geq 0 \\
& \forall i \in \mathbb{A}, s_i = S_i - \sum_j x_{i,j} + \sum_k x_{k,i} \\
& \forall i \in \mathbb{A}, \sum_j x_{i,j} \leq S_i \\
& \forall i,j \in \mathbb{A}, Dist_{i,j} \leq O, x_{i,j} \geq 0, x_{j,i} \geq 0 \\
& \forall i \in \mathbb{A}, h'_i \geq (T_i + T'_i) - (s_i + s'_i), h'_i \geq 0 \\
& \forall i \in \mathbb{A}, \tilde{h_i} \geq h'_i - h_i, \tilde{h_i} \geq 0 \\
& \forall i \in \mathbb{A}, s'_i = \tilde{s_i} + s_i \\
& \forall i \in \mathbb{A}, 0 \leq \tilde{s_i} \leq S'_i
\end{aligned} \tag{3}$$

where $x_{i,j}$ and h_i remains their representation as in the deterministic optimization formulation Equ. 1; T_i and S_i are the lower bounds T_i^l, S_i^l of the confidence intervals of the predicted demand and supply, $[T_i^l, T_i^u]$, $[S_i^l, S_i^u]$, respectively. T'_i and S'_i are the spans of the confidence intervals of the predicted demand and supply, $T_i^u - T_i^l$ and $S_i^u - S_i^l$, respectively. As a result, $\tilde{t_i}$ and $\tilde{s_i}$ represent the usage of demand and supply in their corresponding confidence intervals, respectively.

5 REALTIME MATCHING

Realtime matching at the individual request level corresponds to the second stage decisions in a typical 2-stage stochastic programming. It is a simple, fast heuristic that uses the globally-optimized inter-hexagon supply flows (which are obtained from the global matching) to guide the realtime dispatch for individual demand. This way, our final matching on individual trip requests can leverage the globally-optimized uncertainty-resilient aggregate matches, while being very runtime efficient so as to comply to the realtime matching time limit.

More specifically, for each trip request, the heuristic consists of two main operations.

(1) *Request matching guided by global optimization result*: for each trip request in hexagon j, collect the candidate hexagon set that contains all hexagon i that either is hexagon j itself or having a positive supply flow to j, i.e., $x_{i,j} > 0$, according to the current value of the global optimization result $x_{i,j}$. Rank the candidate hexagons by their supply-to-demand ratio in the order from high to low, and go through the candidate list from top to bottom while trying to find the hexagon to dispatch a supply in it to the trip request.
(2) *Request matching updating the global optimization result*: with the selected hexagon i for the trip request, update the corresponding supply flow value by: $x_{i,j} = x_{i,j} - 1$.

6 EXPERIMENT

This section evaluates the effectiveness and efficiency of GORM's application on the globally-optimized realtime matching problem. We compare GORM against several existing matching heuristics and state-of-the-art matching optimization algorithms. Two publicly-available raw input data sources are used in generating the simulated demand/supply distribution data for the experiments: 1) Uber passenger pickup data in New York City [6], short named pickup data; 2) Google Maps street speed data in New York City [7], short named speed data.

The snapshots of the demand and supply distribution over the marketplace of NYC are generated in 4 parts based on the raw input data: 1) the matched demand in reality, 2) the matched supply in reality, 3) the unmatched demand estimated by simulation, and 4) the unmatched supply estimated by simulation. Parts 1 and 2 are obtained directly from the pickup data, except that the dispatch-time location of a matched supply is randomly assigned within the dispatch radius from the pickup location. Part 3 is estimated by simulation based on the street speed at the corresponding location. Basically, the slower the driving speed around the pickup location is, the heavier the traffic around the location is, and likely the more demand is around the location. Part 4 is estimated based on the volume of the unmatched demand. If there is unmatched demand, the surrounding area should have no unmatched supply; otherwise, the unmatched supply is randomly assigned to locations in proportion to the nearby pickups.

The experiments are performed over the simulated snapshots of demand and supply distribution in 5-minute windows over the marketplace of different sizes. Three sub-regions of NYC in different sizes - mimicking marketplace of different geo sizes - were used in the experiments: city-big with $80k$ hexagons, city-medium with $40k$ hexagons, and city-small with $20k$ hexagons, respectively.

The experiments are set up as follows: For each sub-region of the city, snapshots of 5-minute market dynamics are used as the test time windows. Inside each 5-minute window, individual units of demand and supply surface and expire in their respective time limits. For example, individual demand's expiration time, the realtime supply-demand matching time limit, is set to 10, 15, 20, 25 and 30 seconds in the experiments.

The simulated supply and demand data of each of the 5-minute windows are available to serve as the "historical" data for the market forecasting in the corresponding 5-minute window, should it be needed by an algorithm in the experiment. For simplicity, we assume the chosen matching algorithm will not significantly change supply and demand at the hexagon level over 5-minute windows. This allows us to use the simulated data without running a true experiment with interventions.

We experiment and observe how the matching solution quality responds to changes on the problem parameters, including the error in demand/supply forecasts, realtime matching time limit. On the other hand, some problem parameters are fixed for all experiments. For example, the maximum-allowed geospatial distance in dispatching a supply in a hexagon to another hexagon is 6 hexagons away.

6.1 The algorithms in the experiments

We test and compare our uncertainty-resilient globally-optimized realtime supply-demand match method, GORM, with three baseline algorithms. The three baseline algorithms represent greedy local matching heuristics, and global matching that needs more than a few seconds' runtime to come up with the solution, global matching without treatment to deal with the uncertainty in the market forecasts, respectively. They are briefly described as follows.

(1) Baseline algorithm 1 (LOC-GRD): Nearest neighbor dispatch with a preference of matching supply in hexagons having higher supply-to-demand ratio. This represents greedy, local matching heuristics. While missing global optimization, this method satisfies the realtime matching requirement with ease. It does not need market forecast, and thus it does not face the challenge of prediction uncertainty.

(2) Baseline algorithm 2 (GLO-HIL): A modified implementation of Didi Chuxing's hill-climbing optimization method, which is held against the realtime matching time constraints. Because this method does not depend on a forecast of the market, all the demand/supply considered and optimized by this method is limited to which have occurred in the corresponding matching time window. Thus, the maximum runtime of the optimization algorithm is the matching time limit, and the best solution obtained by the stopping point is used as the matching solution.
For example, if the realtime matching time limit is set to be 15 seconds, then the 5-minute experiment window is divided into 20 15-second sub-windows, during each of which the matching algorithm is performed among all demand and supply occurring in that sub-window, and the optimization has at most 15 seconds to run. This represents globally-optimized matching that usually makes pre-mature sub-optimal matching decision, due to its extensive runtime for matching every trip request with an open supply in a limited market scope. Note that because our supply-demand matching problem ensures a close-to-100% order acceptance rate by the supply, a prediction of driver acceptance rate is unnecessary. For the simplicity of demonstration, we enforce the supply acceptance in all of our supply-demand matching decisions.

(3) Baseline algorithm 3 (GLO-DET): Globally-optimized realtime matching without robust optimization, i.e., global deterministic optimization. This method represents global optimization driven realtime matching, which depends on a market forecast in point estimation style, but does not consider or optimize over prediction error.

(4) Our algorithm (GLO-OUR): Globally-optimized uncertainty-resilient realtime matching. This represents our algorithm that depends on a market forecast with a 95% confidence interval.

6.2 Solution quality evaluation

The solution quality is primarily measured by the total volume of unfulfilled demand (**UFD**), defined as $UFD = \sum_i \max(T_i - S_i, 0)$.

To calculate the theoretical minimum total unfulfilled demand (UFD) for the matching solution, we run the global optimization formulated as Equ. 1 over the true market demand and supply dynamics that is given by the simulation in the corresponding 5-minute window in the market. Summing up the unfulfilled demand over all hexagons in the optimization result gives us the theoretical minimum value of the total unfulfilled demand (UFD), which is the lowest possible UFD for the matching problem.

The effectiveness of a matching algorithm can be better measured in UFD relative to the theoretical minimum UFD. The relative UFD (rUFD) is simply the difference between the UFD yielded by an algorithm and the theoretical minimum UFD. In other words, rUFD is the UFD gap between an algorithm's solution and the ideal best solution.

Next, we review how the 4 algorithms in our experiments perform with their matching solutions. Change problem parameters: demand/supply perturbation, different city sub-regions, and realtime decision time constraint - algorithm runtime limit.

In the following result review, we will use the short names for the 4 algorithms in our experiments. All metric values in the following tables is the percentage of the measurement over the total volume of demand in the whole market. The UFD values from all 4 algorithms are measured as the average over the same 5 snapshots of 5-minute market. Note that the lower the UFD and rUFD values are, the better; the higher the Fulfillment values are, the better.

The columns labeled with "abs" and "relative" indicate the measurement itself and the percentage of its delta relative to the same measurement of the first baseline algorithm (LOC-GRD), respectively. The column named as "Min UFD" indicates the theoretically lowest UFD possible.

Overall, the improvements on rUFD achieved by GORM over the base algorithms range from 24% to 60%. Given the usual unfulfilled demand percentage shown in the experiments, this rUFD improvement by our method translates into roughly $3 - 7\%$ increase on the total fulfilled trips. More detailed solution-quality experiment result on the 3 sub-regions can be found in Tables 1, 2, and 3.

Now we observe how different algorithms respond to the change of problem parameters, namely the uncertainty in demand/supply prediction (perturbed by $0.5\sigma, 1.0\sigma, 3.0\sigma$) and the realtime matching time limit (10, 15, 20, 25, or 30 seconds).

Figure 2 shows our algorithm GLO-OUR and GLO-DET's reaction to forecast uncertainty change in city-medium. GORM's

Table 1: UFD: 5-minute matching in city-small

	Min UFD	UFD		rUFD		Fulfillment	
		abs	relative	abs	relative	abs	relative
LOC-GRD	3.51%	15.07%	↓ 0%	11.56 %	↓ 0%	84.93%	↑ 0%
GLO-HIL	3.51%	13.66%	↓ 9.36%	10.15%	↓ 12.19%	86.34%	↑ 1.7%
GLO-DET	3.51%	11.09%	↓ 26.41%	7.58%	↓ 34.43%	88.91%	↑ 4.7%
GLO-OUR	3.51%	8.64%	↓ 42.67%	5.13%	↓ 55.62%	91.36%	↑ 7.6%

Table 2: UFD: 5-minute matching in city-medium

	Min UFD	UFD		rUFD		Fulfillment	
		abs	relative	abs	relative	abs	relative
LOC-GRD	5.83%	22.52%	↓ 0%	16.69%	↓ 0%	77.48%	↑ 0%
GLO-HIL	5.83%	21.69%	↓ 3.69%	15.86%	↓ 4.97%	78.31%	↑ 1.07%
GLO-DET	5.83%	19.89%	↓ 11.68%	14.06%	↓ 15.76%	80.11%	↑ 3.39%
GLO-OUR	5.83%	18.51%	↓ 17.81%	12.68%	↓ 24.03%	81.49%	↑ 5.18%

Table 3: UFD: 5-minute matching in city-big

	Min UFD	UFD		rUFD		Fulfillment	
		abs	relative	abs	relative	abs	relative
LOC-GRD	6.88%	11.93%	↓ 0%	5.05%	↓ 0%	88.07%	↑ 0%
GLO-HIL	6.88%	11.63%	↓ 22.10%	4.75%	↓ 5.94%	88.37%	↑ 0.34%
GLO-DET	6.88%	10.34%	↓ 30.74%	3.46%	↓ 31.49%	89.66%	↑ 1.81%
GLO-OUR	6.88%	8.91%	↓ 49.03%	2.03%	↓ 59.80%	91.09%	↑ 3.43%

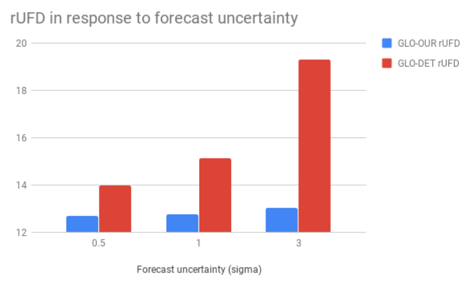

Figure 2: rUFD in response to forecast uncertainty

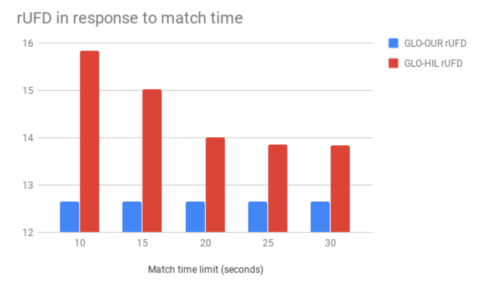

Figure 3: rUFD in response to matching time limit

uncertainty resiliency keeps its solution from significant degradation, like what GLO-DET faces. GORM's fundamental advantage of being resilient to uncertainty usually expands as the uncertainty in the demand/supply forecasting increases.

Figure 3 shows our algorithm GLO-OUR and GLO-HIL's reaction to matching time limit change in city-medium. GLO-HIL's temporally local view of the market cannot improve the solution all the way to GLO-OUR's level, despite the extended optimization runtime. GORM's advantage over baseline algorithms usually expands when the time limit on realtime decision is lower (or equivalently when the trip expiration time is lower).

6.3 Algorithm Runtime

Algorithms in the experiment ran on a single 16-core Xeon machine. The runtime evaluation of all the algorithms is done in two runtime components: 1) offline market forecasting and optimization happening once every 5 minutes, if applicable to a specific algorithm. 2) online supply-demand matching at trip request level happening at realtime upon individual trip requests.

In all experiments, our algorithm - GLO-OUR - always finishes the offline forecasting and optimization under 2 minutes, while takes less than a second to perform the realtime matching. Our algorithm performs both parts with less than 2 GB of memory.

More details of the runtime of all 4 algorithms can be found at Table 4. All the runtime measurements are in seconds.

Note that the runtime of the offline component cannot exceed 5 minutes; the runtime of the online component cannot exceed the matching time limit, which is set to 10 seconds in the solution quality experiments.

Notice that GLO-HIL's online component runtime is always 10 seconds, because it needs more than 10 seconds to complete a fully-optimized solution, but has to be cut off at the realtime matching time limit - 10 seconds.

Table 4: Algorithm runtime in seconds

	small city		medium city		big city	
	offline	online	offline	online	offline	online
LOC-GRD	0	<1	0	<1	0	<1
GLO-HIL	0	10	0	10	0	10
GLO-DET	52	<1	63	<1	79	<1
GLO-OUR	61	<1	76	<1	86	<1

7 FUTURE WORK

There are a range of algorithm/model improvements that can be considered in the GORM framework. Some examples include:

(1) Time-series optimization with minute-by-minute forecast.
(2) Global optimization on sliding time windows.

8 CONCLUSION

This paper presents the Globally-Optimized Realtime Matching framework (GORM) for supply-demand matching in ridesharing market. By applying GORM to a real world ridesharing matching problem, we showed significant improvements on market efficiencies - minimizing unfulfilled trip requests. Compared to several representative existing methods in the field, our method has the advantages of realtime matching runtime performance, globally-optimized supply-demand matching, and uncertainty-resilient robust optimization for forward-looking market optimization.

REFERENCES

[1] G. R. Andrey Glaschenko, Anton Ivaschenko and P. Skobelev. Multi-agent real time scheduling system for taxi companies. In *8th International Conference on Autonomous Agents and Multiagent Systems*, pages 29–36, 2009.

[2] J. Birge. State-of-the-art-survey'stochastic programming: Computation and applications. In *Informs Journal on Computing*, pages 111–133, 1997.

[3] I. Brodsky. H3: Uber hexagonal hierarchical spatial index. In *https://eng.uber.com/h3/*, 2018.

[4] L. C. Chung. Gps taxi dispatch system based on a* shortest path algorithm. In *Ph.D. Dissertation Submitted to the Department of Transportation and Logistics at Malausia University of Science and Technology (MUST)*, 2005.

[5] R. C. Der-Horng Lee, Hao Wang and S. Teo. Taxi dispatch system based on current demands and real-time traffic conditions. In *Transportation Research Record: Journal of the Transportation Research Board 1882*, pages 193–200, 2004.

[6] FiveThirtyEight. Uber trip data from a freedom of information request to nyc's taxi & limousine commission. In *https://github.com/fivethirtyeight/uber-tlc-foil-response*, 2015.

[7] Google. Google maps. In *https://www.google.com/maps*, 2019.

[8] H. K. Y. Y. P. K. J. Lee, G.L. Park and S. Kim. A telematics service system based on the linux cluster. In *International Conference on Computational Science - Springer*, pages 660–667, 2007.

[9] N. H. D. Kiam Tian Seow and D.-H. Lee. A collaborative multiagent taxi-dispatch system. In *IEEE Transactions on Automation Science and Engineering*, pages 607–616, 2010.

[10] K. T. Seow, N. H. Dang, and D. Lee. A collaborative multiagent taxi-dispatch system. *IEEE Transactions on Automation Science and Engineering*, 7(3):607–616, July 2010.

[11] Uber. Hexagonal hierarchical geospatial indexing system. In *https://uber.github.io/h3/*, 2018.

[12] L. Zhang and et al. A taxi order dispatch model based on combinatorial optimization. In *International Conference on Knowledge Discovery and Data Mining*, pages 2151–2159, 2017.