

Methodological issues in:

“Peeking Beneath the Hood of Uber”

Stefan Savage
University of California, San Diego

How Uber works

Phone ride-sharing app

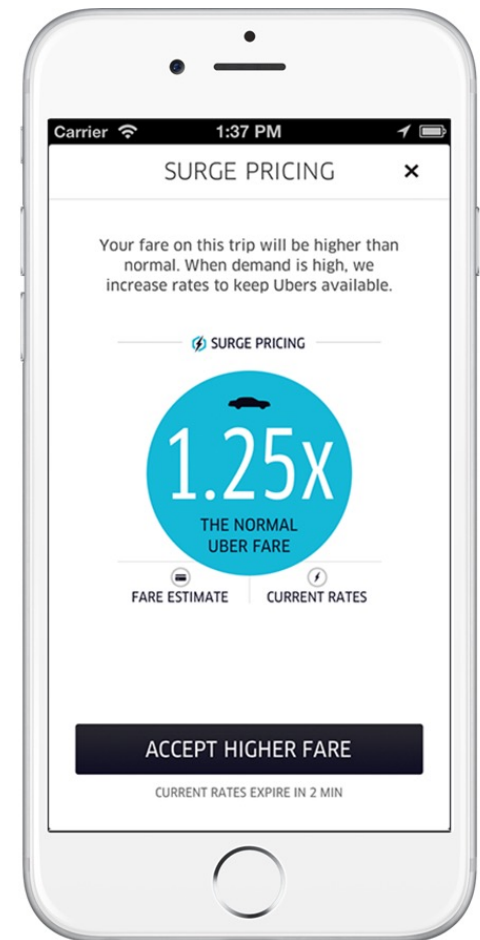
- Runs on customer phones and driver phones (different version)

You enter in destination and pickup location

It provides complete price for ride up front
(function of distance and class and
gives you *estimated wait time* (EWT))

Sometimes (presumably when demand is high)
there is “Surge Pricing”

- Surge pricing is the motivation for the paper



Taking a step back

Lots of marketplaces/services have dynamic pricing models



Frequently this is set by seller

- Bidding on ebay
- Seller-based dynamic pricing on Amazon

Other times it is set by the company

- Automated price matching
- Personalized pricing
- Per-category price adjustment

Concerns about fairness when decision is “black box”

An Empirical Analysis of Algorithmic Pricing on Amazon Marketplace

Le Chen
Northeastern University
Boston, MA USA
leichen@ccs.nyu.edu

Alan Mislove
Northeastern University
Boston, MA USA
amisllove@ccs.nyu.edu

Christo Wilson
Northeastern University
Boston, MA USA
cw@ccs.nyu.edu

ABSTRACT

The rise of e-commerce has unlocked practical applications for *algorithmic pricing* (also called dynamic pricing algorithms), where sellers set prices using computer algorithms. Travel websites and large, well-known e-retailers have already adopted algorithmic pricing strategies, but the tools and techniques are now available to small-scale sellers as well.

While algorithmic pricing can make merchants more competitive, it also creates new challenges. Examples have emerged of cases where competing pieces of algorithmic pricing software interact in unexpected ways and produced unpredictable prices [37], as well as cases where algorithms were intentionally designed to implement price fixing [5]. Unfortunately, the public currently lack comprehensive knowledge about the prevalence and behavior of algorithmic pricing algorithms in the wild.

In this study, we develop a methodology for detecting algorithmic pricing, and use it empirically to analyze their prevalence and behavior on Amazon Marketplace. We gather four months of data covering all merchants selling any of 1,641 best-seller products. Using this dataset, we are able to uncover the algorithmic pricing strategies adopted by over 500 sellers. We explore the characteristics of these sellers and characterize the impact of these strategies on the dynamics of the marketplace.

1. INTRODUCTION

For the last several years, growth in e-commerce has massively outpaced growth among traditional retailers. For example, while retail sales shrank 1.3% in the first quarter 2015 in the US, e-commerce grew 3.7% [21]. Although e-commerce only accounts for around 7.3% of the overall \$22 trillion in global retail spending projected for 2015, this percentage is projected to rise to 12.4% by 2019 [27]. Furthermore, these overall figures mask the disproportionate gains of e-commerce in specific sectors, such as apparel, media, and office supplies.

The rise of e-commerce has unlocked practical applications for *algorithmic pricing* (sometimes referred to as dynamic pricing algorithms or Revenue/Yield Management). Algorithmic pricing strategies are challenging to implement in traditional retail set-

tings due to lack of data (e.g., competitors' prices) and physical constraints (e.g., manually relabeling prices on products). In contrast, e-commerce is unconstrained by physical limitations, and collecting real-time data on customers and competitors is straightforward. Travel websites are known to use personalized pricing [25], while some e-retailers are known to automatically match competitors prices [40, 17].

While algorithmic pricing can make merchants more competitive and potentially increase revenue, it also creates new challenges. First, poorly implemented pricing algorithms can interact in unexpected ways and even produce unexpected results, especially in complex environments populated by other algorithms. For example, two competing dynamic pricing algorithms inadvertently raised the price of a used textbook to \$23M on Amazon [17]; reporters have noted that similar algorithmic pricing also exists in day-to-day commodities [9]. Second, dynamic pricing algorithms can implement collusive strategies that harm consumers. For example, the US Justice Department successfully prosecuted several individuals who implemented a price fixing scheme on Amazon using algorithms [5]. Unfortunately, regulators and the public currently lack comprehensive knowledge about the prevalence and behavior of algorithmic pricing algorithms in the wild.

In this study, our goal is to empirically analyze deployed algorithmic pricing strategies on Amazon Marketplace. Specifically, we want to understand what algorithmic pricing strategies are used by participants in the market, how prevalent these strategies are, and ultimately how they impact customer experience. We chose to focus on Amazon for three reasons: first, Amazon is the largest e-commerce destination in the US and Europe [16]. Second, Amazon is a true marketplace populated by third-party sellers, as well as Amazon itself. Third, Amazon's platform provides APIs that are specifically designed to facilitate algorithmic pricing [1].

To implement our study, we develop a novel methodology to collect data and measure sellers that are likely using algorithmic pricing. We collect four months of data from 1,641 of the most popular products on Amazon. We gather information about the top-20 sellers of each product every 25 minutes, including the sellers' prices, ratings, and other attributes. We use this data to analyze changes in price over time, as well as compare the attributes of sellers. We focus on top-selling products because they tend to have multiple sellers, and thus are likely to exhibit more competitive dynamics.

We begin by analyzing the algorithm underlying Amazon's *Buy Box*. This algorithm determines, for a given product being sold by many sellers, which of the sellers will be featured in the *Buy Box* on the product's landing page (i.e., which seller is the "default" seller). As shown in Figure 1, customers use the *Buy Box* to add products to their cart; sellers not selected for the *Buy Box* are relegated to a separate webpage. The precise features and weights used by the

Copyright is held by the International World Wide Web Conference Committee (IW3C2). IW3C2 reserves the right to provide a hyperlink to the author's site if the Material is used in electronic media.
WWW 2016, April 11–15, 2016, Montreal, Quebec, Canada.
ACM 978-1-4503-1143-1/16/0009.
http://dx.doi.org/10.1145/282747.283009.

Back to Uber

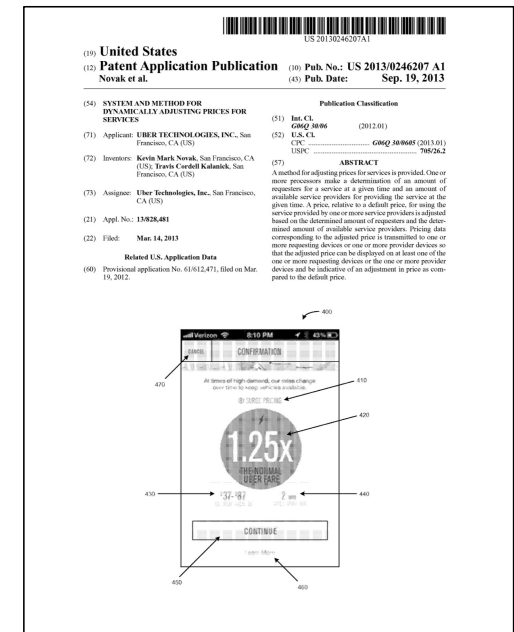
Uber claims

- Surge pricing added temporarily when demand is very high relative to supply
- Goal is to ensure that riders who most need rides can get them
- Does not say how surge multiplier is calculated

But there is a patent application that talks about different ways they *might* do it

Quick aside: patents

- Can be important resource when reverse-engineering a system
 - Some complications about filing dates (provisional, continuations, etc)
 - Rarely describes precisely what was done, but can give clues
- Fine to use for research; but note that if patent is granted and you **USE** material (e.g., in product or open source tool) that you can be liable for infringing (3x damages if you read the patent)



Questions they pose in the paper

Characterization questions

- How much and how often does it surge on Uber?
- How often do surges last?
- How do surge prices vary by location?

Algorithmic reverse-engineering questions

- What features does Uber use to calculate the surge multiplier?

Economic questions

- What is the impact of surge pricing on supply and demand?

Utilitarian questions

- Can we predict surges?
- Can we avoid surges?

What kind of data do you need to answer these questions?

Pricing: Surge multiplier at location x at time t

Supply: Number of Uber rides available near location x at time t

Demand: Number of requests for Uber rides at time t at various locations

Any other variables used for surge pricing calculation
(and ideally, secret source at Uber to tell you algorithm for calculating them)

How might you get pricing data?

Pricing: Surge multiplier at location x at time t

Options? (don't limit yourself to paper)

- Uber API – for 3rd party apps, gives per-location estimated pricing, surge value and EWT
Why can't you do the whole study with this?
- Survey Uber users about prices? Issues?
- App that looks at past Uber "Activity" or e-mail receipts. Issues?
- App that "drives" the Uber app and constantly asks for prices from current location to various destinations. Issues?

What they do:

- Reverse-engineer and emulate Uber client app's communication protocol
- Take measurements "from" the precise locations they are interested in

Aside: is it ok to do this legally?

Generally, yes – but there is some trickiness

Reverse engineering app

- Can you figure it out just from traffic? (probably not, probably under TLS)
- Did you affirmatively agree (clickwrap counts, but not just EULA) that you wouldn't reverse engineer it? (can you get a copy without agreeing?)

Probing Uber

- Does this count as unauthorized access?
 - Do you need to use some kind of unique password to access Uber? No, you're probably fine
 - They did; needed a credit card for each account
 - Can you convince the actual app to issues these requests once its authorized?
(prob fine also, although this isn't what they did)
 - Are you obtaining the same data you could have obtained via the process above?
(probably ok, absent a C&D letter – this is what they did)
- Are you harming Uber? (e.g., poisoning data in a way that they lose money)
 - Lots of civil claims (e.g., tortious interference) are relatively toothless without damages
 - Further aside: they make claim that they don't incur surges, how do they argue that?

How the Uber app works

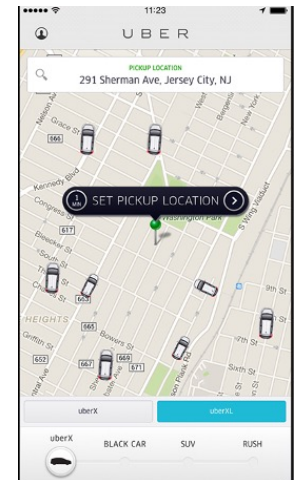
Every 5 seconds it sends a ping with user's **location**

Server responds with list of

- Car types, **nearest eight cars**, EWT and surge multiplier (all before trip booked)
- Look you get supply too!

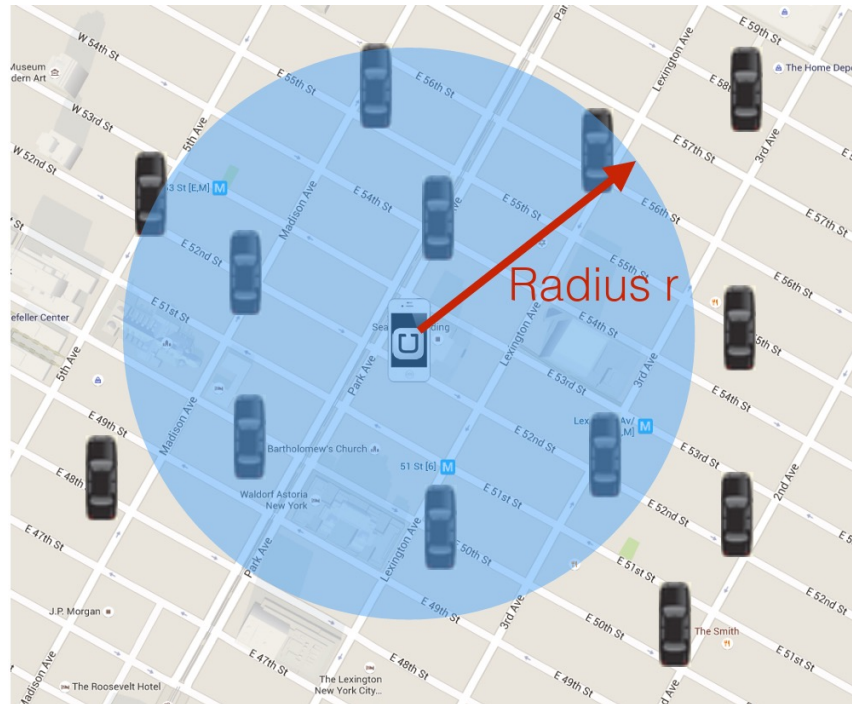
Thus, by varying report location can “see” Uber supply & pricing at different locations

Any potential data quality concerns with such data?



One issue: limited visibility into supply

You only get to see eight cars... you don't know how many cars there are



How to address limited visibility?

Use more than one “fake customer”!

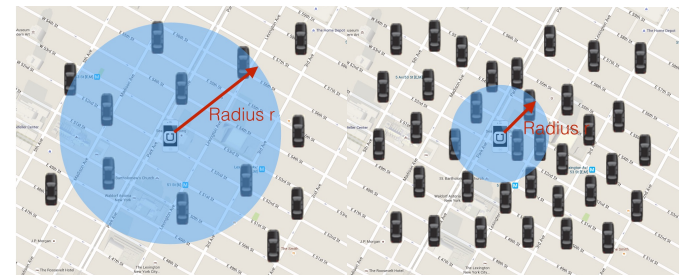
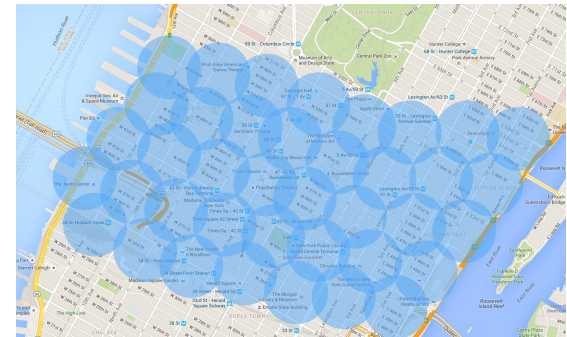
- Distribute to make a “grid” of sampling points
Then you get to see **all nearby** cars

What are assumptions here?

- That you are seeing the 8 closest cars
(i.e., not some other kind of filter... n of m)
- That everyone gets to see the same universe of cars
How would you test these two assumptions?
- That you know density

You don't get to pick r , r is a byproduct of car density near location t

You need to pick client locations to maximize the chance that you cover everything (i.e., r is aspirational)



Pilot study to measure density

Select location x , place four clients

- Every five seconds move each client in different cardinal direction (N,S,E,W)
- Stop when intersection of cars seen across all cars hits zero

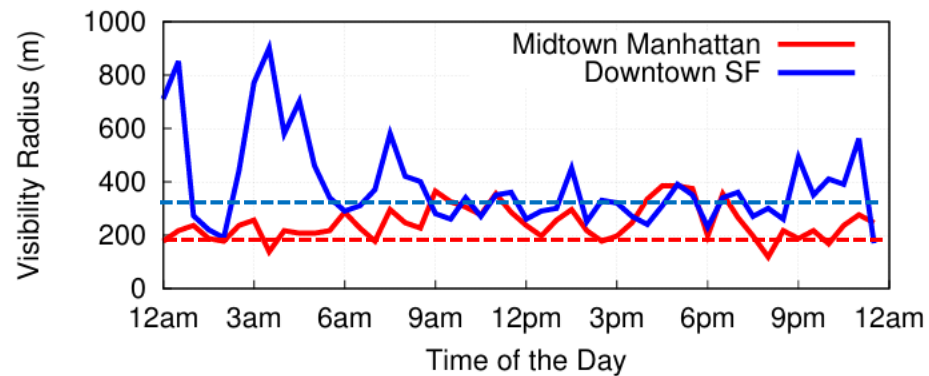


Figure 2: Visibility radius of clients in two cities.

Question: does it matter what location is chosen?

Simulation as validation

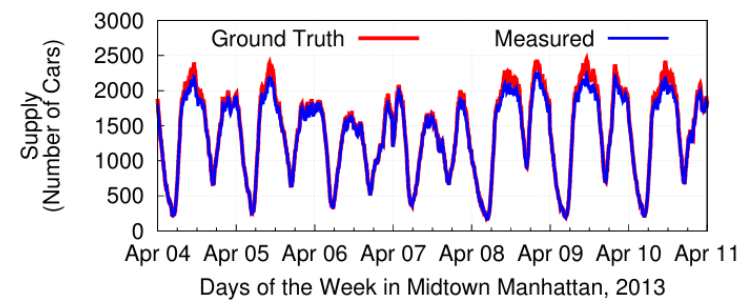
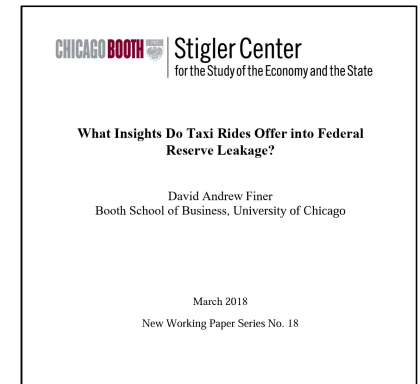
Use NYC TLC data:

- <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
Pickup/drop-off date, time, location (also, distance, fare, etc.)
- Really useful data for lots of research!

Using TLC as ground truth

- Pick-up and drop-off location/time
- Simulate density measurements using same algorithm
(but proportionately more clients)
- Now place grid like you would and
see what fraction of “supply” you capture

What are we assuming?



Ultimate placement of “monitors”



(a) Uber in downtown SF.



(b) Uber in midtown Manhattan.

$R=200\text{m}$ for NYC, 350m for SFO... is that enough to constrain the points?

Aside: about the choice to pick NYC and SFO

Advantages of these choices?

Disadvantages?

Data cleaning issues

50% of UberX cars only appear in dataset once

- Why? How do they show it?
- What else might you do?

90% of cheaper Ubers (X, XL, Pool, Family) have lifetime of < 10 minutes; only true for 65% of higher-priced Ubers (Black, SUV)

- Theories?

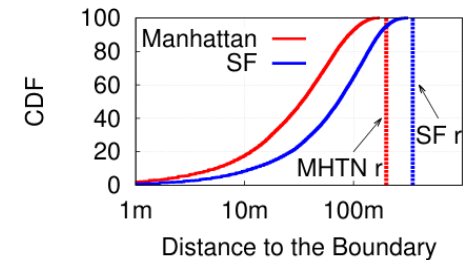


Figure 6: Distance of short-lived insiders to the boundary.

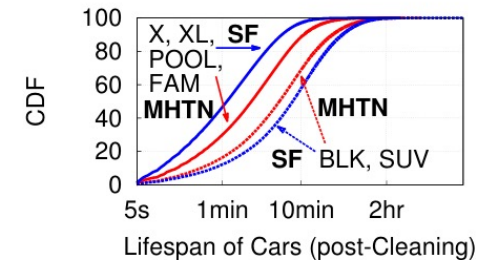


Figure 7: Lifespan of different types of Uber cars after data cleaning.

Last data inference issue: demand

Demand is how many people want an Uber in a location x at time t

Uber knows that, we don't know that

What proxy can we use?

- Ubers that "disappear" are deemed to have picked up rides at the last location they were seen
- Issues/confounders?
 - People could give up (take alternative form of transportation, public transit, lyft, etc)
 - Paper defines term "fulfilled demand" to differentiate this issue
 - Drivers could have gone off duty
 - Drivers could have left measurement area
- How might one test/validate that these confounders are not issues?

Characterization graphs

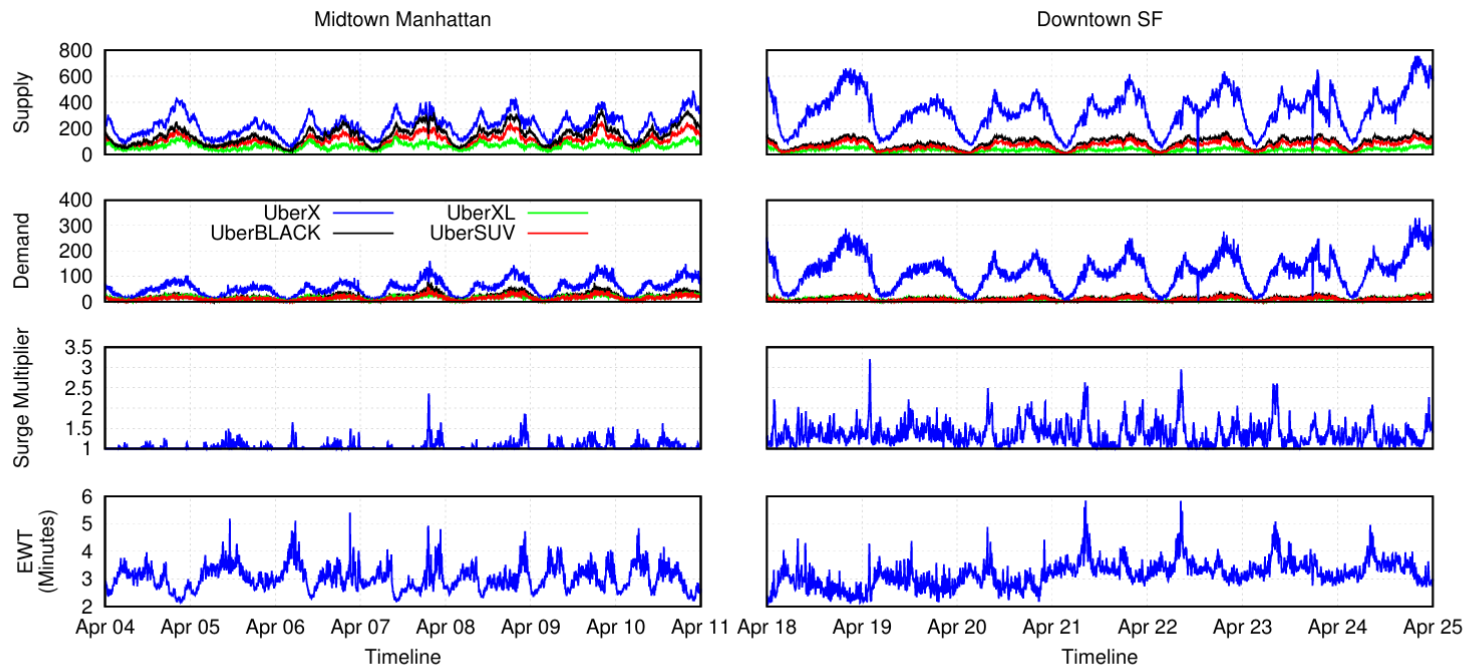
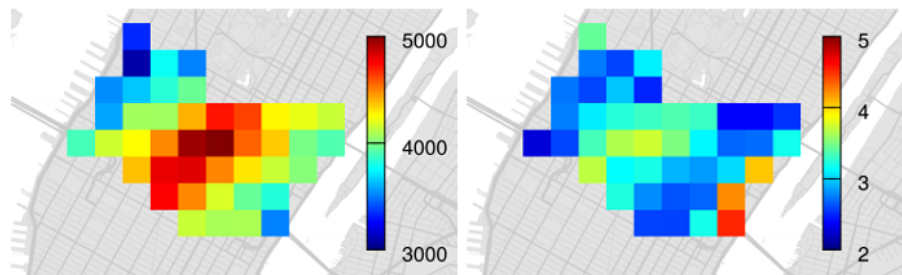


Figure 8: Supply, demand, surge multiplier, and EWT over time for midtown Manhattan and downtown SF. Surge multiplier and estimated wait time (EWT) are only shown for UberX. Diurnal patterns are observed in supply and demand, but the characteristics of the surge multiplier show less predictability.

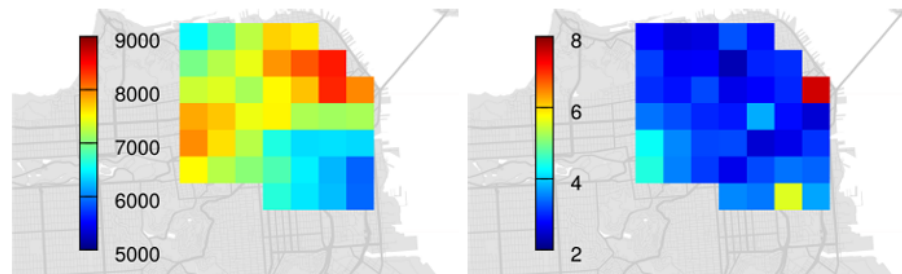
Geospatial dynamics



(a) Avg. Number of Cars

(b) Avg. EWT (Minutes)

Figure 9: Heatmaps for UberX cars in Manhattan.

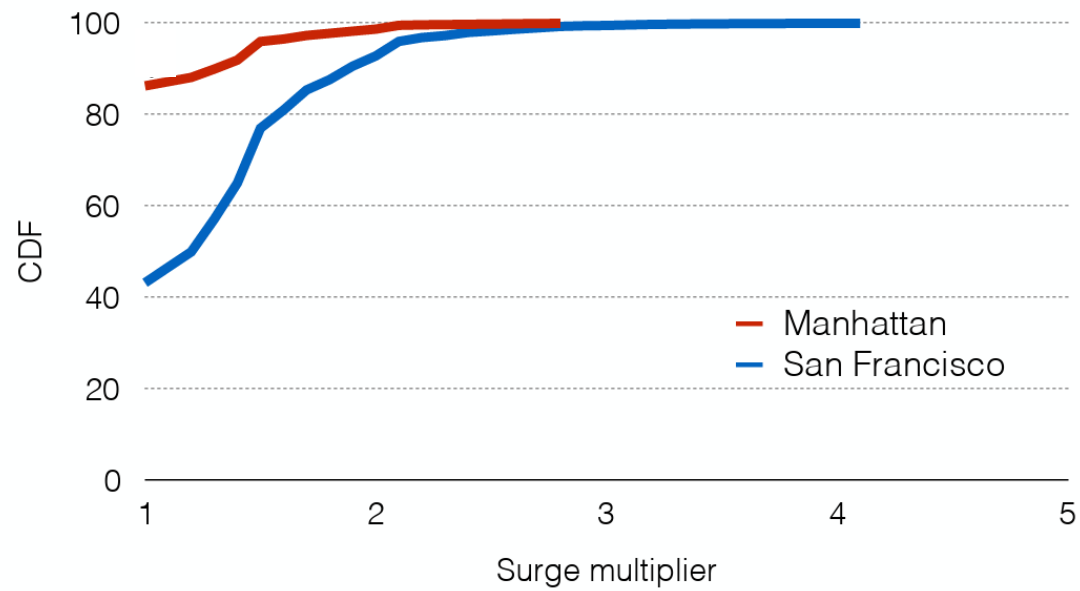


(a) Avg. Number of Cars

(b) Avg. EWT (Minutes)

Figure 10: Heatmaps for UberX cars in SF.

Surges: how often and how much



Life gets complicated for our friends at Northeastern...

Service behavior changes

- Feb -> April
- Updates were at 5min intervals, now is spread out

Jitter

- Sometimes a client gets old surge data (ug)

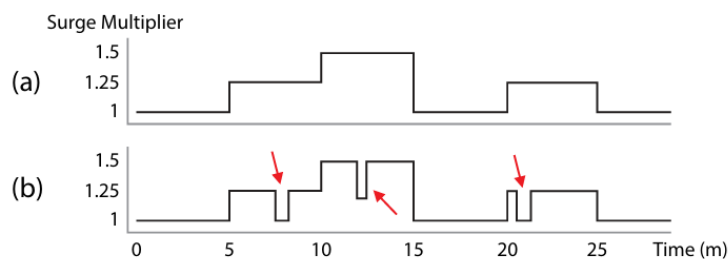


Figure 14: Examples of surge over time as seen from (a) the API and (b) the Client app. Although surge is recalculated every 5 minutes in both cases, clients also observe surge jitters for 20-30 seconds (red arrows).

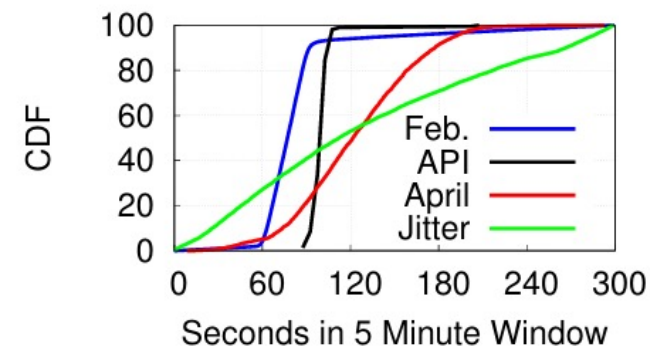
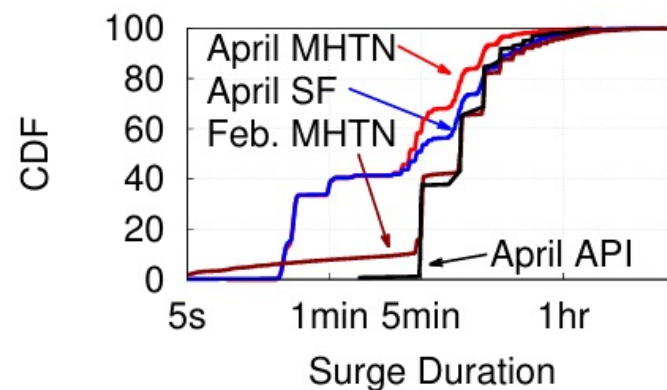


Figure 15: Moment when surge changes during each interval.

Surge areas

How do they determine the “surge areas” in each city?

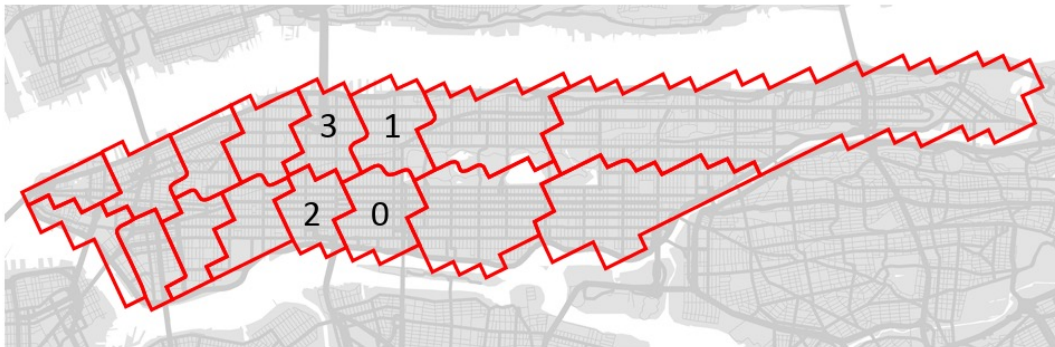


Figure 18: Surge areas in in Manhattan.

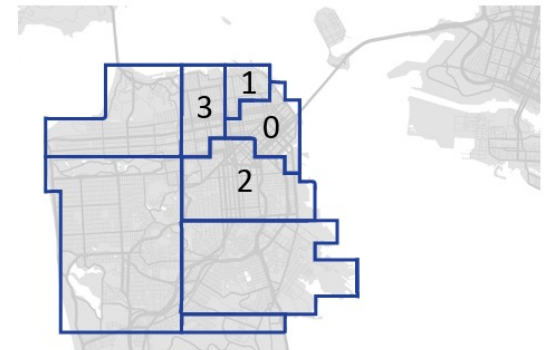


Figure 19: Surge areas in SF.

How are surge prices calculated?

What do you need to figure this out?

- How often surge pricing is updated? And relative to what time period
- Supply, demand and EWT during that time
- The algorithm and any other variables... (dunno)

Their solution: try stuff and see if it works

- Two variables in time: Surge price and test algorithm
- Calculate correlation; slide test algorithm forward/back in time; repeat

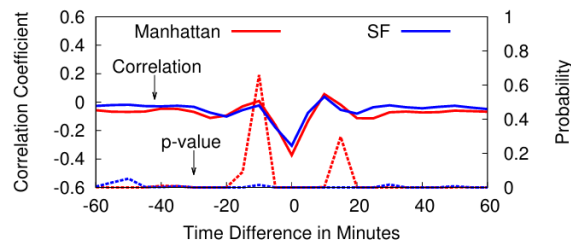


Figure 20: (Supply - Demand) vs. Surge for UberX.

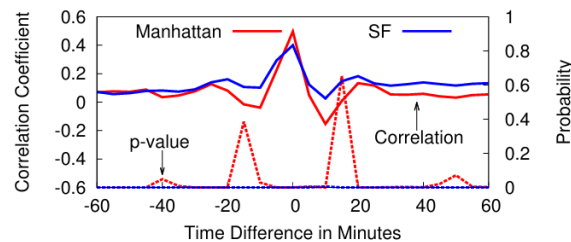


Figure 21: EWT vs. Surge for UberX.

Remaining stuff

Can you forecast surge prices

- Tried a linear regression model – doesn't worry very well

Surge impact on supply/demand

- Look at change when neighboring area is surging
 - Odd results; some new cars come, more cars leave area (unexpected), demand reduces

How often can you walk a bit and get better pricing

- 10-20% of time in NYC; rarely in SFO

Questions?

Next Thursday

Guest Lecture from Alan Mislove

Please read the paper

- Muhammad Ali, Piotr Sapiezynski, Miranda Bogen, Aleksandra Korolova, Alan Mislove, Aaron Rieke. [Discrimination through Optimization: How Facebook's Ad Delivery Can Lead to Biased Outcomes](#), In *Proceedings of the ACM on Human-Computer Interaction*, November 2019, pp. 1–30.

Please think of questions to ask Alan *in advance!*