The Last Few Kilometers: Didi VS. Ofo

Qian Xie
Institute for Interdisciplinary Information Sciences, Tsinghua University

September 21, 2017

Abstract

In this project, I study how the emergence of Ofo impact on the number of people choosing walking and Didi before. I define a cost function to reflect how people make choices with the trade-off between time, price and effort. The effort is assumed as proportional to the distance for walking and Ofo. I study the variation when effort is homogenous and heterogenous respectively using monte carlo simulation.

1 Introduction

With the rapid urbanization and population expansion in Chinese cities, more and more people rely heavily on transportation. Although the rising standard of living makes owning a private car possible for many families, parking and traffic jams are still annoying problems. So public transport means, especially subway, are preferred by citizens and travellers. However, subway service can't take you to the destination. After getting off the subway at the closest station to the destination, people still need to transfer to another mean for the last few kilometers. Alternatives are walking, buses and taxies. Padicabs can also be found in some relative remote subway stations.

Things have become different since "sharing-economy" emerges. In recent years, technology startup firms have created two new kinds of services, ridesharing and bicycle-sharing. Uber and Didi are two representative ridesharing platforms while Ofo and Mobike are the two most popular bicycle-sharing softwares. Unlike Uber, Didi also serves as a taxi-calling application. In these days, people would rather use Didi app instead of waiting for a taxi. Riding a shared-bicycle has also become a new alternative.

A recent study done by Amap showed that subway stations are the places where Amap application is most actively used, indicating that people refer to this kind of map application for information like direction, distance, time and price, in order to decide how to get to their destinations.

One year ago, in 2016, Ofo received strategic investment from Didi, it seems like a cooperation between the two unicorns. However, whether Ofo is a threat to Didi is still an interesting question. The emergence of bicycle-sharing gives people more choices and may result in Didi's loss of customers. In this paper, I'm focusing on the trade-off between Ofo and Didi when people depart from the same subway station but the distances to their destinations are different.

2 Model

2.1 Model Setting

Assume a unit mass of people file out of the same subway station at a particular time. They have different destinations and the distances are drawn from a truncated normal distribution with a minimum and maximum level. Each person can choose from three means of transportation, walking, Ofo and Didi Express. Unlike the taxi-calling service 'Didi Taxi', 'Didi Express' will calculate a suggested reasonable price based on its data and real-time traffic for drivers and customers before their trip and that's why I choose 'Didi Express' instead of 'Didi Taxi'. I don't consider bus here, since bus share similarities with subway in their indirectness to the destination. The only difference is the average distance to the destinations is smaller.

Each person knows the information about distance, time and price of each mean prior to their decisions. I make this assumption based on the observation that map applications such as Google Maps, Baidu Maps and Amap offer estimated distance and time information for walking, cycling and driving. Besides, Didi Express will tell you price once you search for the exact destination and the price rule of Ofo is clear.

Next, I define a cost function related to time, price and effort

$$\text{cost} = \alpha * \text{time} + \beta * \text{price} + \text{effort}$$

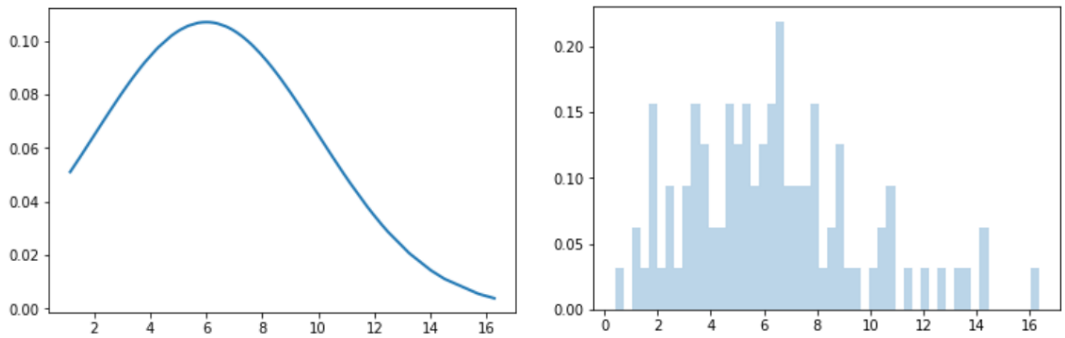
which reflects people's criterion when comparing the three choices. α and β in the formula are elasticity, namely the measurement of how responsive an economic independent variable (time, price, effort) is to a change in the dependent variable (cost). Actually there should be an elasticity before effort, but I can normalize it to 1. Note that effort is hard to measure, which specifies exhaustion (especially

for the old and disabled), discomfort (especially at night or in rainy days) and troubles finding the direction here. To make it computable, I consider it proportional to the distance for walking and Ofo. Now the effort term in the formula becomes

$$\text{effort} = \begin{cases} 0, & \text{Didi Express} \\ \gamma * \text{distance}, & \text{Otherwise} \end{cases}$$

2.2 Parameter Selection

The distance distribution is $d \sim N(\mu, \sigma^2)$ conditional on $d \in (a, b)$. In the subsequent monte carlo simulation, I set the parameters of this truncated normal distribution as $\mu = 6.0,\ \sigma = 4.0,\ a = 0,\ b = 18.0$, according to the average and maximum cover distances of Tiantongyuan North, a subway station in the suburb of Beijing.



The left-hand graph is the probability density function and the right-hand graph is the histogram. The calculation method of time is simply

$$\text{time} = \text{distance}/\text{speed}.$$

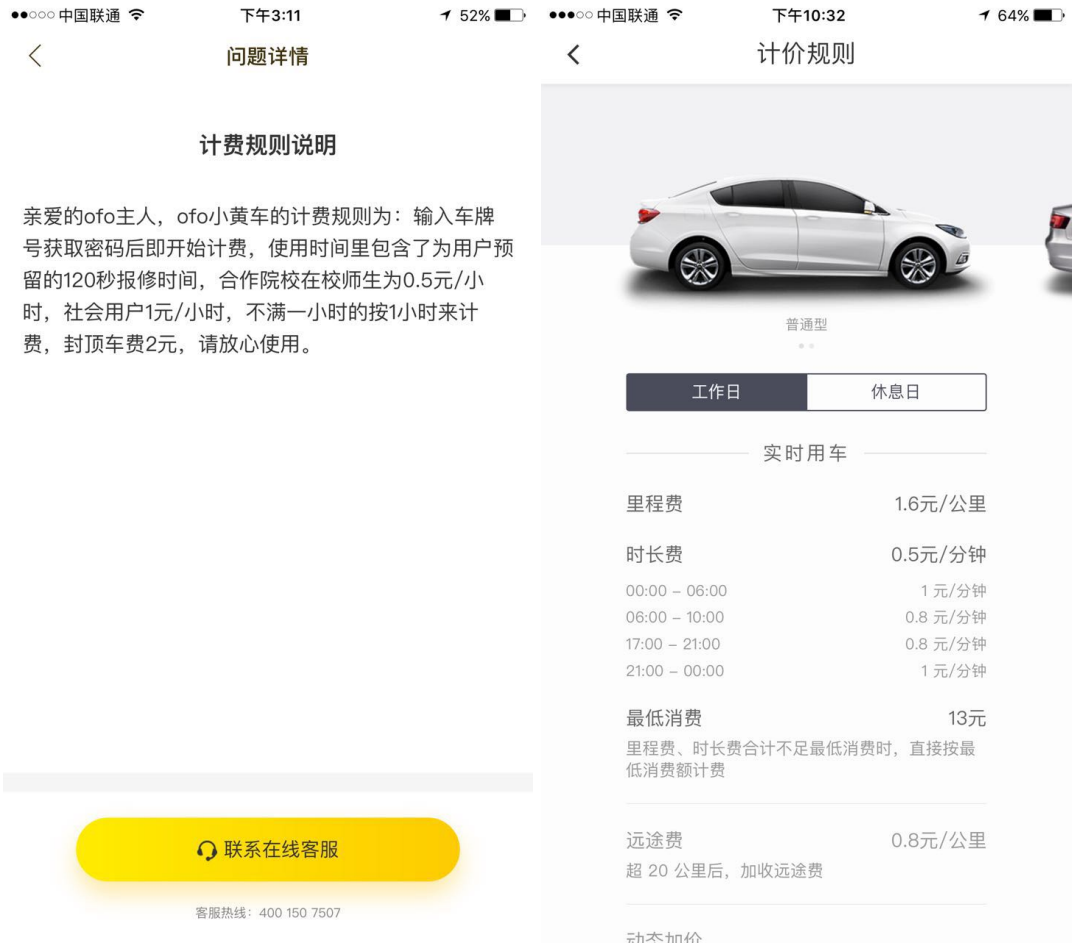
Here speed is assumed homogeneous for the same mean of transportation

$$\text{speed} = \begin{cases} 4.5 \text{ km/h}, & \text{walking} \\ 10.0 \text{ km/h}, & \text{Ofo} \\ 20.0 \text{ km/h}, & \text{Didi} \end{cases}$$

where the specific figures are obtained from Baidu Maps.
As for the price, I refer to the price rules of Didi Express and Ofo,

$$\text{price} = \begin{cases} 0, & \text{walking} \\ \max\{\lceil \text{time} \rceil, 2\}, & \text{Ofo} \\ \min\{1.6 * \text{distance} + 0.5 * \text{time}, 13\}, & \text{Didi Express} \end{cases}$$

The price rule of Ofo follows '1 yuan per hour, at most 2 yuan'. Didi Express considers both distance and time with a minimum charge of 13 yuan. The charge of per unit time 0.5 would be adjusted to 0.6 on weekends, 0.8 in the rush hour and 1.0 during the night. There is also a car-pooling choice that can reduce price but may take a longer time.
Finally in terms of the elasticity, set β to 1.0. Considering the time spent on the way can be viewed as opportunity cost, let $\alpha = 21.0$ according to the minimum hourly wage in Beijing. In the following simulation, I will adjust the value of γ to see how the number of people choosing each mean varied accordingly.



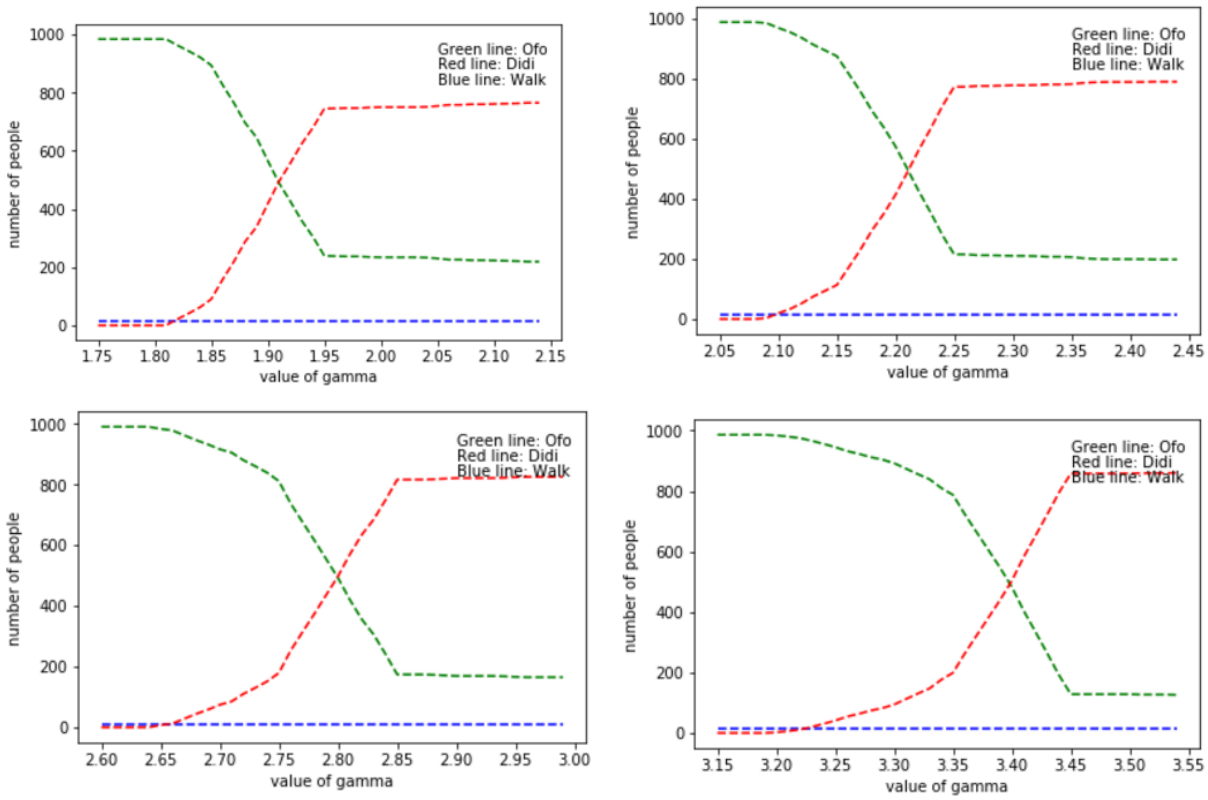
3 Monte Carlo Simulation

3.1 Simulation Procedure

1. Generate 1000 random distances from truncated normal distribution for 1000 people.
2. Set up all parameters.
3. For each person, compare his costs on three means of transportation, choose the mean with minimum cost for him.
4. Count up the number of people choosing walking, Ofo and Didi Express respectively.
5. Gradually adjust the value of γ to see the variation of numbers.

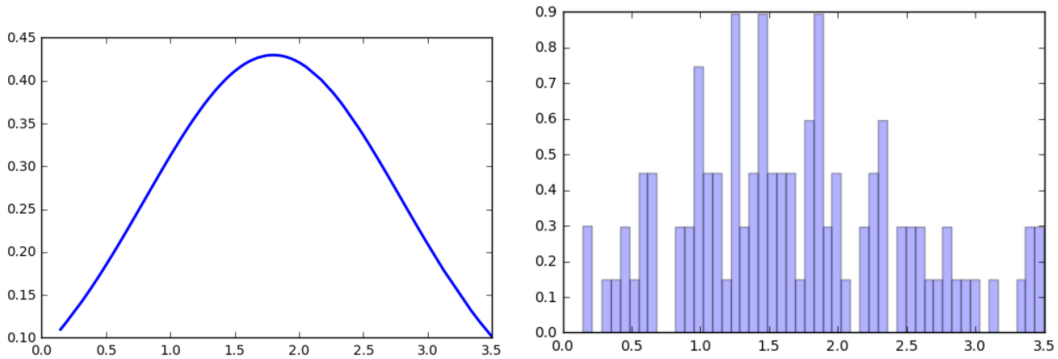
3.2 Simulation Result (γ is homogeneous)

The step of γ I set is 0.01, then I found there is a sharp variation happens at the interval $\gamma \in [1.80, 1.95]$. Before this interval, the number of people choosing Didi Express stays 0 and after this interval, the number of people choosing Didi Express slowly goes up from around 800. We can see from the trends that there is an evident tipping point at around $\gamma = 1.85$. Later I do the similar simulation for $c = 0.6$, $c = 0.8$, and $c = 1.0$ where c denotes the charge of per unit time. There are several differences: the sharp variation interval moves right and becomes wider, the variation becomes smoother, the number of people corresponding to tipping points becomes larger.



3.3 Simulation Result (γ is heterogenous)

After finding out the pattern of how γ impact on the number of people's choices with different c. I assume that people's γ may subject to a normal distribution with $\mu = 1.8$, $\sigma = 1.0$, $a = 0$, $b = 3.6$.



I count up the number of people for each choice under this distribution of γ with different c. Then I delete the choice of Ofo to reflect the situation before bicycle-sharing emerges.

From the above table, I have several findings:

1) Before Ofo emerges, about 80% of people choose Didi Express (or taxi and bus, beyond the scope of my model).

2) After Ofo emerges, no more than 40% of people choose *DidiExpress*. The number declines sharply as c becomes bigger.

3) After Ofo emerges, most of people who choose walking in the past turn to Ofo. The rest of them

	walking	Didi Express
c=0.5	118	882
c=0.6	124	876
c=0.8	161	839
c=1.0	273	727

	walking	Ofo	Didi Express
c=0.5	17	644	339
c=0.6	13	704	283
c=0.8	11	876	113
c=1.0	12	967	21

are those who have quite close destination.

3.4 Explanation

1. Actually before Ofo emerges, not so many people choose Didi Express. They would rather drive on their own, leading to the heavy traffic jams in the rush hour.
2. Ofo contributes a lot to alleviate the congestion and balances the demand of Didi Express.
3. Although the number of people choosing Didi Express declines as c becomes bigger, the supply is not so enough on weekends, in the rush hour and during the night. Besides, the total number of people filing out of subway stations should be larger in the rush hour. So the adjustion of c can be viewed as a balance of demand and supply.

4 Appendix

This appendix is the collection of all codes used in my project, they are written in Python.

4.1 Truncted Normal Distribution

```
from scipy.stats import truncnorm
import matplotlib.pyplot as plt
a, b = 0, 18
mu, sigma = 6, 4.0
X = truncnorm((a-mu)/sigma, (b-mu)/sigma, loc=mu, scale=sigma)
samples = X.rvs(100)
pdf_probs = truncnorm.pdf(samples, (a-mu)/sigma, (b-mu)/sigma, mu, sigma)
plt.plot(samples[samples.argsort()],pdf_probs[samples.argsort()],linewidth=2.0,label='PDF curve
plt.hist(samples, bins= 50,normed=True,alpha=0.3,label='histogram')
```

4.2 Simulation in 3.2

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import numpy as np
from scipy.stats import truncnorm
import math
import matplotlib.pyplot as plt

wave = 4.5 #walk speed average
```

```
#walk speed std
bave = 10.0 #bike speed average
#bike speed std
cave = 20.0 #car speed average
#car speed std
dave = 6.0 #distance average
dstd = 4.0 #distance std
dmax = 18.0 #distance maximum
dmin = 0 #distance minimum
charge = 0.5 #didi time charge
num = 1000 #number of people
start = 1.75 #starting point of plotting
loc = start+0.3 #location of text

def tnormal(mu, sigma, a, b): #truncated normal distribution
    rv = truncnorm((a-mu)/sigma, (b-mu)/sigma, loc=mu, scale=sigma)
    return rv

def utility(alpha, beta, gamma, t, p, e): #t: time, p: price, e:effort, alphabetagamma: paramet
    u = alpha*t+beta*p+gamma*e
    return u

def time(mean, distance):
    if mean == 0: #walk
        return distance / wave
    if mean == 1: #bike
        return distance / bave
    if mean == 2: #didi
        return distance / cave

def price(mean, distance, time):
    if mean == 0:
        return 0
    if mean == 1:
        return min(math.ceil(time), 2) #1 yuan per hour, top to 2 yuan, applicable for ofo
    if mean == 2:
        return max(1.6*distance+charge*60*time, 13)

def effort(mean, distance):
    if mean != 2:
        return distance
    if mean == 2:
        return 0

samples = tnormal(dave, dstd, dmin, dmax).rvs(num)

scale = 40
y0 = [0]*scale
y1 = [0]*scale
```

```
y2 = [0]*scale

x = [start+0.01*elem for elem in range(scale)]
for k in range(scale):
    for i in range(num):
        al = 21.0 #Beijing minimum hourly wage
        be = 1.0
        ga = start+0.01*k
        d = samples[i]
        #print d
        u = []
        for j in range(3):
            u.append(utility(al, be, ga, time(j, d), price(j, d, time(j, d)), effort(j, d)))
        #for j in range(3):
            #print u[j]
        m = np.argmin(u)
        #print m
        if m == 0:
            y0[k] += 1
        if m == 1:
            y1[k] += 1
        if m == 2:
            y2[k] += 1

plt.plot(x, y0, 'b--')
plt.plot(x, y1, 'g--')
plt.plot(x, y2, 'r--')
plt.xlabel('value of gamma')
plt.ylabel('number of people')

plt.text(loc, 930, r'Green line: Ofo')
plt.text(loc, 880, r'Red line: Didi')
plt.text(loc, 830, r'Blue line: Walk')

plt.show()
```

4.3 Simulation in 3.3

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import numpy as np
from scipy.stats import truncnorm
import math
import matplotlib.pyplot as plt

wave = 4.5 #walk speed average
#walk speed std
bave = 10.0 #bike speed average
```

```
#bike speed std
cave = 20.0 #car speed average
#car speed std
dave = 6.0 #distance average
dstd = 4.0 #distance std
dmax = 18.0 #distance maximum
dmin = 0 #distance minimum
charge = 0.5 #didi time charge
num = 1000 #number of people
gave = 1.8
gstd = 1.0
gmin = 0
gmax = 3.6

def tnormal(mu, sigma, a, b): #truncated normal distribution
    rv = truncnorm((a-mu)/sigma, (b-mu)/sigma, loc=mu, scale=sigma)
    return rv

def utility(alpha, beta, gamma, t, p, e): #t: time, p: price, e:effort, alphabetagamma: paramet
    u = alpha*t+beta*p+gamma*e
    return u

def time(mean, distance):
    if mean == 0: #walk
        return distance / wave
    if mean == 1: #bike
        return distance / bave
    if mean == 2: #didi
        return distance / cave

def price(mean, distance, time):
    if mean == 0:
        return 0
    if mean == 1:
        return min(math.ceil(time), 2) #1 yuan per hour, top to 2 yuan, applicable for ofo
    if mean == 2:
        return max(1.6*distance+charge*60*time, 13)

def effort(mean, distance):
    if mean != 2:
        return distance
    if mean == 2:
        return 0

dsamples = tnormal(dave, dstd, dmin, dmax).rvs(num) #generate random distances
gsamples = tnormal(gave, gstd, gmin, gmax).rvs(num) #generate random gamma

scale = 40
y0 = 0
```

```
y1 = 0
y2 = 0

for i in range(num):
    al = 21.0 #Beijing minimum hourly wage
    be = 1.0
    ga = gsamples[i]
    d = dsamples[i]
    #print d
    u = []
    for j in range(3):
        u.append(utility(al, be, ga, time(j, d), price(j, d, time(j, d)), effort(j, d)))
    #for j in range(3):
        #print u[j]
    m = np.argmin(u)
    #print m
    if m == 0:
        y0 += 1
    if m == 1:
        y1 += 1
    if m == 2:
        y2 += 1

print y0, y1, y2
```