

國立成功大學  
資訊工程研究所  
碩士論文

基於賽局理論之計程車共乘推薦機制

Game Theory Based Recommendation Mechanism for  
Taxi-Sharing

研究生：鄭婉君 Student: Wan-Chun Cheng

指導教授：鄭憲宗 Advisor: Sheng-Tzong Cheng

Institute of Computer Science and Information Engineering  
National Cheng Kung University,  
Tainan, Taiwan, R.O.C.

July, 2013

中華民國一百零二年七月

國立成功大學

碩士論文

基於賽局理論之計程車共乘推薦機制

Game Theory Based Recommendation Mechanism for  
Taxi-Sharing

研究生：鄭婉君

本論文業經審查及口試合格特此證明

論文考試委員：

陳俊良 賴威光  
黃志明 鄭憲宗

指導教授：

鄭憲宗

系(所)主管：

謝衡源代

中華民國 102 年 7 月 18 日

# Game Theory Based Recommendation Mechanism for Taxi-Sharing

By

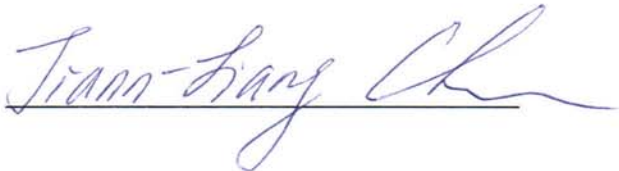
Wan-Chun Cheng

A thesis submitted to the graduate division.

In partial fulfillment of the requirements for the degree of  
Master in Computer Science and Information Engineering,  
National Cheng Kung University, Tainan, Taiwan, R.O.C.



July 18, 2013

Approved by:


Advisor:

Chairman:

## 中文摘要

# 基於賽局理論之計程車共乘推薦機制

鄭婉君\*

鄭憲宗\*\*

國立成功大學

資訊工程研究所

計程車在大都市的交通運輸中，已扮演了一個不可或缺的角色，其衍伸出的問題引起各方學者的興趣與關注。對於乘客而言，如何迅速地尋找到計程車。相對地，計程車要如何避免漫無目的地找尋乘客，在繁雜交通中已成為一大挑戰。

本研究之目的在於提出一套適用計程車與共乘乘客的推薦機制。當計程車提出詢問時，能依據不同的時間地點，推薦計程車一條路徑。對於乘客發出推薦請求時，則是推薦乘客附近區域，讓乘客與計程車能快速找到對方。除此之外，針對需要共乘服務的乘客，也能搜尋適當的乘客搭配共乘。

我們根據 10,357 台計程車在 110 天中記錄下的 GPS 軌跡資料，在不同時段下，以 R-Tree 的方式分析出熱門的行走路線與載客地點；並利用非合作式賽局理論(non-cooperative game theory)，在多輛計程車需要推薦路徑時，有彼此競爭且獲利互相影響的關係下，從中求出多輛計程車選擇推薦路線的納許均衡(Nash Equilibrium)，以獲得雙贏的結果。當計程車已有載客的情況下，且乘客有共乘需求時，則會不斷地篩選候選乘客，並從中挑選出最適合的對象搭配共乘。

在我們的推薦機制下，藉由分析計程車與乘客的歷史資訊，找出不同時段的熱門載客路徑與地點，並給予相對應的推薦，能有效地縮短計程車跟乘客的等待時間。藉由共乘機制，在乘客容許的等待時間下，讓乘客到達目的地又能減少車資；同時計程車也可以藉由共乘機制，延長載客的距離、提高計程車司機的收入。

關鍵字：計程車共乘、軌跡、推薦機制、非合作式賽局

\* 作者

\*\* 指導教授

# ABSTRACT

The taxicab becomes one of the most important public transportations in many big cities. Customers always suffer from waiting a long time for taxis. Similarly, the taxi drivers spend much time on cruising on the road for finding passengers. Therefore, we present a recommendation mechanism for both taxis and passengers. When taxis and passengers have requests for recommendation, the server provides them with paths and locations.

The first aim of our model is to respectively recommend taxis and passengers for picking up passengers quickly and finding taxis easily. The second purpose is providing taxi-sharing service for passengers who want to save the payment. In our method, we analyze the historical Global Positioning System (GPS) trajectories generated by 10,357 taxis during 110 days and present the service region with time-dependent R-Tree. We formulate the problem of choosing the paths among the taxis in the same region by using non-cooperative game theory, and find out the solution of this game which is known as Nash equilibrium. When a taxi is occupied and the on-board passengers who want taxi-sharing service, the taxi checks the proper passengers for sharing periodically.

In order to verify the proposed recommendation mechanism, the simulation of SUMO, MOVE, and TraCI are adopted to fit our model. The results show that our method can find taxis and passengers efficiently. In addition, applying our method can reduce the payment of passengers and increase the taxi revenue by taxi-sharing.

Keywords: Taxi-sharing, Trajectory, Recommendation mechanism, non-cooperative game theory

# Acknowledgement

在成大就學的這個階段，是我人生中不可多得的經驗。求學過程中的酸甜苦辣，皆化成腦海中的美好回憶。首要感謝我的父母，用他們的生命愛我、教導我。在我慌亂無助時給我依靠與指引，讓我即便遭遇挫折也不忘感恩，擁有信念與勇氣面對不同的課題。

還要感謝我的指導教授鄭老師，提供我們優良的學習環境與機會。帶領我們領略學術浩瀚，讓我們懂得在錯誤中學習，引導我們要不斷地提升精進，是我們學習的典範。

這篇論文的完成，要特別感謝博班學長 Polk 與阿國，一路走來耐心地指導我們，讓我們從懵懵懂懂，到學會完成自己的論文。感謝他們在生活上與論文的提點與幫忙，他們在我的碩士生涯中扮演著非常重要的角色。

這兩年的時間與實驗室的夥伴們有著非常多的回憶，Mike、小樞、G、Celia、夏美、阿發、錐錐、洪洋、紀伯、Boss，感謝各位包容我的不足、生活中的關心以及在工作上的協助。此外，我的摯友大豬跟小豬，謝謝你們陪著我走過無數風雨，更謝謝你們理解與支持。很榮幸能有這個緣份能與各位相識，感謝你們也祝福你們。

最後，謹以此著作獻給我最愛的家人。

# CONTENTS

中文摘要 .....	i
ABSTRACT .....	ii
Acknowledgement .....	iii
CONTENTS .....	iv
LIST OF FIGURES .....	vii
LIST OF TABLES .....	ix
<b>Chapter 1 Introduction.....</b>	<b>1</b>
1.1 Motivation .....	3
1.2 Objectives .....	4
1.3 Thesis Overview .....	4
<b>Chapter 2 Related Works.....</b>	<b>5</b>
2.1 Existing Systems .....	5
2.1.1 Taxi Recommender System .....	5
2.1.2 Taxi Ridesharing System .....	6
2.2 Related Methods .....	7
2.2.1 R-Tree .....	7
2.2.2 Game Theory .....	8
2.3 Recent Traffic Simulators .....	10
2.3.1 SUMO .....	10
2.3.2 MObility model generator for Vehicular networks (MOVE) .....	10
2.3.3 TraCI .....	11
<b>Chapter 3 System Structure.....</b>	<b>13</b>
3.1 Overview .....	13

3.2	Parameters Definitions .....	15
3.3	Cognition Phase.....	17
3.3.1	Taxi Trajectories.....	17
3.3.2	Movement Logs .....	18
3.4	Inference Phase.....	18
3.5	Recommendation Phase .....	21
3.5.1	Recommendation for One Taxi.....	21
3.5.2	Non-cooperative Game Model for Taxi Recommendation.....	23
3.5.3	Passenger Recommendation .....	27
3.5.4	Recommendation for Taxi-sharing.....	28
3.5.5	Pricing Scheme .....	33
<b>Chapter 4</b>	<b>Simulation and Result .....</b>	<b>35</b>
4.1	Simulation Settings.....	35
4.1.1	Dataset.....	35
4.2	Parameters .....	36
4.2.1	Evaluation Factor .....	36
4.2.2	Comparing the traditional algorithms with our work .....	38
4.3	Performance Evaluation .....	38
4.3.1	The observation of ‘Average Waiting Time of Taxis and Passengers’	39
4.3.2	The observation of ‘Average Increased Revenue of Taxis’ .....	42
4.3.3	The observation of ‘Average Payment Saved of Passengers’ .....	43
4.3.4	The observation of ‘Average Service Time of Passengers’ .....	44
4.3.5	The observation of ‘Required Average Number of Taxis’ .....	46
<b>Chapter 5</b>	<b>Conclusions and Future Work.....</b>	<b>47</b>
	REFERENCE .....	48





# LIST OF FIGURES

Fig 1	Example of the region of $I$ .....	8
Fig 2	Example of R-Tree .....	8
Fig 3	Example of Nash equilibrium.....	9
Fig 4	Schematic diagram .....	13
Fig 5	System architecture .....	14
Fig 6	Flow Chart.....	14
Fig 7	Example of R-Tree on road network .....	19
Fig 8	Region-partitioned map (in R-Tree $RT_i$ ).....	20
Fig 9	Taxis' Status.....	21
Fig 10	Example of taxi recommendation.....	23
Fig 11	Scenario of recommendation for many taxis.....	24
Fig 12	Example of passenger recommendation.....	28
Fig 13	Definition of a trip.....	29
Fig 14	The bounded area of candidate passengers for taxi-sharing.....	30
Fig 15	Example of pricing scheme for two passengers .....	33
Fig 16	Example of pricing scheme for three passengers .....	34
Fig 17	Service region of the experiment.....	36
Fig 18	Average waiting time of taxis.....	40
Fig 19	Average waiting time of passengers .....	41
Fig 20	Average increased revenue of taxis .....	43
Fig 21	Average increased revenue of taxis for different number of on-board passengers.....	43

Fig 22	Average payment saved of passengers .....	44
Fig 23	Average service time of passengers.....	45
Fig 24	Required Average Number of Taxis .....	46



# LIST OF TABLES

Table 1	Summary of variables.....	16
Table 2	States of a taxi .....	18
Table 3	Representation of transitions .....	22
Table 4	User settings of a passenger .....	29
Table 5	Variables for a passenger .....	30
Table 6	Variables for a taxi.....	32
Table 7	Parameters for simulation.....	36
Table 8	Overall average waiting time of taxis and passengers.....	41
Table 9	Overall average increased revenue of taxis for different number of on-board passengers.....	43
Table 10	Comparison between overall average increased revenue of taxis and payment saved of passengers .....	44
Table 11	Overall average service time of passengers.....	45

# Chapter 1 Introduction

With the rapid development of city, taxis play a vital role in modern public transportation networks. People always suffer from waiting a long time for a taxicab. In fact, taxi drivers also have similar problems for finding passengers. They have no idea where the passengers are but to cruise on the roads at random. Cruising on the roads blindly wastes the time and gas, and decreases the profit of a taxi driver. Besides, many vacant taxis cruising on the roads incur additional traffic and produce more pollution in the city. Therefore, it's an urgent problem how to improve the utilization of taxis and to save energy.

Sometimes passengers are not willing to take the taxicab because of expensive fare unless in an emergency. They would rather take other public transportation system such as bus, train and MRT. Ridesharing is a solution to reduce the energy consumption and the taxi fare of passengers. There are many kinds of collective transport such as car-pooling [9], which is a widespread ridesharing way using private vehicles in the United State and Europe. It always deals with people routine commutes and complements traditional transportation. However, it is more difficult to provide taxi-ridesharing service because the taxis and passengers are dynamic and unpredictable.

In many urban areas, taxicabs are equipped with GPS sensor for safety and can report their location with timestamp periodically. There are a lot of GPS trajectories being generated. Therefore, we can learn the taxis' behavior from these data and predict the possibility of picking-up passengers.

In this thesis, we present a recommendation mechanism for not only finding taxis and passengers but also providing taxi-sharing service. We adopt a centralized scheme for information analysis and recommendation. Using the knowledge of GPS trajectories of

taxis and passengers' mobility patterns, we can learn the places taxis always pick up passengers.

The proposed method is working in three phases: cognition, inference and recommendation. We first collect a large number of information of taxis and passengers. Second, we analyze the historical trajectories of taxis and represent the service region by time-dependent R-Tree. R-Tree [3] is a spatial index data structure containing a set of nodes (leaf and non-leaf nodes).

After the analysis and inference, the server recommends taxis and passengers for different requests. Server provides taxis the waiting places and the routes to these places. On the other hand, server can suggest passengers going to some locations that they can take the taxis easily. The route and location is recommended with a form of region constructed by R-Tree. Moreover, we consider the problem that many taxis in the same region need the route recommendation at the same time. We formulate this problem as a competition game and use a non-cooperative game model to obtain the path choices of taxis. Nash equilibrium [5] is the solution of this game to maximize the profit of all taxis in the same region.

For all of the occupied but not full taxis, they keep searching the proper passengers for taxi-sharing. Our method filters the improper passengers to make sure the increased revenue of taxis and the payment saved of passengers aboard by choosing the passengers for taxi-sharing.

The simulation results can show that the proposed mechanism has better performance than the traditional method for shorter waiting time and higher profit of taxis and passengers. Moreover, our proposed service can decrease the number of taxis required for pollution reduction in the city.

## 1.1 Motivation

With the rapid economic development of the society, the taxicab transport becomes more and more important than other kind of public transportation because of its convenience and flexibility. There are many problems arising from the current system such as long waiting time for finding passengers, traffic congestions and road safety issues. For passengers, they want to take the taxis quickly and reduce expense as much as possible.

The aim of this thesis is to make taxis and passengers find each other easily, and find potential passengers for taxi-sharing. Passengers can request server to provide taxi service; likewise, taxis send a query for paths or passengers recommendation. Server calculates the profit and cost depending on the results inferred from historical information, and chooses the paths or passengers to recommend.

Every taxi is equipped with GPS sensors, and sends the geo-positions and timestamps to server periodically. Moreover, the occupancy information is recorded at the same time according to the weight sensors or the taxi meters. Applying the spatial index R-Tree, the service region is analyzed and time-dependent R-Tree model is established, then we can figure out the popular locations for recommendation.

In this study, we purpose two-fold goal. The primary purpose is to find out the popular regions so that the server can recommend the paths to taxis and the near locations to passengers respectively. Additionally, we try to reduce the fare of passengers and increase the profit of taxis by finding potential passengers for taxi-sharing. Compared with the method without recommendation, the mechanism proposed in this thesis was more advanced, which can be proved by Simulation of Urban Mobility (SUMO) [1] and Traffic Control Interface (TraCI) [11] simulation software.

## 1.2 Objectives

The main goals of our model are as follows: (1) Decreasing the waiting time of taxis and passengers; (2) Providing the taxi-sharing service to passengers; (3) Extending the travel distance of taxis (Increasing the revenue); (4) Increasing the number of passengers with taxi-sharing under the conditions; (5) Reducing the fare of passengers.

This work mainly focuses on solving the problem of long waiting time and high fare. At different time periods, the recommendations are adjusted by time-dependent R-Tree to show the variations in different situations. To eliminate the competition for passengers between taxis, we adapt the non-cooperative game model to formulate this problem and find the Nash equilibrium for the solution of the game.

With this approach, we can make time-dependent recommendations and reach our purposes by taxi-sharing, which is introduced in following sections.

## 1.3 Thesis Overview

The rest of this thesis is organized as follows: Chapter 2 reviews the conventional taxi recommendation systems and applications. Chapter 3 describes the key components of the proposed model, including the inference of historical information and the recommendation mechanism, the strategy of preventing the competition between taxis and the method to choose passengers for taxi-sharing. In Chapter 4, our work demonstrates a simulation of our proposed model with the real world information of taxis and passengers. All of the performance analysis and experiment results are illustrated in Chapter 4 as well. Finally, Chapter 5 draws some conclusions and describes our future work.



## Chapter 2 Related Works

In this chapter, we summarize the background and existing works providing taxi service. At the beginning, the development of taxi recommender and ridesharing system are introduced in section 2.1. Other methods for dealing with related problems are mentioned in section 2.2. Finally, section 2.3 presents the recent traffic simulators briefly.

### 2.1 Existing Systems

#### 2.1.1 Taxi Recommender System

Taxi dispatch and recommender systems are getting more and more attention from researchers with the development of intelligent transportation systems (ITS). The common method for getting a taxi is hailing on the side of the street directly. It is a trivial way, but the customers wait for a taxi depending on luck. Sometimes people have to wait a vacant taxi for a long time.

The other method is ordering a taxi dispatch system for a service. For a dispatch system, the customers need to book a taxi by telephone or internet in advance, and they have to pay for booking. The taxi dispatch system always sends the nearest taxi to pick up the passenger in need of service, and does not take ridesharing into consideration.

Yuan and Zeng et al. [15] proposed a system that suggests some parking places based on a large number of GPS trajectories generated by taxi. The parking places stand for the locations where the taxi drivers always wait for passengers with their taxi parked. The taxi recommender system devises a probabilistic model to formulate the behavior of taxis, and provides the recommendation services to both taxis and passengers without considering the issue of taxi-sharing.

### 2.1.2 Taxi Ridesharing System

With the development of society, the ridesharing transport such as car-pooling [9] becomes the important mobility mode. The aim of ridesharing is to reduce the number of vehicles in the road surface and to share the cost of a trip. Taxi-sharing is the same concept with car-pooling but it relies on taxi as a transportation resource. It is more flexible to apply taxi-sharing than car-pooling. However, the cooperative and coordination between taxis is a real challenge because of the independency and self-employment of taxis.

There are already some existing systems developed for the purpose. M. d'Orey and Fernandes et al. [2] proposed a dynamic (no reservations and immediate assignments) and distributed (no centralized authority) taxi-sharing system which can coordinate between user requests and taxi services. This thesis presented the concept of distributed algorithm for taxi and customer, but did not detail these methods.

The other study about dynamic taxi ridesharing system is called *T-Share*, which is proposed by Ma and Zheng et al [7]. They propose a taxi searching algorithm using a spatial-temporal index to retrieve the candidate taxis that are likely to satisfy a user query. The scheduling algorithm which checks each candidate taxi and inserts the query's trip into the schedule of the corresponding taxi is proposed then.

Different from the study above which partitions the road map into grids, we analyze the trajectories generated by taxis using the spatial index R-Tree. For the consideration of temporal factor, we build up time-dependent R-Tree by the trajectories with timestamp and the state of taxis. Besides, our mechanism aims to reduce the payment of passengers and increase the profit of taxis instead of balancing the revenue among all taxi drivers, which is usually a purpose of dispatch system. The methods adopted in our study are discussed in next section.

## 2.2 Related Methods

### 2.2.1 R-Tree

R-Tree [3] is a height-balanced tree data structure representing the spatial data that contain multi-dimensional information such as geographical coordinates, rectangles or polygons. It is always used in navigation system or the storage of spatial objects in real world. The spatial searching of R-Tree requires visiting only a small number of nodes. For example, we can search nearest places within a specific distance of user's current location quickly.

The key idea of R-Tree is to collect nearby objects and represents them with their minimum bounding rectangle (MBR). Because R-Tree is a height-balanced tree, the leaves in the structure all appear at the same level. On the other hand, the indexing is dynamic which means that the insertion and deletion can be intermixed with searches. An entry  $E$  in non-leaf nodes is defined as

$$E = (I, \text{child-pointer})$$

, where the *child-pointer* points to the child of this node and  $I$  refers to minimum bounding rectangle (MBR). An entry in leaf nodes is defined as

$$E = (I, \text{tuple-identifier})$$

, where  $I$  refers to MBR that encompasses the spatial data pointed to by its *tuple-identifier*.

There is an example of  $I$  in two-dimension displayed in Fig 1. We assume that  $I = (I_x, I_y)$ , where  $I_x = [x_a, x_b]$ , and  $I_y = [y_a, y_b]$ . An example of R-Tree is showed in Fig 2.

Because of the feature of R-Tree such as the storage of spatial data and finding answers of queries quickly, we analyze and store the GPS trajectories generated by taxis for the recommendation. The detail is presented in Chapter 3.

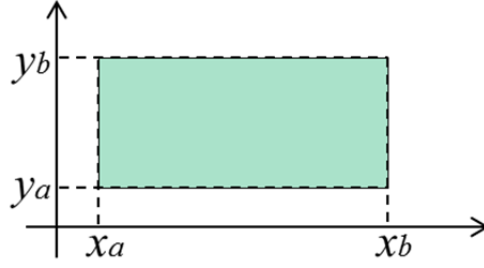


Fig 1 Example of the region of  $I$

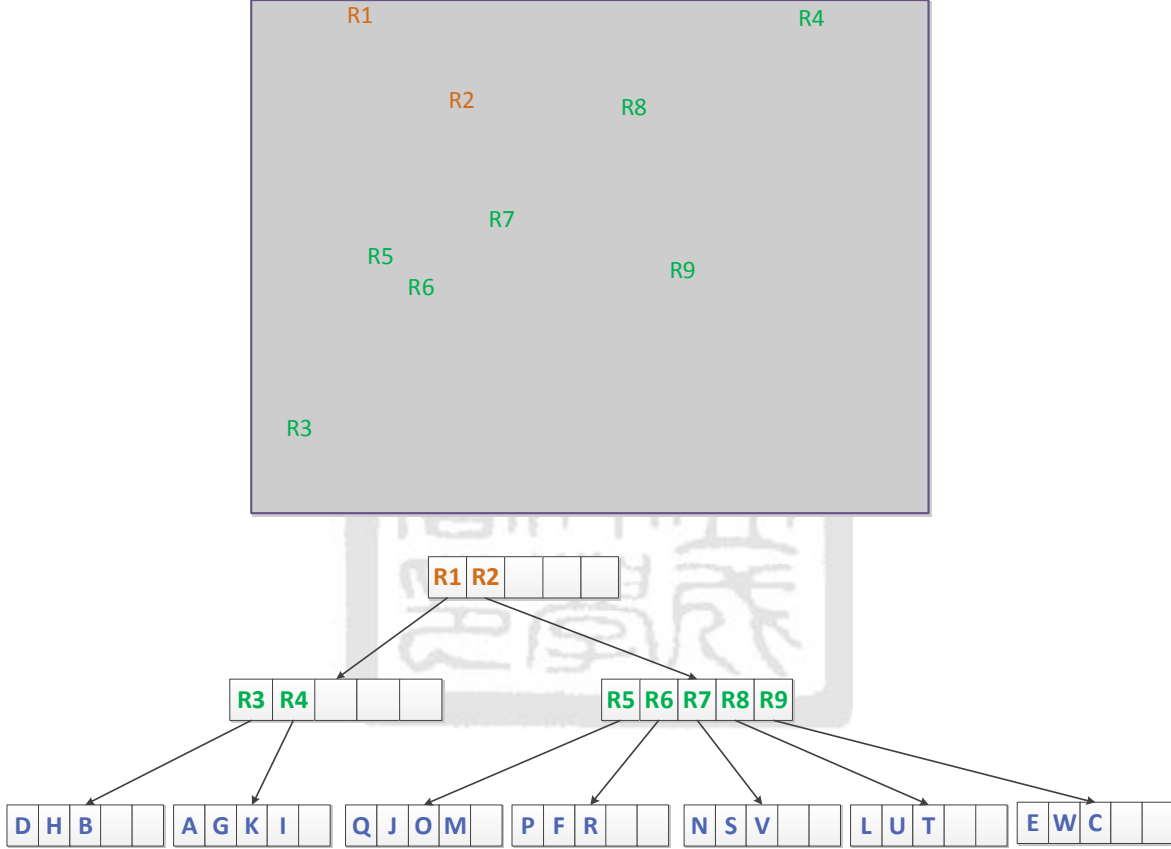


Fig 2 Example of R-Tree

### 2.2.2 Game Theory

Game theory [5] [10] is the study of mathematical models of conflict and cooperation between intelligent rational decision-makers. It is mainly used in economics, political science, psychology, logic and biology. In recent years, more and more researches apply it in the field of computer science, and present many kinds of game model for solving the different issues.

There are three basic elements in every game including players, strategies available to each player, and the payoffs based on one player's decision(s) and the decision(s) of other(s). Cooperative and non-cooperative game are two types of games. If the players can bind commitments, the game is cooperative. On the other hand, the players are not able to have a contract in non-cooperative game.

In non-cooperative game, we have to find an equilibrium solution concept for all players called as Nash Equilibrium [5]. The key idea of Nash equilibrium is that each player chooses the best strategy, given the strategies chosen by the other players. In other words, no player can benefit by changing strategies while the other players keep theirs unchanged. The results of strategy choices and the corresponding payoffs constitute Nash equilibrium. The example of Nash equilibrium represented in a payoff matrix is showed in Fig 3:

		Player 2	
		Strategy C	Strategy D
Player 1	Strategy A	3, 3	2, 6
	Strategy B	6, 2	4, 4 ← Nash Equilibrium

Fig 3 Example of Nash equilibrium

Game formulation can be used for multi-player optimization in some research issues. For example, one of the issues is resource management in wireless networks such as channel allocation. Moreover, the game theoretic model can be used in power control schemes and the problem of competitive spectrum sharing [8].

In our study, we formulate the problem of multi-taxi recommendations at the same time by non-cooperative game model, and find the optimization solution of this game to provide the best recommendation set to all taxis. The problem and the game model applied are detailed specifically in latter chapter.

## 2.3 Recent Traffic Simulators

### 2.3.1 SUMO

SUMO in [1] is an open source, multi-modal traffic simulator. The road network is simulated as a network of nodes connected by edges. The simulation is created on the microscopic level. On the other words, each vehicle is given explicitly, defined at least by an identifier (name), the departure time, and the vehicle's route through the network.

There are two inputs fed in SUMO: the network file and the route file. The network file is an Extensible Markup Language (XML) file describing a network using nodes, edges, lanes and junctions. It can be generated by the tool NETCONVERT. On the other hand, the route file is an XML file containing the description of vehicles with the information such as maximum speed, maximum acceleration, id and type.

The traffic simulation has two versions. First was the basic command-line application for processing the batch of vehicles given as input. In 2006 the simulation was extended to interact with an external application via a socket connection. TraCI is a technique for interlinking road traffic and network simulators. It permits us to control the behavior of vehicles during simulation runtime.

### 2.3.2 MObility model generator for Vehicular networks (MOVE)

MOVE in [6] is a java-based application built on SUMO in [1] with a set of Graphic User Interfaces. The key feature of MOVE is dealing with the configuration of SUMO on traffic level with the simple visualization tool. The user can generate the files fed in SUMO easily without specifying the internal details of the simulator. Thus, it allows users to generate realistic scenarios quickly.

MOVE is consisted of two modules: Mobility Model and Traffic Model. The Mobility model can generate the road map topology and vehicle movement, and contains three parts:

Map Editor, Vehicle Movement Editor and Simulation.

Firstly, creating the topological maps for network scenarios and then generating the movement patterns in the Vehicle Movement Editor automatically or manually. MOVE can generate not only the mobility trace from the TIGER database or Google Earth but also the map graph defined by users. In the Vehicle Movement Editor, we can define the traffic flows, the probability of turning on each junction or set the basic trip for each vehicle type in random way

The second module is the Traffic Model which generates trace files containing the information of realistic vehicle movements that can be directly used by popular simulator tools such as Network Simulator-2 (NS-2) or Qualnet. It can provide simulation of static mobility model or dynamic mobility model with TraCI.

### 2.3.3 TraCI

TraCI [11] is an open source architecture that couples road traffic with network simulator. Different from other simulators, the mobility patterns generated by TraCI are not pre-established as fixed trace files so that they are flexible. They are generated during the runtime and fed to the network simulator. The simulation is divided into many steps. The client sends the request to sever so that the server can move on to the next step. The setting of vehicles' state can be adjusted by the commands from TraCI at any time.

TraCI offers the interface that permits the real-time coupling of the traffic simulator with another program by a client server approach. Thus, we can enable the client application to control and have full knowledge of the simulation in real time.

In general, there are some methods to control the SUMO traffic simulator via TraCI. One is the "SUMO" traffic simulator that is coupled with the "NS2" network simulator through TraCI interface. The traffic simulator is extended as TraCI Server and network

simulator as TraCI Client. The other method to control the SUMO simulator is using the TraCI client Python Application Program Interface (API) which is available in the SUMO package. For our purpose, we use the latter one.

The system simulator written in the Python communicate with SUMO using the TraCI client API, and sends the commands in the form of requests to the traffic simulator. The traffic simulator performs the specified actions and responds to the system simulator correspondingly. As mentioned above, we can control the traffic in SUMO in real time by using the system simulator based on the responses of the traffic simulator in the previous steps.





## Chapter 3 System Structure

### 3.1 Overview

We assume that the taxicabs are equipped with GPS sensors. By some weight sensors or the taxi meter, the taxis can record the occupancy information and send the trajectories with timestamp to server in a certain frequency. Additionally, all of the taxis and passengers are identified.

As Fig 4 shows, there are many waiting places which represent the popular locations for picking up passengers and waiting for taxis in our environment. The taxicabs are cruising on the roads to find passengers and picking them up. On the other hand, the passengers are waiting for the taxis in our service region.

System architecture is presented in Fig 5. The server collects historical data including taxis' and passengers' information for analysis. When the taxis and passengers need recommendations, the server replies them with corresponding information.

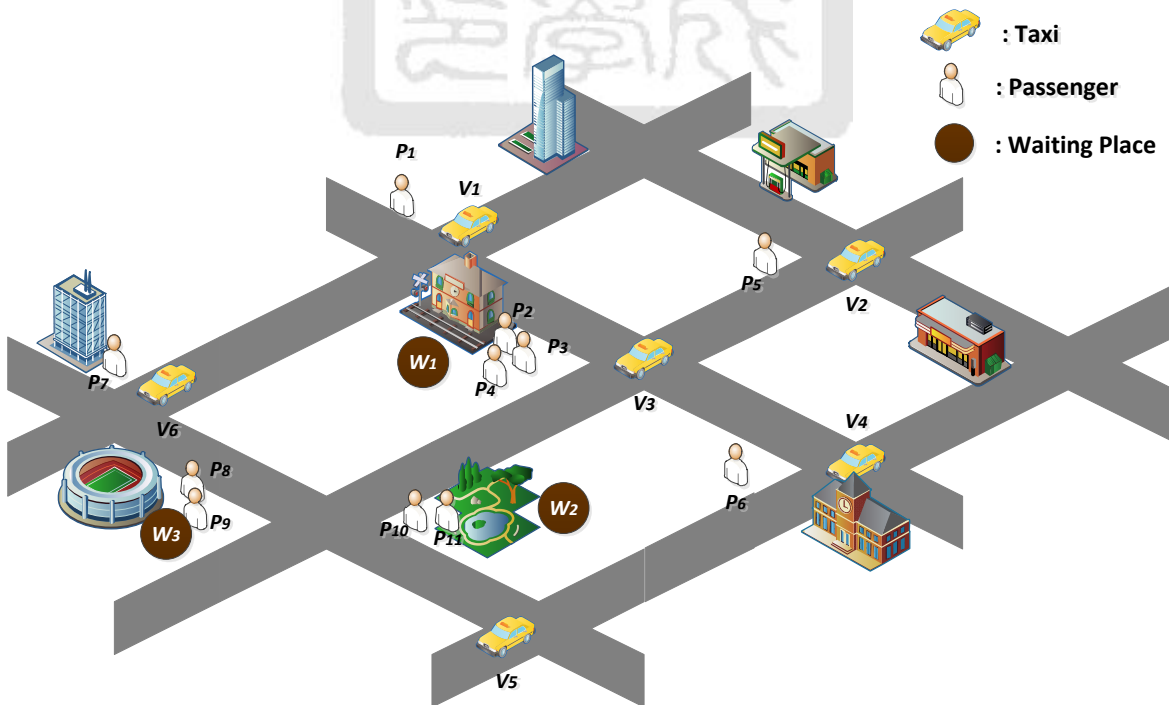


Fig 4 Schematic diagram

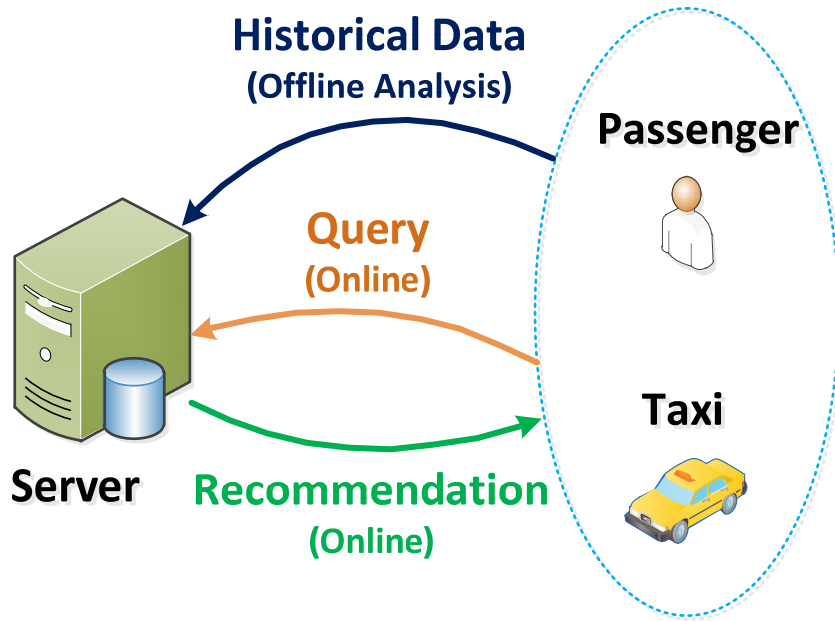


Fig 5 System architecture

The process of our work comprises three phases including Cognition Phase, Inference Phase and Recommendation Phase. The procedure of our method is shown in Fig 6 where each block with dashed line represents each phase in our process.

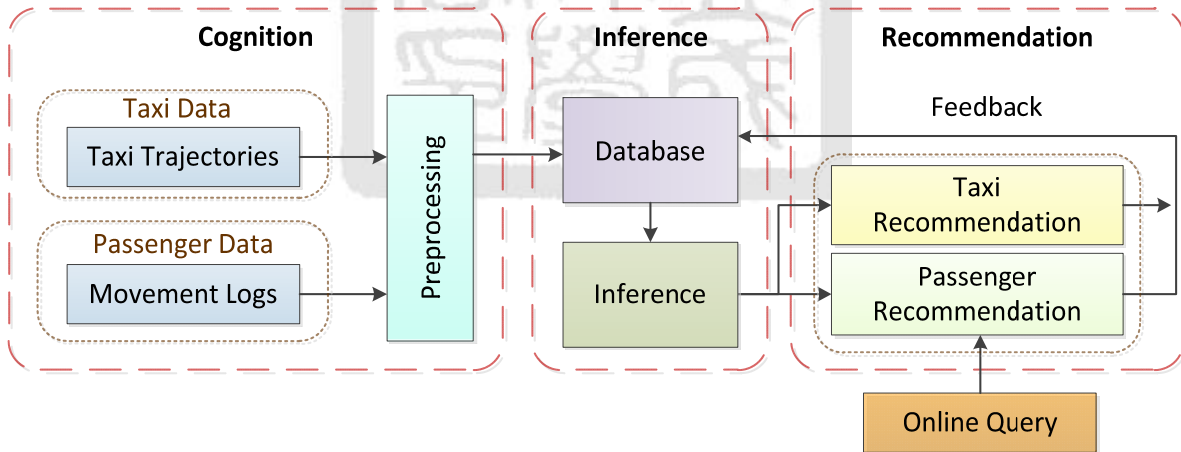


Fig 6 Flow Chart

The first phase is Cognition phase which collects and preprocesses the historical information of taxis and passengers. This information includes taxi GPS trajectories and movement logs of passengers. After preprocessing the information, the results are stored in Database in Inference phase.

Inference module in the second phase analyzes the data stored in Database module. Therefore, we can learn the popular locations for picking up passengers from a large number of trajectories with occupancy information. The time-dependent R-Trees are constructed in this module for recommendation in the third phase.

Recommendation phase contains two modules, Taxi Recommendation module and Passenger Recommendation module. Taxi Recommendation module deals with the requests from the taxis, and Passenger Recommendation module is responsible for tackling the queries from passengers.

When taxis send a request including their location and timestamp for recommendation, the server generates the waiting places and the route to these places based on the time-dependent R-Trees. Passenger Recommendation module recommends the passengers proper locations for waiting taxis easily.

Moreover, taxis keep searching passengers for taxi-sharing if taxis are not full and willing to accept the passengers for sharing. Every taxi and passenger can set their target profit they want to achieve from ridesharing. Our method filters the inappropriate passengers out according to the rules and chooses the best suitable one to rideshare. Under the conditions, taxis find the passengers for ridesharing as many as possible.

## **3.2 Parameters Definitions**

A taxi is advised on some locations and the routes to these locations. For passengers, the server recommends people some locations within walking distance. Before introducing the method we proposed, we define some parameters and make a few assumptions in order to explain our mechanism.

Every taxi and passenger in the system are identified. They record their movement logs and report to server in a specific frequency (e.g. 2 minutes). The taxis and passengers

send queries including their location and time to the system. We suppose that taxis and passengers comply with the suggestion given by the server. If a taxi is occupied but still has available seats, it finds the next passengers for sharing. A taxi only can pick up one person at a time.

We partition the road network using the spatial index R-Tree containing a set of leaf and non-leaf nodes. After analyzing the historical information stored in database, we can build an R-Tree for each time slot. The recommendation for passengers and taxis can be provided in the form of regions within the corresponding R-Tree. To formulate the problem, some notations are used in Table 1.

Table 1 Summary of variables

Symbol	Representation
$P = \{p_1, p_2, \dots, p_n\}$	The set of passengers.
$V = \{v_1, v_2, \dots, v_m\}$	The set of taxis.
$RT^i = \{r_1^i, r_2^i, \dots, r_j^i\}$	R-Tree at time $i$ .
$r_j^i$	Region $j$ in R-Tree $RT^i$ .
$W = \{w_1, w_2, \dots, w_k\}$	The set of waiting places.
$R^{v_i(t)}$	The Route of Taxi $i$ at time $t$ .

### 3.3 Cognition Phase

This phase collects the historical information of taxis and passengers, and preprocesses these data for Inference Phase. There are two modules representing Taxi Trajectories and Movement Logs for gathering the data from taxis and passengers respectively. We discuss it in the following section.

#### 3.3.1 Taxi Trajectories

A taxi trajectory is a sequence of GPS points logged for a working taxi. Each point contains the following terms: taxi ID, time stamp, longitude, latitude, state. The state of taxi indicates whether the taxi is occupied or not. We consider four states: *cruising* ( $\mathcal{C}$ ), *parking* ( $\mathcal{P}$ ), *occupied & shared* ( $\mathcal{O.S}$ ) and *occupied* ( $\mathcal{O}$ ), detailed in Table 2.

When the taxis enter the system, they need the recommendation and follow the routes to find passengers. At that time, the taxis are in cruising state. The parking state proposed in this thesis is the status that taxi drivers wait somewhere (*waiting place* in this thesis) for business, and it doesn't really imply that taxis are parked in a parking lot. The status always appears at airports, train stations and shopping malls..., etc. These places stand for the hot locations for waiting passengers and are called *waiting places* in our work.

The taxi is non-occupied in both the cruising and parking states. When passengers get on a taxi, the state of taxi is turned to *Occupied & Shared* ( $\mathcal{O.S}$ ) or *Occupied* ( $\mathcal{O}$ ) state. A taxi is in the *Occupied & Shared* ( $\mathcal{O.S}$ ) state if it is not full and can accept other passengers for ridesharing. The taxis in the *Occupied & Shared* ( $\mathcal{O.S}$ ) state carry their passengers in the car to their destinations, and keep searching the suitable passengers for ridesharing until the taxis are full. On the other hand, the drivers of taxis in *Occupied* ( $\mathcal{O}$ ) state only drive along the paths they planned for passengers to their destinations.

### 3.3.2 Movement Logs

Movement Logs module records the passengers' mobility patterns including original and destination locations. We can learn some knowledge from these GPS trajectories. One of the knowledge is taxis' behavior containing picking-up and dropping-off passengers. The other is passengers' mobility patterns indicating where and when the passengers get on and off a taxi.

Table 2 States of a taxi

State	Taxi Status
Cruising ( $\mathcal{C}$ )	A taxi is traveling without a passenger
Parking ( $\mathcal{P}$ )	A taxi is waiting for a passengers at some places
Occupied & Shared ( $\mathcal{O}, \mathcal{S}$ )	A taxi is occupied by passengers (not full) and can accept the passengers for sharing.
Occupied ( $\mathcal{O}$ )	A taxi is occupied by passengers and full.

## 3.4 Inference Phase

The main objectives in this phase are historical information storage and analysis. This phase consist of two modules, Database module and Inference module. Database module is responsible for the storage of data from the first phase. Data stored in Database module are studied by Inference module.

In Inference module, we figure out the waiting places representing the popular locations where passengers are picked up frequently. After that, the server can suggest taxis and passengers for better routes and locations respectively according to the results inferred. As the result, the taxis can pick up passengers quickly, and the passengers can find taxis easily, too.

We partition the road network and represent the service region in our system by

spatial index data structure R-Tree, which is illustrated in Fig 7. A point in the road map stands for a GPS trajectory point. The rectangles containing a lot of points represent the regions in an R-Tree. The denser points are in the map, the more number of regions (rectangles) are generated.

Let  $M$  be the maximum number of entries that is able to fit in one node in an R-Tree. When the number of entries in the node is up to  $M$ , the node is regarded as a waiting place. For taxis' recommendation, the server chooses some waiting places and plans the routes to these places. Our recommendations of routes and locations are in the form of regions in R-Trees.

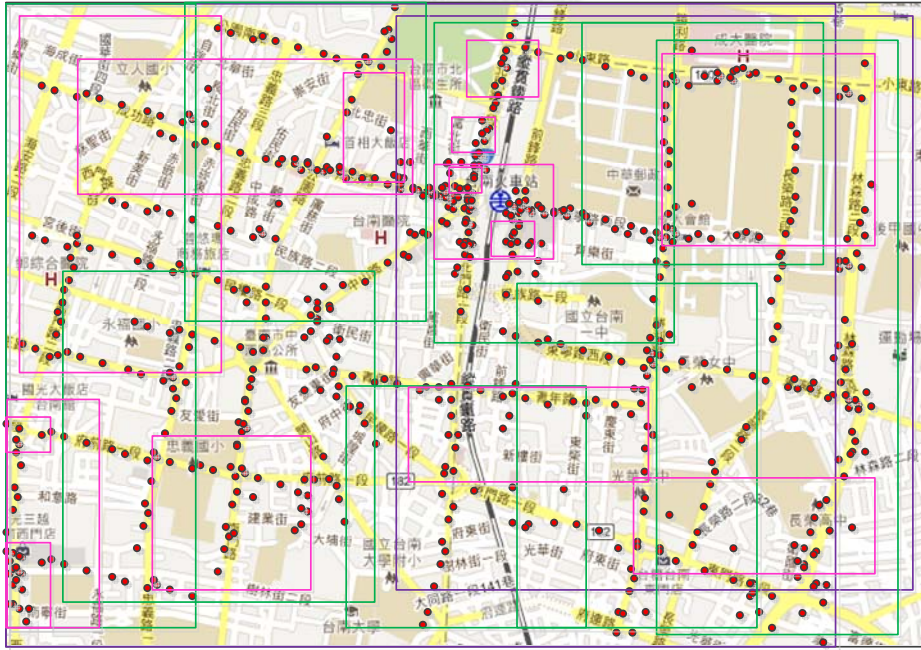


Fig 7 Example of R-Tree on road network

The recommendation need to be different because the popular locations for picking up passengers change as time goes by. Based on a large number of historical GPS trajectories generated by taxis, we build the time-dependent R-Trees offline.

We partition a day into fixed slots (e.g. one hour per slot), and analyzing the trajectories to build an R-Tree for each time slot. There are  $I$  time slots in a day, and the

length of each time slot is  $\varepsilon$ . The  $i_{th}$  R-Tree is represented as  $RT^i = \{r_1^i, r_2^i, \dots, r_j^i\}$ , and  $r_j^i$  is the region  $j$  in R-Tree  $RT^i$ .

As the Fig 8 shows, the center node  $c_j^i$  is the road network node in region  $r_j^i$  whose distance between the center of region  $r_j^i$  is the shortest. The travel time  $t_{jk}^i$  and the distance  $d_{jk}^i$  between region  $r_j^i$  and region  $r_k^i$  are pre-computed for all of the regions in each R-Tree. Using the speed limit of road segment, we can estimate the travel time between two regions. Hence, the server constructs the R-Tree  $RT^i$  for each time slot  $i$  under the offline mining.

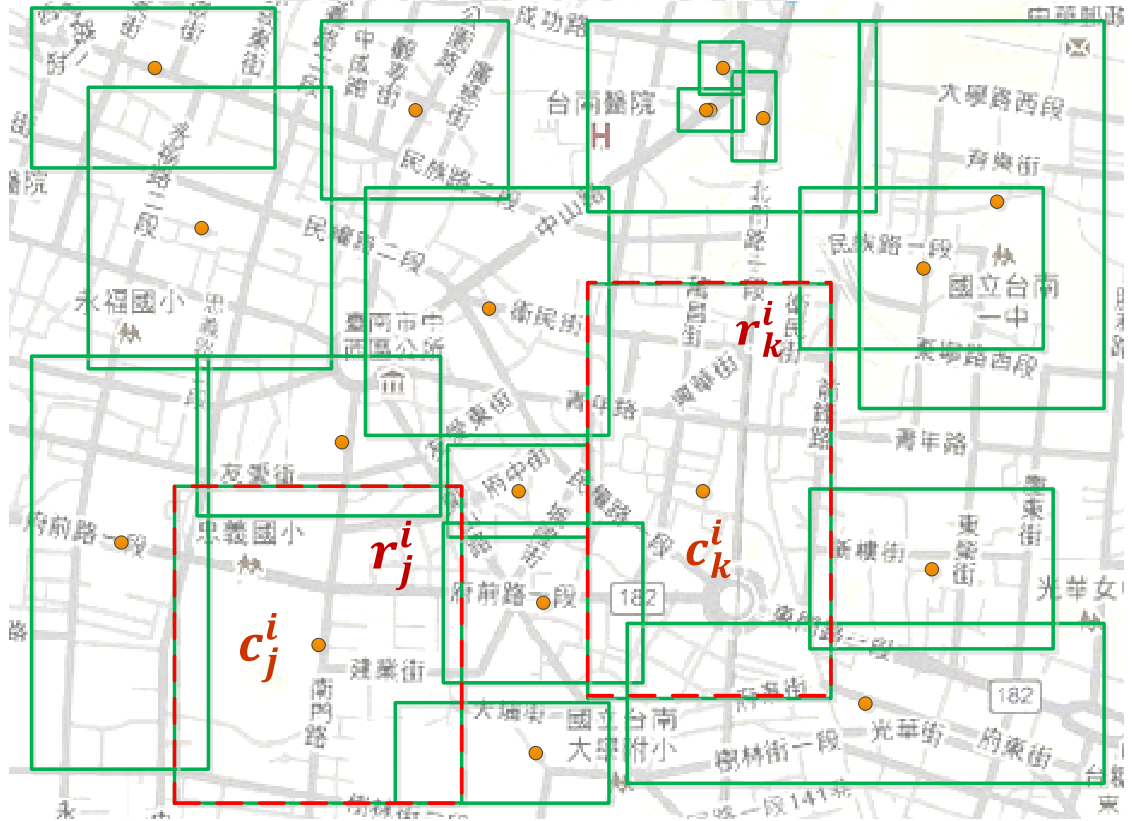


Fig 8 Region-partitioned map (in R-Tree  $RT^i$ )



### 3.5 Recommendation Phase

In this stage, we provide recommendation to taxis and passenger when they need one. Given the location and time of taxi or passenger, the server finds the proper routes or locations for recommendation based on the analysis produced by Inference module. There are two modules in this phase: Taxi Recommendation and Passenger Recommendation.

#### 3.5.1 Recommendation for One Taxi

For the status of a taxi, we consider four states: *cruising* ( $\mathcal{C}$ ), *parking* ( $\mathcal{P}$ ), *occupied & shared* ( $\mathcal{O.s}$ ) and *occupied* ( $\mathcal{O}$ ), detailed in Table 2. The transition graph is depicted in Fig 9, and the representations of transitions are detailed in Table 3.

At the start, a taxi joins in the system and needs the recommendation for finding passengers. The server provides some suggestions to the taxi so that the taxi follows the route in the *cruising* ( $\mathcal{C}$ ) state. If the taxi can't find the passengers through the route, it turns to the *parking* ( $\mathcal{P}$ ) state and keep waiting. Once the taxi picks up a passenger, it changes their state to *occupied & shared* ( $\mathcal{O.s}$ ) or *occupied* ( $\mathcal{O}$ ) depending on whether the passenger wants to share or not.

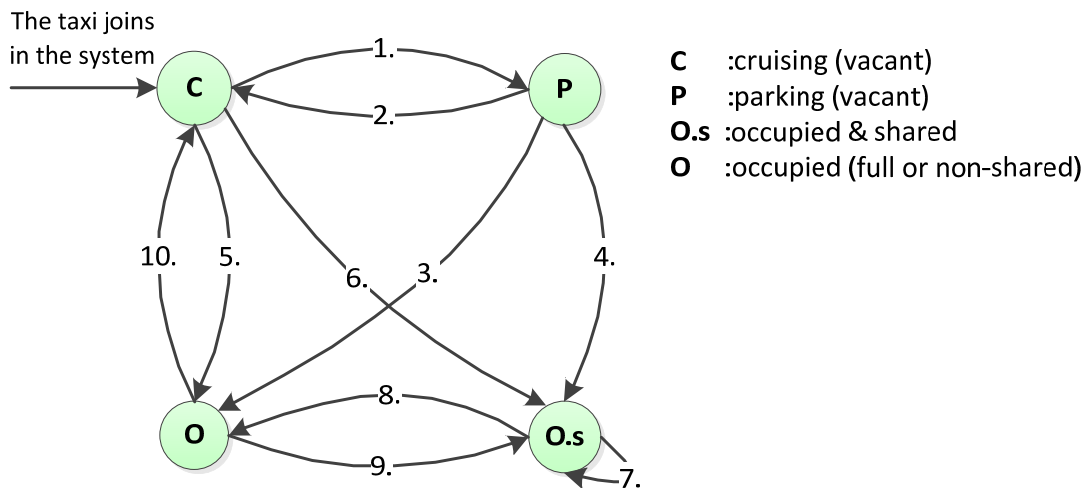


Fig 9 Taxis' Status

Table 3 Representation of transitions

Transition	Representation
Transition 1	The taxi doesn't find a passenger during the route.
Transition 2	The taxi doesn't find a passenger in the waiting place.
Transition 3	The taxi finds a passenger who doesn't want to taxi-share in the waiting place.
Transition 4	The taxi finds a passenger who wants to taxi-share in the waiting place.
Transition 5	The taxi finds a passenger who doesn't want to taxi-share in the route
Transition 6	The taxi finds a passenger who wants to taxi-share in the route.
Transition 7	The taxi finds a passenger who wants to taxi-share and the taxi is not full.
Transition 8	The taxi becomes full.
Transition 9	The taxi drops off a passenger and the passengers in the taxi wants to taxi-share.
Transition 10	The taxi drops off all passengers and becomes vacant.

For the taxi recommendation, we perform a range query according to the location and time of the taxi, and retrieve some waiting places. For each waiting places, the server searches the regions of the R-Tree near the taxi and generates the routes. The recommendation routes are constrained by distance threshold  $D_t$ . If the routes to these waiting places exceed the distance threshold, the server generates the routes based on the corresponding R-Tree. The example of taxi recommendation is shown in Fig 10.

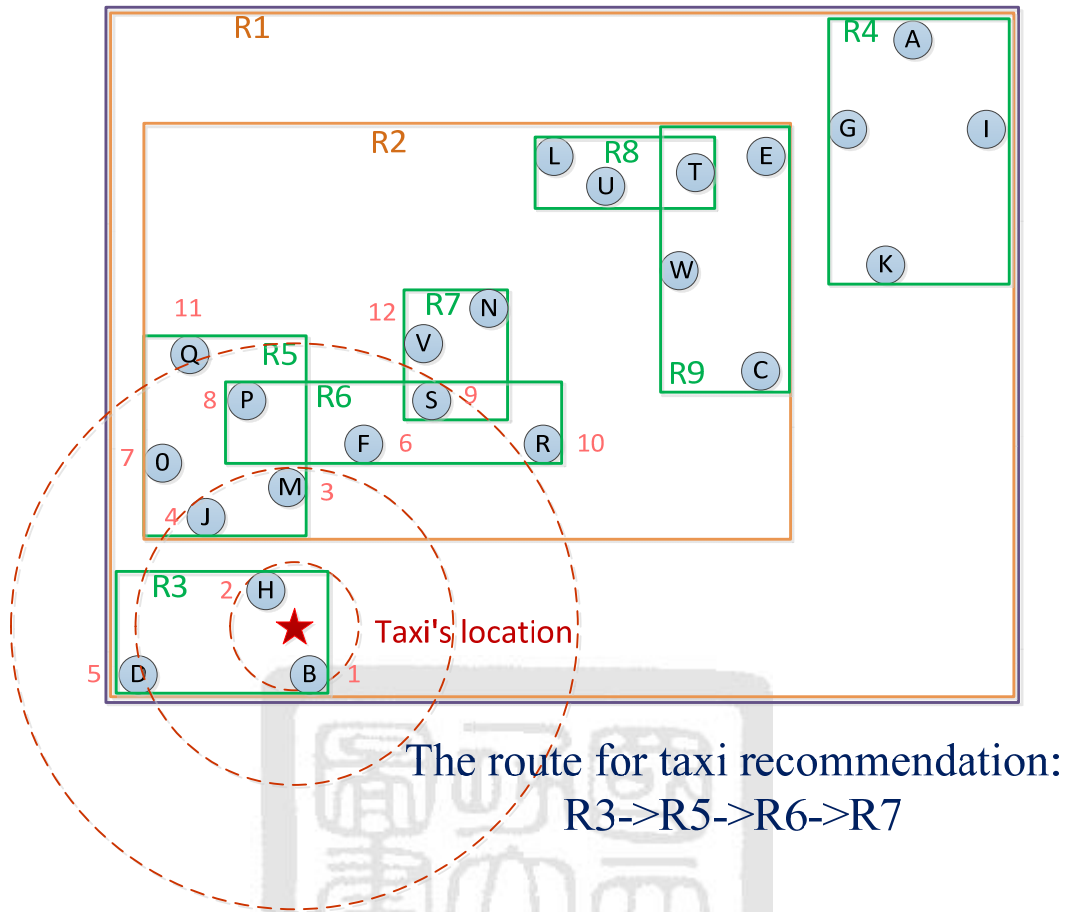


Fig 10 Example of taxi recommendation

### 3.5.2 Non-cooperative Game Model for Taxi Recommendation

At the previous section, we discuss about the recommendation for only one taxi. In fact, more than one taxis need recommendation at the same time. Therefore, in reality we also need to consider the situation which many taxis want to be recommended at the same time. The scenario is illustrated as Fig 11.

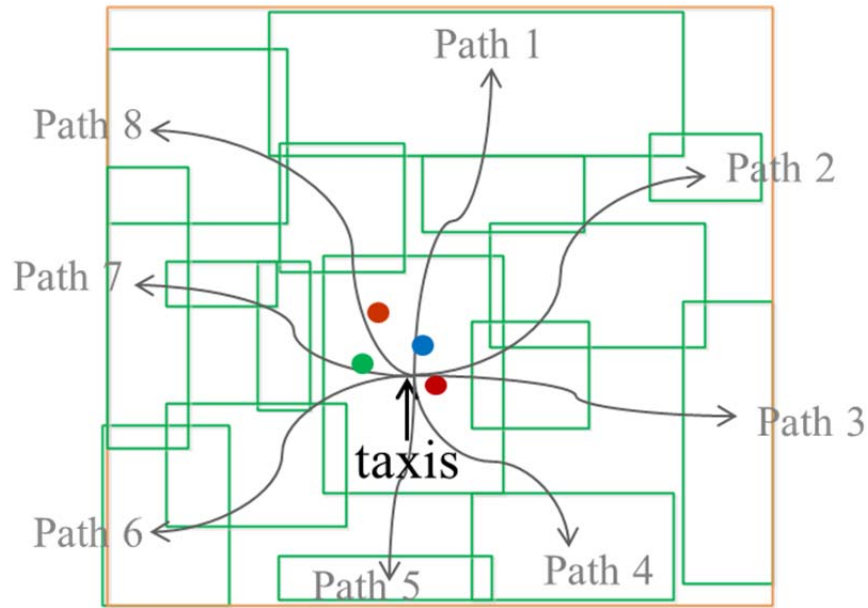


Fig 11 Scenario of recommendation for many taxis

The problem of recommendation for many taxis can be modeled as an oligopoly market competition. In the oligopoly market, there are a few firms controlling the quantity and the price of supplied commodity for highest profit. The firms compete with each other and are likely to be aware of the actions of others. The decisions of one firm influence and are influenced by others’.

In general non-cooperative game model for oligopoly market, all of the firms compete in terms of the product quantity. In our problem, we consider that the taxis in the same region ask for route recommendations. The taxis compete with each other when they choose the same route for finding passengers. The competition among the taxis here is in terms of the probability for picking up passengers. The profit of a taxi in the game model is related to the probability and the cost the taxi needs to bear.

In this section, a non-cooperative game model is applied to provide ideal recommendation. Due to the centralized recommendation scenario, all the taxis which are asking for recommendation in the same region can observe the strategies and the payoffs of

other taxis completely. In the scenario, a taxi observes the change in payoff due to the different revenue and cost, so the taxi adapts its strategy accordingly.

The main objective of the recommendation problem is to maximize the profits of the taxis in the same region by utilizing the concept of equilibrium. As the concept mentioned above, a non-cooperative game model can be formulated as follows. The **players** in this game are the taxis who need the route recommendations in the same region at that time. The **strategy** of each taxi is the choice of route. The **payoff** of each taxi is the profit (i.e., revenue minus cost) related to probability for picking up a passenger and the cost born by the taxi. The **commodity** in the scenario is recommendation routes.

Firstly, we assume there are  $N$  taxis in the same region with the request of recommendation at time  $t$ . Path  $s_i$  for recommendation is considered a strategy of taxi  $i$  in the system model.  $\mathbb{S}$  denotes the set of strategies of all taxis in the region (i.e.  $\mathbb{S} = \{s_1, \dots, s_N\}$ ).

Based on the assumption described above, we represent the function to calculate the cost which taxis need to bear according to the selections of strategies. This cost function is given by

$$c(\mathbb{S}) = \eta_1 \times \left[ \sum_{s_i \in \mathbb{S}} (s_{i,d} + s_{i,t}) \right]^\tau \quad (3-1)$$

, where  $\eta_1$  is a normalized factor to adjust the cost,  $s_{i,d}$  and  $s_{i,t}$  is the distance and travel time for strategy  $s_i$ ,  $\tau$  is a non-negative constant, and  $\tau \geq 1$  (so that the cost function is convex i.e., there exists the extreme value in this function), and  $\mathbb{S}$  denotes the set of strategies of all taxis in the same region.

The revenue of taxi  $i$  is

$$r_i = \frac{E(s_i)}{M(s_i)} = \frac{s_i \times p_{s_i}}{M(s_i)} \quad (3-2)$$

, where  $E(s_i)$  is the expectation (the variable multiplies its corresponding probability) of

finding a passenger by choosing the strategy  $s_i$ ,  $M(s_i)$  is the number of taxis who choose the strategy  $s_i$ . Because more than one taxi choose the same route, the probability that a taxi succeeds in picking up a passenger is divided. In other words, the more taxis go the same way, the less likely a taxi finds a passenger.

As mentioned above, we can present the profit by subtracting the cost from the revenue. Thus, the profit of taxi  $i$  can be rewritten as follows:

$$\pi_i(\mathbb{S}) = \eta_2 \times [r_i - s_i \cdot c(\mathbb{S})] = \eta_2 \times \left\{ \frac{s_i \times p_{s_i}}{M(s_i)} - s_i \{ \eta_1 \times [\sum_{s_i \in \mathbb{S}} (s_{i.d} + s_{i.t})]^\tau \} \right\} \quad (3-3)$$

, where  $\eta_2$  is a normalized factor to adjust the profit.

Let  $\mathbb{S}_{-i}$  denote the set of strategies adopted by all taxis except taxi  $i$  (i.e.  $\mathbb{S}_{-i} = \{s_j | j = 1, \dots, N; j \neq i\}$  and  $\mathbb{S} = \mathbb{S}_{-i} \cup \{s_i\}$ ). The optimal recommendation of routes to each taxi depends on the strategies of other taxis. Nash equilibrium is regarded as the solution of the game. Nash equilibrium of a game is a strategy profile (e.g. lists of strategies in the form of one for each other). The property of Nash equilibrium is that no player can increase his payoff by choosing another strategy according to other players' strategies. It means that no player is willing to change his action for more profit, given the others' actions.

The Nash equilibrium in this case is obtained by using the best response function which is the best strategy of one player, given others' strategies. The best response function of taxi  $i$  given the strategies of the other taxis  $s_j$ , where  $j \neq i$ , is defined as follows:

$$\mathcal{BR}_i(\mathbb{S}_{-i}) = \operatorname{argmax}_{s_i} \pi_i(\mathbb{S}_{-i} \cup \{s_i\}) \quad (3-4)$$

The set  $\mathbb{S}^* = \{s_1^*, \dots, s_N^*\}$  denotes the **Nash equilibrium** of this game if and only if  $s_i^* = \mathcal{BR}_i(\mathbb{S}_{-i}^*)$ ,  $\forall i$ , where  $\mathbb{S}_{-i}^*$  denotes the set of the best responses for taxi  $j$  for  $j \neq i$ .

We can obtain the Nash equilibrium by solving the following equation

$$\frac{\partial \pi_i(\mathbb{S})}{\partial s_i} = \eta_2 \left\{ \frac{p_{s_i}}{M(s_i)} - \{\eta_1 [\sum_{s_i \in \mathbb{S}} (s_{i.d} + s_{i.t})]^\tau\} - \eta_1 s_i \tau [\sum_{s_i \in \mathbb{S}} (s_{i.d} + s_{i.t})]^{\tau-1} \right\} = 0 \quad (3-5)$$

Therefore, the optimization problem with the objective can be defined as follows:

$$\text{Minimize } \sum_{i=1}^N |s_i - \mathcal{BR}_i(\mathbb{S}_{-i})| \quad (3-6)$$

i.e., minimize the sum of the difference between decision variable  $s_i$  and corresponding best response function. The algorithm reaches the Nash equilibrium while the minimum value of the objective function is zero. As a result, the server determines the route recommendation for each taxi in the same region at arbitrary time.

### 3.5.3 Passenger Recommendation

At the beginning, we introduce the passenger recommendation for the purpose of getting in a taxi. Taxi-sharing mechanism which tries to find the second or the third passenger is detailed in the next section.

A passenger who needs the recommendation delivers a range query with location and current time to the server. Accordingly, the server searches the surrounding candidate regions for recommendation in the range of distance threshold  $D_p$ . The threshold  $D_p$  is a walking distance. By calculating the scores of each candidate region, the server selects the region with the highest score for recommendation.

The score function  $S$  of candidate region  $j$  for the  $m$ th passenger can be obtained as follows:

$$S^m(j) = \eta_3 \times (\text{the number of children in } R_j) \times \frac{1}{D_j^m} \quad (3-7)$$

, where  $\eta_3$  is a normalized factor to adjust the score,  $D_j^m$  is the distance between region  $j$  and the location of  $m$ th passenger. The example is shown as Fig 12. The children in candidate region are the trajectory points contained in the region of R-Tree constructed previously. The more children in the region, the more possible finding a taxi is.

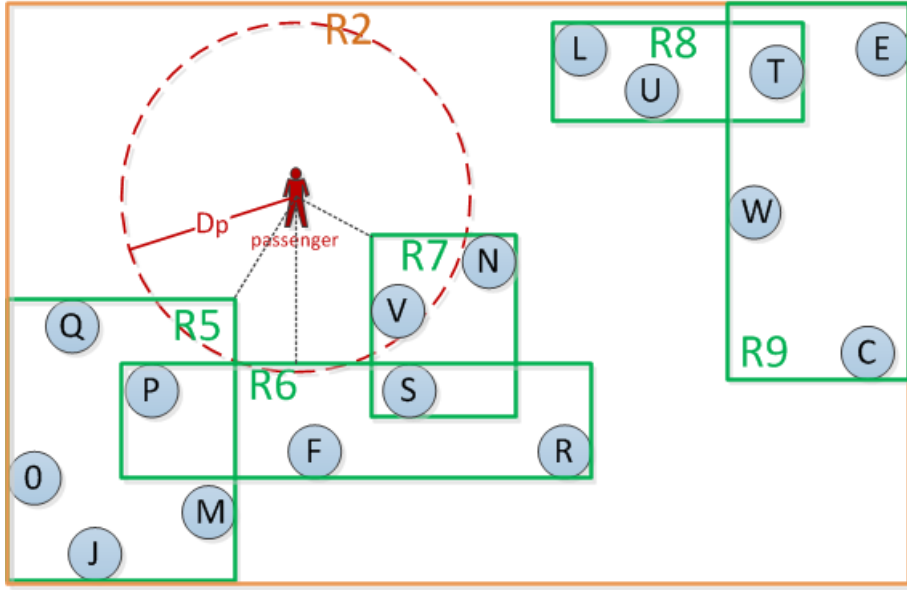


Fig 12 Example of passenger recommendation

#### 3.5.4 Recommendation for Taxi-sharing

As mentioned above, passengers can find the taxi easily and quickly. After getting in a taxi, the passenger may want to find others for taxi-sharing to reduce the payment. The aims of taxi-sharing are two folds of goals. One is reducing the payment of a passenger. Increasing the revenue of a taxi is the second objective. What is more, we also have to decrease the waiting time of taxis and passengers by the action.

Given the passengers  $c_1$  and  $c_2$  are in taxi  $v$  at time  $t$ , and a new passenger  $c_3$  has sent a taxi-sharing query to server. The taxi  $v$  needs to determine whether the passenger  $c_3$  is appropriate for taxi-sharing. Note that a trip of a taxi is defined as the duration from picking up the first passenger to becoming vacant again. A definition of trip of taxi is depicted as Fig 13.



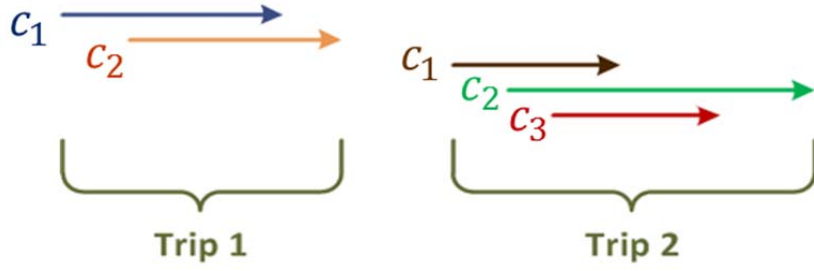


Fig 13 Definition of a trip

Besides original location and destination, every passenger needs to attach the information about taxi-sharing such as the time limit of detouring, the expected value of fare saved and the degree of urgency. The user settings of a passenger are detailed in Table 4.

Table 4 User settings of a passenger

User settings	Representation
$S$	Original location
$D$	Destination
$T_w$	The limit of detour time
$F_e$	The expectation of fare saved
$(w_1, w_2)$	The degree of urgency

The candidate passengers of the taxi  $v$  for taxi-sharing are filtered through the area in the form of rectangle. The height of the rectangle is the distance from current location to destination of the taxi. On the other hands, the width is a distance threshold  $D_s$ . The bounded area is illustrated as Fig 14

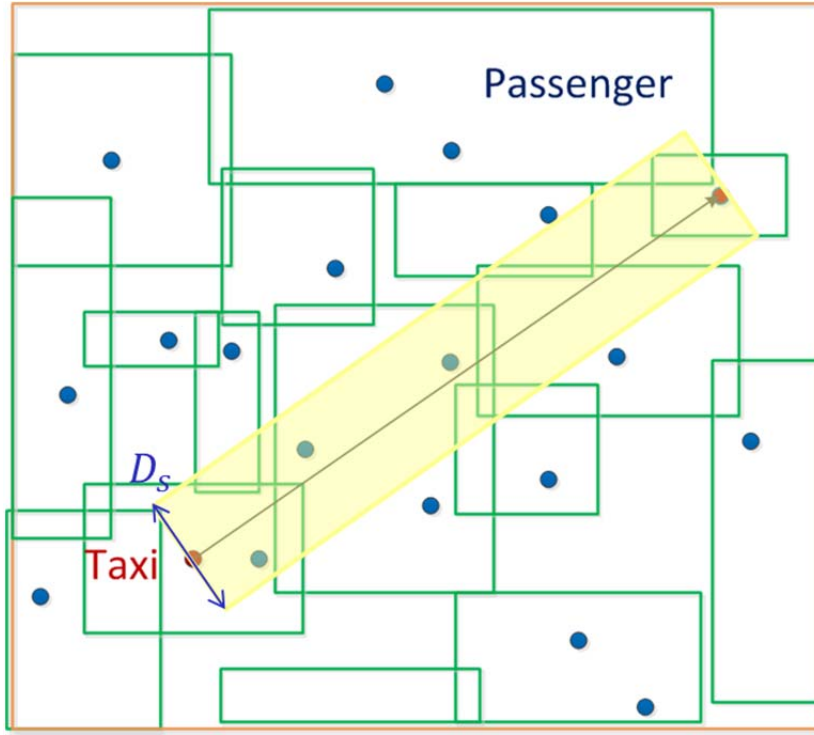


Fig 14 The bounded area of candidate passengers for taxi-sharing

After obtaining the candidate passengers, we propose four rules to check whether a candidate passenger is proper for ridesharing with on-board passengers in the taxi. Three of these rules are from the aspect of passengers, and the last rule is related to taxis. The variable assumptions for each passenger  $c_i$  at time  $t$  are displayed in Table 5.

Table 5 Variables for a passenger

Variable for Passenger	Representation
$l(t)$	Current location at time $t$
$d_t(t)$	Travel distance
$d_r$	Remaining distance
$t_p$	Picking-up time
$t_w(t)$	The maximum delay time for detour at time $t$
$t_d$	The time for detouring
$d_d$	The distance for detouring
$d_s$	The distance for taxi-sharing
$f_b$	The payment before taxi-sharing

$f_a$	The payment after taxi-sharing
$f_p$	The difference between the payment before and after taxi-sharing ( $f_p = f_b - f_a$ )
$Cost_d$	The cost for detouring ( $Cost_d = w_1 \times d_d$ )
$Profit_s$	The profit for taxi-sharing ( $Profit_s = w_2 \times d_s$ )

For each passenger  $c_i$  in the taxi at time  $t$ , the taxi needs to check the following three rules. The first rule meaning the cost for detouring must be less than the profit for taxi-sharing is described as follows:

**Rule 1:**  $Cost_d \leq Profit_s$  (3-8)

If  $Cost_d$  is greater than  $Profit_s$ , the new passenger is not appropriate. The second rule showing the limitation of time for detouring of each passenger can be represented as follows:

**Rule 2:**  $t_d \leq t_w(t)$  (3-9)

$$, t_w(t) = T_w \times (1 - \frac{(t-t_p)}{t_o}) \quad (3-10)$$

If  $t_d$  is greater than  $t_w(t)$ , the new passenger is not appropriate.

We consider the payment saved for taxi-sharing in the third rule. Each passenger sets the expected value of payment saved. The benefit from carrying the new passenger needs to fulfill the expectation so that the taxi-sharing can achieve the goal for saving money. The third rule is

**Rule 3:**  $f_p < 0 \text{ and } |f_p| \geq F_e$  (3-11)

If  $f_p$  is positive or the value of  $|f_p|$  is less than  $F_e$ , the new passenger is not appropriate.

Moreover, we consider the profit of not only passengers but also taxis. A taxi can set the expectation of profit for providing taxi-sharing service which is denoted as  $P_e$ . The new candidate passenger has to let the taxi get more revenue which is not less than  $P_e$ . For taxi  $v$

at time  $t$ , the last rule which needs to be check is

**Rule 4:** 
$$p_p \geq P_e \quad (3-12)$$

If  $p_p$  is less than  $P_e$ , the new passenger is not appropriate. Note that the variable assumption is displayed in Table 6

Table 6 Variables for a taxi

Variable for Taxi	Representation
$S$	Original location of a trip(Given from passenger)
$D$	Destination of a trip(Given from passenger)
$p_b$	The fare before providing the taxi-sharing service
$p_a$	The fare after providing the taxi-sharing service
$p_p$	The profit of providing the taxi-sharing service ( $p_p = p_b - p_a$ )
$A$	Award value

After all of the rules are satisfied, we can compute the fitness degree of the candidate passengers left. The fitness value can be obtained as follows:

$$F_{v[t]}(c_3) = \eta_4 \times p_p \times \sum_{i=1}^3 (|c_{i,fp}| \times \frac{1}{c_{i,td}}) \quad (3-13)$$

, where  $\eta_4$  is a normalized factor to adjust the fitness value, and  $p_p$  means the profit of taxi  $v$ ,  $c_{i,fp}$  and  $c_{i,td}$  are the profit and the delay time for detour of passenger. The taxi chooses the passenger by the highest fitness value for taxi-sharing.

If the taxis want to pick up the same passenger for taxi-sharing at the same time, we can compare *the fitness value  $F$*  of the passenger among these taxis first and *the award value  $A$*  of the taxis if necessary. Once a taxi accepts a passenger for sharing, *the award value  $A$*  increases by 1. Besides, the higher award value a taxi has, the more opportunities to get a passenger for sharing it has.

### 3.5.5 Pricing Scheme

The section is considered that the payment shared by each passenger in the taxi fairly. The previous passengers need to afford the extra payment due to the detour. Therefore, the following passengers have to spend more money than the previous one on the distance with taxi-sharing. For the travel distance for taxi-sharing ( $d_s$ ), the payment of passenger  $c_1$ ,  $c_2$ ,  $c_3$  is proportioned to  $\alpha:\beta:\gamma$ , where  $0 < \alpha < \beta < \gamma$ . For example, there are two passengers  $c_1$  and  $c_2$  for sharing under the constraint  $(\alpha + \beta)r_3 + \alpha(r_4 + r_5) < (\alpha + \beta)r_2 + \beta r_6$  as the Fig 15.

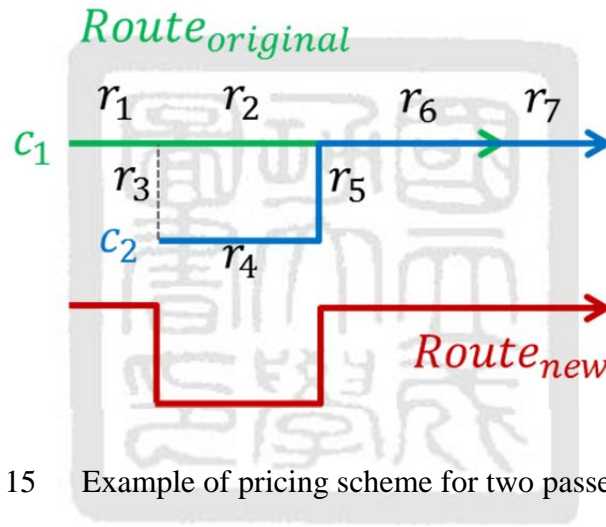


Fig 15 Example of pricing scheme for two passengers

The distance for sharing is  $d_s = r_4 + r_5 + r_6$ . The fare of passengers before sharing:

$$\begin{cases} c_{1.fb} = r_1 + r_2 + r_6 \\ c_{2.fb} = r_4 + r_5 + r_6 + r_7 \end{cases} \quad (3-14)$$

, and the fare of passengers after sharing:

$$\begin{cases} c_{1.fa} = r_1 + r_3 + \frac{\alpha}{\alpha+\beta} \times (r_4 + r_5 + r_6) \\ c_{2.fa} = \frac{\beta}{\alpha+\beta} \times (r_4 + r_5 + r_6) + r_7 \end{cases} \quad (3-15)$$

Thus, we can prove that

$$\begin{cases} c_{1.fa} < c_{1.fb} \\ c_{2.fa} < c_{2.fb} \end{cases} \quad (3-16)$$

Another example of pricing scheme for three passengers with taxi-sharing is presented as Fig 16. There are three passengers  $c_1, c_2, c_3$  for sharing under the constrain  $r_3 \ll r_4 + r_5 + r_6, r_7 \ll r_8 + r_9 + r_{11}$ .

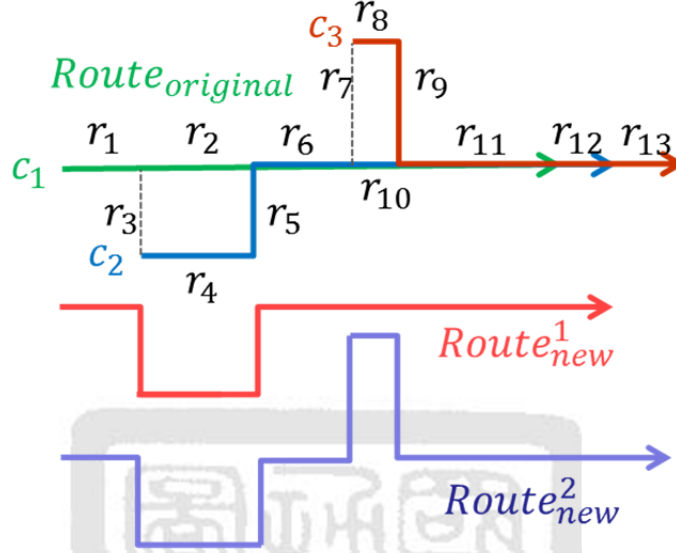


Fig 16 Example of pricing scheme for three passengers

The fare of passengers before sharing:

$$\begin{cases} c_{1.fb} = r_1 + r_2 + r_6 + r_{10} + r_{11} \\ c_{2.fb} = r_4 + r_5 + r_6 + r_{10} + r_{11} + r_{12} \\ c_{3.fb} = r_8 + r_9 + r_{11} + r_{12} + r_{13} \end{cases} \quad (3-17)$$

The fare of passengers after sharing:

$$\begin{cases} c_{1.fa} = r_1 + r_3 + \frac{\alpha}{\alpha+\beta} \times (r_4 + r_5 + r_6 + r_7) + \frac{\alpha}{\alpha+\beta+\gamma} \times (r_8 + r_9 + r_{11}) \\ c_{2.fa} = \frac{\beta}{\alpha+\beta} \times (r_4 + r_5 + r_6 + r_7) + \frac{\beta}{\alpha+\beta+\gamma} \times (r_8 + r_9 + r_{11}) + \frac{\beta}{\beta+\gamma} \times r_{12} \\ c_{3.fa} = \frac{\gamma}{\alpha+\beta+\gamma} \times (r_8 + r_9 + r_{11}) + \frac{\gamma}{\beta+\gamma} \times r_{12} + r_{13} \end{cases} \quad (3-18)$$

, so we can prove that

$$\begin{cases} c_{1.fa} < c_{1.fb} \\ c_{2.fa} < c_{2.fb} \\ c_{3.fa} < c_{3.fb} \end{cases} \quad (3-19)$$

## Chapter 4 Simulation and Result

### 4.1 Simulation Settings

#### 4.1.1 Dataset

We perform the experiment using the dataset of *T-Drive trajectory data sample* [13] [14]

- *Road networks*: the dataset using the road network of Beijing, which contains 106,579 road nodes and 141,380 road segments.
- *Trajectories*: the dataset contains the GPS trajectory record by over 10,357 taxis during the period of Feb. 2 to Feb. 8 in the year of 2010. The total distance of the data set is about 9 million kilometers and the number of points reaches to 15 million. After trip segmentation, there are in total 20 million trips, among which 46% are occupied trips and 54% are non-occupied trips.

Due to the high density of trajectories, we retrieve the center region of *T-Drive trajectory data sample* as the service region of experiment. Fig 17 visualizes the density distribution of the GPS points in the dataset and the service region.

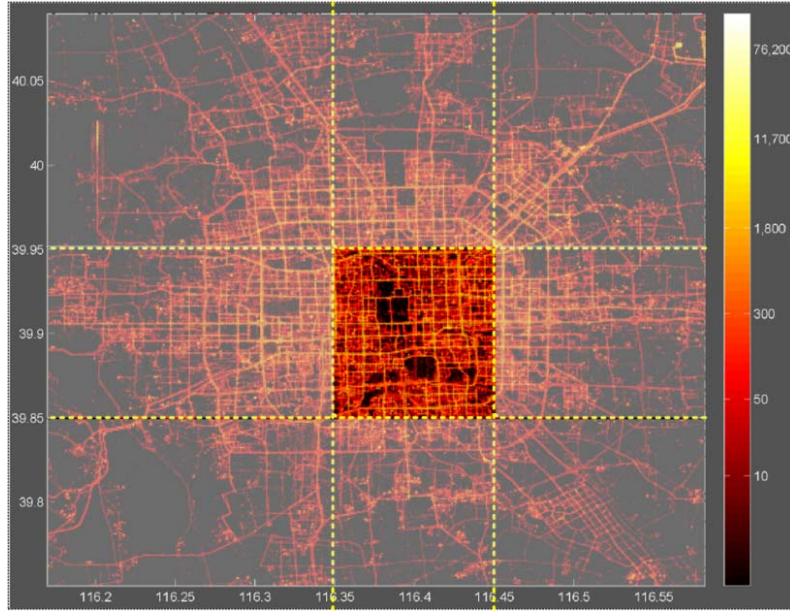


Fig 17 Service region of the experiment

## 4.2 Parameters

Table 7 Parameters of simulation

Parameters	Value	Parameters	Value
The number of Taxis	1376	$D_t$	20 (km)
Simulation area	11(km)×5(km)	$D_p$	30 (m)
Max entries of R-Tree	350 (m)	$\eta_1, \eta_2, \eta_3, \eta_4$	0.5, 0.2, 0.3, 0.7
Min entries of R-Tree	1500 (m)	$(w_1, w_2)$	1, 1
Speed limit	40 (km/hr)	$\alpha: \beta: \gamma$	1: 2: 3
Taxi pricing scheme	56 (NTD/km)	$T_w$	20 (min)
$I$	24	$F_e$	0 (NTD)
$\varepsilon$	1 (hr)	$P_e$	0 (NTD)
$D_s$	20 (m)		

### 4.2.1 Evaluation Factor

In previous work in [2] [15], the majority of experiment was evaluated by the average



waiting time. For taxis, the average waiting time shows how long a taxi driver would take on average for picking up a passenger. Likewise, the average waiting time for a passenger is the duration from entering the system to getting in a taxi.

The main goal of our work is finding taxis and passengers quickly using the recommendations. Thus, the waiting time of taxi and passenger is the first indicator of our study. In addition to the waiting time, the performance is related to the cost such as the payment of passengers. Besides, the increased revenue of taxis is considered the profit of taxis and another evaluation factor.

So we use five indicators to evaluate the performance of taxi-sharing recommendation mechanism:

- ◆ Average waiting time of taxis and passengers
  - The average waiting time (Y-axis) of taxis shows the time for picking up a passenger. Similarly, the waiting time of a passenger is the period from entering the system to getting in a taxi. In the study, a taxi can accept more than one passenger for ridesharing. Thus, a passenger does not have to wait for a vacant taxi so that we can reduce the waiting time. In our experiment, the longer taxis and passengers need to wait, the worse the result is.
- ◆ Average increased revenue of taxis
  - A taxi may get more revenue by providing the taxi-sharing service because of the extended travel distance. It is an incentive for a taxi to spend more time on picking up a taxi-sharing passenger. Each taxi can set the threshold of increased revenue which is one of the rules to determine whether a candidate passenger is appropriate. The more the increased revenue is, the more profit a taxi can get.
- ◆ Average payment saved of passengers
  - The main objective of taxi-sharing is reducing the payment of passengers. For saving money, the passengers may want to spend more time on detouring for

ridesharing with others. The more the saved payment is, the more passengers are willing to rideshare.

◆ Average service time of passengers

- The service time shows the duration of the passenger on board. The ridesharing may reduce the payment of passengers but increase the service time because of detouring. The incurred additional time for picking up another passenger is considered as the cost, and the payment saved is the profit in contrast.

◆ The number of taxis is needed

- For the purpose of saving energy and reducing pollution, collective transport is considered as a solution to reduce the number of car on the road surface. The main goal of taxi-sharing is that a small number of taxis can fulfill all the service demand from passengers because the taxis can provide more than one seat in a trip.

#### 4.2.2 Comparing the traditional algorithms with our work

We compare our recommendation mechanism with the traditional algorithm that finds taxis and passengers in random way. Random algorithm is the simplest method in which the taxis drive with the mobility of random walk, and pick up passengers while finding them. In the traditional algorithm, a taxi does not provide the taxi-sharing service. Thus, each passenger needs to find a vacant taxi to take.

### 4.3 Performance Evaluation

In our study, a taxi needs the recommendation to find the first passenger at the beginning. Then the taxis with one passenger in the car keep searching the proper passengers for taxi-sharing. A taxi only can pick up one passenger at a time.

After picking up passengers, the taxi decides the route based on the destinations of all passengers. Adopting the arbitrary shortest path algorithm as the driving policy makes the

decision. The distance and travel time of the route can be computed by using the tables constructed based on time-dependent R-Trees previously.

In this section, we present the main simulation results using the metrics defined in 4.2.1 to evaluate the performance of our proposed recommendation mechanism for taxi-sharing. The simulation environment is configured with the settings provided in 4.1.

We compare our work with the traditional method, which finds passengers and taxis without recommendation. The taxis in traditional method without recommendation cruise randomly and only pick one passenger in a trip. We simulated the method with and without recommendation from different aspects.

Following graphs show the comparison of the average waiting time and the number of taxis needed for both methods. Besides, the results present the average saved payment, increased revenue and service time. We fixed the number of taxis in the system environment, and conducted the experiment with the varying number of passengers.

For computing the payment of passengers, the charge of taxi fare refers to the pricing principle provided from Taipei City Public Transportation Office [12]. The taxi fare is NT\$ 70 per kilometer in the beginning of journey. In the thesis, we calculate the fare by NT\$ 56 per kilometer for convenience which is presented in 4.2.

Therefore, the result data generated by our simulation were collected and depicted into diagrams by the drawing tool MATLAB. Those graphs are discussed in the following sections later.

#### 4.3.1 The observation of ‘Average Waiting Time of Taxis and Passengers’

Fig 18 shows the average waiting time of taxis with and without recommendation respectively. Each point represents the waiting time on average during 24 hours. With the fixed number of taxis, the average number of taxis in our environment is 1376. X-axis is

the average number of passengers. After the number of passengers increased to 1200, the waiting time of taxis with recommendation diminished sharply, and was almost one half of the waiting time of taxis without recommendation. Our work can provide 25% waiting time reduction on average. The overall average waiting time of taxis is shown in Table 8.

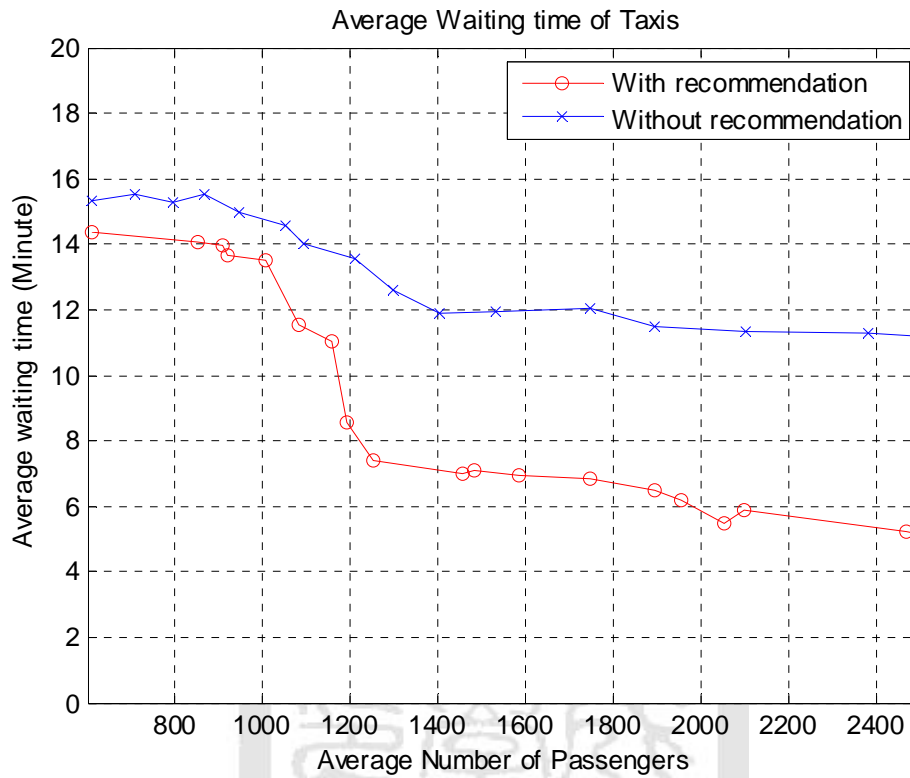


Fig 18 Average waiting time of taxis

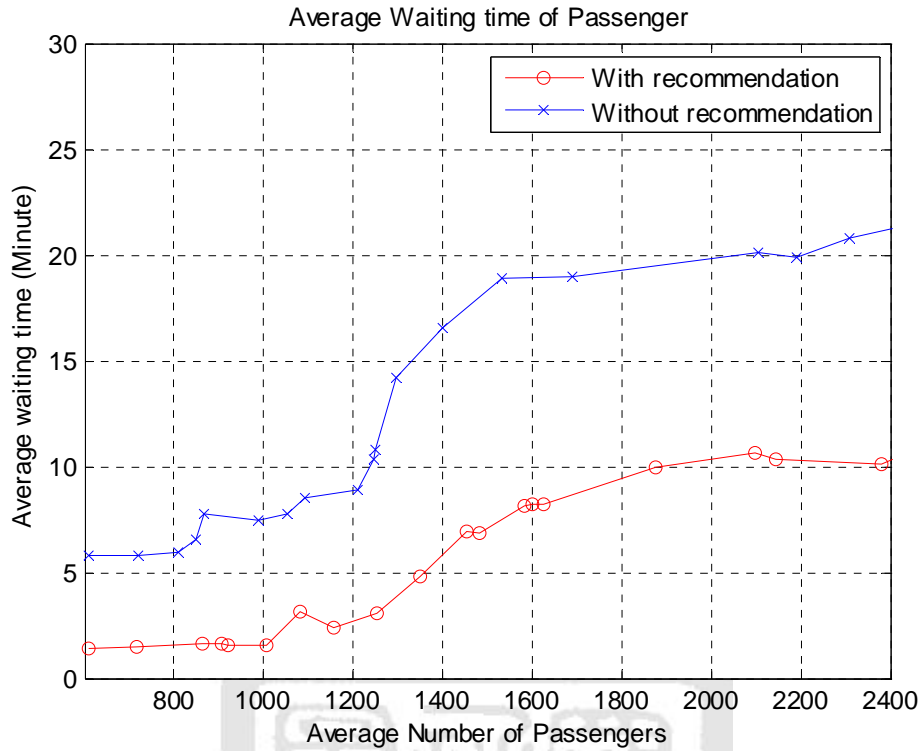


Fig 19 Average waiting time of passengers

Table 8 Overall average waiting time of taxis and passengers

Average Waiting Time	Taxi	Passenger
Without Recommendation	13.06372142 (Min)	12.86021784 (Min)
With Recommendation	9.836891733 (Min)	4.851695442 (Min)

Further, the waiting time of passengers is presented in Fig 19. The difference of waiting time between two methods is quite obvious after the number of passengers increased to 1200, which was consistent with the result of taxi waiting time. The passengers with recommendation had shorter waiting time than the ones without recommendation, because they could rideshare rather than wait for a vacant taxi. The proposed method (with recommendation) had better performance due to 62 percent of waiting time reduced for passengers on average according to the result in Table 8.

### 4.3.2 The observation of ‘Average Increased Revenue of Taxis’

In our study, a taxi can improve their profit by providing taxi-sharing service. The following on-board passengers contribute to the increased revenue by extending the travel distance of a taxi. Thus, taxi-sharing benefits taxi from earning more money.

Fig 20 depicts the average increased revenue of taxis by applying our recommendation mechanism. Along the ascending number of passengers, the increased revenue stayed stable in the range between NT\$30 and NT\$35. The overall average increased revenue of taxis is NT\$32.74. The comparison with the overall average payment saved is shown in Table 9.

For different on-board passengers in the taxis, the average increased revenue is illustrated in Fig 21. As the result, the more passengers the taxi picked up, the more profit they got. The taxis with 3 on-board passengers have more increased revenue than the ones with 2 on-board passengers, which corresponds to a NT\$29 increase approximately.

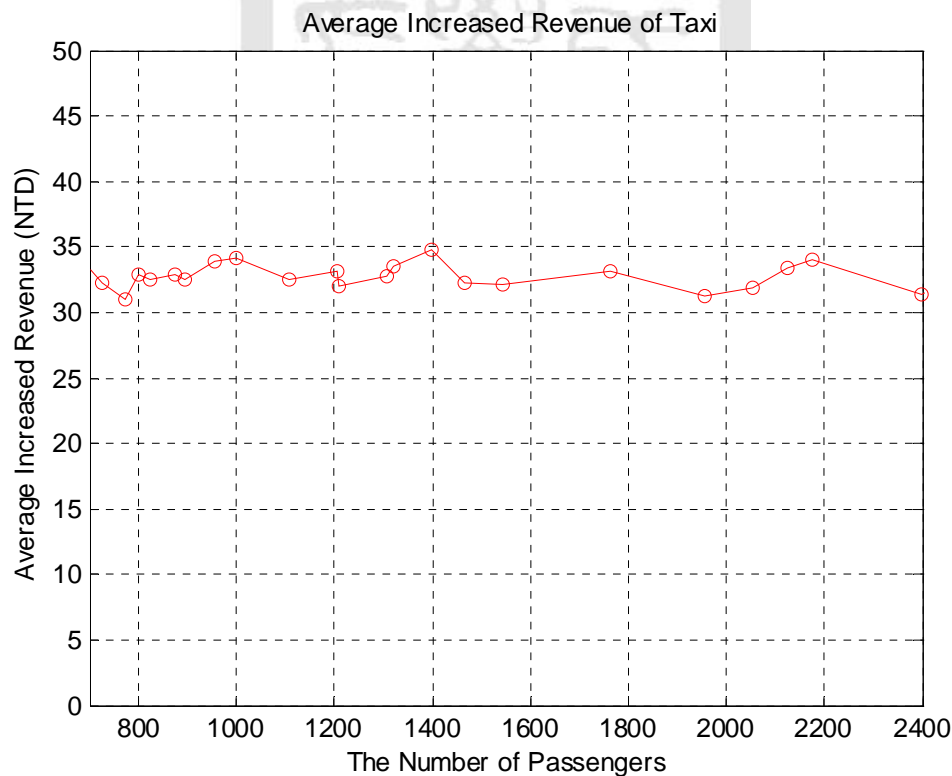


Fig 20 Average increased revenue of taxis

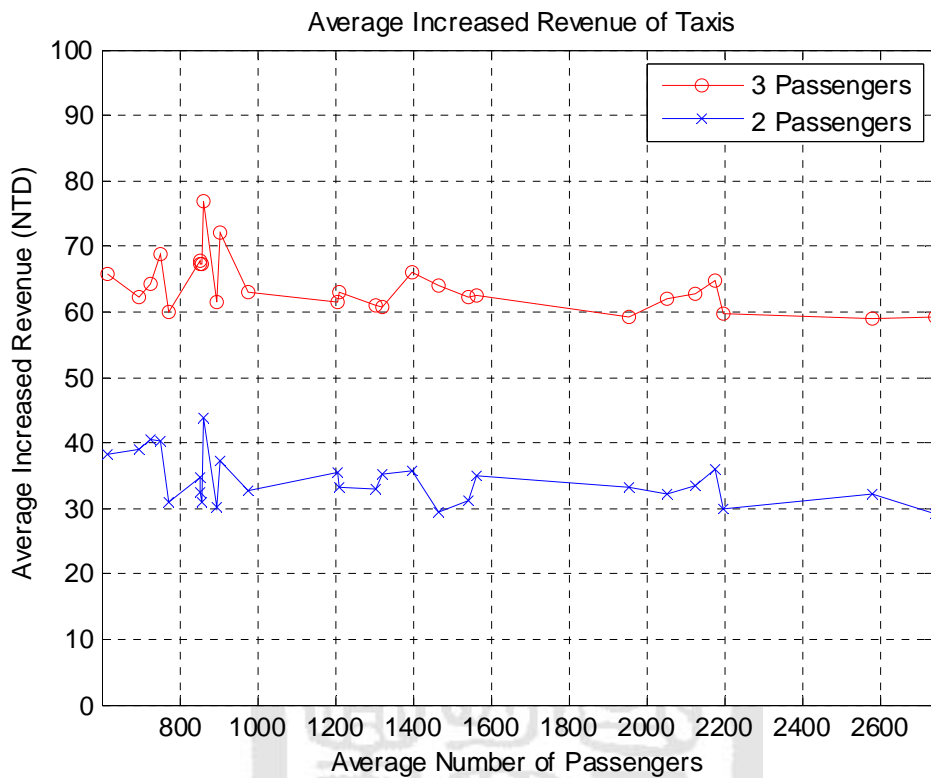


Fig 21 Average increased revenue of taxis for different number of on-board passengers

Table 9 Overall average increased revenue of taxis for different number of on-board passengers

The Number of On-board Passengers	Average Increased Revenue of Taxis
3 Passengers	63.85 (NTD)
2 Passengers	34.24 (NTD)

#### 4.3.3 The observation of ‘Average Payment Saved of Passengers’

In addition to the increased revenue of taxis, the other principal purpose in our work is reducing the payment of passengers. The result with the proposed method indicated a significant effort for saving the payment of passengers as shown in Fig 22. After the number of passengers was up to 1200, the average payment saved raised to almost NT\$130.

Taxi-sharing behaviors had contributed to the climb of the curve due to the share of the fare. Besides, Table 10 reveals that the average payment saved of passengers was greater than the average increased revenue of taxis, because the taxis picked up the ridesharing passengers based on the shortest path algorithm.

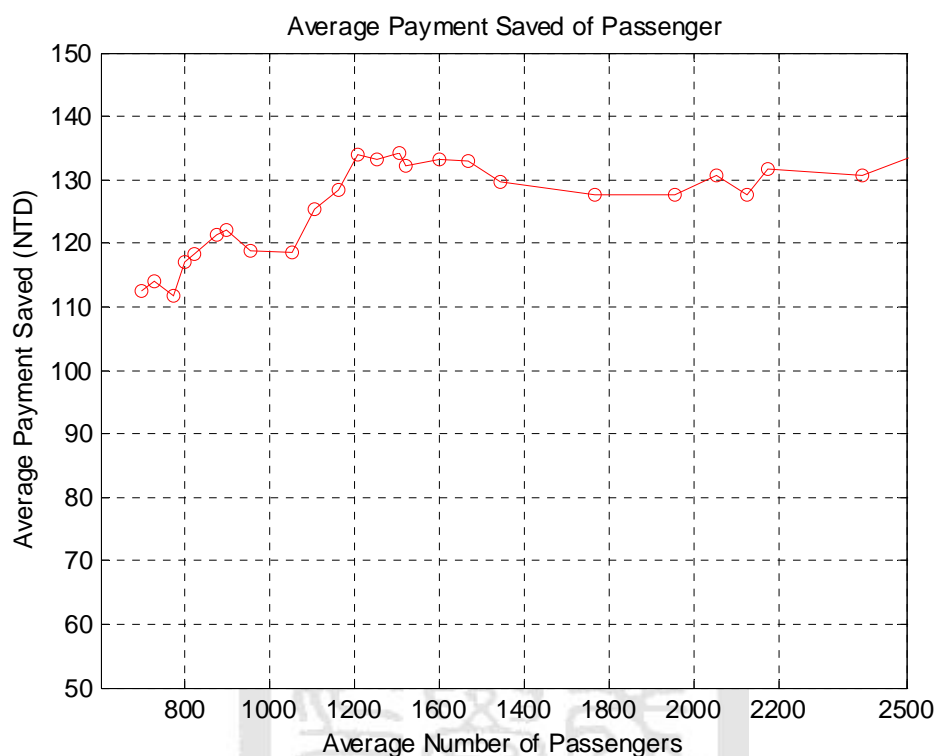


Fig 22 Average payment saved of passengers

Table 10 Comparison between overall average increased revenue of taxis and payment saved of passengers

Average Increased Revenue of Taxis	Average Payment Saved of Passengers
32.73596 (NTD)	126.1997 (NTD)

#### 4.3.4 The observation of ‘Average Service Time of Passengers’

Fig 23 shows the average service time of passengers in the conditions with and without recommendation respectively. The service time of passengers with



recommendation is longer than the one without recommendation because of the additional time spending on detour for taxi-sharing. The incurred additional time for detour is about two minutes on average (see Table 11). However, taxi-sharing could benefit passengers for saving payment as displayed in section 4.3.3. As the result, the passengers just needed to wait a little time for taxi-sharing so that they could get more profit by saving their payment.

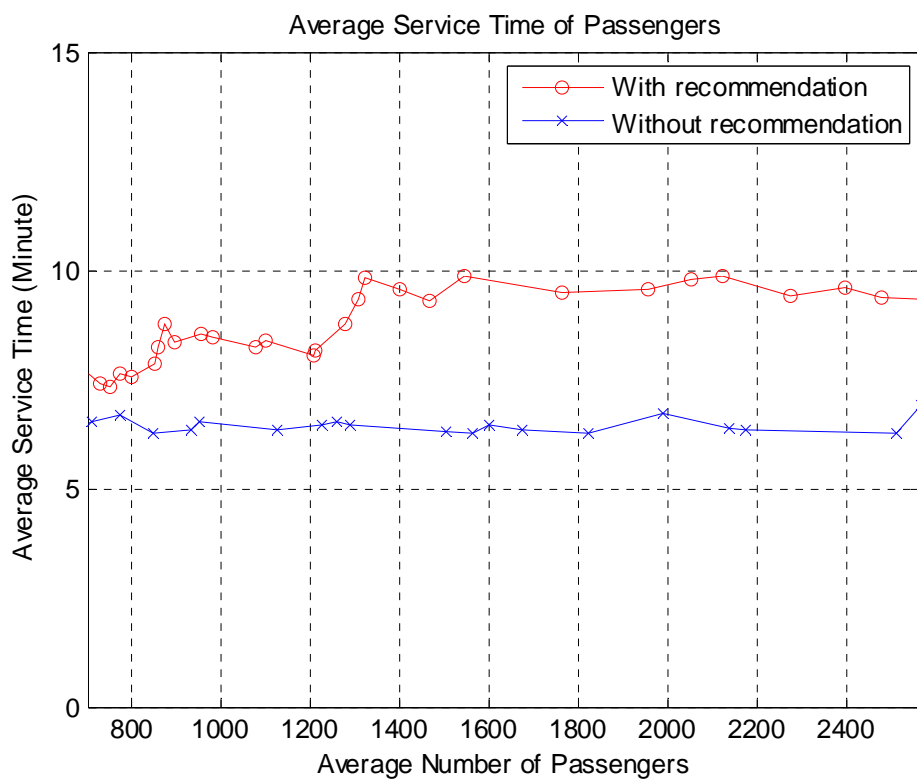


Fig 23 Average service time of passengers

Table 11 Overall average service time of passengers

Average Service Time of Passengers	Service Time
Without Recommendation	6.43304637 (Min)
With Recommendation	8.707401777 (Min)

#### 4.3.5 The observation of ‘Required Average Number of Taxis’

Fig 24 presents the required average number of taxis. As Fig 24 shows, the curve with recommendation had less number of taxis than the one with recommendation. In the traditional method, a taxi can only pick up one passenger in a trip. However, ridesharing can make the passengers share a taxi for saving payment. Along with the increase of taxi-sharing passengers, the difference of the required taxis between with and without recommendation becomes larger. The result indicates that a smaller number of taxis can fulfill the service demands for passengers by taxi-sharing.

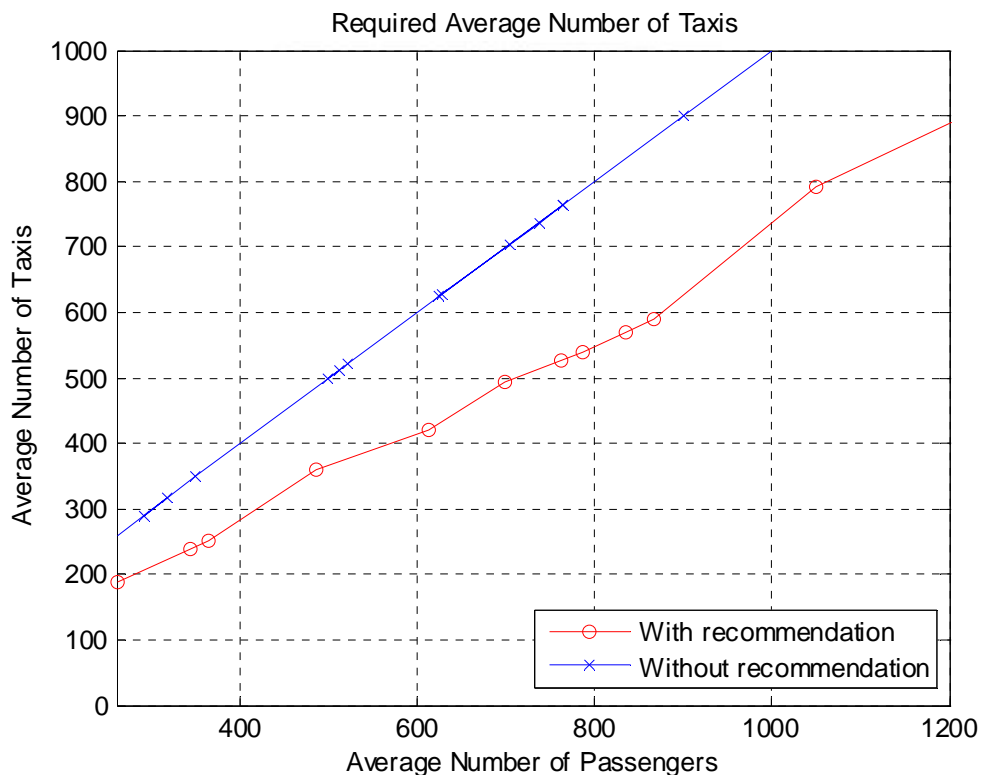


Fig 24 Required Average Number of Taxis

## Chapter 5 Conclusions and Future Work

In this thesis, we propose a taxi-sharing recommendation mechanism for both taxis and passengers, which combines non-cooperative game model to solve the competition among the taxis in need of route recommendations. Based on the historical information of taxis and passengers, we build time-dependent R-Trees to figure out popular locations so that the server can suggest the routes and locations by the R-Trees.

The first step is analyzing the real trajectories according to the dataset in [13] [14], then the R-Trees and the corresponding travel time/distance tables are constructed for each time slot offline. For evaluating the performance of our model, we implement the work using the traffic simulator known as SUMO and TraCI. Furthermore, we perform the experiment with different metrics. The graphs of experiment results are presented in 4.3.

Several research questions were addressed in the study, and the principal findings suggested that (1) the waiting time with our recommendation mechanism could be reduced for both taxis and passengers; (2) the taxis providing ridesharing service had more profit due to the increased travel distance; (3) the passengers got the reduction of payment by the proposed method for taxi-sharing; (4) the study proved that using less taxis could fulfill the service demand from passengers. Our mechanism is objectively considered a good method for not only finding ridesharing passengers and taxis but benefiting the traffic in the city.

We are still in the process refining the time-dependent R-Trees and parameters in the proposed model. After the simulation, we believe that there are a lot of issues can be considered such as the R-Tree with map-matching, the behavior of taxi drivers and the correlation between taxi-sharing passengers. It may be beneficial for the model to take those factors into consideration in the future.

## REFERENCE

- [1] K. D. and R. C., SUMO (Simulation of Urban MObility), German Aerospace Centre, 2007. <http://sumo.sourceforge.net/>
- [2] Pedro M. d'Orey, Ricardo Fernandes and Michel Ferreira. Empirical Evaluation of Dynamic and Distribution Taxi-Sharing System. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2012.
- [3] Antonin Guttman. R-trees a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD '84*, page 47, 1984
- [4] Morgenstern, Oskar and John von Neumann (1947) *The Theory of Games and Economic Behavior* Princeton University Press
- [5] Roger B. Myerson (1991). *Game Theory: Analysis of Conflict*, Harvard University Press, p. 1. Chapter-preview links, pp. vii-xi.
- [6] MOVE (MObility model generator for VEhicular networks): Rapid Generation of Realistic Simulation for VANET., 2007.  
<http://lens1.csie.ncku.edu.tw/MOVE/index.htm>
- [7] Shuo Ma, Yu Zheng, and Ouri Wolfson. T-Share: A Large-Scale Dynamic Taxi Ridesharing Service. In *IEEE International Conference on Data Engineering*, 2013.
- [8] Dusit Niyato and Ekram Hossain. Competitive Spectrum Sharing in Cognitive Radio Networks: A Dynamic Game Approach. In *IEEE Transaction on Wireless Communication*, 2008.
- [9] Quality Carpool Service ,<http://www.qcar.org.tw/>
- [10] Aumann, Robert J. (1987), *game theory*, The New Palgrave: A Dictionary of Economics 2, pp. 460–82.

- [11] TraCI (**Traffic Control Interface**)  
<http://sourceforge.net/apps/mediawiki/sumo/?title=TraCI>
- [12] Taipei City Public Transportation Office, Taxi Pricing System  
<http://www.pto.taipei.gov.tw/ct.asp?xItem=1199267&ctNode=12602&mp=117041>  
<http://english.dot.taipei.gov.tw/ct.asp?xItem=53996843&ctNode=65636&mp=117002>
- [13] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 99-108, New York, NY, USA, 2010. ACM.
- [14] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *The 17th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, KDD'11, New York, NY, USA, 2011. ACM.
- [15] Nicholas Jing Yuan, Yu Zheng, Liuhang Zhang, and Xing Xie. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. In *IEEE Transactions on Knowledge and Data Engineering*, 2013