The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Simulation System Design and An Analysis of The Role of Geographical Location in the Sharing Economy

by

Nguyen Thanh Cong

Dissertation Presented for the Degree of
Master of Science in Operational Research (with Risk)

September 2021

Supervised by
Dr. Burak Büke

This work is a tribute to my late uncle.

Abstract

Ride-hailing services have been on a rapid rise for the past ten years. One of the major key component to the success of their operations is the matching algorithm. This work aims to explore how such matching algorithms can lead to an increase in operations efficiency through simulation design and statistical analysis. The result of our work have shown that the designed matching algorithm is sensitive to changes in factors such as customer arrival rate, taxi arrival rate and etc. Additionally, different matching protocol can play a key role in improving system efficiency. Furthermore, we have found that our system performance can be affected not just by the input parameters but also by the matching protocol. In a realistic setting, if we consider the input parameters as external factors where we have no control over then the result of this dissertation implies that the designed matching algorithm is not just sensitive to changes in external factors but also changes in its inner workings.

Acknowledgments

I want to dedicate this page to express my great appreciations to the people who have helped me make this work possible. Firstly, I would like to express my gratitude to my project supervisor Dr. Burak Büke and my academic advisor Dr. Miguel Anjos. It would not have been possible had they not providing all the necessary supports as well as encouragements. Secondly, I want to give thanks to members of my family especially my mother Nguyen Thi Hong and my sister Nguyen Thai Nhu Quynh (Lea) for always being there and believing in me. Finally I give my thanks to all the great friends that have made my adventure in Edinburgh wholesome: Daniel Adams, Lucas Scally and Anuradha Mukherjee.

Own Work Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

Contents

List of Tables

List of Figures

1 Introduction

With the advancement in the internet and chip technology during the last decade, the popularity of smartphones began to rise in 2012. Alongside its risen popularity, there was also a rise in smartphone applications (apps). These are the utility applications made by either the smartphone providers or individual groups. Individuals and corporations pour time and money into developing these so called apps. The focus on its development and its usage have brought people from the around globe together to form many online based communities. Individuals who participate in these online communities are allowed to trade or exchange goods and services. The participation in such trade or exchange is defined as the sharing economy. This type of engagement lowers the barriers of entries for many service providers because it allows service owners to easily seek and find those who wants to use their services. Additionally, the existence of the sharing economy maximizes the efficiency of many resources in the economy. For instance, under used resources such as empty bedrooms can now be rented out on AirBnB to increase resource efficiency.

Traditional taxi services have existed for a long time. In the age before the internet, their business models have been effective in serving consumers. However, such businesses would require a massive amount of upfront investment into the purchase of vehicles, hiring drivers, hiring employees for a central call center that can dispatch taxis, not to mentions the safety regulations and labour regulations which such business must adhere to. The existence of the internet and the sharing economy have undoubtedly made the business model of traditional taxis obsolete and redundant. The Ethernet has managed to remove the upfront investment costs by acting as a facilitator, like a call center that can deliver the matches between drivers and riders. As mentioned above, the sharing economy eases the movement and the transaction of goods and services across the globe. But for something similar to a traditional taxi service provider to exist in the online environment, there still a need for a central managing body. Even though the internet platform is there, the online community can not ensure quality of services and consumer's safety. Without safety and quality, the online taxi business model can not function. Hence, groups and company like Uber, Lyft, Grab,... have come together to develop a so called ride-hailing app. With the push of a button, these apps give the user control over when and where they want to travel. It also provides information about drivers and fares price before the customer commits. Given options, transparency and information, it provides consumers a sense of trust which no general online community can provide. Therefore, the existence of ride-hailing service providers are necessary and inevitable.

Ride-hailing (or ride-sourcing) (RH) is defined as an online-based platform that utilize information technology to allow users to hire a driver.The platform is operated and maintained by a ride-sourcing company. Companies such as: Uber, Lyft and Grab act as an intermediary that can connect demand (i.e. customers) and supply (i.e. car owner). Upon entering the platform (i.e an app) through a smart device, customers can request a ride by filling out some crucial details including trip origin, trip destination and departure time. Having done that, the app will show the customer a list of offered services at a different price point. After choosing the service, idle taxi drivers (i.e. car owners) who are registered on the platform will then be matched to the corresponding customer by a matching and dispatching algorithm. The algorithm will show the location of the matched driver and an Estimated Arrival Time (ETA). Once the service is completed, the customer will pay a fare to the platform and the platform will split the fare with the driver.

The rapid growth of ride-hailing platforms over the last decade has increased the scale of operations as well as the amount of competition in the market. This gives rise to the need to develop an algorithm that can assign and deploy drivers to customers efficiently. This is dispatching algorithms. The motivation of this dissertation is to investigate how different dispatching protocol may yield different level of efficiencies and we will also investigate the implication of those differences when it comes to designing a complex dispatching or matching algorithm.

In this work, we will often interchange between driver/taxi and riders/customers/passengers.

The lay out of the dissertation is as followed. In Chapter 1, we will discuss about the impacts of the ride hailing businesses (RHB), its general framework, make a comparison between the RHB and the conventional taxi system, and close the chapter with the importance of efficiency. In Chapter 2, we review some of the research which are related to our dissertation. Chapter 3 analyzes the ride-hailing system as a queuing model and concludes by making a distinction between the static system and the spatial system. Chapter 4 explains the need and importance of computer simulation, we clarify and explain our methods for generating random numbers and concludes by explaining the operations of our simulation. In Chapter 5, we present some of the performance measures which will be used in later sections. Chapter 6 explains the different types of models we can create from the original model (ED), estimates the confidence interval estimator, and we conclude with convergence analysis. Chapter 7 compares different types of models by using statistical tools. Finally, in Chapter 8 we conclude and discuss some of the implications of our findings.

We want to note that even though some of the assumptions may have already been stated in one section. We still believe that it is necessary to repeat them in some other section because of the way this work is structured. Therefore, it is not a mistake, in case the reader thinks there is a repeat.

1.1 The impacts of the ride-hailing businesses (RHB)

Due to the sheer scope and scale of the researches, we will mainly focus on the US economy.

The need for travel have existed in the human society for a long time. In the past, most people travelled out of necessity for safety or to seek new opportunities and explore new region. With the advancement of automobile, train and planes, people of today have managed to cross vast distances in a short time. Whatever the reason for travelling in today's world might be, the demand for travel and transport is as high as ever and it is expected to increase as the world population increases and becomes more integrated by the internet. At the push of a button, anyone can travel from anywhere to everywhere.

In today's world, the demand for transport can have wide varieties. It could be a business employee requesting a ride to the next meeting, or a visitor taking a ride to one of the popular tourist attractions. Nonetheless, the necessity for travel remains relevant in today's world. As long as there is demand, there will be supply. Attributed to the advancement of computer technology, firms have managed to provide services that is convenient and efficient. The key importance is convenience. It is due to convenience that the sharing economy prospers. It is also due to this factor that the prominence of the RHB such as Uber, Lyft and Grab,... is inevitable.

UBER impact on the economy and employment. Founded in 2009 with the answer to the question: "what if you could request a ride from your phone", Uber have grown to become the dominant force in the global ride-hailing market. Started mainly in the US, Uber is now operated in more than 65 countries and 600 cities worldwide. With global valuation at \$72 billion and a market share of 37.2%. Uber is directly and indirectly the facilitor for the creation of other ride-hailing firms such as Lyft, Grab and Didi. As a whole, these types of businesses is classified as transportation network companies (TNC) or ride-hailing business (RHB).

Over the year, Uber has undoubtedly made a lot of contribution to the US economy. Such contributions isn't just affecting one business area but it creates a ripple effect throughout the economy. As detailed in the 2018 study by the Economic Development Research Group (EDRG), Uber have contributed to the GDP of US economy with an estimate of \$17 billion in 2017 [4]. In the study, EDRG measured the economic impact base on 2 criteria: namely the Gross impact defined as the contribution of Uber to the U.S. economy and the Net impact defined as changes in the national income (or economic growth). Gross impact can be viewed as the direct and indirect total incomes generated for Uber's and for other workers in different industry. The impact is created when said Uber's drivers spend their income's on things such as foods or rent. The service workers in the foods and rent industry then spend a portion of their income (collected from the spending of the Uber's driver) on things such as electricity and travelling and this create a chain of effects that is best reflected in (Figure 1). In contrast, Net impact can be observed through the loss and gains of other businesses and inflow from tourism spending.

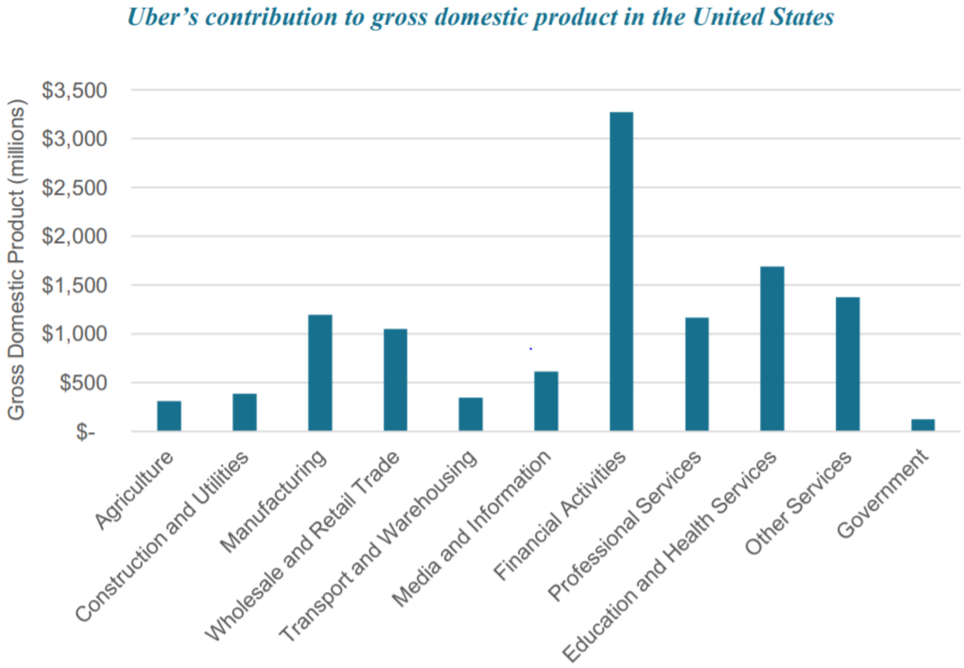


Figure 1: Uber's Gross impact on the US's industries [4]

Aside from economic impact, Uber has also changed the way in which drivers work and the way riders choose to travel. Before the arrival of the ride-hailing businesses, most drivers either has to work for a conventional taxicab businesses with a centralised dispatching servers or they choose to drive for private individuals. The way in which the conventional taxi business work will be detailed in (Section 1.3), but needless to say their operations is cumbersome and it was relatively challenging for all parties involved. Same as drivers, riders didn't have much choice back then, so consumers either have to either choose form of public transport which lower amenity and reduce productivity. In today's world, Uber's drivers benefit greatly from the flexibility provided by the firm. Drivers can start and finish work at the the push of a button whenever and wherever they want. It has been documented that 63% of Uber's drivers in the US drive for less than 20 hours per week. Interestingly enough, study [31] have found that whether an Uber's driver work longer hour than an average Uber's driver make no difference in their earnings level. This of course have factored in the surge pricing strategy which Uber have been using to increase the level of supply within the market when there is high demand. Surge pricing work by increase the fares price of a trip during peak period with low supply of drivers. This in turn motivate off-work drivers to start working and thus increase the number of supply [13]. Surge pricing also work both ways, while it can manage to increase the rapid flow of drivers into the system, it could also restrict the total demand during the surge period. Specifically, those who are willing to pay for the surge price will do so because they deem it justifiable while the others who think otherwise will either choose a different mode of transport or they wait until the surge period have passed [33]. Another benefit which has been documented for being a Uber's drivers is that the company assist new and potentials drivers in the purchases of new cars at a very low interest rate provided that the car is used to drive for the Uber network while the duration of the loan persist [45]. From the perspective of riders the benefits for riding with Uber include: convenience, costs and time. As well documented in the study [4], the monetary values of these benefits are a reduction in roughly $1.68 in fares expenses and a reduction of $5.42 in time savings.

Social impact. Aside from the impact of Uber alone, the RHB as a whole also contribute greatly to the societal changes. If RHB is considered to be a substitute for public transport, then its growth have provided more options for urban travellers. For big city with good area coverage, the study have found that RHB such as Uber compete with other modes of transport during the day, but at night it facilitates and even help boost the use of other modes of transport [34]. This effect is justifiable, because during the day, the number of people who use public transit is much greater compare to at night and that reduce the overall amenity of the ride. Therefore, if better options are available given the price between the two options don't differ by much, then riders would choose the ride-hailing services. At night time, in area where one mode public transit might not provide great coverage but other mode of transit still serves as an viable option for long distance (for instance a train ride from Edinburgh to London is cheaper compare to travelling with Uber.), then ride-hailing provide complementary support by taking riders from their place of origin to the train station or to an airport. Additionally for night time travels, RHB provide a reliable and safe options for travellers. Ride-hailing apps provide riders with information about their drivers and vice versa. With anonymity factor removed, the opportunity for committing crime reduce drastically on the side of both drivers and riders. Moreover, the identity provision from the apps gives both the riders and drivers a form of control and this can guarantee some level of safety and build trust. Another societal benefit brought about by the RHB is the reduction in traffic accident cause by drunk drivings. Study have found that as the number of Uber drivers increase, the number of fatalities in youth cause by drunk driving decrease [48, 21]. Moreover, study have also found an increase in the number of ride hailing trips (not just Uber trips) can cause a reduction in the road crashes and fatalities in general [38].

In contrast all the above benefits, the distribution of the RHB in remote areas or low density area has been somewhat lacking. This raise the the questions on whether RHB is here to provide niche services to rural areas or they are here to replace the conventional taxi businesses. Few studies [11, 27] have pointed out some pattern with discrimination against riders that is of a certain ethnic group. Namely passengers that is of a certain race with a certain name tend have to wait longer and get more trips cancelled by drivers. Other social issues include drivers refusing to accept requests. In the study [53], researchers have found that approximately 40% of ride-hailing request in Beijing receive no respond unless driver's incentives is met. In this case the incentives are trip that cover a large distance. The findings of this research provide information into how the ride-hailing businesses may improve their operations. Furthermore, in contrast to previous findings [48, 21], study [10] have shown that the entry of Uber into the metropolitan area have no effects on the number of traffic fatalities.

Environmental impact. Along with social impact, RHB also cause changes in the environment. The efficiencies of the RHB compared to the conventional taxi is well explained in section. Such efficiencies is important in the face of environmental impact. Hypothetically, ride-hailing help facilitate the use of under use resources within the economy. The higher the utilization of some resources, the less the needs for the creations of a new resources and the need to use other readily available resources. For instances, if car owners choose ride-hailing over the use of their private vehicle, then this reduce the total number of vehicle on the road by one unit. The more cars owners choose ride-hailing services over their own vehicle in any one particular period, the less the total number cars that "exist" during such period. The longer the duration of such period, the more permanent the effect and eventually it can potentially cause a reduction in congestion. As the number of cars ownership reduces and less cars are running on the roads, pollution plummet and public spaces that was being used primarily for parking's can now be used for other purposes. Indeed, study [32] have shown that ride-hailing along side with factor such as parking stress could potentially reduces parking demand and potentially free up parking spaces for other uses. Also, as documented in the study [54], the operations of ride-hailing can cause a reduction in CO_2 and NOx emission as long as ride hailing vehicle's utilization is high and pick up time is short. However, hypothesis and assumption don't often coincide with reality. Instead of reducing congestion, a study in the city of San Francisco from 2010 to 2016 have reported that the congestion level have increased significantly as a result of the ride-hailing services [2]. Additionally, the study [8] in the Paris region have determined that ride-haling services alone is not the cause of car ownership reduction in household that has already has car but rather it is one of the major contributing factor in the reduction of new car purchases.

Impact on existing competition and potential new competitions. Since the day of its inception, the RHB such as Uber has proven to be a disruptive forces. RHB compete directly with the conventional taxi services and over the year they have been accused of unfair competition due to the fact that RHB operate very differently compared to the conventional taxi (More details on this will be explore in (Section 1.3). Other than competitions, the RHB has caused a shift in driver's income. Some may argue that this shift benefited drivers as a whole because they now earn more while working less hour. However, this does not factor in the fact that RHB do not consider their drivers as employees. As a company employee, individuals are entitled to a certain benefits, compensation and protection by the law. However if Uber's drivers are not considered employees, this causes tension as they then are not entitled to worker's rights and company benefits. Hence, when an accident occurs during work, drivers may not be compensated adequately due to lack of worker's rights.

Since ride-hailing businesses aren't running a labour intensive operations similar to a call centers, or the conventional taxi services. They have to improve their operations efficiencies and minimize cost in a different manner. Namely by improving the matching algorithm, businesses like Uber can increase the total number of time customers spend in a cars. Note that just because customers spend longer time travelling in a vehicle does not mean it will increase the total average number of trips completed. This could depend on the destination of the customers. But as a whole, if all customers trips averaging to be 3km, then on average the more time the customers spend travelling inside a vehicle, the more likely the number of trip completed will also be high. The recent impact of COVID 19 have also pushed the ride-hailing businesses towards the adoption of the work-from-home model. With this implementation, ride-hailing employees aren't required to be present at the office and since no office spaces is needed, operations costs get reduced while operation efficiency is still maintained.

Despite some of its disadvantages, the RHB have undoubtedly bring about positive changes to the economy and society. Society might have been better with or without it but that is not a question we intend to answer in this work. One thing for certain is that as society progress become more technologically advance, the arrival of the ride-hailing services is inevitable. With this we conclude the section on the impact of the RHB.

1.2 Ride-hailing platform and the general framework:

1.2.1 Overview of ride-hailing operations:

A ride-hailing system comprises of the RHB/ platform maintainers/modelers, taxi drivers (i.e. servers) and riders/customers. However, for ease of explanation, we will only focus on the interaction between drivers and customers. Analogous to previous literature, we will affix a gender role for both drivers and customers for ease of reference. Drivers will be classified as female and customers will be classified as male.

(Figure 2) is a graphical description of the interactions between different variables within the ride-hailing system. The first rectangle on the very top shows the different objectives a RHB might choose to prioritise. The objective can be short or long term depending on the business. However, it is important for businesses to be able to switch objectives depending on the circumstance. When the objective has been decided, a set of strategies will be employed that can assist the business in reaching that objective optimally. The strategy used depends on the side which the RHB want to target. If the target is on the demand side, then RHB could utilize surge pricing, promotions and bonuses to increase the number of trip completions and service adoptions rate. If the target is on the supply side, then RHB may want to employ strategies that in general could control the flow of supply. Strategy such as surge pricing have proven to be effective when there is peak demand (as explained in Section 1.1). Alternatively, if the RHB is concerned with other objectives such as system efficiency then strategies that prioritise improvement in the dispatching methods and repositioning may be used.

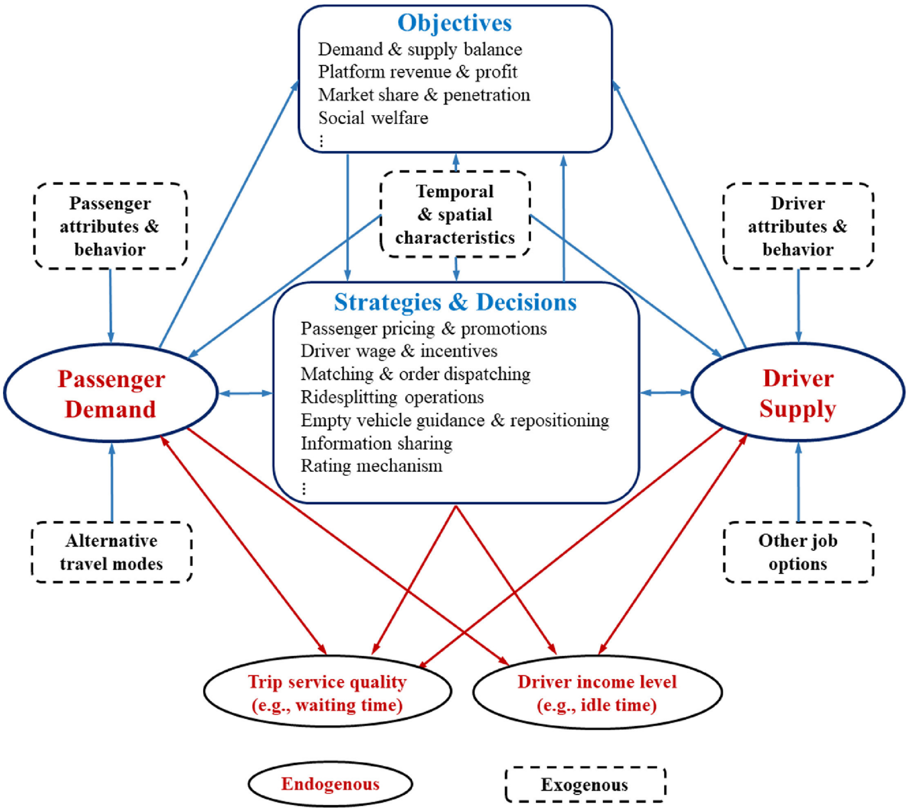

Objectives
Demand & supply balance
Platform revenue & profit
Market share & penetration
Social welfare
⋮
Passenger attributes & behavior
Temporal & spatial characteristics
Driver attributes & behavior
Strategies & Decisions
Passenger pricing & promotions
Driver wage & incentives
Matching & order dispatching
Ridesplitting operations
Empty vehicle guidance & repositioning
Information sharing
Rating mechanism
⋮
Passenger Demand
Driver Supply
Alternative travel modes
Other job options
Trip service quality (e.g., waiting time)
Driver income level (e.g., idle time)
Endogenous
Exogenous

Figure 2: The general framework [47]

Two vital components of the ride-hailing system are the Passenger Demand (PD) and the Driver Supply (DS). Without these two, the ride-hailing system would not be possible because the RHB are not capable of producing the supply themselves. On the other hand, if there is no demand, then it does not matter how efficient and well priced the supplies are, it does not return the benefits back to the RHB. The supply is almost always guaranteed due to monetary incentives but the demand is influenced by other factors not just from monetary incentives. If we consider the PD alone, then PD is affected by factors such as passenger's patience and other cheap alternatives for travel. Additionally, PD can also be influenced by trip service quality through system ratings.Trip service quality is measured by a certain level of passenger wait time and pick-up time. If a RHB has consistently performed poorly then demand will shift to other options. However, PD can also influence trip service quality. If there are too many customers in the system then waiting time increases and so does pick-up time. The lower the wait time and pick-up time, the higher the trip service quality and therefore system ratings. From the DS side, the DS is influenced by driver behaviour. Driver behaviour is influenced mainly by incentive such as income. Other alternative factors such as job security, comfort and job pressures can also affect driver's decision to work for a RHB. However, mainly income will be the driving force that provide supply. Furthermore, income is affected by PD, the duration of work, working cost, fractions of idle time and etc. Similar to PD, DS can also influence trip service quality. The higher the number of supply, the less the wait time and the higher the trip service quality. Alternatively, DS can also affect driver income level. This factor can be measured by the fraction of time the taxi is idle. Similar to the trip service quality, driver income level is also affected by the supply and demand. High passenger demand and low supply increase drivers income (i.e potentially surge pricing). In contrast, high supply and low demand reduces the income factor greatly. Indeed, the two components PD and DS are the driving force that allows the ride-hailing system to function.

With these explanation, we conclude the section on the general frame-work of the ride hailing system.

1.3 The costs and benefits of the matching algorithm for the conventional taxicab and ride-hailing platforms.

There are many advantages to designing a ride-hailing algorithm. Not only does it benefit the RHB that implemented it (such as Uber, Lyft, Grab,...) but it could potentially save operational costs and help improve system efficiency of the conventional taxicab businesses. Most conventional taxi cab businesses in the US relies on the two-way radio dispatch system which was developed in 1940 or the sight-based street hailing [18]. For the two-way radio system to work, it would require a reasonably size call center. Moreover, the down side of this system is that conventional taxi drivers have to depend on the call center worker to assign them a driver. When comparing the order processing time and efficiency of the call center workers against a computer algorithm, the outcome is indisputably in favour of the computer algorithm. Furthermore, the cost of hiring and training a call center worker are also quite expensive. Not to mention, all the businesses that hire employees need to adhere to regulations such as minimum wages, the right to works, employees right to work, employee's insurances and healthcare. Additionally, there are cost incurred by the drivers for the conventional taxi businesses as well. Specifically, taxi drivers have to have occupational licensing in order to transport passengers. A driver with an occupational licensing driving in a jurisdiction named B can not pick up a passenger from the jurisdiction named C. This driver can drop off their customers at any jurisdictions but they can only pick up their passenger in the jurisdiction where their occupational licensing is registered. Furthermore, drivers also need to have a permit to operate their cab which are called the Certificate of Public Necessity and Convenience (CPNC) or taxi medallion. The purposes of the taxi medallion is to restrict the supply of taxi drivers that can operate. Furthermore, fares in conventional taxi system is centrally regulated by an external body. With these restrictions it is ideal for the conventional taxi to transition partially or completely towards the ride hailing model.

In comparison with conventional taxicab businesses, RHB allows private car owner to be flexible in designing their own working schedule. Driver can choose to work whenever and for however long they prefer. This creates a level of job satisfactions that might lead to an overall improvement of customer service and trip safety. Additionally, RHB don't require their drivers to go through rigorous verification and employee's checks before they can start working.[14] Instead each service provider is a "company" which functions on their own because most drivers are self-employed. So in a way, the RHB removes the barriers of entry and ease up access to the market. Moreover, unlike in the conventional taxi, RHB have incentives and bonuses system such as the surge pricing which can help boost driver's earnings in the short run. Some may argue that the bonuses system and the efficient design of the matching algorithm may help off-set and even out the level of earnings of the ride-hailing drivers. After all expenses have been accounted for, perhaps ride-hailing drivers on average earn more than conventional taxi drivers while they work for less hour.

Furthermore, another difference between the two system is the wage and compensation. In the conventional taxi system of the US, drivers have the right to drive (by leasing a medallion) and they make a fixed payment toward the business based on either the weekly or daily medallion lease. The fixed payment is independent of their earnings and drivers get to keep every fare dollar. In contrast,the RHB drivers have to pay a proportion of the fares (called: Uber fees) to the platform. According to article [6], the percentage split from the fare can range from 20 to 25% of the total fare.

Aside from all the seemingly disadvantages of operating a conventional taxi cab business, we will now analyze from the perspective of efficiency and examine how the conventional businesses improve their operations efficiency and lower their costs. As we have stated before, when comparing the performance of the call center workers against a computer algorithm, the algorithm excelled in all categories. Such increase in performance translate to the improvement of the welfare of the drivers. Specifically, when comparing the two-way radio system of the conventional businesses against the algorithm of a ride-hailing business, half the time when the RH drivers work, they get a customer compared to only 30%-50% of the time for the conventional taxi drivers. This is measured according to the so called capacity utilization of the paper [18]. Note that the capacity utilization have also taken into account the fact that ride-hailing drivers in general work less hours compared to the conventional taxi drivers [31]. Additionally, in the mentioned research, capacity utilization is defined either by the fraction of time or by the fraction of miles in which a passenger sit in their car. For instance, if a driver work a whole day and traverse a 100 miles, then for the ride-hailing drivers, there will be a passenger occupying their vehicle for 50 miles. Whereas, for the conventional driver, passengers only occupy their vehicle for about 30-50 miles.

As analyzed, it is obvious that one of the major problem in any operations is the problem of efficiency. An efficient ride-hailing system leads to an increase in revenues for both the RHB and the drivers. Additionally, it could bring about improvements in customer satisfaction and thus increase the likelihood of repeated service. This paper [22] concluded that ride hailing system efficiency performs better than conventional taxi services. Most conventional taxi business model incur too many additional costs while RHB operate at a lower cost. Hence, they will be able to offer their fare price at a much more competitive level. The problem is not that it is too costly to run a conventional taxi cab, but rather such businesses do not adapt and incorporate some of the approach of the RHB model. It is predicted that conventional taxi might be driven out of the market if they can not adapt [3]. Furthermore, when discussing efficiency, the conventional taxi businesses do not have as much control over how distributed their drivers are. Whereas for the case of the RHB, the matching algorithm can provide guidance and detours by predicting future demand and minimise travel time. Undoubtedly it is much more efficient and relatively cheaper to design and maintain a ride-hailing system. Hence this might be one of the reason that explains the cheaper fare of Uber compared to conventional taxis [46]. Additionally, firms that hires programmers to design a matching algorithm do not have to provide additional training unlike the conventional taxi businesses since a computer programmer is required to have University graded qualification. This give rise to another interesting question which is whether it is cheaper to train workers to work at a centralised dispatch center or to hire a qualified computer programmer. However, if a business is concerned about the cost of hiring a long time programmer then they might choose to outsource the design of the algorithm to a third party and run a base-free operations. That would minimise the cost of renting an office space or a building.

2 Motivation and Literature review

Our dissertation is inspired by the conference paper on Spatial Capacity Planning [9]. In [9], researchers aimed to help platforms developers in the spatial environment to determine the desired level of capacity (or driver supply). The author argued that due to the complexity of the "spatial environment", the square root staffing rules (SRS) that has been used in the classical system need to be need to be updated. In contrast to [9], our work will focus more on the computational aspect.

Over the year, there have been many research into the structure and operations of the ride-hailing system. Many have chosen to focus on aspect such as price surging, balance supply and demand. The bulk of this work, will revolve around designing the algorithm and statistical anlysis.

In our work we will use "shortest straight-line distance to the customer location"

Order dispatch. Somewhat similar to our work, paper [39] also uses computer simulation to measure the effectiveness of the existing dispatching system (shortest straight-line distance to the customer location) and the proposed dispatching algorithm (the shortest-time path). Although this research was done on the conventional taxi system, its results are still relevant today. The author argued that, the shortest straight-line distance is not always the most optimal protocol due to city layouts and traffic rules; A rider can hail a taxi that is just on the opposite side of the road. However, to reach that customer, the driver would have to make a U-turn at a specific road intersection that allows a U-turn. If this intersection happens to be far away, then the driver would have to traverse a greater distance than originally anticipated by the (shortest straight-line distance). To remedy this issue, researchers have come up with a new dispatching algorithm called the proposed dispatching algorithm. As a result, this paper shows that the proposed dispatching algorithm cause a significant reduction in the amount of time taken for the driver to reach the customer. From the perspective of the customer, this reduces waiting time significantly and improves customer satisfaction. On the side of the driver, idle time as well as travel time are also reduced which leads to lower fuel costs and increase in earnings in long term. As a whole the system operates more efficiently and both drivers and riders benefit greatly. Furthermore, this research assumed that the rate of idle taxi remained constant throughout the simulation. Whereas in our work, the rate of idle taxi fluctuates, it depends on the number of customer as well as the number of taxi arrival during a certain state.

Another research with a similar goal to [39] is the paper [52]. Both were still trying to minimizing passenger waiting time in the conventional taxi system where taxi are in constant radio contact with a control center. However paper [52] aimed to minimize the expected total wait over the rolling horizon. The author have argued that the proposed dispatching algorithm in [39] may sometimes result in a sub-optimal driver assignment due to the fact that the driver could have reach other subsequent customers better and system efficiency would have been improved as a result. In [52], the author approached the problem from the perspective of dynamic programming and the look-ahead approach. The look-head approach use the probabilistic profile to predict future requests. Three case scenario were set up including: *taxi assignment without look-ahead*, *taxi assignment with 1 and 2-step look ahead* and *a heuristic algorithm.* Each scenarios were run by computer simulation and then being compared against one another. The result showed that the 2-step look ahead perform better than the 1-step look head, followed by the scenario without any look ahead. The performance of the heuristic was dependent largely on the alpha value and the demand level.

With great advancement in mobile sensor and data processing technology, [42] have managed to further improve the efficiency of the dispatch methods by utilizing real time GPS and occupancy data. Researchers proposed a *Receding Horizon Control* (RHC) framework with the aim of directing idle taxi to customers with minimal pick-up time. The proposed framework would operate as followed: the dispatching center would collect and store data from the GPS unit and trip recorder through cellular radio. The center run the dispatching algorithm, generate a solution, and then informing the drivers their tasks. As a result, customer wait time improved sufficiently and taxi supply was being redistributed efficiently while reducing the total pick-up time of idle taxis. Additionally, research

[55] also studied the order dispatch problem from a data-heavy approach; with the goal to improve user experience as well as to maximise the matching success rate between drivers and customer. The problem was formulated as a combinatorial optimization problem with the objective is to maximise the the probability of an order being accepted (success rate) by each driver. This probability (2.1) can be estimated by using a machine learning model such as linear logistic regression model (LR) or gradient boosted decision tree (GBDT). The author chose LR as their predictive model for accuracy reason. Then, the probability matrix will be feed into the combinatorial optimization problem (2.2) with the objective is to maximise the average success rate. Many combinatorial optimization are NP hard problems, hence the author had to proposed a heuristic called the Hill-Climbing algorithm. As a result, [55] have showed that customer's waiting time, average dispatch time and other performance metrics improve significantly compare to the previously deployed model. One of the challenges of the "data-heavy" methods is that it is inherently computationally expensive, for [42] to be applicable, the computers have to re-compute the dispatching algorithm numerous time. Furthermore, the "data-heavy" approach in both paper [42], [55] would require a huge amount of upfront capital investment to build and maintain multiple data centers to collect, store and analyze information. Not to mention, if the size of the computational problem is too large (i.e. too many ride requests take place too quickly), the algorithm might take too long to run and dispatch-time might suffer as a consequence.

$$p_{ij} = p(y = 1|o_i, d_j) = \frac{1}{\exp(-\mathbf{w^T x}_{ij})} \tag{2.1}$$

$$\begin{cases} \max_{a_{ij}} E_{\text{SR}} = \frac{\sum_{i=1}^{N}[1-\prod_{j=1}^{M}(1-p_{ij})^{a_{ij}}]}{N} \\ \text{s.t}, \sum_{i=1}^{N} a_{ij} \leq 1, a_{ij} \in 0, 1 \end{cases} \tag{2.2}$$

In our work however, we will generate our own data via simulation.

System performance under different regime with state dependent service rate. The dissertation also wish to highlight some researches that look at the ride-hailing related problems under a different perspective and address some of the other aspects of the ride-hailing problem that is not being focused in this work. Over the past decade, there has been a growing stream of literature focusing on how the effects on the state dependent service rate affect system performance as a whole. [12] looked at the service rate of the Erlang-R queuing model, using fluid approximation they could observed how service speed-up during service congestion may affect system performance in long term. They conclude that service rate speed-up can sometimes be beneficial as well as detrimental depending on the speed-up threshold N^*. Another seemingly related research that also use fluid approximation to analyze system performance is [20]. In contrast to [12], [20] looked at how service slow-down during congestion affect system performance as a whole. They analyzed the system of the modified Erlang-A model to take into account the slow-down effect and found that system performance are sensitive to different load sensitivity level.

Queuing analysis. There are many approaches to solving the many different aspects of the ride hailing system, one can either view it strictly as a mathematical problems (i.e. a queuing model) and solve using mathematical methods (such as fluid or diffusion approximation) or it can be viewed as a computational problems and solve through computer simulation. If one chooses to approach from the analysis of the queuing model point of view, we may refer the reader to [30], [44], [25], [24], [49]. These researches work on finding a good apporximation for the waiting time of the *GI/GI/1* queue. This queue type is closely related to the design of the simulation model of this work.

Ride-hailing matching. Another approach that has been gaining traction over the year is the application of bipartite matching on the ride-hailing system. In the field of graph theory, **matching** between vertices in an un-directed graph is defined to be a set of edges which share no same vertices

(Figure 3). A vertex is matched if it is an endpoint of a matching. Let V_1 be a set of all vertices with color red and V_2 be a different disjoint set of vertices with all the vertex colored black. Then a bipartite graph is one in which no two adjacent vertices share the same coloring [7] (for example, Figure 4 show that no two adjacent vertices share the same color.). The bipartite graph problem arises when a matching operation occur; for instance, assignment of items,tasks, or jobs to another set of items,tasks or jobs. In the case of ride-hailing, the problem of matching driver to rider or vice versa can be formulated as a bipartite graph matching problem with the objective is to find the maximum cardinality bipartite matching (MCBM) (i.e. maximising the pairs that can be matched with each other). One approach to solving the MCBM problem is to employ the *greedy algorithm.* As the name suggest, this method of matching attempt to match customer/agents the moment they have arrived. However, the down side to this approach is that it is not optimal in minimising the fraction of unmatched agent (we will refer the readers to paper [5] for more details). Researchers in [5] and its extension work [19] analyzed the problem they have shown that under the condition which the planner know when the agent is about to depart (i.e. the critical time of the customer), it is better for the system to wait until it has been filled with a certain amount of agents (i.e. the *Patient algorithm* . Additionally, [19] also arrived to the same conclusion with Monte-Carlo simulation. However, one drawback of the *patient algorithm* is it requires the planner to have the knowledge of the critical time; unfortunately this information is not readily available in a realistic setting. So, if the planner don't know the agent's critical time, then the *greedy algorithm* is suffices. Another disadvantages of these two algorithms is that they are local in it's approach (i.e. they only look the current neighborhoods of the agents that is being considered for matching instead of the global network structure). The researchers have also showed that the *Patient algorithm* is also not optimal when because it ignores said global structure. Another drawback of [5] is that its results depend heavily on the fact the market is one-sided (i.e. agents don't have preferences). If agents have preferences it is uncertain whether the local algorithms still performs well. Another study on the two sided model of [5] done by [40] have concluded that the more the variety of preferences (example: agents can be of two types hard to match and easy to match), the more impact it will have on the local algorithms. Namely, in the low variety of preferences setting, *Patient algorithm* still outperform the *Greedy algorithm.* In contrast, *Greedy algorithm* outperform the *Patient algorithm.* The result in [40] have also showed that preferences affect the findings in [5] in no trivial way. With this we conclude our literature review.

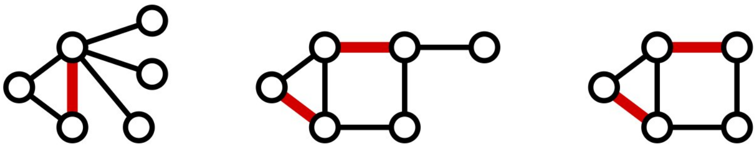


Figure 3: Example of a matching in undirected graph [16]

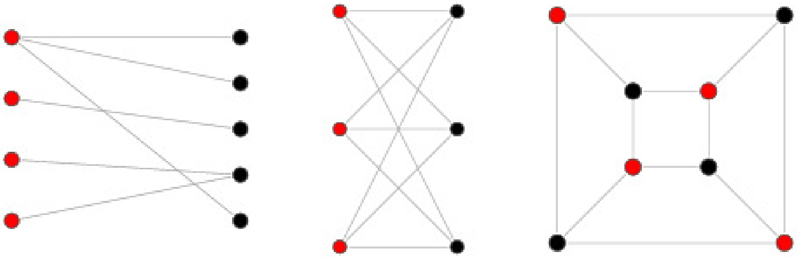


Figure 4: Example of a bipartite graph [50]

3 Ride Hailing as a queuing system:

To begin to understand the queuing model that we will employ in later chapter. The audience should try to familiarise themselves with some key definitions and terminologies.

Definition 3.1. *Stochastic process:*
A ***stochastic process*** *is a collection of sequence of random variables* $\{X_\theta\}$*, indexed by a parameter* θ*, where* θ *belong to the index set* Θ*.[43]*
For instance, if Θ *is a set of integer, then we have a discreet time stochastic process. However, if* Θ *is the real line or a subset of the real line then we will have a continuous time process.*

Definition 3.2. *:*
A stochastic process is said to have Markov property if given the present state, the future events are independent of the past. For discrete time processes $(X_n)_{n\in\mathbb{N}}$ *this property can be stated as:*

$$P(X_{n+1} = j|X_n = i_n, X_{n-1} = i_{n-1}, ..., X_0 = i_0) = P(X_{n+1} = j|X_n = i)$$

$\forall j, i_n, i_{n-1}, ..., i_0 \in$ *the state space S and* $n \in \mathbb{N}$,
and we define the one step transition probability from state i to state j at time n to time n+1 to be:

$$p_{ij}(n) = P(X_{n+1} = j|X_n = i)$$

Definition 3.3. *Exponential random variables:*
A continuous non-negative random variable X is exponential with rate λ *if its cumulative distribution function (cdf) and its probability density function (pdf) are of the form:*

$$F(x) = P(X \leq x) = 1 - e^{-\lambda x}$$

$$f(x) = \lambda e^{-\lambda x}$$

, respectively.

Definition 3.4. *Memoryless property:*
The exponential distribution is unique in the analysis of the stochastic process due to its memoryless property, i.e.:

$$P(X > s+t|X > s) = \frac{P(X > s+t, X > s)}{P(X > s)} = \frac{e^{-\lambda(s+t)}}{e^{-\lambda s}} = e^{-\lambda x} = P(X > t) \tag{3.1}$$

Intuitively speaking, suppose that the lifetime of a light bulb is exponentially distributed with rate λ*. When this lightbulb is installed the expected lifetime is 1/*λ*. After this lightbulb has been operating for s years, the remaining lifetime is still exponential with rate* λ *and the expected remaining lifetime is 1/*λ*. This means we can forget about the history of the light-bulb and treat it as a brand new one.*

The understanding of the above key concepts in crucial in understanding what is to come next. Because the upcoming queuing system utilize the memoryless property.

3.1 The basic of queuing theory:

3.1.1 *M/M/1 Queue*

Queuing theory is a sub discipline in the field of probability theory. A queuing system consist of two-side: the customers and the servers. Upon arrival, customers will have to enter a queue before they can be served and they leave the system once the service is completed. Both term "customer" and "server" are merely used in a general sense. In different application, customer and server are represented as different object. In this dissertation, customers represent riders and taxi drivers will represent the servers. According to the Kendall's notation [35], since the two-side system can be expressed as a queuing process, such process can be described by a series of letter $A/B/c$. The very first letter A represent the inter-arrival time distribution, B stand for the probability distribution of the service time and finally c usually represent the number of servers in the system. In many queuing theory literature, $A/B/c$ is replaced with $M/M/1$ where the first M describe the customer **Arrivals** process is Markovian-Poisson distributed, the second M means the distribution of the **Service** time is exponential and the number 1 indicate the quantity of servers in the system. [29]

Under this system, customers are served on the *First come first serve* (FCFS) basis. Customer's **Arrivals** is Poisson distributed with rate λ. Each arrivals occur independent of the previous arrivals and the time between each arrivals is defined to be the inter-arrival time. Inter-arrival time is independent and identically distributed (i.i.d) exponentially with parameter λ and the expected value $\frac{1}{\lambda}$. Due to the nature of the exponential distributions, arrivals has the memoryless property 3.1. Equivalently, **Service** time is also exponentially distributed but with rate μ and the expected service time is $\frac{1}{\mu}$. The service time of each customer are independent of one another. From the memoryless property of the exponential distribution, knowing how much time it took to service a previous customer, give no extra information on how much much service the next customer will require.

An extension of the $M/M/1$ system is the M/M/n queue (or can be called the Erlang-C model in some literature). This set up is similar to their predecessor except now it models **n** different servers as well as **n** queues (serving and departing at different rate).

3.1.2 *G/G/1 Queue*

The details process of mathematical analysis and explanation of the GI/GI/n queue is beyond the scope and scale of this dissertation. Therefore, if the readers are still interested then we refer them to [36, 29, 28, 23]

3.1.3 Operational regimes of the ride-hailing system

Before going into the ride-hailing system, we would like to first introduce the classic static environment systems. The static system can be thought of as banks, call centers, movie theater queue, etc... The system is static because servers are stationary in space and arrivals have to first reach servers before any queuing or services can begin. The most common queuing models which supports the static nature of said system are the Erlang-A and Erlang-C models. Erlang C as mentioned before is an M/M/n queuing system, this model is the simplest queuing model for call center because unlike the Erlang-A it does not taking into accounts the customers abandonment factor. Both the Earlang-A and Erlang-C assume constant arrival rates λ and services rate μ. For the Erlang C, defined the average offered load/traffic intensity to be: $R = \frac{\lambda}{\mu}$, where: $1/\mu$ is the average service time. According to the Halfin-Whitt regime (1981) or (a.k.a the quality-efficiency drive (QED) regime), while the average service rate stay constant, as the traffic intensity increase along with the arrivals rate λ, the right staffing level (or the right number of servers) is set to be: $n = R + \beta\sqrt{R}$ where: β is the service grade and $\beta = (1 - R)\sqrt{n}$. Additionally, the quantity n is also called the Square-Root Safety Staffing Rule (SRS rule) and it provide us the optimal number of servers that is needed to have a good system performance. Notice that the term $\beta\sqrt{R}$ can described as the extra capacity needed to take into account the stochasticity factor of the real world environment. In the upcoming paragraph, we will observe how the Erlang-A can be split and operate under different regime (due to its abandonment factor).

For the Erlang-A model, the patience time is incorporated into the arrival process. Arrivals will abandon (with individual abandonment rate called α) if their waiting time exceed the patience time. Define the probability of customer having to wait upon entering the system to be $P(W > 0)$ and the probability that an arrival will abandon without service to be P(Ab). If the SRS rule derived from the Erlang C system is applied to Erlang-A, then as the system is operated under the QED regime, for a large amount of arrivals, $P(W > 0) \to \alpha(\beta)$ where α depend on the service grade β and for a small amount of arrivals $P(Ab) \to 0$ [20, 26]. Aside from the QED regime, other operating regime is also considered. Namely, when the staffing cost is high, the efficiency driven (ED) regime is employed and the the SRS rules is set to be $n = R - \epsilon R$ where ϵ is chosen according to [51]. Under the ED regime, servers utilization reach 100% but the probability of waiting $P(W > 0) \to 1$ and $P(Ab) \to \epsilon$ [26]. In contrast to the ED regime, the quality driver (QD) regime prioritise service quality and set its SRS rule to be $n = R + \epsilon R$ where ϵ is chosen in a similar fashion [51]. Under QD operation, $P(W > 0) \to 0$ and $P(Ab) \to 0$. [26]

Similar to static servers such as call centers and bank queuing, the spatial multi-servers system also follow three operational regime. The three regimes are as before Quality driven (QD), efficiency driven (ED) and Quality-Efficiency driven (QED). The cost of operating in each regime is measured by the waiting time probability; denoted by $P(W > 0)$. This quantity represent that probability that a customer have to wait for a certain time before receiving services. Conversely, the probability that a customers get served without waiting is $1 - P(W > 0)$.

As the research [9] have put it, the difference between the static and the spatial environment is an additional work load called the **Pick-up** time. Upon entering the system, all customers will experience a so called *Time in the system* defined as followed:

For the static environment the customer time in the system is

$$\text{Time in the system (Static)} = \text{Waiting time} + \text{En-route (Service time)} \tag{3.2}$$

For the spatial environment the customer time in the system will be:

$$\text{Time in the system (Spatial)} = \text{Waiting time} + \text{Pick-up} + \text{En-route (Service time)} \tag{3.3}$$

To simply put, Waiting time is a duration in which a customer spend in the system until he is assigned a server. Pick-up time correspond to the time when a server/taxi have to travel her current location to the customer location before beginning service. En-route time is the time when the customer occupy the taxi and is on his way to a destination point. Additionally, en-route time can be referred to as the Euclidean distance between any two points in the set $\mathcal{C}$ and it does not depend on the state of the system because once the server has picked up the customer the only thing it cares is completing its task. In contrast, pick up time has to depend on the state of our system because the platform are the one can either choose to allocate server to customers or customers to servers. And whichever approach the platform decide is based the number of customers versus the number of servers that is currently in the system at that time.

Due to the Pick-up time factor, the SRS used widely in the Erlang-A and the Erlang-C model can not be applied to the Spatial environment. The Pick-up time factor introduce an extra workload on the system because servers/taxi have to travel to pick up customers. During travelling time, the taxi are busy despite not having customers on board. If we observe from the perspective of the drivers then on average, the service time of the servers (i.e. Pick-up time + En-route) in the spatial environment is much greater compare to the servers in the static environment. Another extra set of challenges introduced by the Pick-up factor is the that it depends on that state of the system as mentioned above and the dispatching protocol. A dispatching protocol is defined to be a method of assigning drivers/taxis to customers and vice versa. Different dispatching protocol may increase or decrease the pick-up time and the state of the system determine how far taxis have to travels to pick up customers. In attempt to reduce the the pick-up time variability and to approximate its value, researchers in paper [9] choose the Nearest neighbor (NN) dispatching protocol.

– If the number of servers > customer $\implies$ then NN assign the next arriving customer to the closest available server/taxis.

- If the number of servers < customers $\implies$ then NN assign the an idling server/taxis to a closest waiting customer.

The author chose the NN algorithm because it is simple, intuitive and near optimal. The behavior of NN give rise to the state dependent mean service rate $= \frac{1}{\mu(q)}$ and it creates a spatial friction in the following way. If the number of taxis in the system exceed the number of customers, then there are much flexibility in how the NN algorithm can match drivers and riders (because of excess supply and the algorithm need only to match optimally for the customers), as a result pick-up time will be almost negligible and the mean service time will be closed to the en-route time. If the number of customers is much larger than the number of drivers, then the same scenario will also happen (because of the excess supply in demand and the algorithm need only to match optimally for the taxi) and the algorithm can flexibly match customers to drivers optimally so the pick-up time for the driver who received a match is also negligible and the mean service time is also the en-route time. However, when the numbers of taxis and customers are almost identical, then the pick-up time will be much more significant because the algorithm now have to optimally assign from both side of supply and demand.

Since the pick-up time can be approximated, the stochastic dynamic of the $M/M_Q/n$ spatial system can be also be approximated using a fluid model. This allow us to study the probability of having to wait as n grow large (i.e. $\lim_{n\to\infty} P(\text{Wait}_n > 0)$. Additionally, the analysis of the fluid model allow us to understand how the probability having to wait might behaves under different equilibrium. These equilibrias are associated to the difference operations regimes which we have mentioned before. Thus, the result suggest that, if the number of servers/taxis n is set such that:

$$n = \frac{\lambda}{\mu} + \beta.(\frac{\lambda}{\mu})^r, \text{with}, r > 2/3 \tag{3.4}$$

then $\lim_{n\to\infty} P(\text{Wait}_n) = 0$ the system will reach an equlibria where more taxis will eventually becomes idle and the system will operate under the QD regime. If the number of servers/taxis n is set such that:

$$n = \frac{\lambda}{\mu} + \beta.(\frac{\lambda}{\mu})^r, \text{with}, r < 2/3 \tag{3.5}$$

then $\lim_{n\to\infty} P(\text{Wait}_n) = 1$ the system will reach an equlibria where more customers will eventually have to wait and the system will operate under the ED regime.

As shown above, if the SRS rule of the Erlang-A and the Erlang-C system were to be applied to the spatial environment. Then $r = 1/2(< 2/3)$. That means that eventually, all customers will have to wait. On the other hand, for the probability of having to wait to be strictly between (0,1) (i.e. $\lim_{n\to\infty} P(\text{Wait}_n) \in (0, 1)$). The researchers argue that r value need to be set to 2/3 and the number of servers n to be:

$$n = \frac{\lambda}{\mu} + \beta.(\frac{\lambda}{\mu})^{2/3} + \gamma_n.(\frac{\lambda}{\mu}), \text{with}, \gamma_n \to 0 \tag{3.6}$$

To sum up, the SRS rule in the static environment need to be adapted to take into account the pick-up up and hence instead of the SRS rule set ($r = 1/2$), the spatial environment will set its SRS rule to be ($r = 2/3$).

Even though the capacity planning paper only perform their operations regime analysis for the $M/M_Q/n$ queue. The concept of different operational regime work can still apply to the simulation model of this dissertation. When it comes to assessing the operational regime of a system, most literature relies on the Waiting time of the customers. Due to the complex nature of our algorithm, Waiting time will not be used as a performance measure, but rather we will use the abandonment rate. In a multi-servers, two side matching system, it is challenging (at least for the scope and scale of this project) to keep track of the waiting time of each customer, even if such is possible, our system are designed to operate solely in the QED regime, hence such step is unnecessarily. Some may wonder why the rate of abandonment is a reasonable substitution ? If the rate of abandonment is bigger than

0, then the the probability of customers having to wait is equal to 1. (i.e. P(waiting) = 1) and it implies that the system is operating in ED regime. If the rate of abandonment is 0, then we know that the P(Waiting) = 0. Additionally, even if the rate of abandonment is 0, it still can not guarantee that the Probability of waiting is 0, it may approach close to zero but it will never be zero.

3.2 The queuing model of this work:

In the queuing model of this work, customers arrive and upon entering the system, will be immediately assigned to any empty server based on the minimum distance policy. If all the servers are busy, customers will wait for a fixed duration and they will leave the system either after their patience has run out or after the service has been completed. Note that during waiting time, taxi is only assigned if she meets the requirement of the minimum distance, if not then the customer will wait until either another taxi finish her service close to our current customer or a new taxi is joining the system and she happens to be close by.

Even though the queuing model of this work is GI/GI/n, we believe that the algorithm can simply be adapted and transformed into the M/M/n queuing system by choosing different distributions for the arrival rates or just by calling our system GI/GI/n queue but with an exponential distribution. Most of the experiments done in this work have assumed exponential arrivals and abandonment rates. We did not find it is necessarily to adapt our system to model the GI/GI/n with a general distribution because the purpose of this dissertation is to design a working matching algorithm and analyze its behaviour according to the performance measures produced. Hence it was of no importance whether our system was operated as a GI/GI/n or M/M/n queue.

Later on, when the model is introduced, the pick-up time will be defined as followed: Pick-up time = `time_coeffiecent` × `Min_distance`.

Over the years, there have been many other work focusing on the abandonment factor of the system (Erlang-A model dubbed the $M/M_n/n$ queue). However so far, most papers have only discussed customers abandonment. The difference between our work and other research is that, we take into account the taxi's abandonment factor. Taxi's abandonment here means that taxi has left the system after having finished its assigned working hours. Hypothetically, when the taxi left the system, it may increase the work load on the system and can have negative impacts on the performance measures at any one state. Perhaps if not too many taxis leave the system (i.e the number of taxis do not drop below a certain threshold), then we can maintain the stability of system performance under the QED regime.

4 Why simulation ?

Real world operation systems are usually costly and time consuming to implement. Operations managers have to figure out a way in which a production chain can run efficiently. For example: how many workers do they need without hiring too many, how many machinery each section need without them being redundant, how should each section be laid out so as to minimise workers travel time ? etc. Hence, real world systems are restrictive when it comes to testing out different configurations but operation managers want to find out the most optimal way they can operate a plant. This is where computer simulation can help because it allows the operation managers to implement such systems without the cost of actually implementing it.

A simulation is a computer based program that can model a real-world system. Due to time constraints and the complexity of real world situations, computer simulations usually perform a much more simplified version of a real world system. Despite its simplicity, it still includes all the necessary factors that a real world scenario would have and it is a perfect tool to test our assumptions.

Computer simulation is vital in the making of optimal decisions by allowing user to implement a hypothetical situations without actually implementing it in real life. As a result, it provides helpful predictions, saves time and costs of operations. Therefore the upcoming section of this work will utilize the power of computer simulation to try to understand how different dispatching methods affect system performance as a whole.

4.1 Random number generation

For our simulation system to function, it either requires real world data or it has to be able to generate its own data. Contrary to other research [55, 42, 39] which involve real world data, the simulation model of this work will generate its own data. The `numpy` library in Python will provide some crucial random number generators (RNG) such as the `np.random.exponential` and the `np.random.uniform`. The `np.random.exponential` will mainly be used to generate inter-arrival time and abandonment time. its probability density function is of the following form:

$$f\left(x;\frac{1}{\beta}\right)=\frac{1}{\beta}\exp\left(\frac{-x}{\beta}\right) \tag{4.1}$$

where β is the scale parameter and $\beta = \frac{1}{\lambda}$ (where λ is the rate of the exponential distribution. The generator will draw samples from an exponential distribution depending on the rate specified. In terms of ride-hailing simulation, according to [9] the exponential distribution can be used to describe the inter-arrival time as well as abandonment time due to its independent and memoryless property. Thus our choice of the inter-arrival and abandonment time distribution is a sensible one.

The `np.random.uniform` is used to generate the uniform(0, b) random variable. This generator will sample from the (0,b) uniform distribution and it will determine randomly the x (coordinate) and the y (coordinate) arrival point of the drivers and riders. For the moment, we choose the uniform distribution for simplicity. In a later section, we will show the reader another interesting way in which we can restrict how our distribution can sample and it will confine the arrival location of our subjects to a bounded zone. Note also that for almost all of our upcoming experiments, the uniform(0,2) is the distribution of choice.

Inevitably, there will be issues with designing the simulation in Python and to correctly identify and debug such issues, an important property of RNG is required and that is its reproducibility. Additionally, not just for troubleshooting, reproducibility property is key because it allows the experiment to be rerun and retested and to be able to yield the same performance measure repeatedly so others can verify our findings. To ensure reproducibility is satisfied, we will utilize the `set.seed` function provided by Python. With different `seed` value as input, we will be able to generate a different stream of random numbers. Each number stream generated by each seed is independent from the other stream and we are going to see in a later section as to why this is important. Thus, with this guarantee, the result of our work will be reproducible as long as a specific seed value is set before performing any experiments.

4.2 Details of the Python model

As mentioned, Python will be the programming language of choice due to ease of comprehension as well as ease of access. Additionally, Jupyter notebook will be the platform of choice for running the simulations and performing analysis. Python will be used to construct an algorithm that can hypothetically generate riders and drivers, record their time of entries and exit, assign riders and driver an arrival rate, arrival location and abandonment time, matching riders to drivers and vice versa. The algorithm stops when it reaches the termination parameter (`T`). To generate the required parameters, a built-in package `numpy` can be "imported". By calling the function `np.random.exponential`, the program will sample from an exponential distribution and pick out a value and assign that fixed value to one of the model input parameters. Our model parameters will be detailed below.

4.2.1 Model variables

The parameters are:

- Taxi arrival rate: (`lambda1`)
- Customer arrival rate: (`lambda2`)
- Taxi abandonment rate: (`aban1`)
- Customer abandonment rate: (`aban2`)
- Time coefficient: (`time_coef`)
- Recording the total number of trip completion: (`Num_trip`)
- Total number of taxi in one replication: (`total_T`)
- Total number of customer in one replication: (`total_C`)
- The number of taxi at each system state: (`NumT`)
- The number of customer at each system state: (`NumC`)
- Termination time: (`T`)

4.3 Model assumptions

Unlike other works, our model assume that driver cannot reject ride requests. The distribution of the inter-arrival and service time is a general distribution and chosen to be exponential. We are aware that sampling from exponential distribution might indicate that our simulation is not the GI/GI/n queuing model. But this can be adjusted by simply changing our code.

4.4 Description of the simulation task:

Riders appear on a 2-dimensional square with the location coordinate (x, y) that is uniformly distributed between (0,2). Riders arrival is deterministic and assumed to be exponentially distributed with rate λ_1. Additionally, each rider will have a fixed abandonment rate μ_1, which is generated from an exponential distribution. This abandonment quantity is created whenever a customer arrives and they don't immediately get a match.

Analogous to riders, drivers will also appear uniformly on the rectangle (0,2) with arrival rate λ_2 that is sampled from an exponential distribution. Drivers leave (or abandon) the rectangle when her shift is finished with rate μ_2. The rate for driver abandonment exists because if not, then the rectangle would eventually be filled up with an infinite amount of driver points and after some time new riders would not be able to arrive for work. From the moment a driver is assigned to the moment that driver takes a rider to his desired destination is the distance which the driver has to travel which is $(L_1 + L_2).v$. L_1 is denoted by the distance between driver's current location and the assigned customer's location. L_2 is distance between the location of a customer and the destination location. For instance, if a driver has to travel 20 miles in total to complete a service, and it takes on average 2 minutes to cover a distance of 5 miles, then it would take that driver in total 8 minutes to complete a service for that particular customer. Additionally, driver abandonment time (or driver's time when she finishes her shift) is independent of the en-route time. For example, if a driver work from 9am to 4pm, and at 3:45pm she was assigned to a customer, then she would still be in the system even if the system clock have passed 4pm.

To record (or log) the data generated, an event calendar called (`EC`) will be set up. `EC` will initially be a 3-dimensional vector with entry of all 0's. The first entry will record the time of driver arrival, the second entry will record the time of rider's arrival and the third entry will keep the Termination time. In Python language, each entry in a list can be referred to by an index, and index start counting from 0. Hence the first entry is the zeroth index, the second entry is the first index, and so on. For example, when we write `EC[0]`, it means that we want to pick out the value at the 0-th position of this vector `EC`. Note also that in this work, a replication is defined to be one run of a simulation. During the simulation, each iteration will be cycled by the "while loop". At each iteration, the `min` function will check for an index which hold the smallest value of the EC. Then that value will be assigned to the variable `Tnext` and the smallest index (i.e. `min_index`) of the EC vector is computed by calling `EC.index(Tnext)`.

The `min` function is employed to make sure that events are happening "accordingly" and the simulation is not skipping over events that are supposed to take priority (i.e events with the smallest value index has to take precedent). Furthermore, this also helps in schedule the next event appropriately. As the simulation progress through time, the dimension of the EC vector will increase and the EC vector will be updated. For example: in one replication, if min.index is zero, then it means that the current event is an arrival of a driver.

Alongside EC, we are also interested in keeping track of the total of number of taxi and customers in the system. To do that, we will set up the LT and LC matrix. Initially, the two matrix will be an empty list but as the simulation progress, LT and LC will become a $4 \times D$ (with $D \geq 4$) matrix and a $6 \times D$ matrix (with $D \geq 6$) respectively. The 0-th row of both LT and LC will hold the current status of the taxi and the customer (i.e: if the 0-th entry display the value of "1" then it indicates that the taxi or the customer is "busy" serving or being served and "0" otherwise). Additionally, the first and second row of both matrix will holds the x and the y-coordinates of the taxi and of the customers respectively. Finally, the extra two rows of the LC matrix the will hold the x and y destinations coordinates respectively.

With EC, LT and LC, the program now need to keep track of the event, the drivers and the customer that are associated to that event. Originally, ET is a 3×3 matrix with entry of all 0's. But with each iteration, the matrix columns will expand to become a $3 \times D$ matrix (with $D \geq 3$). Each column D of the ET matrix will contain 3 rows: the first row will record the types of event, the second row correspond to the ID of the associated driver and finally the third row will note the ID of the customer.

There are six different event types and each one will be assigned a numeric value:

- The event of driver arriving to the 2-dimensional square will be assigned the value of 1.

- The event customer arriving is assigned to the value of 2
- The termination event is assigned to the value of 3
- The event driver abandonment or (driver time of finishing shift)is assigned to the value of 4
- The event customer running out of patience and leave the system is assigned to the value of 5
- The event driver taking rider to his destination is assigned to the value of 7.

The reader may wonder as to why there is no event of type 6. In this work, we have chosen to skip from `ET == 6` to 7. The reason this happen is because in our computational model, we have decided to not consider the event which driver is on route to picking up customer. We have instead incorporated the event of type 6 into the process of updating EC with the time until service completion. In essence, "the time until service completion" will be of unit of time and its calculation include: `TNOW`+ `(time_coefficient_factor)`×[`(minimum_distance)`+`distance_between_customer_and_destination`].

In (Figure 5), after the min_index value has been determined, the algorithm will decide the order of execution. The algorithm will run until the variable TNOW is no longer less than the termination time.

If the `minimum` index generate the event of type 1 then the algorithm will follow the process as described in (Figure 6). At first, Python will store in their memory the fixed parameters which we have defined earlier. Then it will check whether or not `TNOW` is less than `T`, the program will keep on running for as long as this condition hold true. Next, it will pick out the index number of the smallest value of the `EC` vector. And that index number will execute the event of type 1.

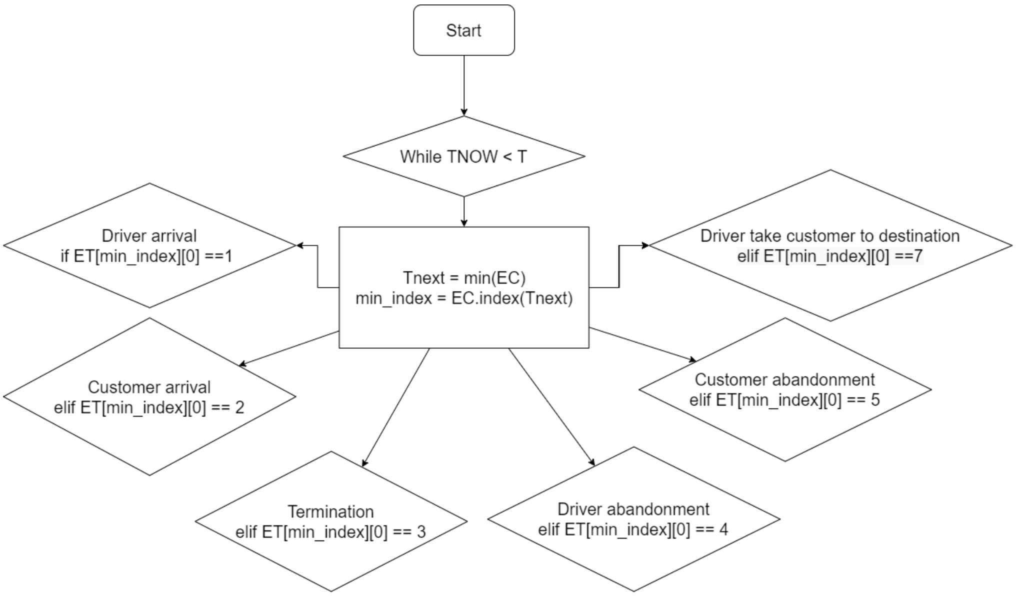

Start
While TNOW < T
Driver arrival
if ET[min_index][0] ==1
Tnext = min(EC)
min_index = EC.index(Tnext)
Driver take customer to destination
elif ET[min_index][0] ==7
Customer arrival
elif ET[min_index][0] == 2
Customer abandonment
elif ET[min_index][0] == 5
Termination
elif ET[min_index][0] == 3
Driver abandonment
elif ET[min_index][0] == 4

Figure 5: Simulation operation flow chart for each event type

In the event of type 1, the algorithm need to update the time `TNOW` as well as other key processes such as generating destinations points for the taxi (more details can be found in (Figure 6). After the initial set up of the preamble, we need to make sure that the total number of customer in the system is non-negative otherwise we cannot create a match Then `for` loop will check for "idle" customer that is currently in the `LT` list. If they are "idle", the algorithm will assign that customer to the current taxi base on the minimum distance rule (i.e nearest customer assignment). After assigning that customer, we will need our system to update the relevant parameters such as the status of taxi and customer need to switch from 0 to 1, schedule an event where this taxi is taking this customer to his destination.

Finally, we need to make sure that the customer who have already been assigned to a taxi can not abandon mid way, (This can sometimes happen because some customers will have to wait before they get assigned a taxi and when they wait we in essence create an event of type 5) i.e. the system need to delete the corresponding abandonment event in EC and ET.

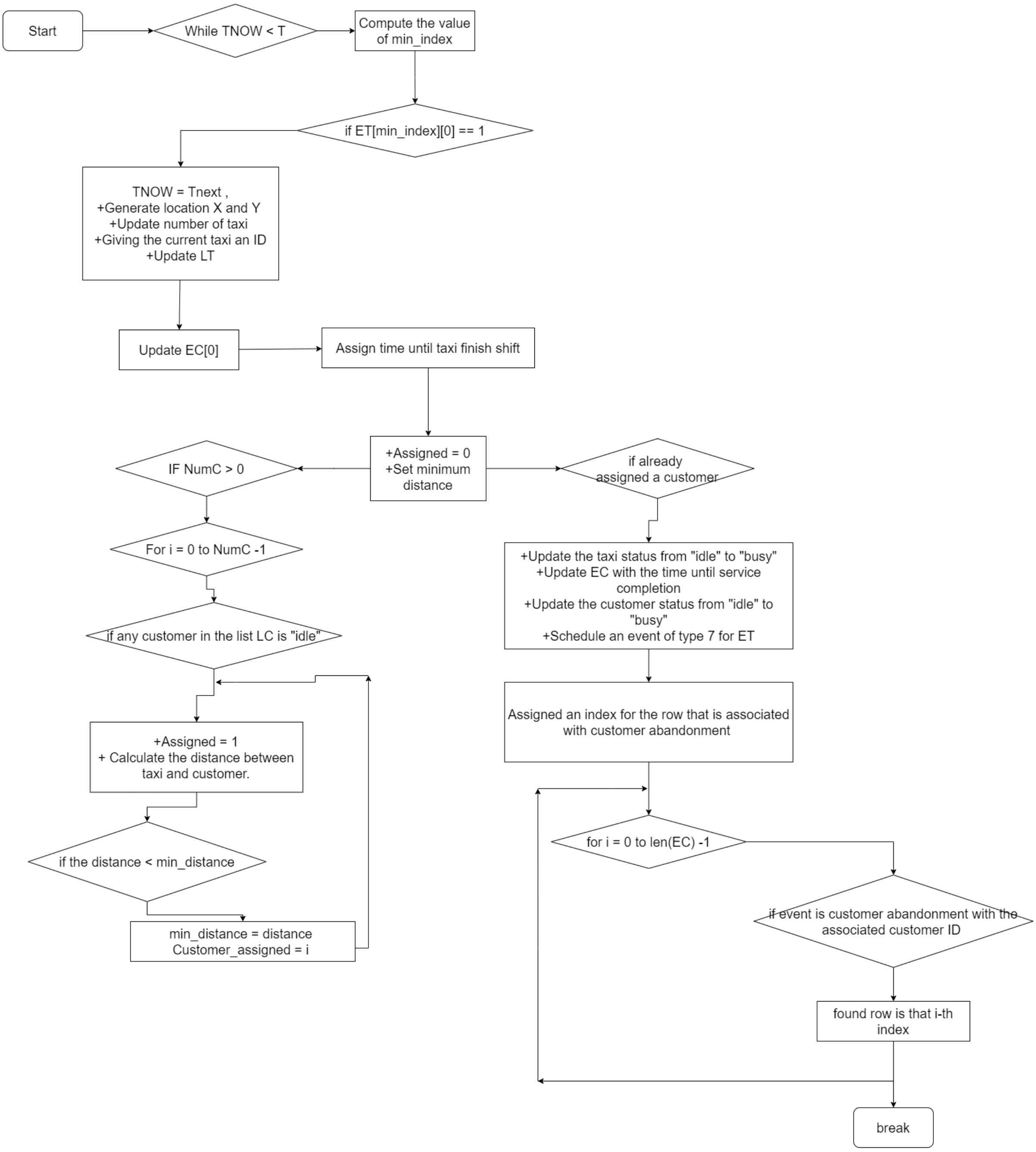

Start
While TNOW < T
Compute the value of min_index
if ET[min_index][0] == 1
TNOW = Tnext ,
+Generate location X and Y
+Update number of taxi
+Giving the current taxi an ID
+Update LT
Update EC[0]
Assign time until taxi finish shift
+Assigned = 0
+Set minimum distance
IF NumC > 0
For i = 0 to NumC -1
if any customer in the list LC is "idle"
+Assigned = 1
+ Calculate the distance between taxi and customer.
if the distance < min_distance
min_distance = distance
Customer_assigned = i
if already assigned a customer
+Update the taxi status from "idle" to "busy"
+Update EC with the time until service completion
+Update the customer status from "idle" to "busy"
+Schedule an event of type 7 for ET
Assigned an index for the row that is associated with customer abandonment
for i = 0 to len(EC) -1
if event is customer abandonment with the associated customer ID
found row is that i-th index
break

Figure 6: Flow diagram operation of event of type 1

After a taxi has arrived, we have to set up her abandonment schedule. As mentioned earlier, the event type that is associated closely with driver abandonment is ET = 4. As per usual, the program need to update the current clock of the simulation (i.e. TNOW). Additionally it needs to assign a dummy value to a variable called t_found, this variable will be used to identify "the index that is associated with the current driver". Note that the order of the process is important here (Figure 7 read from left to right), because at first, our program need to be able to identify the index of the taxi that it works with, then it will use that index to search if that same taxi is currently busy, and if not then it can leave at its scheduled departure time (when this taxi leave we have to remove the values that are associated with it in the EC, ET and LT vector). However, if the taxi that we are considering is indeed busy, then the system can not yet make it depart until it finishes serving. For instance, if a taxi number 6 is scheduled to leave the system at time point 11 but it is busy serving customer and won't finish until time point 14. Then, we should request the program to search along the column of the EC vector and find the time until trip completion. If "taxi scheduled departure time" don't coincide with the "trip completion time" (i.e. 11 $\neq$ 14) then the taxi has to stay in the system for an additional 3 minutes and the program need to "turn" the 11 values into 14. Otherwise the taxi will just leave the system at time point 11 and the customer won't reach his destination. In contrast, if the two values are equal, then it means that "the scheduled departure time" happen to coincide with the "trip completion time", the taxi can leave at time point 14 and the program need to remove the customer that is associated with this taxi and the event type of 7.

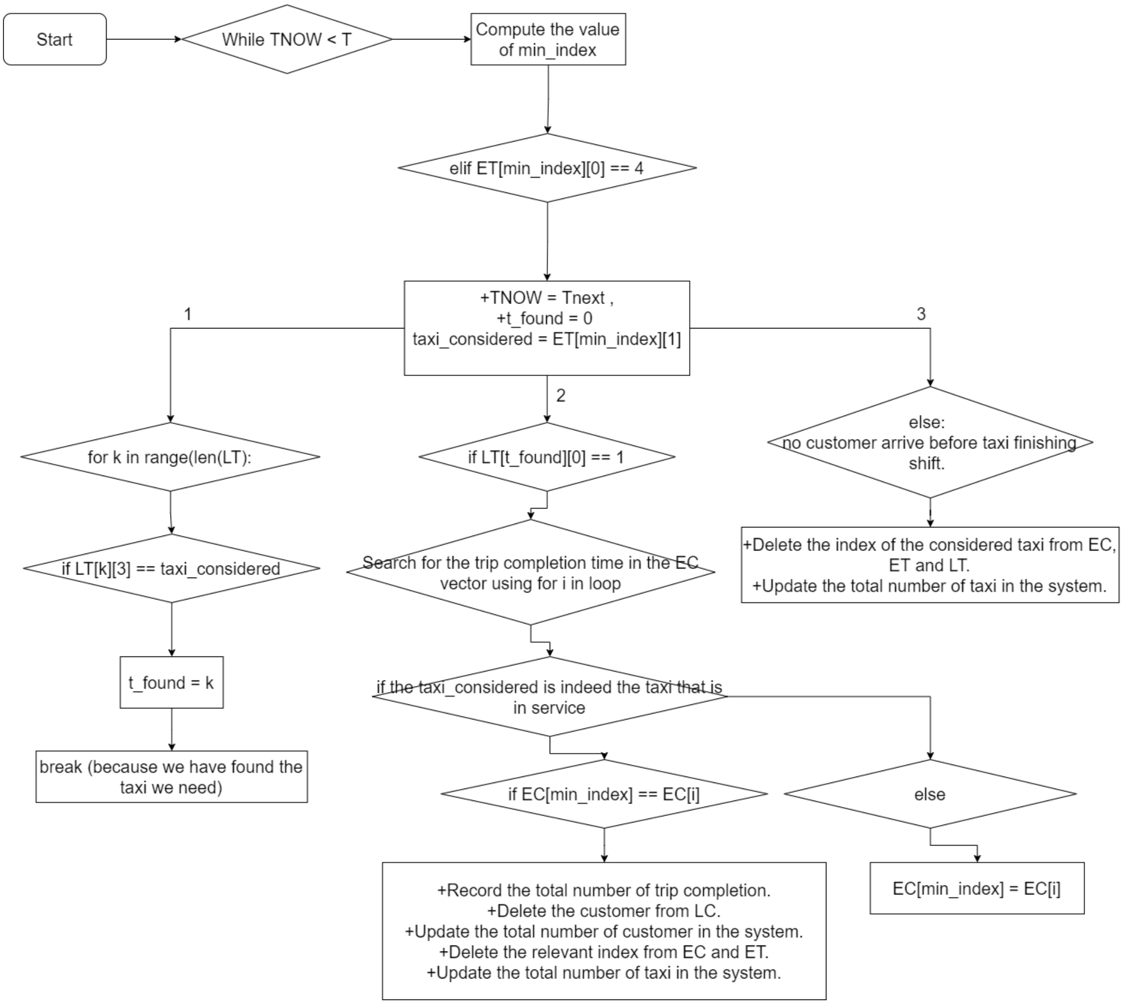

Start
While TNOW < T
Compute the value of min_index
elif ET[min_index][0] == 4
+TNOW = Tnext ,
+t_found = 0
taxi_considered = ET[min_index][1]
1
2
3
for k in range(len(LT):
if LT[k][3] == taxi_considered
t_found = k
break (because we have found the taxi we need)
if LT[t_found][0] == 1
Search for the trip completion time in the EC vector using for i in loop
if the taxi_considered is indeed the taxi that is in service
if EC[min_index] == EC[i]
+Record the total number of trip completion.
+Delete the customer from LC.
+Update the total number of customer in the system.
+Delete the relevant index from EC and ET.
+Update the total number of taxi in the system.
else
EC[min_index] = EC[i]
else:
no customer arrive before taxi finishing shift.
+Delete the index of the considered taxi from EC, ET and LT.
+Update the total number of taxi in the system.

Figure 7: Flow diagram operation of event of type 4

The event of type 2 (Figure 8) is almost identical to the type 1 with a slight difference. When we are in the situation of customer's arrival, we have to observe from the perspective of the customer. This mean that we now have to consider programming the operation of assigning a taxi to our current arrival customer. Different events types will require us to observe the process from different perspective. As mentioned, one notable difference between ET of type 1 and ET of type 2 is that we schedule the customer abandonment time after we have tried to search for the closest driver. Whereas in the case of taxi arrival, we immediately schedule the taxi abandonment rate just right before any operations can take place. This happen due the assumption of our construction of the system. The fact that driver finishing her shift has nothing to do with whether or not there is a customer in the system. In another words, taxi abandonment is independent of the total number of customer and the total number of taxi in the system at any state.

The simplest ET that we can program is the event of type 3. The event of type 3 is the termination event, when the simulation have reached this operation, it needs to updating the simulation time, outputting the performance measure statistic, and stop the program. As ET of type 3 is the simplest and it does not have any complex operations, we won't be including its flow diagram.

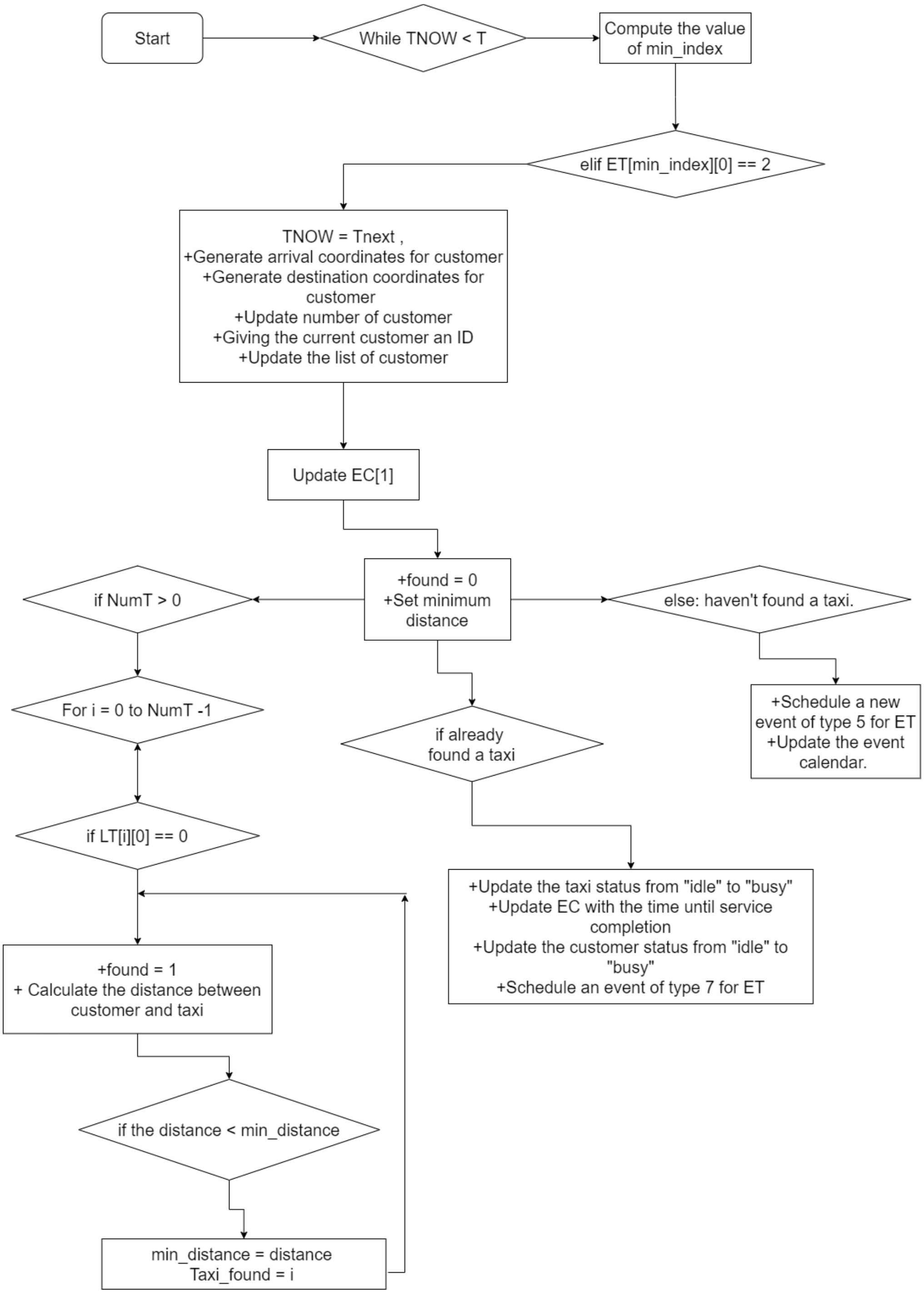

Start
While TNOW < T
Compute the value of min_index
elif ET[min_index][0] == 2
TNOW = Tnext ,
+Generate arrival coordinates for customer
+Generate destination coordinates for customer
+Update number of customer
+Giving the current customer an ID
+Update the list of customer
Update EC[1]
+found = 0
+Set minimum distance
if NumT > 0
else: haven't found a taxi.
+Schedule a new event of type 5 for ET
+Update the event calendar.
For i = 0 to NumT -1
if already found a taxi
if LT[i][0] == 0
+Update the taxi status from "idle" to "busy"
+Update EC with the time until service completion
+Update the customer status from "idle" to "busy"
+Schedule an event of type 7 for ET
+found = 1
+ Calculate the distance between customer and taxi
if the distance < min_distance
min_distance = distance
Taxi_found = i

Figure 8: Flow diagram operation of event of type 2

Comparable to the taxi departure event, our system must also taken into account the event of customer departure (Figure 9). This happen because the algorithm is unable to find any available drivers. However, we still need it to keep the system clock up to date, assign a dummy values to the variable `Customer_found` and finally it needs to assign the ID of the current customer to the variable `Customer_depart`. These steps are taken to ensure that we are removing the "right" customer. In addition, the order of the process is very important (i.e. branch 1 take place before branch 2) because the removal process can not take place before that program have identified that it is indeed the "right" customer to remove.

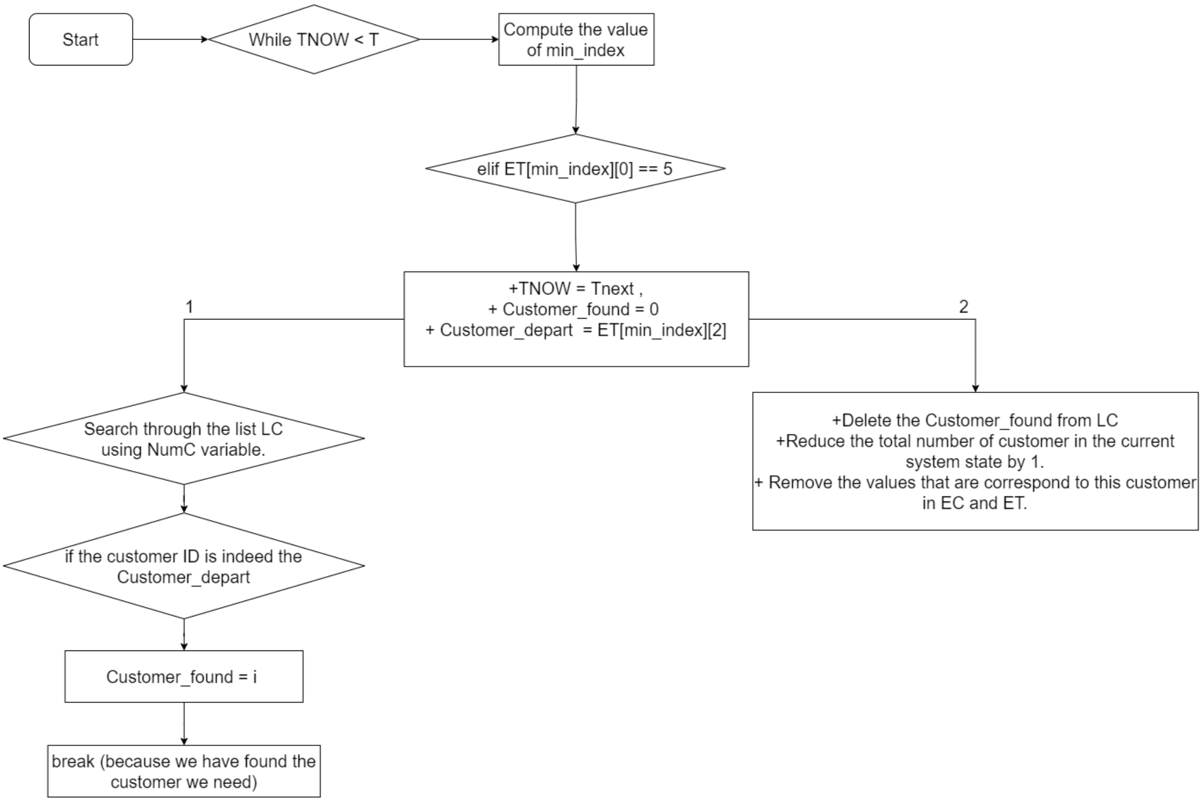

Start
While TNOW < T
Compute the value of min_index
elif ET[min_index][0] == 5
+TNOW = Tnext ,
+ Customer_found = 0
+ Customer_depart = ET[min_index][2]
1
2
Search through the list LC using NumC variable.
if the customer ID is indeed the Customer_depart
Customer_found = i
break (because we have found the customer we need)
+Delete the Customer_found from LC
+Reduce the total number of customer in the current system state by 1.
+ Remove the values that are correspond to this customer in EC and ET.

Figure 9: Flow diagram operation of event of type 5

Finally, we can not finish our model description without explaining the event of type 7 (Figure 10). The process will begin from the `start` point and it behaves similar to the previous process. Once again, the audience should note that the order of the branches important. We will start off with branch 1 and 2: As taxi and customers leave the system at every state, the index that denoted their entry in the EC and ET matrix changed due to the shift; however, their ID's stay the same. Essentially, the first 2 branches ensure that our current event type is handling the "right" customer and taxi. Branch number 3 attempt the operation of matching the current driver to a new customer in the system (whether he is just arrived or he has been waiting) base on the minimum distance rule which we have implemented before. Furthermore, the 4th branch is a continuation of the 3rd branch, if the algorithm managed to match the current driver to a new customer, then our program should update the status of our new customer from "idle" to "busy" and schedule another event of type 7 in our `ET` matrix. Note also that we only change the status of our customer but not our driver because we assume the algorithm will be able to match our driver to a new customer (immediately) once it has dropped off its previous customer. Branch number 5 guarantee that if our previous condition could not find the closest customer, then it will proceed to changing the status of the current driver from "busy" "idle" and updating its current position. To conclude, branch 6 ensure that when an event of type 7 is finished, it is removed from our the `EC` and `ET` matrix.

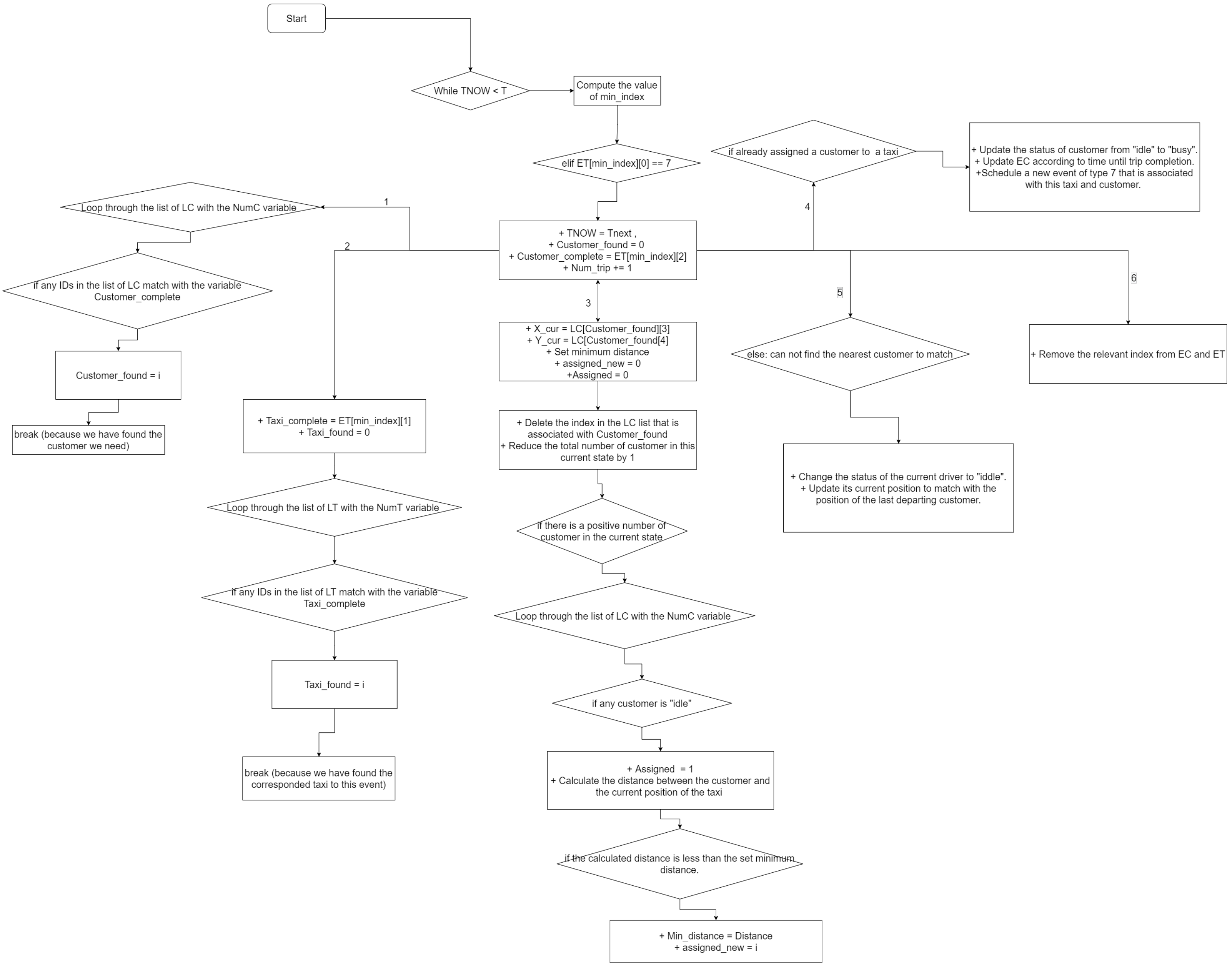
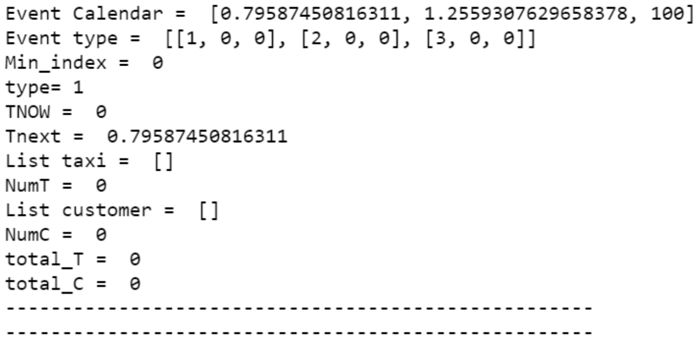

Start
While TNOW < T
Compute the value of min_index
elif ET[min_index][0] == 7
if already assigned a customer to a taxi
+ Update the status of customer from "idle" to "busy".
+ Update EC according to time until trip completion.
+Schedule a new event of type 7 that is associated with this taxi and customer.
Loop through the list of LC with the NumC variable
1
2
4
+ TNOW = Tnext ,
+ Customer_found = 0
+ Customer_complete = ET[min_index][2]
+ Num_trip += 1
6
5
if any IDs in the list of LC match with the variable Customer_complete
3
+ X_cur = LC[Customer_found][3]
+ Y_cur = LC[Customer_found][4]
+ Set minimum distance
+ assigned_new = 0
+Assigned = 0
else: can not find the nearest customer to match
+ Remove the relevant index from EC and ET
Customer_found = i
+ Taxi_complete = ET[min_index][1]
+ Taxi_found = 0
+ Delete the index in the LC list that is associated with Customer_found
+ Reduce the total number of customer in this current state by 1
break (because we have found the customer we need)
+ Change the status of the current driver to "iddle".
+ Update its current position to match with the position of the last departing customer.
Loop through the list of LT with the NumT variable
if there is a positive number of customer in the current state
If any IDs in the list of LT match with the variable Taxi_complete
Loop through the list of LC with the NumC variable
Taxi_found = i
if any customer is "idle"
break (because we have found the corresponded taxi to this event)
+ Assigned = 1
+ Calculate the distance between the customer and the current position of the taxi
if the calculated distance is less than the set minimum distance.
+ Min_distance = Distance
+ assigned_new = i

Figure 10: Flow diagram operation of event of type 7

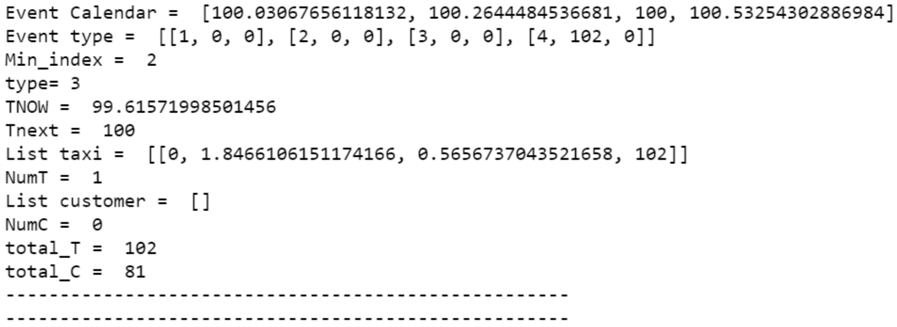
5 Model expected outcome demonstration and performance measure:

When designing our algorithm, we have a certain expectation for how each iteration (state) of one replication should look like. This provide us with a point of reference to debug and adjust our code accordingly. In the figure below, the audience should expect to see a stream of the simulation output and that every line should match accordingly to the other output above as well as below. In order to test and compare the output of our simulation, it is necessary and important that it can reproduce this stream of output provided with the same starting condition (i.e. set the seed value of the function `np.random.seed` equal to 0).



```
Event Calendar =  [0.79587450816311, 1.2559307629658378, 100]
Event type =  [[1, 0, 0], [2, 0, 0], [3, 0, 0]]
Min_index =  0
type= 1
TNOW =  0
Tnext =  0.79587450816311
List taxi =  []
NumT =  0
List customer =  []
NumC =  0
total_T =  0
total_C =  0
----------------------------------------------------
----------------------------------------------------
```

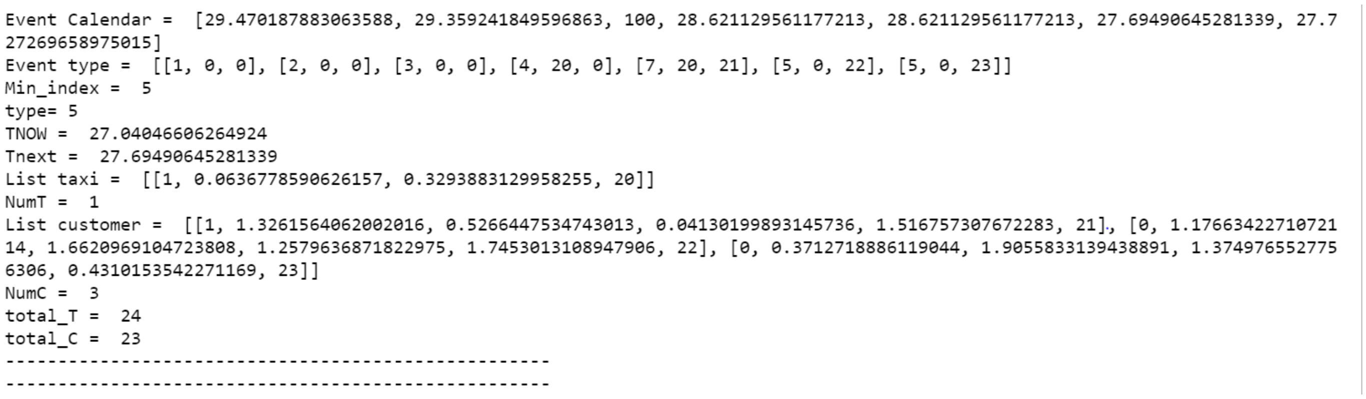
Figure 11: First simulation iteration



```
Event Calendar =  [100.03067656118132, 100.2644484536681, 100, 100.53254302886984]
Event type =  [[1, 0, 0], [2, 0, 0], [3, 0, 0], [4, 102, 0]]
Min_index =  2
type= 3
TNOW =  99.61571998501456
Tnext =  100
List taxi =  [[0, 1.8466106151174166, 0.5656737043521658, 102]]
NumT =  1
List customer =  []
NumC =  0
total_T =  102
total_C =  81
----------------------------------------------------
----------------------------------------------------
```

Figure 12: Last simulation iteration

To explain further what we mean by "the output should match accordingly", we will choose one arbitrary iteration as an example and attempt to clarify the output of that specific iteration. In (Figure 13), the simulation produce us this iteration when the simulation time `TNOW` is at roughly 27.040. The crucial point here is what is going to happen at the time `Tnext`, because that's when algorithm is deciding on the process of the next operation. When `Tnext` $\approx$ 27.694, the algorithm inform us that it is the event of type 5 (i.e. this is the event when customer run out of patience and leave the system) If we can skim through the list of the `Event Calendar` and started counting from 0 (left to right) to 5, then we will find the rough value of 27.694. If we also perform the same counting process for the `Event type` (`ET`) list, then we should expect to see that it is indeed the event of type 5 and customer number 22 is about to leave our system. Additionally, if we search through the list of customer (`LC`), we should expect to see the status of customer number 22 is indeed (0 = idle) reflecting his "frustration" of having to wait for so long without being assigned a driver. To confirm that the other process also work without issues, we can perform same counting process mentioned above. For instance, in the list of taxi (`LT`), it informs us that there is currently only one taxi (taxi number 20 is serving) in our system. If we check the list of `ET`, there is indeed only one event of type 4 (because there is only one taxi working, hence we should expect to only see that taxi working schedule. Right next to that taxi scheduled depart time, the event of type 7 is telling us that that taxi is currently serving customer number 21 and the both the taxi and customer status confirm this. For this specific example, the audience should also be aware that the `Min_index` is not always necessarily be equal to the type of event.



```
Event Calendar =  [29.470187883063588, 29.359241849596863, 100, 28.621129561177213, 28.621129561177213, 27.69490645281339, 27.7
27269658975015]
Event type =  [[1, 0, 0], [2, 0, 0], [3, 0, 0], [4, 20, 0], [7, 20, 21], [5, 0, 22], [5, 0, 23]]
Min_index =  5
type= 5
TNOW =  27.04046606264924
Tnext =  27.69490645281339
List taxi =  [[1, 0.0636778590626157, 0.3293883129958255, 20]]
NumT =  1
List customer =  [[1, 1.3261564062002016, 0.5266447534743013, 0.04130199893145736, 1.516757307672283, 21], [0, 1.17663422710721
14, 1.6620969104723808, 1.2579636871822975, 1.7453013108947906, 22], [0, 0.3712718886119044, 1.9055833139438891, 1.374976552775
6306, 0.4310153542271169, 23]]
NumC =  3
total_T =  24
total_C =  23
--------------------------------------------------
--------------------------------------------------
```

Figure 13: Output of an arbitrarily simulation run

Even though these output values are important in providing us with a reliable and functional simulation, we should remember that our initial goal is to test and compare the performance measure of the simulation under different parameters. Therefore our point of interest should lies solely on the performance measure output. There are many ways to assess the performance of our simulation, but for the moment, let's restrict ourselves to only a few popular performance measure that have been used over the year, such as: taxi utilization, system throughput, rate of customer abandonment (without receiving service) and average number of customer in the system, etc.

5.1 Efficiency driven performance measure

When it comes to designing and maintaining the matching algorithm, the platform designer will usually have to contemplate on the question of what type of optimal outcomes does he/she wants to reach. If the designer is concerned with system efficiency then he/she will focus on making sure that the number of taxis and customers don't stay idle for prolong period of time. If they are concerned with customer's satisfaction, then they wouldn't want the number of customer who left the system without receiving service becoming too high. After all the ultimate goal is to increase profit and so should the designer prioritise, however; the objective of this work isn't centering around determining which optimal outcomes (efficiency driven vs quality driven) yield better profits. But to rather create a functional algorithm and testing its performance under different parameters that can be used for later research and development. Hence, let us now move on to defining some important performance measures.

Despite the simplistic naming, taxi utilization is challenging to defined because the calculation involved is often not so straight forward. In the general, if the utilization is 70% then it means that 30% of the time, our taxis are idle. Utilization can be calculated as followed (5.1).

$$\texttt{utilization} = \frac{\texttt{total\_utilization}}{\texttt{total\_taxi}} \tag{5.1}$$

For the lack of a better naming choice, `total_utilization` is the total area under a step function. There is the step function because every time a taxi's status changes from "0" to "1", the area under that same step function increase slightly and decrease otherwise. In essence, the area below this step function tell us the total usage of all the taxi up until time `T`. Similarly, `total_taxi` is the area under another step function which tell us the total number of taxi arriving up until time `T`. If we divide `total_utilization` by `total_taxi`, we should expect to get a number that is strictly between 0 and 1 due to the fact that the area which explain the status of all taxi up until some time s can not exceed the the total area of the number of taxis that have arrived up to that same time point. Indeed, This ratio inform us of how often the taxis are being utilized up until time `T`. The audience should be aware that the `utilization` level here is the utilization of all the taxis in our system after the termination time `T`. If we decide to treat all taxis in our system after `T` as "one" big single server with n capacity (i.e. a shopping mall that can process or intake a large number of people at any one time), and every time a taxi become busy, the utilization level increase slightly by (let's say 1%). Then the utilization is well defined and it is indeed the quantity of (5.1).

For demonstration purposes, we have plotted the utilization graph. It is the ratio between **a)** the total number of taxis at each time step and **b)** the utilization of those taxis at each time step. In (Figure 14), a) is the step function which colored red and b) is the step function which colored green. This figure is more useful for demonstration purposes compare to the (Figure 13) and (Figure 15) because we can see "a bigger picture" instead of just seeing the raw outputs. For instance if we look at a the time when TNOW = 50, we can say that there are 7 taxis in our system and only 3 of them are being utilized and if we look further into the future where `TNOW` $\approx$ 85, we see that there are in total 4 taxis in our system at around that time with 100% utilization.

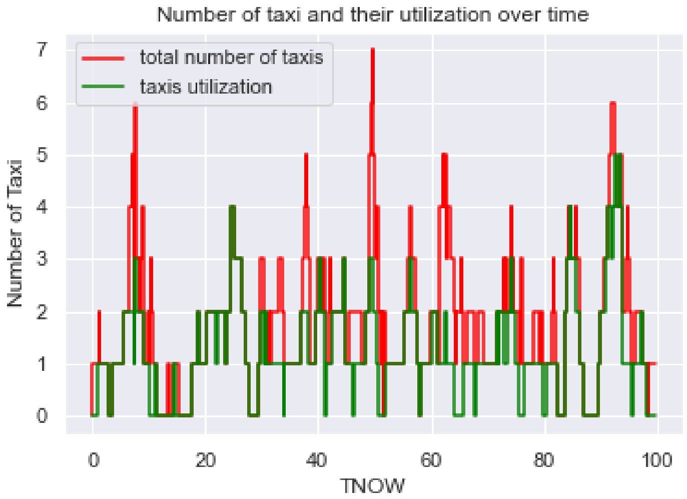


Figure 14: Number of taxis and their utilization over time

Aside from the utilization of taxi, we would also like to assess the rate at which the customer is being processed. In many literature, this quantity is defined as customer throughput we will sometimes use the term customer utilization. Customer utilization appraise us the proportion of time the customer are being served. If customer utilization is roughly 70%, then 30% of the time all of our customers spend waiting for a car. Additionally if one customer's total time spent in our system is 5 minutes, then base on the (70-30) utilization, he will have to wait for an average of 1 minute and 30 seconds (note that this consist of both waiting and pick-up time as described in (3.3)) .

$$\texttt{Customer\_throughput} = \frac{\texttt{total\_throughput}}{\texttt{total\_customer}} \tag{5.2}$$

`total_throughput` is the area under a step function which describe the number of time the customer are busy. `total_customer` is the area under an ever increasing step function which describe the total number of customer up until a certain time s. Even though the ways in which we compute these two quantity: (5.1) and (5.2) appear to be same, the fundamental difference between the two is the ways in which we design the taxi and the customer abandonment process in our code. Whenever a taxi enter our system, its abandonment is automatically scheduled. In essence, all of our taxis will have a designated leave time irrespective of whether it has served any customer and it will leave when that time come. Whereas, the customer abandonment time is only being scheduled if the system can not find an "appropriate" driver immediately. Another important condition is that if the algorithm manage to find a suitable driver within that abandonment time frame, then that customer abandonment scheduled is removed right at the instance of the match. To a certain extend, customer abandonment depend steadily on the number of taxi that is in the system at that state as well as the way in which we design the matching criteria (i.e. Euclidean distance, Manhattan distance). Using the same analogy as above, if we now consider all taxis to be the input and the customer to be the giant shopping mall, then the customer utilization tell us about how often our customer (i.e. many small shops inside a shopping mall) are being utilized as a whole by the drivers. Furthermore, the difference between this shopping mall and the shopping mall above is that many small shops within our mall will shut after some s if the algorithm can not manage to redirect the right driver to the those shops.

5.2 Quality driven performance measure

In addition to the efficiency driven measure, we also have the quality measure which allow us to observe and record the the level of customer's satisfaction. Such assessment metric are often referred to as the rate of customer abandonment, proportion of abandonment and the number of trip completion. If a platform designer's goal is to improve that rate of customer's satisfaction then he/she should make sure that these performance measure don't fall below a certain threshold.

Let us now define a simple performance measure which is the rate of customer abandonment. As defined in our coding language as:

$$\texttt{abandonment} = \frac{\texttt{NumC\_aban}}{\texttt{T}} \tag{5.3}$$

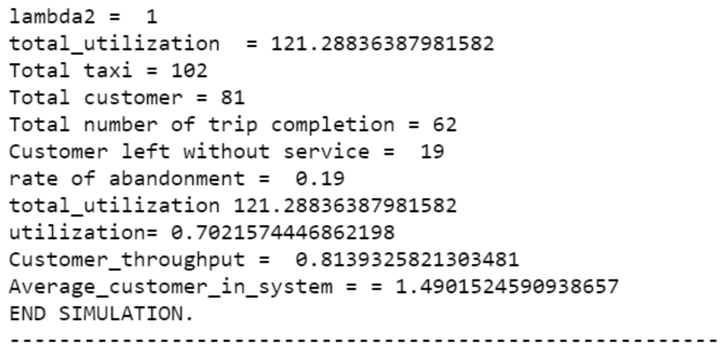
This ratio is self explanatory because it should describe the rate at which our customer is leaving the system per simulation unit time. `NumC_aban` is a variable use in our code to keep track of the accumulated number of customer that has left the system up until a certain time s. If we divide, `NumC_aban` by the termination time `T`, then we should get the explained ratio. Ideally, if platform designer's goal is boosting customer's satisfaction and increase in profit, he/she would want this ratio to be as small as possible if not zero. Unfortunately though, the ratio `abandonment` should increase linearly as the rate of customer arrival exceed the rate of taxi arrival. This is well justified because eventually there will be many customer for taxi to pick up at any one state.

Somewhat identical to the ratio `abandonment`, the proportion of abandonment tell us the number of customer that has left our system in relation to the total number who have arrived. It is calculated based on the rate of abandonment (5.3) divided by the rate of customer arrival (`lambda2`). For instance, if we set `lambda2` to be equal to 4 and the proportion of abandonment is calculated to be equal to 0.5, then it is advising us that half of those who have arrived will abandon without being served.

$$\texttt{Proportion\_abandonment} = \frac{\texttt{abandonment}}{\texttt{lambda2}} \tag{5.4}$$

Lastly, a performance measure that is considered to be partially in between efficiency and quality driven regime is the total number of trip completion. This quantity is calculated by adding 1 to the variable `Num_trip` every time a taxi have successfully taken its customer to his destination. Although simple, we could still compare this quantity to the total number of taxi that have arrived to our system and derive the ratio that can inform us of the average number of taxis that has successfully received a match. With this last definition, we can now proceed to the experiment phase.

To reiterate, one of our goal is to design the algorithm and then assess its performance measure under different starting parameter. Hence, it would not be prudent to conclude this section without showing the output of interest (Figure 15).



```
lambda2 =  1
total_utilization  = 121.28836387981582
Total taxi = 102
Total customer = 81
Total number of trip completion = 62
Customer left without service =  19
rate of abandonment =  0.19
total_utilization 121.28836387981582
utilization= 0.7021574446862198
Customer_throughput =  0.8139325821303481
Average_customer_in_system = = 1.4901524590938657
END SIMULATION.
----------------------------------------------------------
```

Figure 15: Simulation output

We have designed and formatted our simulation into a function (called `sim_taxi`) that can produce a different replication depending on the input parameters. Figure 15 above, is a solid demonstration of when `sim_taxi` take the inputs `sim_taxi(lambda1 = 1, lambda2 = 1, aban1 = 1, aban2 = 1, time_coef = 1, T = 100)`. As mentioned earlier, it is vital that these outcomes are reproducible and indeed with such input and the `seed` value set to 0, we can reliably reproduce this outcome. The audience may notice that despite the similarity in arrival rates, our system still have more taxi than customer at time `T`. This deviation is expected and negligible because after all this is just one replication. As we change the `seed` value or allowing our simulation to run for longer time (i.e. up until `T` = 1000, or even 10000 simulation step), then we should expect a ratio of 1-to-1 between drivers and customers.

6 Numerical experiments and models comparison

When setting up simulation, modelers usually have to decide which type of simulation to perform. The types of simulation depend whether our interest lies solely on the performance of the system over some period of time or the system performance in the steady state. In general, there are two types of simulation namely, steady state and non-steady state simulations. The nature of a system does not determine whether a simulation is a steady state or a non-steady steady simulations. In short, steady state simulation is a special case of non-terminating simulation. It is a simulation that have no definite star time and ending condition. The name does not imply that simulation won't terminate, but rather the termination time is determined by the modeler. Moreover, a system which is *not* a steady state simulation can also be a non-terminating simulation. For example, a queuing system where customers inter-arrival time is exponential distributed with rate 8 and services time are exponential with rate 4. After some time, we observe that the system will blow up and the probability that it will visit state k in the future converges to 0

$$\lim_{t\to\infty} \frac{1}{t} \int_0^t 1_{\{X(s)=k\}}(s)ds = p_k > 0 \tag{6.1}$$

$$\lim_{t\to\infty} \frac{1}{t} \int_0^t 1_{\{X(s)=k\}}(s)ds = 0 \tag{6.2}$$

Equation (6.1) describe the long run average of a steady state simulation, it simply informs us that our simulation will eventually revisit a certain state k with a probability p_k, regardless how long the system might take. Equation (6.2) simply mean that, if our process run for an infinitely long time, then the probability that it will visit stake k will converge to 0.

Along side steady state system, we also have non-steady state simulation (also known as terminating simulation). This type of simulation have a definite starting and ending conditions. For example, customers arrive to the bank at 9am when it's open and system end at 5pm. If we want to run the bank simulation for one week then the terminating time can be determined by multiplying the bank working time by the number of working days which will yield 40. Note that the run time of 40 is a simulation unit time but it can also be treated as though 40 hours of work have passed in real time.

Our simulation model is designed to be a steady state simulations because the rate of arrivals of taxis and customers is being cancelled out by the rate of abandonment of taxis and customers. For example, this statement implies that regardless of the time and the seemingly large number of customers and drivers in the system at a particular state, if the simulation can run for "long enough", the system will eventually revisit a state where it will have less number of drivers and customers. In a later section, we are going to determine the exact approximation for this "long enough" conditions (a.k.a the terminating conditions) by analyzing more closely the steady state nature of our system.

6.1 Distance measure

Before diving deep into some of our experiments. We would like to define the distance measure between any 2 points on a plane. After all, if we want to assign a taxi to a customer, we ought to know a way to measure how far apart those two points are. The way in which we choose the distance measure is important because it has an impact the performance measure of our system. It has been a well known fact that the shortest distance between any 2 points in a two-dimensional space is the Euclidean distance (ED). Since this work revolve around matching taxi driver to customer, it would not be realistic to utilize such measure because it does not taking into account the fact that there are obstacles that can prevent a vehicle from traveling straight from point A to point B. Nonetheless, we will still use ED in our experiment due to its simplicity and ease of computation. A different distance measure which is more realistic is the Manhattan distance (or another name Taxicab geometry). Manhattan distance measure the shortest path a taxi driver will take between city blocks. This measure is more realistic due to the fact that it taken into account the grid like structure of a city and object travelling from point A to point B cannot "move through" those city blocks. The red, yellow and the blue line demonstrate the shortest path under the Manhattan metric and the green line represent the shortest path taken under the ED (Figure 16).

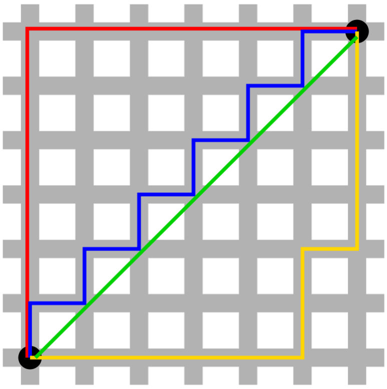


Figure 16: Manhattan distance and Euclidean distance [17]

For this part, instead of mathematically defining the distance measure for the Manhattan and the Euclidean distance, separately. We will define a generalized distance measure for both which is the Minkowski distance.

For any two points $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$ in $\mathbb{R}^n$, the Minkowski distance of order p is defined to be:

$$d(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{6.3}$$

For:

- $p = 1$, we yield the metric for the Manhattan distance.
- $p = 2$, we yield the metric for the Euclidean distance.

6.2 Experiments with the Euclidean metric under the nearest dispatch regime

In this section, we intend to present the result of some of our experiments when the algorithm is operated under the ED. When a customer or a taxi arrive, it greedily attempts to create a match as quickly as possible while still following the minimum distance rules. The initial goal of designing of the algorithm is to ensure that it produces an intended outcome. Once that is guaranteed, we would like to know how it will behaves given different inputs. As suspected, if we reduce the rate of taxi arrival and keep increase the rate of customer arrival, then the total number of customer abandonment should increase proportionately. Ultimately, it is our desire that the algorithm of this work be extended and developed further and help contribute to any future researches. We intended to design such algorithm so that it can reflect some aspect of the real world scenario while obeying the "Principle of Parsimony". We believe that an unnecessary complicated model is time consuming to develop and would require extreme computational power. Hence the simpler the better. The success of this work lies in the fact that the model have function as intended and that can be used as blue print for other research purposes.

In many of our subsequent experiments, the audience should be aware of the termination time of our simulation. Larger termination time mean the simulation is being run for longer time and that might be time consuming. If reliable results and analysis can be produced with a shorter simulation length then it will be done so and we will inform the audience of such choice. Additionally, the experiments will mainly focus on the two performance measures which are the utilization rate and the rate of abandonment. Even though we could have performed more experiments and producing more figures for the other performance measures such as the trip completion, customer utilization, etc. We believe that taxi utilization and rate of abandonment are a sufficient representation for our findings because from the two performance measures, we can deduce the behavior of the other performance measures. For example, utilization can somewhat give information about the number of trip completed and the rate of abandonment can reasonably describe the level of customer's satisfaction.

To first showcase the functionality of the algorithm, a simple result will be produced. As stated earlier, the algorithm can simply be called by using the `sim_taxi` function. This time, the input of `lambda2` will be changed slightly. Instead of fixing it to a constant number, we will vary it by running the function through a `for` loops (the function is named drange and more details about such function can be found in the appendix) with different rate of `lambda2`. The system configuration is sim_taxi(lambda1 = 1, lambda2 = index, aban1=1, aban2=1, time_coef=1, T=1000) for index in (0.01, 20, 1). More details about the reason for such small starting step is explained below. Undoubtedly, different values of lambda2 will yield a slightly different performance. We will also show using plot how different rate of customer arrival affect the rate of customer abandonment in (Figure 17).

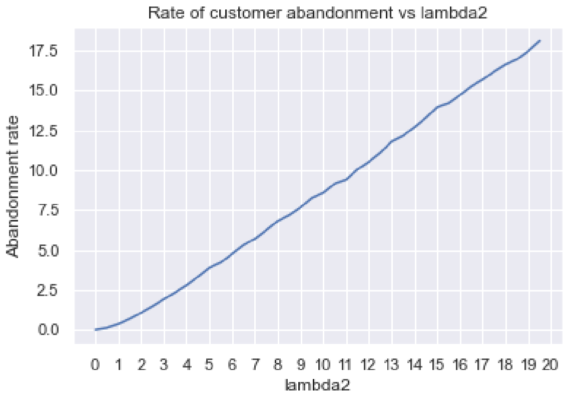


Figure 17: Abandonment rate vs lambda2

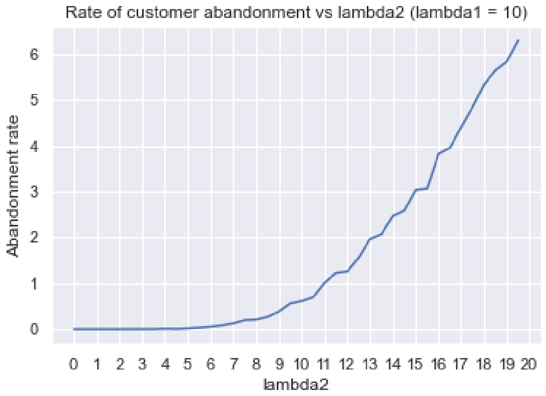


Figure 18: Abandonment rate vs lambda2 with (lambda1 = 10)

As the rate of taxi arrivals is fixed, we observe that the rate of abandonment increase linearly with the rate of customer arrival. Due to the linear relationship, abandonment rate appear to be depend on the total number of customer in the system at any time. This result indeed correspond with our expectations. Furthermore, an interesting extension of the current experiment would be to increase the rate of of taxi arrival `lambda1` from 1 to 10 (Figure 18). With such an increase in the number of drivers, we should expect the abandonment rate to be extremely low until `lambda2` exceed `lambda1` (i.e. `lambda2` $\gg$ `lambda1`). As seen from (Figure 18), the performance of the taxi is indeed struggled slightly while `lambda2` $<$ `lambda1`. However, as `lambda2` increase beyond the value of 10, the abandonment rate rise sharply afterwards.

In the following experiment, the input parameters is set as followed: `lambda1 = 1, lambda2 = index, aban1= 1, aban2=1, time_coef=1, T=1000`. The `index` variable will iterate through the `drange` function in the range `(start, stop, step)` of values that start from 0.01, stops before reaching the value of 11 with the step size = 0.2 (i.e. `(0.01, 11, 0.2)`). This mean that when the function `sim_taxi` have finished iterating through 0.01 and produce a replication, it will then continue to generate the performance measure for when `lambda2` = 0.21 (= 0.01 + 0.2) and produce the second replication, the same procedure apply for the next index: 0.41 and so on and so forth. After the function has cycled through all the indexes, it will produce a figure which can explain how changes in customer arrival rate will affect the rate of taxi utilization.

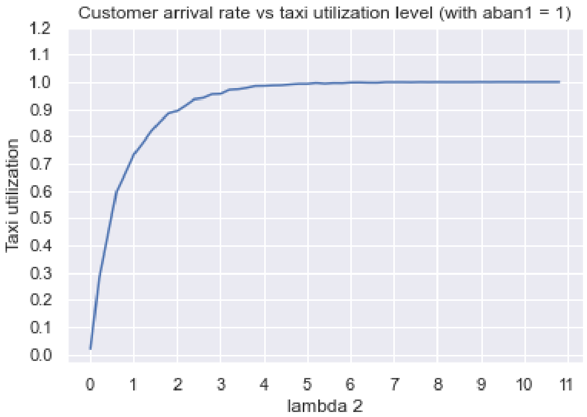


Figure 19: Utilization vs Lambda2

Indeed, we should expect a concave function (Figure 19) because utilization rate is a values that lies strictly between 0 and 1 (inclusive). While the rate of taxi arrivals may stay constant, any extra increases in the input `lambda2` can only in either increase utilization rate sharply (if the value of `lambda2` is small) or increase it slightly (if the value of `lambda2` is large). Moreover, we can also observe that when `lambda2` is small, modest changes to the input *lamda2* (from 0.01 to 0.21) will result in a steep increase in utilization level. This is expected because the more customers arrive, the more there is to occupy the empty taxis. This also explained our reason for choosing such a small starting step, because we wanted to showcase the practicality and the sensitivity of the algorithm. Indeed, it's behavior is in coherent with most realistic settings. On the other hand, if we look at the tail-end of this concave function, the gradient approaches to almost zero as `lambda 2` plateau at around the value of 4 because once all the taxis have reached saturation (i.e. all of them is occupied by at most one customer), then any additional customer arrivals to the system could only result in an increase in waiting time along with the rate of customer abandonment. To demonstrate this point further, another experiment is conducted with the configuration: `sim_taxi(lambda1 = aban, lambda2 = index, aban1 = 1, aban2=1, time_coef=1, T=1000)`. This particular experiment involve varying the rate of taxi arrival (`lambda1` = aban where (aban= 1,5 or 10) and the rate of customer arrival (`lambda2`) by cycle(/loop) the latter through the drange function and cycle the former through the fixed value (aban= 1,5 or 10) with the `enumerate` method. We present the result of this experiment in (Figure 20).

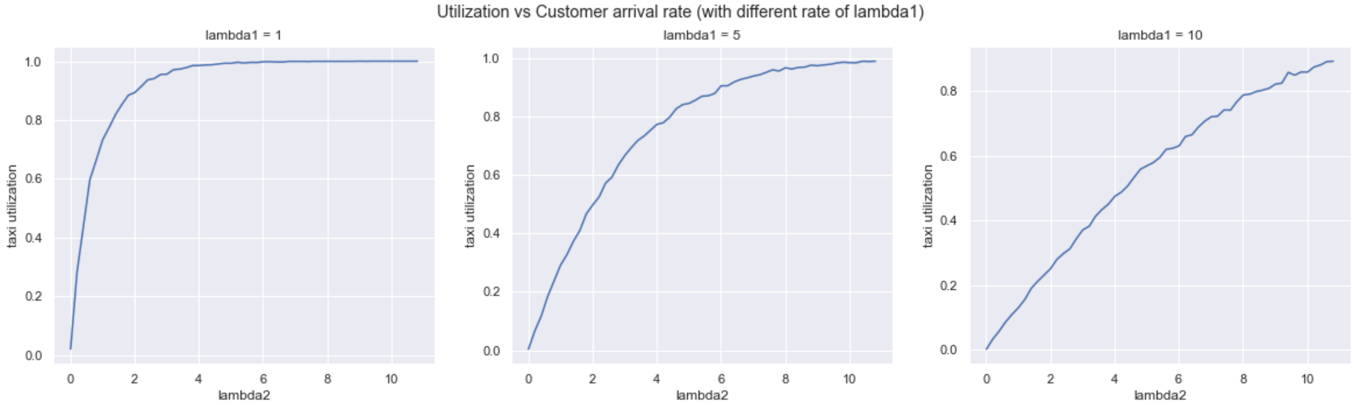


Figure 20: 3 graphs: Utilization vs Lambda2 (changes in lambda1)

As the audience might observe, the far left figure is identical to (Figure 19). However, when the input parameter `lambda1` is increased from the value of 1 to 5, the rate of utilization rises less sharply compare to the far left figure in (Figure 19); it would require the input `lambda1` to exceed the value of 5 for the utilization rate to reach 100%. This occur due to the fact that when the input `lambda1` (= 5 or 10) is large, the system have more resources to accommodate for an ever increasing in the number of customer arrivals. Furthermore, as long as the value of `lambda1` is significantly greater than `lambda2`, then the utilization rate won't likely to reach 100%. As a side note, higher utilization rate is not always optimal in a realistic environment because drivers will often require a short break every few hours and it is dangerous for our drivers to operate at 100% utilization at all time. Hence, it is not our goal to design a system that can maximise utilization rate, we merely use utilization rate as a point of reference just to measure the accuracy and the behavior of our model.

After realizing how utilization rate depend on the number of taxi, the number of customer and the ratio between the two. We can proceed with our next experiment which is almost identical to some of the previous experiments but instead we will now varying the rate of taxi abandonment by iterate it through the `enumerate` method (i.e. cycle through these three specific value `aban1` = 1, 5 and 10, while at the same time still iterating through `lambda2`).

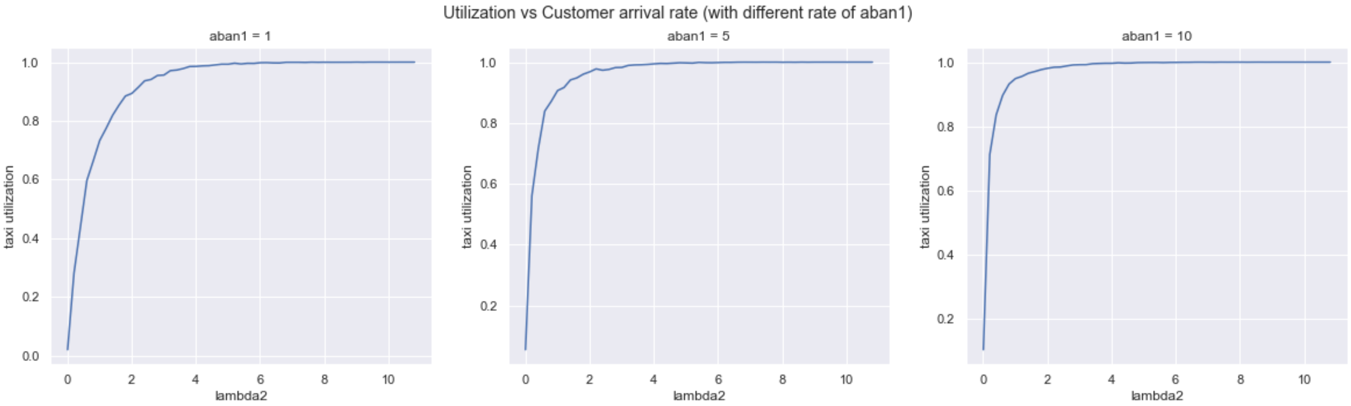


Figure 21: 3 graphs: Utilization vs Lambda2 (changes in aban1)

From the initial observation, when the input `aban1` is low, the concave function is less steep compare to when the input `aban1` take the value of 5 or 10. With `aban1` = 1, the function plateau right after `lambda2` have outpaced the value of 4 whereas when `aban1` = 5 or 10 the function plateau more rapidly. Notably, in most of our experiments, the rate of utilization plateau at 100% even before `lambda2` reach the value of 4. This should be expected because the more often the taxi leaves the system, the less competition there are for the remaining taxi; while the rate of customer arrival increase steadily, the number of taxi reduce sharply, in essence this creates almost a balancing effect because the less taxis there are to occupy, the more rapidly the utilization level to reach 100%. This bring us to an interesting strategy for algorithm designers. Irrespective of the rate of customer arrival, if at any point the designer wants to increase the utilization rate, they can do so by restricting the total number of hours a driver can work; provided that the designer have control over the number of driver who is currently working and the information on the number of driver who is going to enter work soon.

So far we have mainly observed how the rate of utilization and the rate of abandonment is influenced by the changes in the input `lambda2`. Now we are going to explore how those performance measures is affected by changes in the input `lambda1`. The system configuration for the upcoming experiment is: `sim_taxi(lambda1 = index, lambda2 = 1, aban1= 1, aban2= 1, time_coef=1, T=1000)`; index will cycle through the `drange` function which start from 0.01, with step size = 0.2 and end before reaching the value `lambda1` = 20.

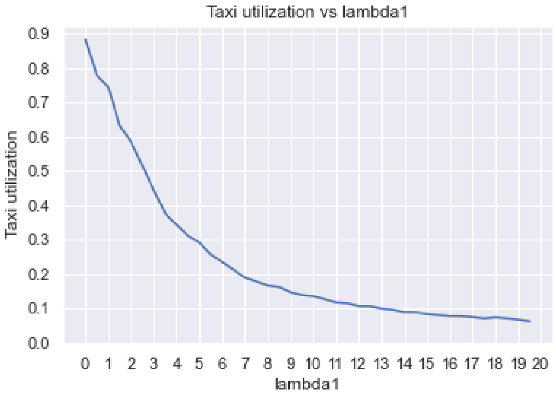
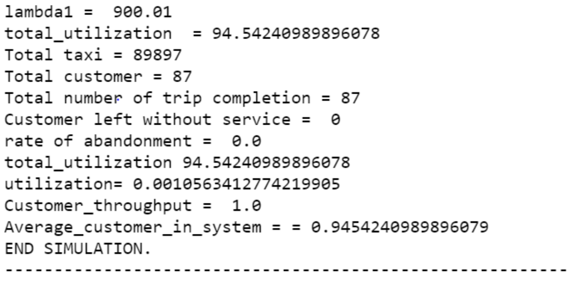


Figure 22: lambda1 vs taxi utilization



```
lambda1 =  900.01
total_utilization  = 94.54240989896078
Total taxi = 89897
Total customer = 87
Total number of trip completion = 87
Customer left without service =  0
rate of abandonment =  0.0
total_utilization 94.54240989896078
utilization= 0.0010563412774219905
Customer_throughput =  1.0
Average_customer_in_system = = 0.9454240989896079
END SIMULATION.
--------------------------------------------------------
```

Figure 23: Experiments results for large lambda1

In contrast to earlier experiment, changes in `lambda1` create an opposite effect compare to changing in `lambda2`. This time utilization decrease as the rate of taxi arrivals increases (Figure 22). This effect is counter-intuitive but reasonable, because the higher the number of taxis in the system, the less the rate of utilization. As more taxis arrive to the system, it requires more time as well as more customers for utilization to remain high. The number of customer arrivals is fixed so after a certain point, the overwhelming number of taxis is going to overtake. Recall that we are considering the utilization of *all* taxis. Certainly, for a small fraction of *all* those taxis, their total utilization will be close to 100% due the fact that there are a customers arriving (i.e. `lambda2` = 1). But because utilization rate is calculated for all taxis, hence the utilization of those small group of taxis is being over-shadowed by the utilization of those that don't have any customers. Despite its insignificant, these small group of cars are the reason why the concave function will approach the line x = 0 as an asymptote but it will never reach it (unless the simulation run for infinite time). Indeed, this claim is backed by the result of the experiment: `sim_taxi(lambda1 = index, lambda2 = 1, aban1 = 1, aban2 = 1, time_coef = 1, T = 100)`, where index will loop through `(0.01, 1000, 100)` (Figure 23). Clearly, we can observe that even for when `lambda1` = 900, the utilization rate is 0.00105.

Additionally, we would also like to see how drastic changes in the input `lambda2` might affect the rate of utilization. Note that we are still observing the effect on utilization as `lambda1` changes, but we would like to know whether higher rate of `lambda2` might alter the decreasing trend of taxi utilization.

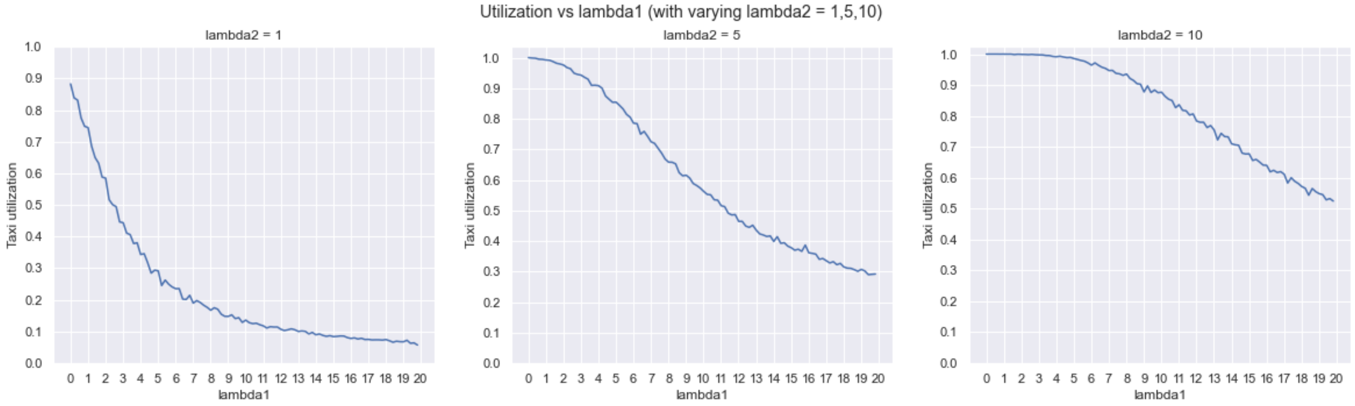


Figure 24: Utilization vs Lambda1 (with changes in lambda2)

From the result of the experiments, we can conclude that utilization rate decrease less drastically as long as there are a reasonable size of customers in the system or is going to arrive to the system. Indeed, the far left figure confirm that with a high number of customer arrivals, we see a slower rate of reductions in utilization level as compared to when `lambda2` = 1 or 5.

6.3 Experiments with the Euclidean metric under the nearest dispatch regime (with delay matching)

For this section, we want to demonstrate ways in which we can affect the utilization level other than just changing the arrivals input, the abandonment input, etc. By slightly altering how the algorithm matches driver to customer, we have managed to reduce the utilization level while keeping the input parameter constant. To explain, instead of greedily matching the driver and customers as soon as they arrive, we instruct the program to hold out and wait for the number of drivers and the number of customers have reached a certain level. Note that this is quite similar to instructing the algorithm to wait for a certain time before beginning the matching process. To be precise, when a taxi has arrived, the algorithm check whether there are more than 5 customers in the system and if not, the system will not begin the matching process. On the other hand, when the customer arrive, the algorithm will check if there are already more than 5 taxis in the system and if yes then, the system will begin the matching process. This one sided cross checking matching protocol is different from the matching protocol where it has to wait until the system have accumulated more than 5 drivers and 5 customers because for the one-sided cross checking protocol, if there are 2 taxis and 6 customers, then the algorithm will start matching them. whereas with the latter matching protocol, the system will have to wait until exactly 5 customers and 5 drivers have arrived. To add also, the one-sided cross checking matching protocol depend heavily on the fact that arrivals will eventually come. For the newly arrived drivers and customers, the algorithm will only perform this one-sided cross check for them. If there is no drivers and no customers to arrive then the matching process won't begin. In hindsight, this could undoubtedly create a scenario where there are a high number of taxi and customer abandonment if the input `aban1` and `aban2` isn't set to a small value. As a point of reference, we will call the described one-sided matching protocol the delayed-system.

For these upcoming experiments, we would like to observe how might the original Euclidean distance system (non-delayed system) behaves under the same parameters compare to the delayed-system. Note that we purposely only altering the matching protocol while keeping the simulation input parameters the same so in a later section, we could make a fair comparison on the efficacy of the different dispatching protocol. If there are too much variations and disparities between the two systems (such as, changes in the distribution of taxis, customers or changes in the input parameters), it might be too challenging in pin pointing the differences in the two system and the cause of the changes.

For the delayed-system experiment, we will choose a relatively large waiting condition (i.e. the one-sided cross checking work as followed: the system wait until a new taxi have arrive and at the moment of its arrival if there are more than 5 customers waiting in the system, the matching process begin, the process work in a similar fashion as for when a new customer arrive). If the result of our experiment is significant, we will make a comparison in a later section. But for now, we are simply just going to perform a few experiment and presenting the results under the configuration with the inputs as followed: `lambda1 = 1, lambda2 = index, aban1= 1, aban2=1, time_coef=1, T=1000`. Similar to experiment (Figure 19), step size will be chosen to be: `(0.01, 11, 0.2)`).

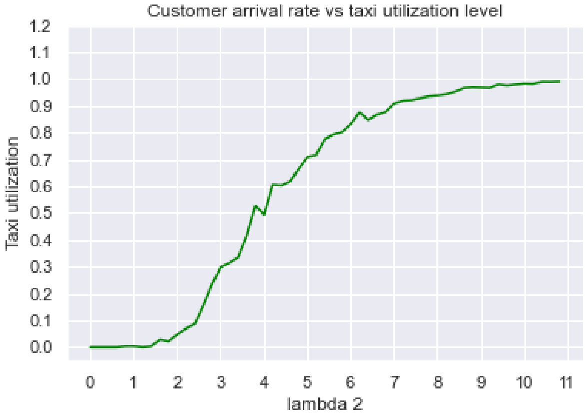


Figure 25: Utilization vs Lambda2 (with delayed)

According to the result, we are observing a slightly new trend. For low number of customer arrival (lambda2), the utilization rate stay flat at 0 because customers have a certain patience time. If they have to wait until the system can reach 5 taxis then they are just going to leave. Additionally, the drivers are also leaving because they have a designated work time that is controlled by the input (`aban1`). If we delve deeper into the results produce by these outputs, then for relative small value of `lambda2`, the taxis and the customers are indistinguishable. Meaning when taxis arrive for work, their abandonment/shift (`lambda1`) time is immediately scheduled. Similarly, when the customers arrive, their abandonment time are also being scheduled immediately because of the condition of the delayed system. So only for a relatively large enough value of `lambda2` or `lambda1` or both, then our system could possibly revert back to the "original" greedy matching (without waiting for accumulations) setting. Note also that the utilization rate only begin to improves once the ratio of customer to taxi have surpassed 2-to-1. This make sense because the higher the rate of customer arrivals the more likely that the newly arrived taxi can jump start the algorithm matching process. Additionally, if our prediction is accurate, then irrespective of the slight deviation at the beginning, as `lambda2` become larger, we should expect an almost identical concave function to the one in experiment (Figure 19).

Next we would also like to know how this delayed system might respond according to the different rate of `lambda1`. The same configuration and the step size will be identical to experiment (Figure 25).

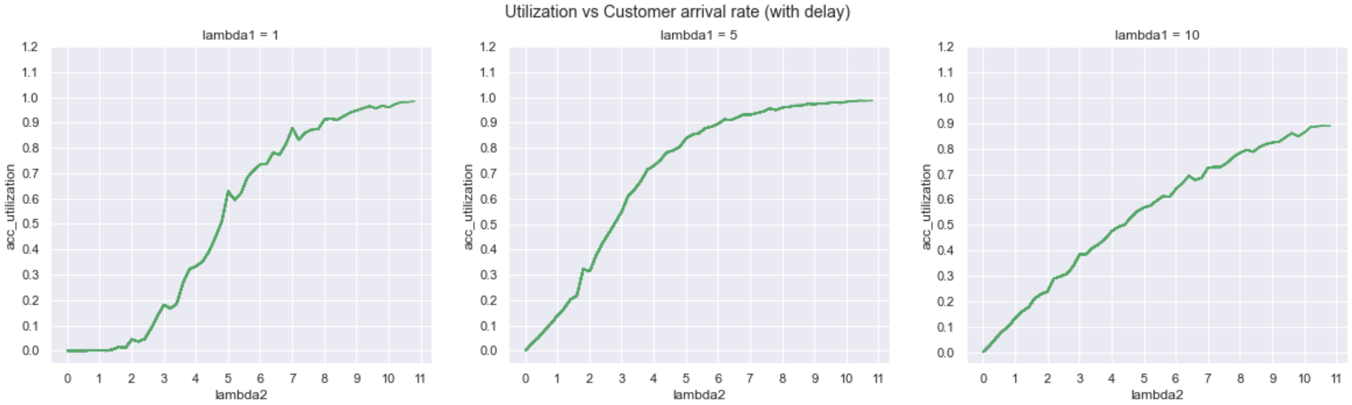


Figure 26: Utilization vs lambda2 (with changes in lambda1) delayed system

Notice that the very far left figure in (Figure 26) is identical to figure in (Figure 25). However, as the input `lambda1` was set to equal the delayed (hold out) conditions, the delay effect starting to wear off and we see that the function start to shift towards a familiar concave function just as in (Figure 19). The center figure demonstrate this point succinctly with the utilization rate rise more sharply as compare to (Figure 25). Indeed, this should happen because the matching environment is less strict (i.e. for every newly arrived customers, the rate at which a match will happen is higher for a high rate of `lambda1`). If the system already have a large number of taxis arriving then more likely a match will occur. As we have anticipated earlier, indeed our system is going to revert back to the original greedy matching system (original ED) when `lambda1` and `lambda2` are relatively large because when the two inputs rate are larger than 5, the speed at which the delayed-system match drivers and customer should almost be identical to the matching rate of the non-delay system; i.e. the far right graph in (Figure 26) is almost identical to the far right graph of the (Figure 20).

6.4 Experiments with the Manhattan metric under the nearest dispatch regime

The purpose of this section is to familiarise the reader with the Manhattan system configuration. We plan to perform comparisons and output analysis in a later section.

The reason why we presented the performance measure results for the Euclidean distance (ED) system first was because we started out designing the algorithm that way. As we have progressed through this work and grown more knowledgeable, we begin to acknowledge some of our earlier shortcomings when designing the algorithm. Indeed, the shortest distance between any two points on an empty surface is the Euclidean distance. However, if this work were to applied to a realistic settings, we have to taken into consideration the nature of the environment of the application. One thing for certain is that a taxi can not travel through any objects nor buildings, but rather they operate in a city environment which has grid like structure. Hence, the distance calculations have to be adjusted to take into account that fact and the Manhattan distance mentioned in (Section 6.1) is the right candidate.

As per anticipation, slight modification in the calculation of the distance should not cause any great fluctuations in how the system perform. Namely the performance measures such as the rate of utilization or the rate of abandonment should not be too different between the Manhattan system and the Euclidean system.

6.5 Experiments with the hot-spot distribution of the locations of taxi under the nearest dispatch regime

The upcoming experiment involve changing the distribution of the locations of customers arrival. An example of this is when people are trying to get home on a heavy rainy day after a visit to the concert hall. In a way, this create a sort of "hot-spot" location where customers coordinates are mainly concentrated inside a bounded region (i.e. customers concentration around the concert hall waiting for taxi to pick them up) of the two dimensional Euclidean plane. So far in the algorithm, such Euclidean plane have been chosen as the square with height and width of 2 units starting from 0. As mentioned before, this region is generate by using the Python function: `np.random.uniform(0,2)` (not inclusive of the end point). Now, to generate the so called hot-spot location, we simply adjust the X and Y coordinates arrival of the customers from uniform(0,2) to triangular(left,mode,right) = triangular(1.0,1.1,1.2). *Left* represent the lower limit, *mode* represent where the peak of the distribution and *right* is the upper limit. All samples drawn from the `triangular(1.0,1.1,1.2)` will be totally contained inside by the *left* and the *right* limit. By selecting the left, the mode and the right to be a specific value, we ensure that the triangular distribution only sample from within this range and furthermore, this range is guaranteed to be contained within the (0,2) square. For demonstration purposes, 50 samples (Figure 30) will be generated from the triangular distribution with left = 0.8, mode = 1.0 and right = 1.2. The samples will be contained inside a bounded region and that region is also being contained in the (0,2) square.

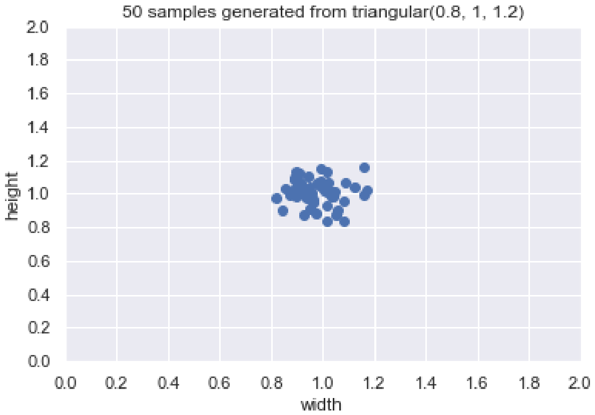


Figure 27: 50 samples generated by triangular(0.8, 1, 1.2) contained in the (0,2) square

In a later section, this concept is going to be implemented into our model. We will called this model the hot-spot system and it will then be compared with the original Euclidean distance (ED) system.

6.6 Experiments with the ED system under the furthest dispatch matching protocol

As already discussed, the ED system is a greedy matching algorithm that assign the closest taxi to the closest customers. One may raise the question such as what would happen if the algorithm instead assign the furthest away taxi ? Remember that, this ought to be more specific because the designed algorithm perform two-side matching. Indeed it can match the furthest taxi to a customer but it ought to know where that customers location is in relation to the taxi. For this section, we would like to show case a different type of assignment protocol, namely the furthest dispatch. As the name suggest, this matching protocol assign the furthest away taxi to the furthest away customer. The system operate as followed, when a taxi arrive to the system, if there are idle customers, the algorithm calculate a distance between that taxi and all the idle customers (by pairing them sequentially). If the distance between that taxi to any particular customer is the largest among the distance calculation to other customers then a match is made. When a new customer arrive, a similar matching process is applied.

6.7 Output analysis

When trying to analyze the output of our simulations (Figure 15), it is not satisfactory to just provide a single value or a point-estimate of the performance measures. Every time the simulations is cycling through a different input parameter of a different `seed` value (/a different stream of random number), we obtain a slightly different outputs. For instance, the performance measure outputs of any systems on a Monday is different from the performance measure outputs of those system that is being run on a different day. Therefore, to reliably produce an accurate estimates for the system, we need to run our simulations multiple times with different starting condition and use the Central Limit Theorem (CLT) to produce a confidence interval estimates for the outputs of interest. This simply mean that we ought to run our simulation with different `seed` value and build a confidence interval.

Theorem 1 (Central Limit Theorem)**.** *Let $X_1, X_2, \ldots$ be independent identically distributed (i.i.d) random variables with mean μ, variance σ^2 and n is the number of sample that is included in the sample mean $\left(i.e., \bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}\right)$. Then*

$$\frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}} \tag{6.4}$$

approaches to a standard normal random variable. Note that the quantity $\frac{\sigma}{\sqrt{n}}$ can also be referred to as the standard error.

What theorem (1) tell us is that, if we have a sequence of i.i.d random variables with finite mean and variance. Then for large n, (6.4) will converge to a normal distribution with mean μ and variance σ^2. Additionally, if we know $\bar{X}$ and σ, then for large value of n, the probability that the confidence interval (CI) (or the critical region) $(-z_{1-\alpha/2}, z_{1-\alpha/2})$ will contain the quantity (6.4) is roughly $1-\alpha$. Rearranging the equation (6.5):

$$P\left(z_{1-\alpha/2} \leq \frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \leq z_{1-\alpha/2}\right) \tag{6.5}$$

we yield (6.6):

$$P\left(\bar{X} - z_{1-\alpha/2}\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + z_{1-\alpha/2}\frac{\sigma}{\sqrt{n}}\right) \tag{6.6}$$

Lemma 2. *Supposed that we know $\bar{X}$ and unknown σ, then for n large, the quantity $\frac{\bar{X}-\mu}{\frac{S}{\sqrt{n}}}$ will instead follows a student t-distribution with $n-1$ degree of freedom; where α is the significant level and S is the sample standard deviation, that will replace σ, calculates as followed:*

$$S^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}) \tag{6.7}$$

Proof.

$$\begin{aligned} 1-\alpha &\approx P\left(-t_{1-\alpha/2,n-1} \leq \frac{\bar{X}-\mu}{\frac{\mathsf{S}}{\sqrt{n}}} \leq t_{1-\alpha/2,n-1}\right) && (6.8)\\ &\approx P\left(-t_{1-\alpha/2,n-1}\frac{\mathsf{S}}{\sqrt{n}} \leq \bar{X}-\mu \leq t_{1-\alpha/2,n-1}\frac{\mathsf{S}}{\sqrt{n}}\right) && (6.9)\\ &\approx P\left(\bar{X} - t_{1-\alpha/2,n-1}\frac{\mathsf{S}}{\sqrt{n}} \leq \mu \leq \bar{X} + t_{1-\alpha/2,n-1}\frac{\mathsf{S}}{\sqrt{n}}\right) && (6.10) \end{aligned}$$

□

Similar to the case with known σ, if we can calculate both the left and the right hand side of (6.10), then we can conclude that with $(1-\alpha) \times 100$ confident, our interval (6.10) contain the true expected value.

Equip with this knowledge, we will can now attempt to produce a more precise estimate for the utilization rate that was produce in experiment (Figure 15). This time the output of interest will take the form of (6.11). We will use the Python `range` function (from (0,50)) on the configuration: `sim_taxi(lambda1 = 1, lambda2 = 1, aban1 = 1, aban2 = 1, time_coef = 1, T = 100)`. $\bar{X}$ and S will be the sample mean and the sample standard deviation computed from a list which was set up to collect and store the utilization rate produced from each seed value. Each seed for which the function `np.random.seed` will cycle through produce a different performance measures. Furthermore, the student t-distribution can be calculated either through a manually written function or by using the `scipy` library in Python (more details about the code of the function and the `scipy` command can be found in the appendix). Regardless of the methods, both approaches should still yield the same result.

$$\left(\bar{X} - t_{1-\alpha/2,n-1}\frac{\mathtt{S}}{\sqrt{n}}, \bar{X} + t_{1-\alpha/2,n-1}\frac{\mathtt{S}}{\sqrt{n}}\right) \tag{6.11}$$

Hence, with 95% confidence, the following interval will contain the true utilization rate of the experiment (Figure 15).

$$\left(\bar{X} - t_{1-\alpha/2,n-1}\frac{\mathtt{S}}{\sqrt{n}}, \bar{X} + t_{1-\alpha/2,n-1}\frac{\mathtt{S}}{\sqrt{n}}\right) = \left(0.72544 - t_{1-\alpha/2,49}\frac{0.04047}{\sqrt{50}}, 0.72544 + t_{1-\alpha/2,49}\frac{0.04047}{\sqrt{50}}\right) \tag{6.12}$$

$$= (0.71394, 0.73694) \tag{6.13}$$

This confidence interval gives us more assurance of the accuracy of the experiment as the true performance measure lies inside this interval. However, the results are limited by the method. So far, in most of our experiments, the terminating time is chosen arbitrarily to provide a rough analysis on the behaviour of our model. This is also the case for the CI estimation of our true performance measure. Some can argue that perhaps, this arbitrary choice is acceptable, but unfortunately there are negative implications for the results. There are two problems with choosing the termination time arbitrarily. The first is outlined by Ergodic theory [1, 28] which states that, the performance measures of a system converges to the true performance measure as time t approaches infinity. This creates a conflict as we can not run our system until infinity because it would take an infinite amount of time. However, as our model is a steady state simulation, which means that we want to observe the behaviour of our simulation under the steady state so that we can obtain the true performance measure, we must allow our simulation to run for "long enough". If we terminate our simulation before it reaches the true steady state then our result is unreliable and if we run for too long, we will be wasting time and computational power. But one might ask, how long is "long enough"? Hence, as we cannot choose infinity, an arbitrary amount of time must be chosen. As a result, we must settle for running our system for a finite time and obtaining the performance measure estimator based on this limitation. As we produce the performance measure estimator, we get the second problem which is the initialization bias. This occurs when the performance measure of the first few customers negatively weights down the estimated performance measure of our system. For instance, if we start our steady state simulation at time point 0, and customer 1 arrives, then we can expect that the rate of abandonment for this particular customer will be lower than the true performance measure (if not 0) because they have no waiting time. The same will apply for the second, the third and the fourth customer and so on. As a whole, if we include the abandonment rate of these customers in the calculation abandonment rate estimator calculation, then it will negatively bias our estimator. In other words, the abandonment rate estimator will be smaller than the true rate. If we could hypothetically run our system until infinity, then this initialization bias is negligible. But because we can not run for infinity, hence we have to devise a heuristic approach to address both the issue with choosing the right termination time and dealing with the initialization bias.

One way to resolve the initialization bias is to discard them from the performance statistic calculations. There are two methods that will deal with the initialization bias problem in this manner. Namely, the Truncated Replications (TR) and the Batch Means Methods (BMM). For the TR, the initialization bias period or so called the warm-up period (i.e. t_0 to t_w) is discarded while multiple replications are being performed and the statistic collected will be used to calculate the estimated performance measure. For the BMM, one long replication is performed, the warm-up period is still there but the length of such period might be different compare to the TR (denote this length to be t_0 to t_w = cut off point). Similar to TR, the statistic collected up until the end of the warm up period will be discarded. Instead of carry on collecting the performance statistic after the warm-up period, we will instead reset all the collected statistic at the the cut-off point t_w. From the time t_w to another fixed time point which we might choose to be t_1, the simulation is run, the statistic is collected and then reset once again at the t_1 cut-off point. The same will be done for the statistic collected in interval (t_2, t_3), (t_3, t_4) and so on. The disadvantage of TR and BMM is that both methods require us to discard the warm-up period which mean computational time is wasted. If a simulation system happen to be complex, the interval of the warm-up period might be lengthy (otherwise the initialization bias would still persist) which mean that more computational power will be wasted. Not to mention the fact that the amount that we discarded when using the TR methods is equivalent to the number of replications produced. Additionally, the BMM have issue with dependency between collected statistics because the performance statistic depend on state of the system. We are concerned with the dependency because according to the CLM, the collected statistic (collect performance measure from each replication) have to be independent and identically distributed for building our CI. It is irrelevant whether the performance statistic is reset because the BMM perform only one long replication and the future statistic (i.e performance measure) depend on the past statistic (performance measure) to a certain degree. For instance, the average cycle time of the 10th customer will be almost identical to the average cycle time of the 9th customers because they are facing the same queue length. Therefore, we have to someone how find a way to eliminate the dependency as well as the issue with the initialization bias.

Due to the disadvantages of both the TR and the BMM, we will circumvent the initialization bias problem with a different approach. Namely, We will use the fact that our system is a steady simulation to remedy the fluctuations introduced by the initialization bias. By performing another experiment with the configuration: `sim_taxi(lambda1 = 1, lambda2 = 1, aban1= 1, aban2=1, time_coef=1, T=20000)`, we wish to observe whether our system performance measures will reach the steady state. Steady state mean that the performance measures cease to fluctuate greatly and the collected statistic is quantifiable as the simulation length approach infinity. If our system can reach said steady state in finite time, then we claim that the initialization error is negligible. Because irrespective of the fluctuations during the warm-up period, as long as our performance measures converges to a certain value at the steady state, then we can neglect the initialization bias and the warm-up period altogether.

As a remark, for our particular system, the convergence to the steady state in finite simulation length is quite versatile. In essence, our system will converge to a steady state as long as one of our input parameters is fixed. This statement is valid because it does not matter how much the other input parameters changes, as long as say for instance, `lambda1` is constant then those finite number of taxi could only be utilized up to a certain threshold and to go above that threshold would require an increase in `lambda1`. Moreover, if we were to observe the number of trip completed as `lambda2` changes, then it does not matter what we set our simulation length to be, the number of trip completed will eventually reach the steady state.

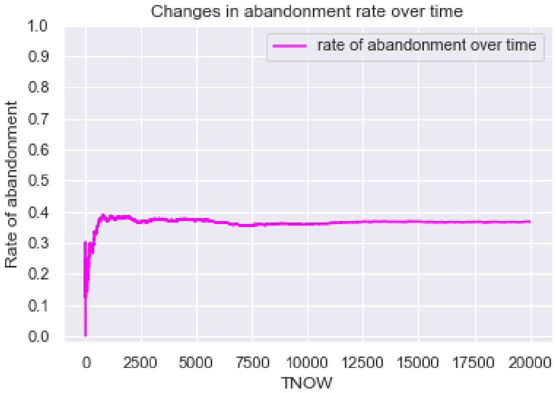


Figure 28: Changes in abandonment rate over time

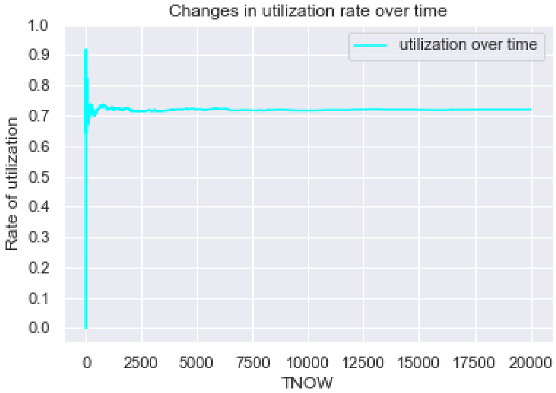


Figure 29: Changes in utilization rate over time

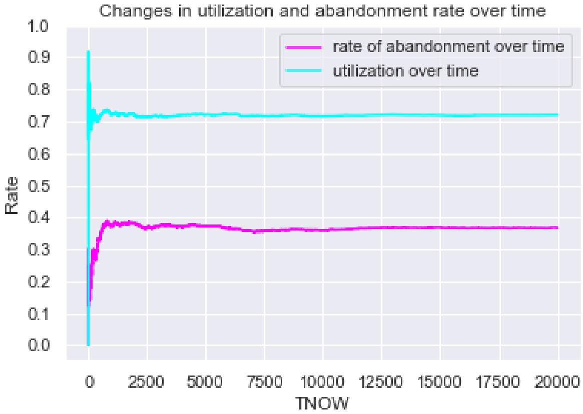


Figure 30: Abandonment and Utilization vs TNOW

As the audience might observe, (Figure 28) and (Figure 29) show that both the rate of abandonment and the rate of utilization fluctuate greatly during the warm up period (from `TNOW` $= 0$ to `TNOW` $= 2500$ due to the initialization bias. However, the oscillation cease after `TNOW` have passed the 7500-th step. This prove our original conjecture about the convergence to steady state. The result of this experiment also implies that we need not run our simulation up until infinity and the initialization bias is indeed negligible and hence can be ignored only if we perform our simulation up to at least 7500 steps. Since the initialization bias is negligible, we won't have to concern ourselves with determining the warm-up period. We have chosen the 7500-th steps because if the system run any longer, we would simply be wasting computational time and power and we expect that the true performance measure of our system (from the Ergodic theory) would not deviate too much from our current estimated performance measures. Additionally, we have chosen this specific value for our termination time because if we cease our termination at the time point 2500, there would still be some slight fluctuations (such as for the case of the abandonment rate in (Figure 28)) before reaching the steady state. Perhaps not for the case of the utilization rate but if there are other performance measures other than the utilization and the abandonment rate that we would like to estimate based on the CLT and the Ergodic theory, then we think it might be more "safe" to choose the 7500 time step. From now on, we will run and perform output analysis for all of our simulation with the termination time `T` chosen to be 7500.

With these resolutions, we can finally perform experiment (Figure 15) for the last time and we expect that the yield performance measure estimator will be closely match with the true performance measure (according to the Ergodic theory). Base on the CLT, we are going to perform roughly 50 replications to estimate the true performance measure of the Euclidean metric system with the configuration `sim_taxi(lambda1 = 1, lambda2 = 1, aban1 = 1, aban2 = 1,time_coef = 1, T = 7500`. With 95% confidence, the confidence interval (6.14) will contain our desired performance measure estimator:

$$\left(\bar{X} - t_{1-\alpha/2,n-1}\frac{\mathtt{S}}{\sqrt{n}}, \bar{X} + t_{1-\alpha/2,n-1}\frac{\mathtt{S}}{\sqrt{n}}\right) = \left(0.7231 - t_{1-\alpha/2,50-1}\frac{0.00644}{\sqrt{50}}, 0.7231 + t_{1-\alpha/2,50-1}\frac{0.00644}{\sqrt{50}}\right) \tag{6.14}$$

$$= (0.72129, 0.72495) \tag{6.15}$$

As the audience may observe, the variance of the estimator in equation 6.14 is more refined compare to the estimator in 6.12. The lesser the variance or tighter the confidence interval bound, the more accurate our result. With this calculations we conclude the current section on output analysis.

7 Comparing alternative system

There are many ways in which we can modified, test and compare the performances of our original algorithm (ED). Due to sheer varieties and complexities, we believe that it is not necessary to adjust and compare every slight modification of our systems. Hence we are going to be more selective in presenting our modified our system and experimenting them. Undoubtedly, there are others interesting adjustments and improvements which can be done, however; we intend to leave that for future research and we are going to present some of the modification that we have performed down below.

Before making our comparison, to ensure their independent, we are going to to use a different stream of random number for every replications produced. This can be done by assigning a specific `seed` value for each replications. For example, we will set `seed` $= 0, 1, 2, ...50$ to produce 50 replications for the two sytem with configuration: `sim_taxi(lambda1 = 1, lambda2 = 1, aban1 = 1, aban2 = 1,time_coef = 1, T = 7500` without delay and then we will set seed value to be equal to `seed` $= 50, 51, ..., 100$ for the system configuration `sim_taxi(lambda1 = 1, lambda2 = 1, aban1 = 1, aban2 = 1,time_coef = 1, T = 7500` with delay in matching.

7.1 Comparing the mean difference in utilization between ED without delay and ED with delay

For the two systems, we are going to perform hypothesis testing on whether there is a difference in the mean utilization between them. The desired output is the confidence interval (CI). If our CI result include the value of zero, then we conclude that we can not reject the null hypothesis and there is not enough evidence to say that these two system on average perform differently when it comes to the utilization level of the taxi. In contrast, if our CI does not include the value of 0,then we can conclude that we reject the null hypothesis with $(1 - \alpha) \times 100$ percent confident. In other words, we say that we reject the fact that on average these two systems perform similarly when it comes to the utilization level. We will assign the numeric value of **1** to denote the non delay system and the numeric value of **2** to denote the delay system.

- Null hypothesis H_0: $\mu_{U_1} = \mu_{U_2}$
- Alternative hypothesis H_1: $\mu_{U_1} \neq \mu_{U_2}$

We choose utilization rate as a point of comparison because we believe it is best at showcasing and representing the ways in which we have designed the algorithm.

We are going to employ the Welsh's two sample test (calculated as followed (7.2)) as our test statistic because it performs much better than the Pair t-test. Welsh's test provide much credibility to our conclusion because it requires not just pairwise independent between replications of a configuration but it also requires pairwise independent between replications from different configuration. The seed will be set as outlined in (Section 7) above to ensure independent. For the moment, we will not concern ourselves too much with the reason for choosing 50 replications, obviously the number of replications perform will have an influence on the variance of our confidence interval. Our immediate goal is to observe and see if there is any differences in the performance measures between our two systems. Hence any arbitrary number of replications will be suffice and the number of replications will be enough to draw some meaningful conclusions to our findings.

To calculate the degree of freedom:

$$\hat{f} = \frac{(S_1^2/n_1 + S_2^2/n_2)^2}{\frac{(S_1^2/n_1)^2}{n_1-1} + \frac{(S_2^2/n_1)^2}{n_2-1}} \tag{7.1}$$

Where:

- where S_1 and S_2 are the sample standard deviation of the system 1 and 2 respectively.

- n_1 and n_2 are the number of replications of system 1 and 2 respectively. Note that for Welch's test, the number of replications don't have to be identical between the two system.

To form the confidence interval (CI):

$$\left(\bar{X}^1 - \bar{X}^2 - t_{\hat{f},1-\alpha/2}\sqrt{S_1^2/n_1 + S_2^2/n_2}, \bar{X}^1 - \bar{X}^2 + t_{\hat{f},1-\alpha/2}\sqrt{S_1^2/n_1 + S_2^2/n_2}\right) \tag{7.2}$$

Where:

- t is the student-t distribution with $\hat{f}$ degree of freedom. The t distribution has $\hat{f}$ degree of freedom due to the fact that the number of replications performed on the two system are not always the same.

- $\bar{X}^1$ and $\bar{X}^2$ are the sample mean of system 1 and 2, respectively.

As analyzed earlier in (Section 6.2) and (Section 6.3), for small value of `lambda1` and `lambda2` (i.e. less than or equal to 5), we don't require hypothesis testing to know that there is indeed a massive different in the mean utilization of the delay system and the non delay. What we might be interested in however, is to see if these differences still persist after the waiting (or "hold out") conditions have been exceed by setting the input parameter `lambda1` and `lambda2` to be large enough (i.e. larger than 5). In other words, we are going to verify our conjecture regarding how the delay system will revert back to the original (ED) system provided that the input parameters can "exceed" the "hold out" condition. For these upcoming experiment, we want to compare the two system configurations: sim_taxi_with_delay(lambda1 = 10, lambda2 =5, aban1 = 1, aban2 = 1,time_coef = 1, T = 7500) and sim_taxi_without_delay(lambda1 = 10, lambda2 =5, aban1 = 1, aban2 = 1,time_coef = 1, T = 7500). We will perform 20 replications each with seed for the non delay system starting from (0,20) and the delay system with seed starting from (20,40) both not including the endpoint. The module `statmodels` in Python will assist in producing the CI. Hence our CI will be: (-0.001792, 0.001903).

Observe that our interval does indeed contain the value of 0. Hence with 95% confident, we can not reject the null hypothesis and that there is not enough evidence to suggest that the mean utilization is different between the non-delay (system 1) and the delay system (system 2). In other words, on average the utilization rate of the delay matching system is similar to the non-delay system as long as the input parameters `lambda1` and `lambda2` are both larger than the "hold out" conditions. Additionally, our conclusion is also being supported by the p-value = 0.951579 (calculated with the same `statmodels` module found in (Appendix B)). In statistic, p-value is defined to be the probability of obtaining the test results that is at least as extreme as the results we are actually observed. Since this probability is quite high (0.95) for our case, it means we are very much likely to arrive to the same conclusion if we were to perform the this particular experiment multiple times. Since we have set our confidence level (CL) α to be equal to 0.05, a p-value that is larger than the CL is statistically insignificant and it implies that we can not reject the null hypothesis H_0. Thus it confirms the conjecture that the delay system does indeed revert back to the original non-delay system.

7.2 Comparing the differences in performance measures between Euclidean (ED) and Manhattan distance (MD)

As introduced in section, Manhattan metric appear to be a reasonable implementation that could make our algorithm more applicable. However a reasonable implementation is not a "good enough" measure. When something is good, the specificity is important because it has to be "good" in comparison to something that is of similar nature. In other words, the system performance of the Manhattan metric need to be tested against some of the already existed configurations. More specifically, we would like to observe how the new Manhattan metric implementation might perform against the original Euclidean distance (ED) measure.

For the demonstrative purposes of the upcoming experiment, it is not necessary to perform the experiment for 7500 simulation length. We simply want to showcase the difference in how changes in the rate of utilization is affected by changes in `lambda2` for the: MD system and the ED system. We expect that there won't be any substantial fluctuations and that we should expect to see two concave functions which will overlap one another. Below, we present a plot for the MD system with the input parameters: sim_taxi_Manhattan(lambda1 = 1, lambda2 = index, aban1= 1, aban2=1, time_coef=1, T=1000) and the ED system with the input parameters: sim_taxi_Euclidean(lambda1 = 1, lambda2 = j, aban1=1, aban2=1, time_coef=1, T=1000). Where both `index` and `j` will loop through the range (0.01,11,0.2).

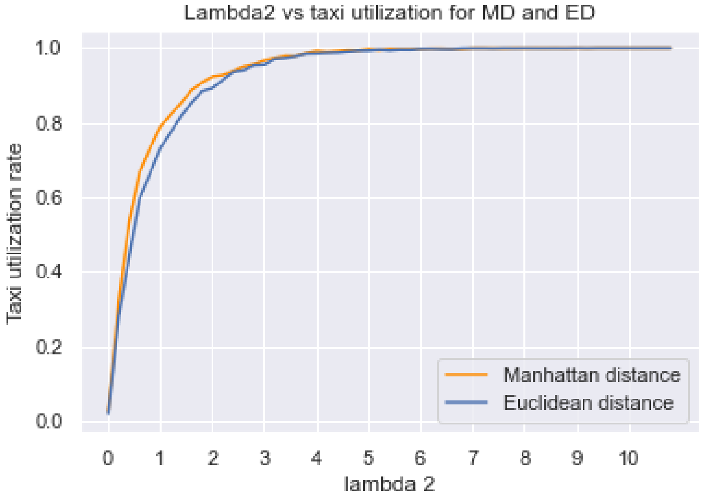


Figure 31: Taxi utilization vs lambda2 for the Manhattan distance and the Euclidean distance

Evidently, we note no substantial differences in the utilization rate of the MD and the ED systems. Perhaps utilization is not an adequate measure for comparing these two systems. After all the utilization rate can only describe "how well used are our taxis". Changing the environment/landscape in which the taxis operate should not affect how often they are busy. A better performance measure which are better suited for the task is the number of trip completions. The total number of trip completed depend on how "fast" our taxis can traverse in the environment. Hypothetically, if all obstacles in its way are removed which is the case for the ED system, then the total amount of time taken for that taxi to complete a trip on average is going to be shorter compare to the total amount of time taken for the taxi that has to operate in the MD environment. Of course, this assume the fact that the moment one of the taxi becomes available, it immediately get reassigned by the algorithm.

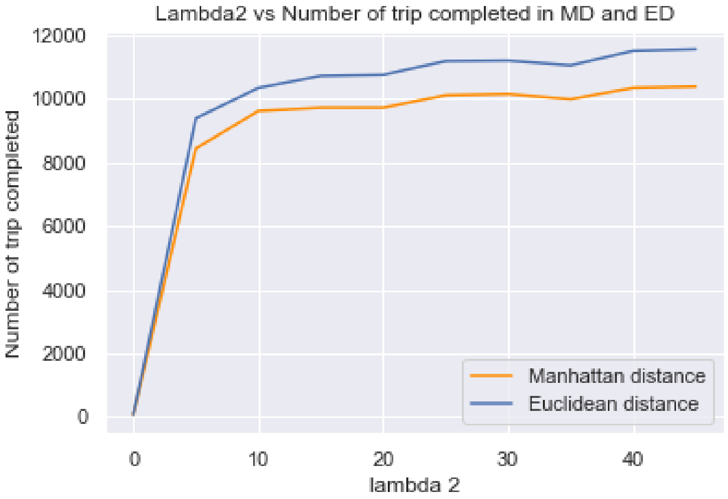


Figure 32: Lambda2 versus Number of trip completed in the Manhattan distance and the Euclidean distance system

As `lambda2` increase, the number of trip completed should also increase. Hence that explain the ever increasing trend. As anticipated, ED taxis on average take shorter time to complete one trip compare to MD taxis hence on average the number of trip completed for the the ED taxis system are higher compare to the trips completed by the taxis in the MD system. Notably, the difference is estimated to be roughly equal to 1000 trips.

Coming up, let's explore how the MD implementation is going to affect one the performance measures of the system. As explained earlier, the rate of utilization is not a reliable metric when comparing the MD implementation against the ED system. However, we expect that the new system might have a significant effect on the rate of abandonment of our customers. It is logical because taxis have to traverse a more complex terrain in the MD system, hence they complete less trips and if the number of customers stay constant for both the MD and ED systems, then there should be a significant difference in the rate of abandonment in the MD system compare to ED. Namely, we predict that on average, the abandonment of the MD system is higher than the abandonment of the ED system. We also believe that some of our predictions will gather more credibility if we can confirm it with hypothesis testing than plotting graphs and compare the differences. Hence we are going to perform another hypothesis testing which is similar to the one above in section.

- Null hypothesis H_0: $\mu_{A_1} = \mu_{A_2}$
- Alternative hypothesis H_1: $\mu_{A_1} \neq \mu_{A_2}$

Our objective is to see if there is a significant different in the mean of the abandonment rate for both MD (system 1) and ED (system 2). To ensure the stability of the experiment, all replications will have the simulation length of 7500. As a reminder, if the duration of the simulation length is not at least of this value, then for all of our replications, there will be initialization bias and that will affect the result of the hypothesis testing. The input parameters for the two system are as followed,

- The MD system: sim_taxi_Manhattan(lambda1 = 1, lambda2 = 1, aban1= 1, aban2=1, time_coef=1, T=7500)
- The ED system: sim_taxi_Euclidean(lambda1 = 1, lambda2 = 1, aban1=1, aban2=1, time_coef=1, T=7500)

In order to generate independent sample that will obey the CLT, we set the seed for system 1 to run through the range (0,20) and producing 20 replications and the seed for system 2 to run through the range (20,40) and producing another independent set of 20 replications. Thus with 95% confidence, our CI will contain (0.0099494, 0.0226372) and the p-value = $6.438458e^{-6}$. Observe that our interval does not contain 0 and with the p-value less than the CL: $\alpha = 0.05$. We reject the null hypothesis H_0 with 95% confident and as predicted the mean rate of abandonment for the two system is indeed different.

Like all statistical tools, hypothesis test are bound to have some errors and one may criticize how much error are we incurring and how precise is the result of our conclusion. Additionally, the choice of the number of replications is also questionable and that will inherently have an effect on some the errors. In hypothesis testing, there are two types of errors:

- Type I error: is rejecting the null hypothesis H_0 when it is in fact *true* "false positive"
- Type II error: is accepting the null hypothesis H_0 when it is in fact *false* "false negative"

The probability that one is making a Type I error is the confident level (CL): α and the probability of a Type II error denoted: P(Type II error) = 1 - (Statistical Power of a test). Usually, we set the confident level before performing hypothesis testing and we most often choose it to be $\alpha = 0.05$, this choice of alpha have some effect because the lower the CL the narrower our CI and vice versa. So in our recent experiment, there is only a 5% chance that we rejected the null hypothesis when it is in fact true. What about the type II error ? For the Type II error, we need to be able to compute the value of the so called Statistical Power (SP). In statistic literature, SP can often be referred to as the true positive rate. SP indicate the probability of correcting rejecting the null hypothesis H_0 when the alternative hypothesis H_1 is true. Additionally, the Type II error is also affect by the sample size (i.e. the number of replications that we do). The smaller the sample size the larger the type II error and vice versa. In the recent experiment, if we are confident that we are rejecting the right H_0, then we can set SP value to be 0.8. In statistical literature, SP usually is chosen to be either high = 0.8, medium = 0.5 and low = 0.2.

We are going to use Kernel density estimation (KDE) to observe and give an informed estimation of the statistical power. To put it simply, for a data set which may contains multiple observations (i.e. abandonment rate result from each replications), KDE calculate the Gaussian kernel function and center it at around each data points. Hence each observations will now have a "bump" made by the red curve (Figure 33).The KDE curve (blue) is obtained by adding up all the y-value for each "bumps" along the x-axis. The y-axis is the density function created by the overlapping "bumps". Many overlapping bumps, when being added up, create a high peak which will result in higher density value. Additionally, the smoothness of the KDE curve is controlled by a bandwidth parameter, the large the bandwidth, the smoother the KDE curve. There are methods for choosing a bandwidth but we are not going to dive deep into this area because that is beyond the scope of this work. We want to use the result from the KDE (or the estimated distribution plot) so we can make a better comparison between the two data set. Nonetheless, the bandwidth chosen for most of our graphs will be set to the default bandwidth which is the `scott` = 1.059 [37].

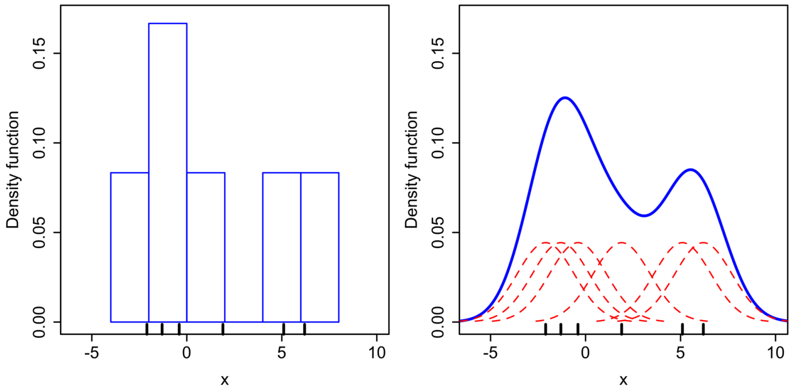


Figure 33: Kernel density estimation (right) [15]

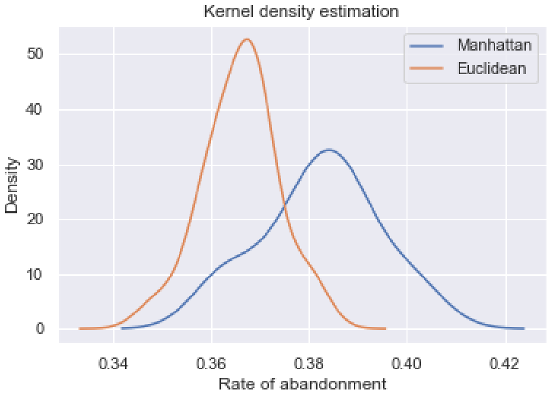


Figure 34: Kernel density estimation for the abandonment rate of MD and ED (bandwidth = Scott)

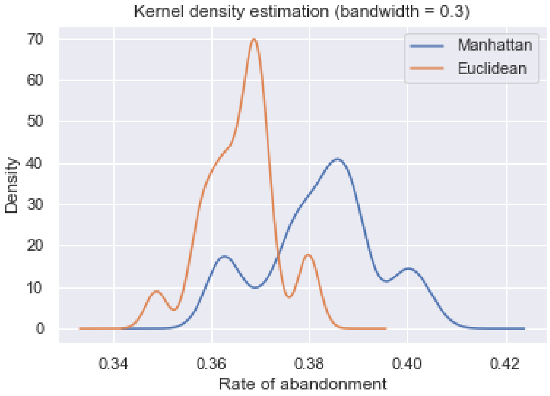


Figure 35: KDE for the abandonment rate of MD and ED with (bandwidth = 0.3)

The above Figure 34 is a Kernel density estimation of the probability density of the mean abandonment rate of the two system MD and ED. Due to the Ergodicity nature of our system and the simulation length that have been chosen to be 7500, the two distributions above is close if not almost identical to the true distribution. In other words, it the best approximation of the true distribution for the mean abandonment rate of both the MD and ED system with the configuration: sim_taxi_Manhattan(lambda1 = 1, lambda2 = 1, aban1=1, aban2=1, time_coef=1, T=7500) with seed range from (0,20) and sim_taxi_Euclidean(lambda1 = 1, lambda2 = 1, aban1=1, aban2=1, time_coef=1, T=7500) with seed range from (20,40).

First, we need to understand the concepts of how distribution overlapping affect the result of hypothesis testing. If we have two distributions that are overlapping substantially, then most of the time, our hypothesis test will lead us to the conclusion of not being able to reject the null hypothesis H_0. Even though the two distributions might be distinct in nature, large overlaps will cause the probability of making a Type II error to be large (i.e. accepting H_0 when it's false) and such effect cause the hypothesis testing to lead us to the conclusions of accepting the false H_0 repeatedly. Only in a few rare instances, the hypothesis test will yield a p-value or a confidence interval (CI) that could cause us to reject that false H_0 but because of such large overlaps, it's high unlike with the probability of 80% of the time we will keep accepting that false H_0. So in essence, if we could somehow decrease the magnitude of such overlaps then, the hypothesis test will lead us to a more precise conclusion which is rejecting H_0. Why sample plays a key role in affecting the probability of correctly rejecting the null hypothesis ? The more sample we can collect, the better we can estimated the true mean of the abandonment rate of the MD and the ED system. The closer our estimated abandonment rate to its true mean, the less likely the estimated mean from the two distributions of the MD and ED system will overlaps. When the estimated mean of the two systems overlap less or not overlapping at all, it increases the probability that the hypothesis test have led us to accurately rejected the null hypothesis (or in other words increase the true positive).

Despite such large overlaps in the two distributions, observe that they are still distinct. But because of such large overlaps and the small sample size (i.e. the number of replications = 20), for now we will have to accept the fact that our conclusions for the recent experiment is highly likely to have a low SP and that the probability of accurately rejected the null hypothesis is small. We are not concerning with determining how small this probability might be but rather we want to increase the SP of our hypothesis testing. In other words, we want to increase the probability that said we are making the right decision by rejecting the null hypothesis H_0. We can achieve this by increasing the number of replications for each system. But one may ask how many replications should we produce ? Power Analysis will assist in determining the least number of replications in order to obtain a reasonably good statistical power value. As a remark, if the distribution for the abandonment rate of MD were instead identical the the distribution of the abandonment rate of ED, then the null hypothesis would have been true and we would have accepted it. Then, the concept of “correcting reject the null hypothesis” would not have had much validity and the concept of SP wouldn’t apply here.

To sum up, Type I error can be dealt with by choosing a reasonable value for the CI. The issue with type II can be solve by performing many replications and Power Analysis will help us determine the number of replications to perform.

The Power Analysis require 4 components. Namely, a significant level α, Effect size, the Sample size and the Statistical power.Most of these value can be calculated or chosen provided that we have a certain knowledge about the behaviour of the distribution of the data set. First of all, the significant level α can be chosen and it usually take the value of 0.05. Secondly, the Effect size is an estimations of overlaps between the two distributions. For example, if hypothesis test lead us to conclude that there is indeed a significant difference between the mean rate of abandonment for MD and the mean rate of abandonment for ED, then the Effect size can inform us of the magnitude of this difference. Moreover, overlaps is affected by the distance between the sample mean and the sample standard deviation. By combining these two metric we can calculate the Effect Size of our experiment. The are multiple measures for Effect size such as the Pearson correlations coefficients which quantify the linear relationship between two variables, Cohen’s D which quantify the magnitude of the difference between the mean of two groups, and etc. For our upcoming experiment, Cohen’s D measure will be chosen to calculate the Effect size and finally, if we set the Sample size to be an unknown and choose SP to be large (i.e 0.8) then the result of the upcoming Power Analysis can tell us whether the number of replications in the previous experiment can guaranteed us a high probability of correctly rejecting the null hypothesis. Or at least it will provide an in formations on how many replications we should have performed in order to have the probability of 80% of rejecting the null correctly.

Cohen’d formula for our current experiment will be defined by:

$$\texttt{Cohen's D} = \frac{|\bar{X}_{A_1} - \bar{X}_{A_2}|}{\text{Pooled Standard deviation}} \tag{7.3}$$

$$\text{Pooled Standard deviation} = \sqrt{\frac{(n_1 - 1)Sd_{A_1}^2 + (n_2 - 1)Sd_{A_2}^2}{(n_1 + n_2 - 2)}} \tag{7.4}$$

where:

- $\bar{X_A}$ is the sample mean of the collected abandonment rate.
- Sd_A is the standard deviation of the collected abandonment rate.

- n_1 and n_2 is the number of replications from system 1 and 2.

If we want to know the Power or in another words the probability that we have accurately rejected the null hypothesis in our recent experiment. With Python `stat` module, the function `statsmodelsstats.power.TTestIndPower.solve_power`, $n_1 = 20$ and $n_2 = 20$, Effect size = 1.69716 and $\alpha = 0.05$. The Power Analysis confirm that with the probability of 1, we have correctly rejected the null hypothesis and indeed there was a different in the mean of abandonment rate between MD and ED. If we perform another experiment to determine the the least number of replications that was needed for each system. Then the Power Analysis give the value of 6.565. Rounding this value up to 7, this mean that we would need to perform at least 7 replications per systems to guarantee a 0.8 probability of correctly rejecting the null hypothesis. Although there are still 20% chance of making the type II errors, the more replications the simulations can produce the less likely this error will pose a problem. With this we conclude our analyses for the MD and ED system.

7.3 Comparing the differences in performance measures between hot spot distribution of locations and uniform(0,2) distribution of locations (original ED system)

Similar to the some of previous experiments, we want to explore how the performance measures of the hot-spot system is different from the performance measures of the original ED system. Note that whether, we compare between the hotspot vs ED or the hotspot vs MD does not make a difference because if there is indeed a difference between hotspot vs ED, then that difference is expected to be somewhat similar to the comparison between hotspot vs MD.

Before performing hypothesis testing, we would like to present the KDE of the mean of the utilization and the mean abandonment rate distribution of the hot-spot and the ED system. We obtained this by performing 20 replications for the two configurations sim_taxi_Triangular(lambda1 = 1, lambda2 = 1, aban1= 1, aban2=1, time_coef=1, T=7500) with seed value ranging from (100,120) and sim_taxi_Euclidean(lambda1 = 1, lambda2 = 1, aban1=1, aban2=1, time_coef=1, T=7500) with seed value ranging from (120,140). Additionally, a new seed range have been chosen to ensure independent observations between experiments.

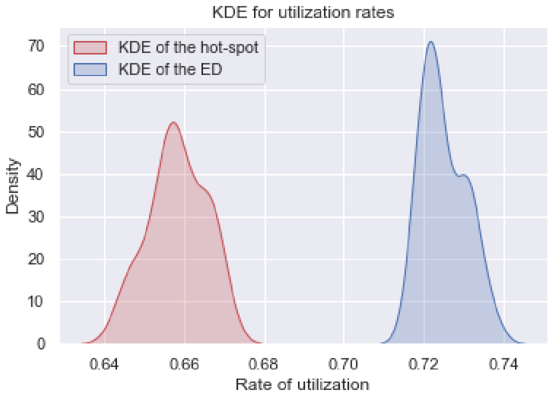


Figure 36: KDE for the utilization rate of hot-spot and ED (bandwidth = scott)

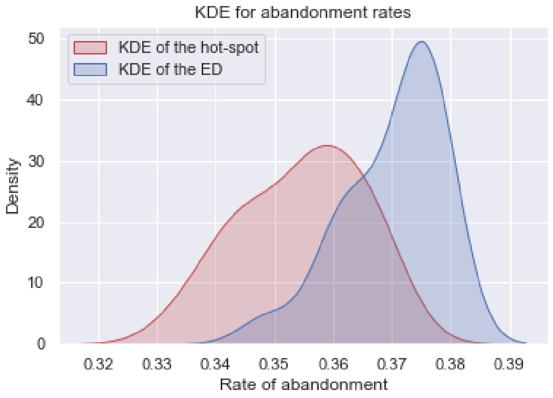


Figure 37: KDE for the abandonment rate of hot-spot and ED with (bandwidth = scott)

Even though the variations between the hot-spot and the ED system is insubstantial. There is a massive different between the utilization rate of the two. As mentioned before, the KDE of the distribution of the utilization rate between the hot-spot and the ED is a close approximation if not almost identical to the true distribution of the utilization rate attribute to the Ergodicity nature of the system. Hence, graphically we can detect such differences and hypothesis testing is not required to verify this fact. If Power Analysis were to be performed for the data in (Figure 36), the effect would be extreme large which indicate a favourable conclusion towards the alternative hypothesis H_1. On the other hand, when comparing the abandonment rate, this differences is less pronounced and to arrive at a more precise conclusions for the case with the abandonment rate. Once again, hypothesis testing is required. To avoid wasting computational power, Power Analysis therefore performed before hypothesis testing. Power Analysis provide an information on the number of replications that is required so that the probability of making the type II errors is minimized. With SP = 0.8, $\alpha = 0.05$ and Cohen's D Effect size = 1.717882. The number of replications that is needed to ensure the 0.8 probability of correctly rejecting the null hypothesis H_0 is 6.437 replications for each configurations. This mean that if the hypothesis testing is performed with at least 7 replications, then the probability of making type II error is going to be at most 20%.

- Null hypothesis H_0: $\mu_{A_{\text{hot-spot}}} = \mu_{A_{ED}}$
- Alternative hypothesis H_1: $\mu_{A_{\text{hot-spot}}} \neq \mu_{A_{ED}}$

Use a different seed here because 100-110 is already used for plotting distribution and the power analysis, use seed (210,220), (220,240) *redo the experiment maybe.*

For the hypothesis testing, 10 replications will be performed for each configurations to further minimize any potentials impact of the type II error. To ensure that the seed value used in the Power Analysis is independent from the seed use for hypothesis testing, the replications produce for the hot-spot system will have the seed value in the range (200,210) and the replications produce for the ED system will have the seed value in the range (210,220). The significant level $\alpha = 0.05$ and with 95%, the CI is: (-0.029301, -0.012751). Observe that this interval don't contain the value of 0 and from additional calculation, the p-value yield $4.3111e^{-5}$. Since the p-value is much less than 0.05, it implies statistical significant in favour of the alternative hypothesis H_1. Furthermore, with the type I error to be 5%, we can confidently reject the null hypothesis H_0 and conclude that the mean abandonment rate of the two systems is indeed different.

7.4 Comparing the differences in performance measures between Furthest dispatch (FD) and the closest dispatch (original ED system)

As the tittle suggested, we believe that there might be intrinsic different between FD and ED. Hence it might be interesting to compare the performance measures of these two system. As stated, FD operate in the opposite manner compare to the ED system. Due to the fact that the furthest taxi now have to travel to the furthest away customer, we anticipate that the utilization rate of FD is much higher compare to the utilization rate of ED. However, it is probable that this higher utilization level is not an indication of an improvement in overall system efficiency. As explained ,utilization a quantity that measure how often the taxis is occupied. In our system, the moment a taxi is assigned a customer that is when it is considered occupied. Because of that, it does not take into account factor such as the pick-up time. Pick-up time is important because it increase the workload of the system. Indeed, higher utilization imply better system efficiency but when considering the system under a metric such as the number of trip completions. Then utilization may not imply a higher number of trip completed. Hypothetically, this create an inverse effect meaning FD has higher utilization but the number of trip completed is not much significant or even less in some case compare to the ED system with lower utilization rate but can potentially has a higher number of trip completed.

Before verifying some of the our predictions with hypothesis testing, we will use KDE again to observe the differences between the mean utilization distribution of the FD and the ED system. The goal is to first observe the phenomenon graphically. If there there is indeed a difference in between the two distribution then we can perform Power Analysis and then from the result of the Power analysis, we can carry on to the hypothesis testing. Be ware that if the differences in the two distribution is too large, (i.e. the distribution don't overlap or very far apart) then the effect size will also be large. This mean that irrespective of the number of replications produced, the hypothesis test will always lead to the conclusion of rejecting the null hypothesis H_0. If the chance of rejecting H_0 is always close to 100% then it does not matter how confidence we are in correctly reject H_0 because it will always be rejected and so the process of Power Analysis is not required. As mentioned before, on the other hand if the two distribution are indeed overlaps to a great degree and the data collected from the two are actually came from the same distribution, then it is highly likely that the hypothesis test will lead to the conclusion of not being to reject H_0 and once again the Power Analysis process is not required because we will never reject H_0.

To ensure minimal bias in drawing our conclusion in a later hypothesis testing analysis, the data for mean utilization of the distribution is collected from producing 20 replications for both FD and ED system using 2 sets of seed: FD with seed range from (400,420) and ED with seed range from (420,440).

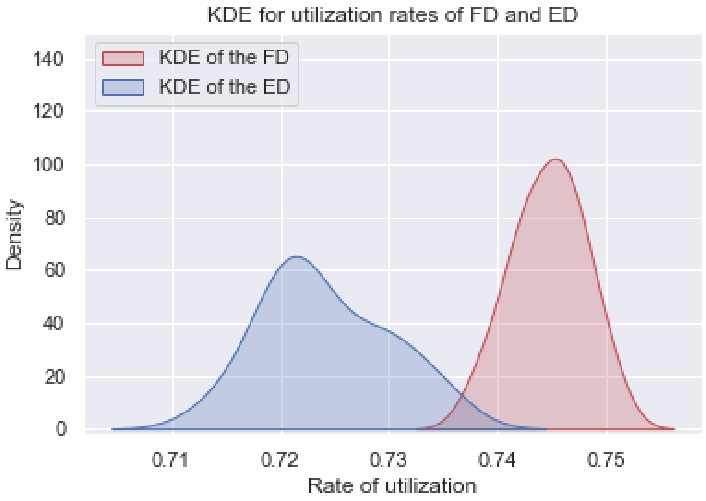


Figure 38: Kernel density estimation for FD and ED system

As the audience may observe, there are indeed some differences in the mean utilization rate of the two systems. However, the magnitude in such differences is not yet conclusive and this can be verified with the process of power analysis. Note also that Power Analysiscan only inform us of the magnitude of the difference but it cannot inform us on which distribution from which system is "better" than the other. Choosing the confidence level $\alpha = 0.05$, statistical power (SP) = 0.8 ,the number of sample to be 20 (for each distribution to produce a plot) and the effect size (d) = 4.4884. The number of replication (n) that we must at least produce to guarantee a 0.8 probability of confidence if we are going to reject the null hypothesis is n = 2.24456. Thus as long as we can produce more than 3 replications, then we can be fairly confidence in our later conclusion.

To verify some of the above hypothesis, once again we will perform hypothesis testing. 10, replications will be produce for both the FD and ED system. Since this experiment will reuse the ED system, to ensure independence across all tests, and independence from the the power anaylsis test, the seed value set for the FD system will take the range from (500,510) and the set set for the ED system will take the range from (510,520). The two system configurations will take the input parameters as followed: sim_taxi_Furthest(lambda1 = 1, lambda2 = 1, aban1= 1, aban2=1, time_coef=1, T=7500) and sim_taxi_Euclidean(lambda1 = 1, lambda2 = 1, aban1=1, aban2=1, time_coef=1, T=7500).

- Null hypothesis H_0: $\mu_{U_{\text{FD}}} = \mu_{U_{ED}}$
- Alternative hypothesis H_1: $\mu_{U_{\text{FD}}} \neq \mu_{U_{ED}}$

The hypothesis is testing for the difference in the utilization between the FD and the ED system. With 95% confidence, our confidence interval (CI) will be in the range: (0.011894, 0.022165) and the p-value $= 1.65524e^{-6}$. Since the CI does not contain zero and the p-value point favourably toward the alternative hypothesis H_1. We conclude that with 95% confidence, we reject the null hypothesis H_0 and that the mean utilization rate is difference between FD and ED.

To test the hypothesis regarding the number of trip completed, a new performance measure is required. Defining the rate of trip completed to be the ratio between the total number of trip completed per replications divided by the total run length of the simulation (Termination time). This new quantity measure the rate of trip completion per unit of simulation time. For instance, if the rate of completion is calculated for one run of the simulation to be 0.6, then on average a trip is completed roughly every two minutes of simulation time.

$$\texttt{rate\_complete} = \frac{\texttt{Num\_trip}}{\texttt{T}} \tag{7.5}$$

With the newly defined performance measure, KDE will be used to estimate the distribution the mean rate of trip completion. Note that the replications have already been produced when we estimated the mean utilization rate above. Therefore, we present the KDE for the mean rate of completion:

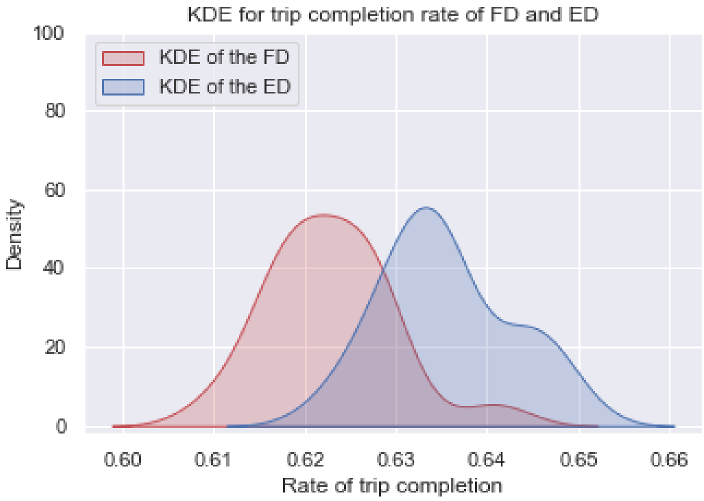


Figure 39: Kernel density estimation for the mean rate of trip completion for FD and ED system

From the KDE plot, we can perceive some slight differences between the rate of trip completion between FD and ED. To make this more conclusive, a Power Analysis and hypothesis testing is required. First for the Power Analysis to provide the number of replications, we choose the confidence level ($\alpha = 0.05$), SP = 0.8 and the effect size (d) = 1.89561 is calculated from the 20 replications with the seed: FD with seed range from (400,420) and ED with seed range from (420,440). With these components, the Power Analysis provide the number of replications that is require of each group, n = 5.51631.

With the above information, a new hypothesis testing will be performed base on 10 replications produce for both system configurations: FD sim_taxi_Furthest(lambda1 = 1, lambda2 = 1, aban1= 1, aban2=1, time_coef=1, T=7500) with seed ranging from (600,610) and ED sim_taxi_Euclidean(lambda1 = 1, lambda2 = 1, aban1=1, aban2=1, time_coef=1, T=7500) with seed ranging from (610,620).

- Null hypothesis H_0: $\mu_{Rate\_completion_{\mathrm{FD}}} = \mu_{Rate\_completion_{ED}}$
- Alternative hypothesis H_1: $\mu_{Rate\_completion_{\mathrm{FD}}} \neq \mu_{Rate\_completion_{ED}}$

As stated above, we are testing for the difference in the mean of the rate completion between FD and ED. Thus with 95% confidence, the critical region will be: (-0.011401, -0.000198) and the p-value = 0.040247. Notice that this interval does not contain the value of 0 and the p-value quite high but it is still less than the confidence level α. With 80% confidence from the Power Analysis test, we reject the null hypothesis H_0 and conclude that there is in fact a mean different in the rate of trip completion.

So far, in most of our experiment,we have managed to introduced a technique for data analysis. The first step require the plotting of the distribution of the data, the second step require observing the behaviour of the distribution, the third steps is making prediction/hypothesize on the performance of the system and then finally perform hypothesis testing to verify the predictions. With this we conclude this chapter.

8 Discussions and conclusions

As the RHB grow and compete among one another, the need to increase system efficiency is becoming more prevalence. The matching algorithm that have made firms such as Uber to become a phenomenon that it is today have become even more important than ever before. To understand its importance in system efficiencies as well as its challenges, we have set out to design our own matching algorithm and test its efficiency. From the design of our own algorithm, we have found that that incremental changes in the matching algorithm can have a huge impact on system performance.

The goal of this dissertation were two fold: firstly, we aimed to design a matching algorithm that can simulate the process of matching taxis and customers. Secondly, we aim to use the data generated from that simulation to apply statistical analysis technique to analyze a specific performance measure. Base on the graphical representation as well as the statistical analysis on our generated data, it can be concluded that minor changes in the input parameters such as taxi arrivals rate, abandonment rate can result in significant changes in the performance measures. More specifically, the higher the number of taxi arrivals the lower the utilization rate. In contrast, the higher the number of customer arrival rate the more rapid the system reach maximum utilization level. Additionally, changes in the dispatching protocol can also affect the utilization rate in an undesirable manner. It has also been found that the delay system will behave similarly to the original (ED) system for large values of the input parameters. Given these findings this information can be applied to live dispatching algorithms to provide valuable insights to operation efficiency. However, the application may be limited due to the ability to control factors we can not control in a real setting such as the input parameters (i.e. `lambda1`, `lambda2`, etc).

The first goal has been achieved the moment our simulation can run for any extended period of time and being able to produce the desired outputs without producing errors. With the output produced as intended, the performance measure derived from the outputs are in line with our expectations.

The second goal of performing analysis on the different configurations of our systems has also been achieved to a large extent. In total, five systems configurations were being used for comparison. Although better statistical methods may be applied such as the analysis of variance (ANOVA), we believe that the approach was a sensible one given the total number of configurations.

As a point of discussion: some may argue that the process of Power Analysis in some of the experiment is redundant and not necessary because the model is simple enough to produce multiple observations in reasonable time. However, we disagree with that statement. The design of the simulation although simple, it has tremendous opportunity to be expanded and developed for further research. In our current model, we have only considered how to match and assign taxi to customers efficiently. We did not take into account factors such as fuel cost, drivers income, minimal pick-up time, etc... Note that these factors can be incorporated into the current model under the form of optimization problems. However, if the system need to perform an optimization step before it can match taxi to customers every time, then the time the system require to produce one replication is going to take much longer compare to our ED system. Hence, to produce multiple replications and assess the performance of the new model, it would take longer than a “reasonable time”. As a result, Power Analysis needed to help save computational power.

As a suggestion, a different performance measure could have been chosen such as the customer wait time. As mentioned earlier, due to complexities of our model, we chose to measure the rate of abandonment as we found that keeping track of the customer wait time will add another level of unnecessary complexity. However, if we could manage to keep track of the customer’s wait time then this would translate well with the study of convergence to steady state of the GI/GI/n queue. Additionally this would also allow for tracking of the last idle time of either the driver or the customer that would enable an alternative dispatching algorithm. This may provide valuable information as it gives an alternative perspective. Therefore it may be worth pursuing.

A direction for future research is to employ machine learning techniques onto the data generated by our model to determinate the most optimal configurations.

As a philosophical discussion: In hindsight, the existence of RHB remove the need for a centralised service provider (at least for the transportation service). However, the online-based platform is still being run and maintained by the RHB. This begs the question: did RHB actually remove the need for a centralised structure by removing the barriers that prevented users and service providers from interacting or did they simply remove the responsibility from the centralised company and moved it towards their employee.[**?**]. With this, we leave this question for future research.

Even though our project is mainly about designing and testing the efficiency of the different types of configurations and matching protocol. We would like to bring the audience's attention to some of its shortfalls. In reality, the algorithm can be as efficient as we make it; however, taxi drivers could only be as efficient as they want to be. Therefore, they could not operate at the level of efficiency that is required by the algorithm. Potentials factors such as traffic congestion, safety and long term health problems are the reason that preventing them from serving trips-to-trips. Additionally, our current algorithm does not take into account factors such as when drivers can reject an assignment. This could explain the reason for the high rate of utilization in many of our experiments. Additionally, the fact that drivers can choose to accept a match can have profound impact on system utilization and more often, we believe it has a negative impact on the system utilization.

Furthermore, the result of this work might also apply well to the deployment of autonomous taxi. With autonomous vehicles, the "randomness" factors such as driver's behaviours are less prominent because such vehicles do not require rest nor do they care about their own safety and job security. Hence, we only need to factor in refuelling time or charging time for each vehicles but it cant be easily managed due to it being a matter of incorporating a linear optimization solver into the model.

The work on simulation design and statistical analysis on such system has resulted in a basic but already useful model. Ultimately, the goal of this work is to contribute to the process of building the most optimal matching algorithm that could adapt and switch between different matching protocols (i.e. the closest dispatch, the furthest dispatch, the stable marriage matching, etc.) depending on the state of the system. The exploration of the performance measures has provided a new perspective that will be a valuable contribution to the literature.

References

[1] https://www.statslab.cam.ac.uk/~james/Markov/s110.pdf, Oct 2001. [Online; accessed 16. Sep. 2021].

[2] Tnc's and congestion. https://www.sfcta.org/sites/default/files/2019-05/TNCs_Congestion_Report_181015_Finals.pdf, Dec 2019. [Online; accessed 15. Sep. 2021].

[3] Uber and the economic impact of sharing economy platforms | Bruegel. https://www.bruegel.org/2016/02/uber-and-the-economic-impact-of-sharing-economy-platforms/?utm_content=buffer256ea&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer+(bruegel)#_ftn3, Sep 2021. [Online; accessed 14. Sep. 2021].

[4] Uber's Economic Impacts Across the United States | EBP | US. https://www.ebp-us.com/en/projects/ubers-economic-impacts-across-united-states, Sep 2021. [Online; accessed 14. Sep. 2021].

[5] M. Akbarpour, S. Li, and S. O. Gharan. Thickness and information in dynamic matching markets. *Journal of Political Economy*, 128(3):783–815, 2020.

[6] J. D. Angrist, S. Caldwell, and J. V. Hall. Uber versus taxi: A driver's eye view. *American Economic Journal: Applied Economics*, 13(3):272–308, 2021.

[7] A. S. Asratian, T. M. J. Denley, and R. Häggkvist. *Bipartite Graphs and their Applications.* Cambridge University Press, Cambridge, England, UK, Jul 1998.

[8] A. Bekka, N. Louvet, and F. Adoue. Impact of a ridesourcing service on car ownership and resulting effects on vehicle kilometers travelled in the paris region. *Case Studies on Transport Policy*, 8(3):1010–1018, 2020.

[9] O. Besbes, F. Castro, and I. Lobel. Spatial capacity planning. *Operations Research*, 2021.

[10] N. Brazil and D. S. Kirk. Uber and metropolitan traffic fatalities in the united states. *American journal of epidemiology*, 184(3):192–198, 2016.

[11] A. E. Brown. Prevalence and mechanisms of discrimination: Evidence from the ride-hail and taxi industries. *Journal of Planning Education and Research*, page 0739456X19871687, 2019.

[12] C. W. Chan, G. Yom-Tov, and G. Escobar. When to use speedup: An examination of service systems with returns. *Operations Research*, 62(2):462–482, 2014.

[13] M. K. Chen and M. Sheldon. Dynamic pricing in a labor market: Surge pricing and flexible work on the uber platform. *Ec*, 16:455, 2016.

[14] X. Cheng, S. Fu, and G.-J. de Vreede. A mixed method investigation of sharing economy driven car-hailing services: Online and offline perspectives. *International Journal of Information Management*, 41:57–64, 2018.

[15] Contributors to Wikimedia projects. Kernel density estimation - Wikipedia, Jun 2021. [Online; accessed 11. Sep. 2021].

[16] Contributors to Wikimedia projects. Matching (graph theory) - Wikipedia. https://en.wikipedia.org/w/index.php?title=Matching_(graph_theory)&oldid=1032560546, Jul 2021. [Online; accessed 17. Aug. 2021].

[17] Contributors to Wikimedia projects. Taxicab geometry - Wikipedia, Jul 2021. [Online; accessed 27. Aug. 2021].

[18] J. Cramer and A. B. Krueger. Disruptive change in the taxi business: The case of uber. *American Economic Review*, 106(5):177–82, 2016.

[19] S. Das, J. P. Dickerson, Z. Li, and T. Sandholm. Competing dynamic matching markets. In *Proceedings of the Conference on Auctions, Market Mechanisms and Their Applications (AMMA)*, volume 112, page 245, 2015.

[20] J. Dong, P. Feldman, and G. B. Yom-Tov. Service systems with slowdowns: Potential failures and proposed solutions. *Operations Research*, 63(2):305–324, 2015.

[21] M. A. D. Driving and Uber. New Report from MADD, Uber Reveals Ridesharing Services Important Innovation to Reduce Drunk Driving. *PR Newswire*, Jun 2018.

[22] G. Feng, G. Kong, and Z. Wang. We are on the way: Analysis of on-demand ride-hailing systems. *Manufacturing & Service Operations Management*, 2020.

[23] S. Foss. The G/G/1 queue. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[24] S. Foss. The G/G/1 queue. http://www.math.nsc.ru/LBRT/v1/foss/gg1_2803.pdf, May 2010. [Online; accessed 8. Aug. 2021].

[25] D. Gamarnik and D. A. Goldberg. Steady-state GI/G/n queue in the halfin–whitt regime. *The Annals of Applied Probability*, 23(6):2382–2419, 2013.

[26] O. Garnett, A. Mandelbaum, and M. Reiman. Designing a call center with impatient customers. *Manufacturing & Service Operations Management*, 4(3):208–227, 2002.

[27] Y. Ge, C. R. Knittel, D. MacKenzie, and S. Zoepf. Racial and gender discrimination in transportation network companies. Technical report, National Bureau of Economic Research, 2016.

[28] G. Grimmett and D. Stirzaker. *Probability and random processes.* Oxford university press, 2020.

[29] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris. *Fundamentals of Queueing Theory.* John Wiley & Sons, Ltd., Chichester, England, UK, Jul 2008.

[30] S. Halfin and W. Whitt. Heavy-traffic limits for queues with many exponential servers. *Operations research*, 29(3):567–588, 1981.

[31] J. V. Hall and A. B. Krueger. An analysis of the labor market for uber's driver-partners in the united states. *Ilr Review*, 71(3):705–732, 2018.

[32] A. Henao and W. E. Marshall. The impact of ride hailing on parking (and vice versa). *Journal of Transport and Land Use*, 12(1):127–147, 2019.

[33] C. B. Insights. How Uber Makes Money Now. *CB Insights Research*, Apr 2021.

[34] S. T. Jin, H. Kong, and D. Z. Sui. Uber, public transit, and urban transportation equity: a case study in new york city. *The Professional Geographer*, 71(2):315–330, 2019.

[35] D. G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

[36] D. G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

[37] J. Kim and C. D. Scott. Robust kernel density estimation. *The Journal of Machine Learning Research*, 13(1):2529–2565, 2012.

[38] E. Kontou and N. McDonald. Associating ridesourcing with road safety outcomes: Insights from austin, texas. *PloS one*, 16(3):e0248311, 2021.

[39] D.-H. Lee, H. Wang, R. L. Cheu, and S. H. Teo. Taxi dispatch system based on current demands and real-time traffic conditions. *Transportation Research Record*, 1882(1):193–200, 2004.

[40] T. Liu, Z. Wan, and C. Yang. The efficiency of a dynamic decentralized two-sided matching market. *Available at SSRN 3339394*, 2019.

[41] J. Mezuláník, L. Durda, M. Civelek, and L. Malec. Ride-hailing vs. taxi services: a survey-based comparison. *Journal of Tourism and Services*, 20:170–186, 06 2020.

[42] F. Miao, S. Han, S. Lin, J. A. Stankovic, D. Zhang, S. Munir, H. Huang, T. He, and G. J. Pappas. Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach. *IEEE Transactions on Automation Science and Engineering*, 13(2):463–478, 2016.

[43] U. of Kent. Introduction to stochastic processes. https://www.kent.ac.uk/smsas/personal/lb209/files/notes1.pdf. Accessed 09/Jul/21.

[44] J. Reed. The G/GI/N queue in the halfin–whitt regime. *The Annals of Applied Probability*, 19(6):2211–2269, 2009.

[45] K. Roose. Uber Might Be More Valuable Than Facebook Someday. Here's Why. *Intelligencer*, Dec 2013.

[46] D. Sperling, S. Pike, and R. Chase. Will the transportation revolutions improve our lives—or make them worse? In *Three Revolutions*, pages 1–20. Springer, 2018.

[47] H. Wang and H. Yang. Ridesourcing systems: A framework and review. *Transportation Research Part B: Methodological*, 129:122–155, 2019.

[48] X. Wang, H. M. Ardakani, and H. Schneider. Does ride sharing have social benefits? 2017.

[49] A. R. Ward and P. W. Glynn. A diffusion approximation for a GI/GI/1 queue with balking or reneging. *Queueing Systems*, 50(4):371–400, 2005.

[50] E. W. Weisstein. Bipartite Graph. *Wolfram Research, Inc.*, Oct 2002. [Online; accessed 8. Aug. 2021].

[51] W. Whitt. Efficiency-driven heavy-traffic approximations for many-server queues with abandonments. *Management Science*, 50(10):1449–1461, 2004.

[52] K.-I. Wong and M. G. Bell. The optimal dispatching of taxis under congestion: A rolling horizon approach. *Journal of advanced transportation*, 40(2):203–220, 2006.

[53] K. Xu, L. Sun, J. Liu, and H. Wang. An empirical investigation of taxi driver response behavior to ride-hailing requests: A spatio-temporal perspective. *PloS one*, 13(6):e0198605, 2018.

[54] M. Xue, B. Yu, Y. Du, B. Wang, B. Tang, and Y.-M. Wei. Possible emission reductions from ride-sourcing travel in a global megacity: the case of beijing. *The Journal of Environment & Development*, 27(2):156–185, 2018.

[55] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye. A taxi order dispatch model based on combinatorial optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 2151–2159, New York, NY, USA, 2017. Association for Computing Machinery.

Appendices

A Terminologies, Definitions and Theorem:

A.1 Picard-Lindelöf theorem:

Consider the initial value problem:

$$\frac{dy}{dt} = f(t, y(t)) \; and \; y(t_0) = y_0 \tag{A.1}$$

Suppose also that f is continuous in t and uniformly Lipschitz continuous in y $\Rightarrow$ for some value $\epsilon > 0$, $\exists$ a unique solution y(t) to the initial value problem on the interval $[t_0 - \epsilon, t_0 + \epsilon]$.

Definition A.1. *Poisson process:*
Let $X_1, X_2, \ldots, X_n$ be independent exponential RV's with rate λ. Let $S_0 = 0, S_n = X_1 + X_2 + +X_n$ and $N_t = max\{n \geq 0 : S_n \leq \; n\}$.Then $N_1, N_2, \ldots$ is a Poisson Process with rate λ. Here N_t is the number of arrivals by time t. Additionally, for any t, if the process $(N_t)_{t\geq 0}$ is a Poisson process, then it follows a Poisson distribution with rate λt. $\left(i.e.P(N_t = k) = \frac{(e^{-}\lambda t(\lambda t)^k}{k!}\right)$.

Definition A.2. *Spatial economies of scale:*
As the number of customer increase steadily without exceeding the number of servers (n), the number of idle cars decrease and thus lead to higher pick up time for newly arriving customers. Similarly, as the number of customers increase beyond the number of cars (n), the number of customers waiting in the system will increase. This lead to the next idle server will have lesser time being match with new arriving customer. Thus, the service rate increase.

Definition A.3. *Metric space:*
A metric space is a set with a metric on that set. A metric is a function that defines a distance between any 2 members within that set. Definition: a metric space M (a set) with the metric d is an ordered pair (M, d) (i.e. $d : M \times M \to \mathbb{R}$) such that for any $x, y, z \in M$, the following hold:

1. $d(x, y) = 0 \Leftrightarrow x = y$.
2. $d(x, y) = d(y, x)$ *symmetry.*
3. $d(x, y) + d(y, z) \leq d(x, z)$ *triangle inequality*

Definition A.4. *Lipschitz continuity:*
Given 2 metric spaces (X, d_X) and (Y, d_Y), a function $f : X \to Y$ is Lipschitz continuous if $\exists$ a real constant $K \geq 0$ such that $\forall x_1, x_2 \in X$,

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2)$$

Any K that satisfied the above relations is called the Lipschitz constant of the function f.

B Appendix: Code listings

```
import time
import numpy as np
import pandas as pd
import random
import math

def sim_taxi_Euclidean(lambda1, lambda2, aban1, aban2, time_coef, T):

#     lambda1 = 1
#     lambda2 = 1.2
#     aban1 = 1
#     aban2 = 1
#     time_coef = 1
#     T = 100 #termination

    NumC_aban = 0 # keep track of the total number of customer leave without being served.

    total_utilization = 0 # this varialbe calculate the total utilation of all taxi.
    total_taxi = 0 # the area under the taxis arrival graph (red graph).
    total_customer = 0



    total_throughput = 0


    #event calendar vector of 3
    EC = [0,0,0] # explain each entry: 1st entry is taxi arrival, 2nd customer arrival and 3rd entry is termination time.


    #event types
    # 1st entry: is the event type, 2nd entry: is the ID of a taxi, 3rd entry: is the ID of customer
    # list grow as simulation changes.
    ET = [[0,0,0],
         [0,0,0],
         [0,0,0]]

    #ET[0][0]

    LT = [] #list of taxi
    LC = []  #list of customers

                    #check len(LC) = NumC at arbitrary point

    NumT = 0 #hold: the total number of taxi in the system at any particular state.
    total_T = 0 #hold: the total number of taxi in the system through out the simulation + the ID of the current taxi

    Num_trip = 0 # RECORD THE TOTAL NUMBER OF TRIP COMPLETED

    NumC = 0 #number of customers
    total_C = 0 #Customer ID



    EC[0] = np.random.exponential(1/lambda1) #arrival rate of taxi
    EC[1] = np.random.exponential(1/lambda2) #arrival rate of customers
    EC[2] = T #termination

    ET[0][0] = 1
    ET[1][0] = 2
    ET[2][0] = 3

    TNOW = 0

    list_utili = []


    while TNOW < T:
        Tnext = min(EC)
        min_index = EC.index(Tnext)

        total_taxi += NumT*(Tnext - TNOW) # if you plot this against TNEXT then you should expect to see a step funcion that show how each taxi is being utilized.

        total_customer += NumC*(Tnext-TNOW) #area under the customer arrival graph


        num_of_busy_taxi = 0
        for i in range(len(LT)):
            num_of_busy_taxi += LT[i][0] #this variable keep tracks of the number of time the taxis become busy.

        num_of_busy_customer = 0
        for j in range(len(LC)):
#              print("here is numbusy = ",LC)
            num_of_busy_customer += LC[j][0] #this variable keep tracks of the number of time the customers are busy




        total_utilization += num_of_busy_taxi*(Tnext-TNOW)


        total_throughput += num_of_busy_customer*(Tnext-TNOW)


        list_utili.append(total_utilization)
        # this varialbe calculate the total utilation of all taxi (area under the green curve).






#         print("Event Calendar = ",EC)
#         print("Event type = ",ET)
#         print("Min_index = ", min_index)

#         print("type=", ET[min_index][0])
#         print("TNOW = ",TNOW)
#         print("Tnext = ",Tnext)
#         print("List taxi = ",LT)
#         print("NumT = ", NumT)
#         print("List customer = ",LC)
#         print("NumC = ", NumC)

#         print("total_T = ", total_T)
#         print("total_C = ", total_C)

#         print("-----------------------------------------------------")
#         print("-----------------------------------------------------")


        if ET[min_index][0] == 1: #event type: Taxi arrival
            TNOW = Tnext
```

```python
        X = np.random.uniform(0,2) #add arrival locations
        Y = np.random.uniform(0,2)
        NumT += 1 #update number of taxi
        total_T += 1
        LT.append([0, X, Y, total_T]) #update list taxi

        EC[0] = TNOW + np.random.exponential(1/lambda1) #update EC
        ET.append([4, total_T, 0]) #taxi finishing shift/abandonment of taxi
        EC.append(TNOW + np.random.exponential(1/aban1))
        Assigned_1 = 0
        Min_Distance = 10000
        Customer_assigned_1 = 0

        #change all customer_assigned to the ID of the customer instead total_C

        #loop to find which taxi is the closest when NumC is not zero.
        if NumC > 0:
            for i in range(NumC): #range take value from 0 -> NumC-1
                if LC[i][0] == 0:
                    Assigned_1 = 1
                    Distance = math.sqrt((LC[i][1]-X)**2 + (LC[i][2]-Y)**2)
                    if Distance < Min_Distance:
                        Min_Distance = Distance
                        Customer_assigned_1 = i

        # assignd that closest taxi to a customer and update LT(change from iddle = 0 to busy =1), EC, LC and ET.
        if Assigned_1 == 1: # this mean that if Assigned == True
            #LT[Customer_assigned][0] = 1 #make ID which is total_T become the index
            num_of_busy_taxi += 1
            num_of_busy_customer += 1
            LT[NumT-1][0] = 1 # reassigning the 0-th index of this matrix to the value of NumT line 52.
            EC.append(TNOW + time_coef*(Min_Distance + math.sqrt((LC[Customer_assigned_1][1]-LC[Customer_assigned_1][3])**2 +
                                                                  (LC[Customer_assigned_1][2]-LC[Customer_assigned_1][4])**2)))

            LC[Customer_assigned_1][0] = 1
            ET.append([7, total_T, LC[Customer_assigned_1][5]])

            FR1 = -1 #found row
            for i in range(len(EC)):
                if ET[i][2] == LC[Customer_assigned_1][5]: #this condition check if this event is an abandonment and it also check if the event is correspond to the customer we are deal with.
                    # Customer assigned + 1 is because if Customer assigned = 0 then we know that it is the first customer that is going to abandon.
                    FR1 = i # want to find which row of the EC and the ET correspond the abandonment of the customer
                    break

                #Delete/ remove and item from a list of EC and ET
            EC.pop(FR1)
            ET.pop(FR1)

        #check this line above

    #assigning customers to taxi
    elif ET[min_index][0] == 2: #event type: Customer arrival
        TNOW = Tnext
        X1 = np.random.uniform(0,2)
        Y1 = np.random.uniform(0,2)
        Des_X = np.random.uniform(0,2) #Generate destination points when customer arrive
```

```python
            Des_Y = np.random.uniform(0,2)

            NumC += 1
            total_C += 1
            LC.append([0, X1, Y1, Des_X, Des_Y, total_C])
            EC[1] = TNOW + np.random.exponential(1/lambda2)


            Min_Distance = 10000
            found_2 = 0
            Taxi_found_2 = 0

            #Check if is there an idle taxi ?
            if NumT > 0:
                for i in range(NumT): #loop through the list of taxi to find the closest taxi and assign it to the customer.

                    if LT[i][0] == 0: #if taxi i-th is idle, then we found one but still need to check for min distance.
                        found_2 = 1
                        Distance = math.sqrt((LT[i][1]-X1)**2 + (LT[i][2]-Y1)**2)
                        if Distance < Min_Distance:
                            Min_Distance = Distance
                            Taxi_found_2 = i

                            #dont use break here because if we break then we would never find the closest taxi, but rather we found a taxi.

            if found_2 == 1:
                #LC[Taxi_found][0] = 1
                LC[NumC-1][0] = 1 #customer i-th become busy
    #                 print("Taxi_found = ", Taxi_found)

                num_of_busy_taxi += 1
                num_of_busy_customer += 1

                EC.append(TNOW + time_coef*(Min_Distance + math.sqrt((LC[NumC-1][1]-LC[NumC-1][3])**2 +
                                                                        (LC[NumC-1][2]-LC[NumC-1][4])**2)))
                ET.append([7, LT[Taxi_found_2][3], total_C]) #keep track of the ID of the taxi.
                LT[Taxi_found_2][0] = 1
                #ET.append([7, Taxi_found, NumC]) =  ET.append([7, NumT, Customer_assigned])

            else: #if found == 0 then assign departure time for customers.
                ET.append([5, 0, total_C])
                EC.append(TNOW + np.random.exponential(1/aban2))


            #We need to assign the 0-th taxi to customer.
            #The first bracket denote the index for each taxi.

                #Think about assigning a taxi that is going to abandon but has to finish service




        elif ET[min_index][0] == 3: #event type: Termination

            # out put performance measure
            TNOW = T
            Trip_complete = Num_trip #record the total number of trip completed by all taxi.
#             print("total_throughput = ",total_throughput )
#             print("total_customer = ", total_customer)
#             print("total_taxi = ", total_taxi)

            Customer_throughput = weird_division(total_throughput,total_customer)# denote the proportion of served customer who has left the system.

            Average_customer_in_system = total_customer/T


            #this variable tells you the utilization rate of all the taxis if you treat all taxis like one single taxi.
            #you can treat all taxis like a single sever becuase if you add up the 0-th entry of the LT's,it tells you
            #the number of times all taxis is busy.

            print("total_utilization  =", total_utilization)

            utilization = total_utilization/total_taxi # Area of the green curve divided by the area under the red curve.



            abandonment = NumC_aban/T


            Proportion_abandonment = abandonment/lambda2

            rate_complete = weird_division(Trip_complete,T)

            #try to find the utilization for all customers ?


            print("Total taxi =", total_T)
            print("Total customer =", total_C)

#             print("TOTAL BUSY TAXI =", num_of_busy_taxi) #this variable keep tracks of the number of time the taxis become busy


            print("Total number of trip completion =", Trip_complete)
            print("Customer left without service = ", NumC_aban)

            print("rate of abandonment = ", abandonment)

            print("total_utilization", total_utilization)

            print("utilization=", utilization)

#             print("total_taxi = ", total_taxi)


            print("Customer_throughput = ", Customer_throughput)
            print("Average_customer_in_system = =", Average_customer_in_system)

            print("rate_complete = ", rate_complete)


            print("END SIMULATION.")
            print("--------------------------------------------------------")

            # #total_trip_complete =
            # #num_cus_abandon =
            # #num_taxi_abandon =
            # #average_customer_cycle_time = number_customer/total_cus_left
            # #average_taxi_cycle_time = number_taxi/total_taxi_left
            return total_T, total_C, num_of_busy_taxi, Trip_complete,\
            abandonment, utilization, NumC_aban,\
            list_utili, Customer_throughput, Average_customer_in_system, Proportion_abandonment, rate_complete




        elif ET[min_index][0] == 4: #event type: taxi finish shift and leave
            TNOW = Tnext
            t_found_4 = 0 #identify the taxi that is going to leave
            taxi_considered_4 = ET[min_index][1]

#             print("HERE I AM IN ET == 4, TRYING TO DELETE OR ELSE THIS TAXI:",ET[min_index])
#             print("ET = ",ET)
#             print("ET[min_index] = ", ET[min_index])
#             print("taxi_considered = ",taxi_considered_4)
#             print(LT)
#             print("EC_min_index = ",EC[min_index])

            for k in range(len(LT)):
    #                 print("k=",k, "LT[k][3]=", LT[k][3] , "taxi_considered =",taxi_considered_4)
                if LT[k][3] == taxi_considered_4: #this check the id: id(LT[k][3]) == id(taxi_considered)
                    t_found_4 = k #index of the taxi ID.
                    break
    #             print("taxi status = ", LT[t_found_4][0])

            if LT[t_found_4][0] == 1: #if the taxi we are considering is busy
    #                 print("taxi busy = ", taxi_considered_4)

                for i in range(len(EC)):
    #                     print("i=",i)
    #                     print("lenET = ",len(ET))
                    if ET[i][0] == 7 and ET[i][1] == taxi_considered_4:
                        if EC[min_index] == EC[i]: #if EC of taxi abandoment = to EC of taxi complete the final trip.
    #                         print("index is", i, EC[min_index], EC[i])
                            Num_trip += 1 #record the total trip completions.
#                             print("A TRIP HAS COMPLETED.")
#                             print("ET[i][2] = ",ET[i][2])
#                             print("LC=",LC)

                            for j in range(len(LC)):
                                if ET[i][2] == LC[j][5]: #check if the ID of the considered customer is actually the customer considered
                                    LC.pop(j)
                                    NumC -= 1

            #                                   print("deleted", i)
                                    break
                            num_of_busy_customer -= 1
                            EC.pop(i) #delete i-th index of the trip completed that is associated with EC and ET.
                            ET.pop(i)

                            EC.pop(min_index) #delete index of the taxi abandonemnt event that is associated with EC and ET.
                            ET.pop(min_index)
                            LT.pop(t_found_4)
                            NumT -= 1
                            num_of_busy_taxi -= 1



                        else:
                            EC[min_index] = EC[i]
        #               print("postion 2")
                        break


            else: #if there is no customer to arrive before taxi finish shift then just delete taxi
                EC.pop(min_index)
                ET.pop(min_index)
                LT.pop(t_found_4)
                NumT -= 1
    #       print("Im here")


        elif ET[min_index][0] == 5: #event type: customer run out of patience
            #delete appropriate column from LC, ET, EC, update numC.
            TNOW = Tnext
            NumC_aban += 1
            Customer_found_5 = 0
            Customer_depart_5 = ET[min_index][2]
            for i in range(NumC):
                if LC[i][5] == Customer_depart_5:
                    Customer_found_5 = i
                    break
                    EC.pop(i)
                    ET.pop(i)

            LC.pop(Customer_found_5) # NumC - 1 should be after LC.pop and the same thing apply for LC
            NumC -= 1
            EC.pop(min_index)
            ET.pop(min_index)





        elif ET[min_index][0] == 7: #event type: taxi taking customer to destination
            TNOW = Tnext
            Customer_found_7 = 0 #dummy variable which will hold the index of the serving taxi.
            Num_trip += 1 #this variable hold the total number of trip completion.

            #Find the customer who is going to finsh his trip
            Customer_complete_7 = ET[min_index][2] #denote the ID of the customer who is going to finsh his trip.
            for i in range(NumC):
                if LC[i][5] == Customer_complete_7:
                    Customer_found_7 = i
                    break
#             print("Customer_found_7=", Customer_found_7)

            #Find the driver that is taking this customer to finish the trip
            Taxi_complete_7 = ET[min_index][1] #denote the ID of the taxi who is taking the customer to finish her service.
            Taxi_found_7 = 0

            for j in range(NumT):
                if LT[j][3] == Taxi_complete_7:
                    Taxi_found_7 = j
                    break
#             print("Taxi_found_7=", Taxi_found_7)
#             print("TAxi: complete, found", Taxi_complete_7, Taxi_found_7)
#             print("ET[min_index][2]",ET[min_index][2])
#             print("HERE I am at ET 7 inside the very first FOR loop")
#             print("Customer_found", Customer_found_7)
#             print("I HAVE JUST FINISHED SERVING THIS CUSTOMER: ", Customer_complete_7)

            X_cur = LC[Customer_found_7][3] #update thecurrent positition of taxi i-th after dropping off customer j.
            Y_cur = LC[Customer_found_7][4]

            #Update the status and the location of the current taxi to the dropped off coordinates

#             print(ET[min_index])
#             print("Taxi_found_7", Taxi_found_7)
#             print("LT", LT)

            LT[Taxi_found_7][0] = 0
            LT[Taxi_found_7][1] = X_cur
            LT[Taxi_found_7][2] = Y_cur

    #             print("X_cur = ", X_cur)
    #             print("Y_cur = ", Y_cur)

            #assign a different customer to same taxi
            Min_Distance = 10000
            assigned_new_7 = 0
            Assigned_7 = 0 # this indicate whether or not we have assgined a customer. This is as same as Assigned= 0.

#             print("IM going to delete: ", Customer_found_7)
            LC.pop(Customer_found_7)
            EC.pop(min_index)

            #if we don't record ET[min_index] to the variable taxi_index then after we pop min_index from ET, the algorithm
            # will assign a wrong taxi to an idle customer.

            taxi_index = ET[min_index][1]
            ET.pop(min_index)
#             print("I just deleted: ", Customer_found_7)
            NumC -= 1
#             print("Customers after deleting:", LC)

            # the problem is we try to assign customer to driver, try assign driver to customer instead. This don't work because
            # the problem is not in how we are assigning them, but rather which taxi is being assigned to a customer.
            # and we don't want to assign a busy taxi to a idle customer.

            if NumC > 0: #check to see if there is any other customer close by.
                 for k in range(NumC): #range take value from 0 -> NumC-1
                    if LC[k][0] == 0:
                        Assigned_7 = 1
                        #check if the location of the i-th taxi that just dropped off customer 1 is close to customer 2
                        Distance = math.sqrt((LC[k][1]-X_cur)**2 + (LC[k][2]-Y_cur)**2)
                        if Distance < Min_Distance:
                            Min_Distance = Distance
                            assigned_new_7 = k
#                             print("In computing distance; this is the favourite:", LC[assigned_new_7])

            if Assigned_7 == 1:

                num_of_busy_taxi += 1
                num_of_busy_customer += 1

                LT[Taxi_found_7][0] = 1
                LC[assigned_new_7][0] = 1 # make customer i-th become busy
    #                 print("assigned_new=",assigned_new_7)
                EC.append(TNOW + time_coef*(Min_Distance + math.sqrt((LC[assigned_new_7][1]-LC[assigned_new_7][3])**2 +
                                                                        (LC[assigned_new_7][2]-LC[assigned_new_7][4])**2)))
                ET.append([7, taxi_index, LC[assigned_new_7][5]])

                #The below step make sure that we delete the abandonment of customer once they have been assigned a taxi.
                FR7 = -1 #found row
                for m in range(len(EC)):
#                     print("im in FR7")
#                     print("m =",m)
#                     print("assigned_new_7 = ",assigned_new_7)

                    if ET[m][2] == LC[assigned_new_7][5]: #this condition check if this event is an abandonment and it also check if the event is correspond to the customer we are deal with.
                        #Customer assigned + 1 is because if Customer assigned = 0 then we know that it is the first customer that is going to abandon.
                        FR7 = m # want to find which row of the EC and the ET correspond the abandonment of the customer
                        break

                #Delete/ remove and item from a list of EC and ET
                EC.pop(FR7)
                ET.pop(FR7)
#                 print("I have just executed if Assigned = 1 ")
#                 print("FR =", FR)
#                 print("m = ", m)

            else:
#                 print("Taxi_found_7", Taxi_found_7)
                LT[Taxi_found_7][0] = 0 # min_index never used on LT and LC
                LT[Taxi_found_7][1] = X_cur
                LT[Taxi_found_7][2] = Y_cur
                num_of_busy_taxi -= 1
#                 print(" I COULD not find a customer to match and I have just executed else condition of ET==7")

#             ET.pop(min_index)
#             EC.pop(min_index)




#         print("Event Calendar = ",EC)
#         print("Event type = ",ET)
#         print("Min_index = ", min_index)

#         print("TNOW = ",TNOW)
#         print("Tnext = ",Tnext)
#         print("List taxi = ",LT)
#         print("NumT = ", NumT)
#         print("List customer = ",LC)
#         print("NumC = ", NumC)

#         print("total_T = ", total_T)
#         print("total_C = ", total_C)
#         print(Num_trip)

#         print("----------------------------------------------------")
#         print("----------------------------------------------------")



    #Number of customer should be equal to the total number of of Event type 5 and 7.
    #Number of Taxi should be equal to the total number of of Event type 4.
    #Don't schedule a new event when you delete: dont do ET or EC.append
    # NumC -1 or NumT - 1 always after pop.LC and pop.LT
```

Listing 1: Python main simulation code

```
def weird_division(n, d): # for when dividing by zero due to low rate of arrivals.
    return n / d if d else 0


def drange(start , stop , step): #not including the end point
    r = start
    while r < stop:
        yield r
        r += step
```

Listing 2: Python function that do some loop operations and function that assist with division by zero when the input parameters are small which may cause the value of some of our performance measures to be zero

```python

#1 PLOT DISTRIBUTION FIRST
#2 THEN DO POWER ANALYSIS
#3 HYOPETHESIS TESTING USE DIFFERENT SEED.

import numpy as np, statsmodels.stats.api as sms
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import scipy
from scipy.stats import ttest_ind
from scipy.stats import t

acc_utilization_1 = []
acc_utilization_2 = []


acc_abandonment_1 = []
acc_abandonment_2 = []

acc_trip_complete_1 = []
acc_trip_complete_2 = []


for seed in range(600,610):

    np.random.seed(seed)

    total_T, total_C, num_of_busy_taxi, Trip_complete,\
    abandonment, utilization, NumC_aban,\
    list_utili, Customer_throughput, Average_customer_in_system, Proportion_abandonment, rate_complete\
    = sim_taxi_Furthest(lambda1 = 1, lambda2 = 1, aban1= 1, aban2=1, time_coef=1, T=7500)



    acc_abandonment_1.append(abandonment)
    acc_utilization_1.append(utilization)
    acc_trip_complete_1.append(rate_complete)

print("Next model in coming")


for seed_2 in range(610,620):

    np.random.seed(seed_2)

    total_T, total_C, num_of_busy_taxi, Trip_complete,\
    abandonment, utilization, NumC_aban,\
    list_utili, Customer_throughput, Average_customer_in_system, Proportion_abandonment, rate_complete\
    = sim_taxi_Euclidean(lambda1 = 1, lambda2 = 1, aban1=1, aban2=1, time_coef=1, T=7500)

    acc_abandonment_2.append(abandonment)
    acc_utilization_2.append(utilization)
    acc_trip_complete_2.append(rate_complete)

###########################################################

X1 = acc_trip_complete_1
X2 = acc_trip_complete_2

print((X1))

print((X2))

cm = sms.CompareMeans(sms.DescrStatsW(X1), sms.DescrStatsW(X2))
print(cm.tconfint_diff(alpha = 0.05,  alternative='two-sided', usevar='unequal'))


res = ttest_ind(X1, X2, equal_var = "False")

print(res)
```

Listing 3: Python code for confidence interval analysis (we mainly use this code for Welch's t-test)

```python
import pandas as pd
import seaborn as sns

sns.set()

# df = pd.DataFrame({
#      'Manhattan': X1,
#      'Euclidean': X2,})


# ax = df.plot.kde(bw_method = 0.3,title='Kernel density estimation (bandwidth = 0.3)')



#PLOT USING SEABORN
fig = sns.kdeplot(X1, shade=True, color="r",  bw_method="scott", label = "KDE of the FD")
fig = sns.kdeplot(X2, shade=True, color="b",  bw_method="scott", label = "KDE of the ED")
# plt.yscale("log")
plt.ylim((-2, 100))
plt.xlabel("Rate of trip completion")
plt.title('KDE for trip completion rate of FD and ED')
plt.legend(loc="upper left")


#SAVE FIGURE HERE

fig = fig.get_figure()
# fig.savefig("KDE_rate_complete_FD_ED.png")



plt.show()
```

Listing 4: Python code for distribution plot

```python
#2

from math import sqrt
from statsmodels.stats.power import TTestIndPower
import numpy as np

#calculation of effect size
# size of samples from 2 distributions.
n1 = 20
n2 = 20

# Sample Standard deviations
sd1 = np.std(X1)
sd2 = np.std(X2)

# calculate the pooled standard deviation
# (Cohen's d)
s = sqrt((((n1 - 1) * sd1**2 + (n2 - 1) * sd2**2)) / (n1 + n2 - 2))

# means of the samples
m1 = np.mean(X1)
m2 = np.mean(X2)

# Cohen'd effect size:
d = abs(m1-m2) / s
print(f'Effect size: {d}')

# factors for power analysis
alpha = 0.05
power = 0.8

# perform Power Analysisto find sample size
# for given effect
obj = TTestIndPower()
n = obj.solve_power(effect_size=d, alpha=alpha, power=power,
                    ratio=1, alternative='two-sided')

print("d = ", d)
print("alpha = ", alpha)
print("power = ", power)

print("mean 1 =",m1)
print("mean 2 =",m2)
print("SD = ", s)
print("number of sample for each group = ",n)


#CALCULATE THE POWER OF A HYPOTHESIS TEST GIVEN n, EFFECT SIZE AND ALPHA.

from statsmodels.stats.power import TTestPower

power = TTestPower()
n_test = power.solve_power(nobs = 10, effect_size = d, power = None, alpha = 0.05)

print('Power: {:.3f}'.format(n_test))
```

Listing 5: Python code for Power Analysis

```python
import numpy as np
from scipy.stats import ttest_ind
from scipy.stats import t
import pandas as pd

def welch_ttest(x1, x2,alternative):

    n1 = len(x1)
    n2 = len(x2)
    m1 = np.mean(x1)
    m2 = np.mean(x2)

    v1 = np.var(x1, ddof=1)
    v2 = np.var(x2, ddof=1)

    pooled_se = np.sqrt(v1 / n1 + v2 / n2)
    delta = m1-m2

    tstat = delta /  pooled_se
    df = (v1 / n1 + v2 / n2)**2 / (v1**2 / (n1**2 * (n1 - 1)) + v2**2 / (n2**2 * (n2 - 1))) # f_hat or degree of freedom.

    # two side t-test
    p = 2 * t.cdf(-abs(tstat), df)

    # upper and lower bounds
    lb = delta - t.ppf(0.975,df)*pooled_se
    ub = delta + t.ppf(0.975,df)*pooled_se

    return pd.DataFrame(np.array([tstat,df,p,delta,lb,ub]).reshape(1,-1),
                         columns=['T statistic','df','pvalue 2 sided','Difference in mean','lb','ub'])




# REMEMBER TO CHANGE THE VALUE OF ALPHA/2 IN LB AND UB





acc_utilization_1 = []
acc_utilization_2 = []


for seed in range(0,20):
    np.random.seed(seed)

    total_T, total_C, num_of_busy_taxi, Trip_complete,\
    abandonment, utilization, NumC_aban,\
    list_utili, Customer_throughput, Average_customer_in_system, Proportion_abandonment\
    = sim_taxi_without_delay(lambda1 = 10, lambda2 = 5, aban1=1, aban2=1, time_coef=1, T=7500)

    acc_utilization_1.append(utilization)

for j in range(20,40):
    np.random.seed(j)

    total_T, total_C, num_of_busy_taxi, Trip_complete,\
    abandonment, utilization, NumC_aban,\
    list_utili, Customer_throughput, Average_customer_in_system, Proportion_abandonment\
    = sim_taxi_with_delay(lambda1 = 10, lambda2 = 5, aban1=1, aban2=1, time_coef=1, T
    =7500)

    acc_utilization_2.append(utilization)


x1 = acc_utilization_1
x2 = acc_utilization_2


welch_ttest(x1, x2,"not equal")
```

Listing 6: Python code for calculating Welch t-test in a different way

```python

#plot graph side by side in this cell


import matplotlib.pyplot as plt
import seaborn as sns

sns.set()

fig, axs = plt.subplots(1,3, figsize=(20,5)) # 1x3 so you have a row of plots


for i, aban in enumerate([1,5,10]): # loop through 3 values first

    acc_total_T = [] #accumulated list of total taxi of each simulation run.
    acc_total_C = []

    acc_num_of_busy_taxi = []

    acc_Trip_complete = []

#     acc_taxi_time_arrival = []

    acc_abandonment = []
    acc_utilization = []

    acc_lambda = []
    acc_throughput = []
    acc_NumC_aban = []

    for index in drange(0.01,20,0.2):

        print("lambda2 = ",index)

        total_T, total_C, num_of_busy_taxi, Trip_complete, abandonment, \
        utilization, NumC_aban, list_utili, Customer_throughput, \
        Average_customer_in_system, total_T_accum, taxi_arrival, t_now_acc,\
        utilization_acc, rate_aban_acc, utilization_each_taxi\
        = sim_taxi(lambda1 = index, lambda2 = 1, aban1 = 1, aban2= aban, time_coef=1, T=1000)


#         acc_taxi_time_arrival(variable_taxi_arrival_time)
        acc_total_T.append(total_T)
        acc_total_C.append(total_C)
        acc_num_of_busy_taxi.append(num_of_busy_taxi)
        acc_Trip_complete.append(Trip_complete)
        acc_abandonment.append(abandonment)
        acc_utilization.append(utilization)
        acc_NumC_aban.append(NumC_aban)
        acc_throughput.append(Customer_throughput)

        acc_lambda.append(index)
#         fig = ax.figure()
    print("acc_utilization =", acc_utilization)



    axs[i].plot(acc_lambda, acc_utilization, 'b')
    axs[i].set_title("lambda2 = " + f'{aban}')




#         axs[i].xlabel('lambda2')
#         axs[i].ylabel('Utilization')
```

```python
for ax in axs.flat:
    ax.set(xlabel='lambda1', ylabel='Taxi utilization')
    ax.set_xticks(np.arange(0, 21, 1))
    ax.set_yticks(np.arange(0, 1.1, 0.1))


fig.suptitle('Utilization vs lambda1 (with varying lambda2 = 1,5,10)')

# SAVE FIGURE HERE:


# plt.savefig('uti_vs_lambda1_(vary_lambda2).png')
plt.show()



# print("Printing accumulated elements")
# print("acc_total_T = ",acc_total_T)
# print("acc_total_C = ",acc_total_C)
# print("acc_utilization =",acc_utilization)




# # Hide x labels and tick labels for top plots and y ticks for right plots.
# for ax in axs.flat:
#     ax.label_outer()


# for index in np.arange(0.01, 20, 0.5):
#     sim_taxi(lambda1 = 1, lambda2 = index, aban1=1, aban2=1, time_coef=1, T=100)
#     print(index)

# for index in drange(1,10,1):
#     sim_taxi(lambda1 = 1, lambda2 = index, aban1=1, aban2=1, time_coef=1, T=100)
#     print(index)
```

Listing 7: Python loop with `enumerate` and plot functions side-by-side

```

import matplotlib.pyplot as plt
import seaborn as sns

sns.set()



total_T, total_C, num_of_busy_taxi, Trip_complete, abandonment, \
utilization, NumC_aban, list_utili, Customer_throughput, \
Average_customer_in_system, total_T_accum, taxi_arrival, t_now_acc,\
utilization_acc, rate_aban_acc,  utilization_each_taxi\
= sim_taxi(lambda1 = 1, lambda2 = 1, aban1= 1, aban2=1, time_coef=1, T=100)

print("total_T_accum = ",total_T_accum)


# Can not append at the end of our function because it will not taken into consideration
# of the time when there are no taxis. If you append at the end then before the simluation can reach the very end
# our NumT would have increased from 0 to 1 and it would have forgotten to taken into  account the NumC = 0

utilization_acc = utilization_acc


print("utilization_acc = ", utilization_acc)

total_T_accum = total_T_accum

taxi_arrival =  taxi_arrival
# print(taxi_arrival)

t_now_acc = t_now_acc


print("t_now_acc = ", t_now_acc)

print(len(total_T_accum))
print(len(t_now_acc))


# utilization_each_taxi = [100*x for x in utilization_each_taxi]

# x = total_T_accum
# y  = taxi_arrival
# plt.plot(total_T_accum, taxi_arrival)

# utilization_acc = [100*x for x in utilization_acc] # scale utilization by 100%

# plt.step(t_now_acc, total_T_accum, color='red', label='total number of taxis') # plotting total number of arrving taxi over time.
# plt.step(t_now_acc, utilization_acc, color='green', label='taxis utilization') # plotting utilization of the taxis over time.


# PROOF OF CONVERGENCE TO STEADY STATE.

plt.plot(t_now_acc, rate_aban_acc, color='fuchsia', label = 'rate of abandonment over  time') # this prove steady state convergence.
plt.plot(t_now_acc, utilization_each_taxi, color='cyan', label = 'utilization over time')

#LABEL

```

```python
plt.legend()
plt.xlabel('TNOW')
plt.ylabel('Number of Taxi')


# plt.step(x, y, where='mid', label='mid')
# plt.plot(x, y, 'o--', color='grey', alpha=0.3)

# #         fig = ax.figure()
#     print(acc_utilization)
#     axs[i].plot(acc_lambda, acc_utilization, 'b')
#     axs[i].set_title(f'{aban}')
# #         axs[i].xlabel('lambda2')
# #         axs[i].ylabel('Utilization')
# for ax in axs.flat:
#     ax.set(xlabel='lambda2', ylabel='acc_utilization')
# fig.suptitle('Utilization vs Customer arrival rate')




# SET FIGURE SIZE AND THIS WILL APPLY TO THE WHOLE FILE.
# plt.rcParams["figure.figsize"] = (10,15)

#SET THE RANGE FOR X AND Y COORDINATE.
# plt.yticks(np.arange(0, 1.1, 0.1))
# plt.xticks(np.arange(0, 101, 10))

plt.title('Number of taxi and their utilization over time')


#SAVE FIGURE HERE:
plt.savefig('NumT_utilization_over_time.png')


plt.show()







# # Hide x labels and tick labels for top plots and y ticks for right plots.
# for ax in axs.flat:
#     ax.label_outer()


# for index in np.arange(0.01, 20, 0.5):
#     sim_taxi(lambda1 = 1, lambda2 = index, aban1=1, aban2=1, time_coef=1, T=100)
#     print(index)

# for index in drange(1,10,1):
#     sim_taxi(lambda1 = 1, lambda2 = index, aban1=1, aban2=1, time_coef=1, T=100)
#     print(index)
```

Listing 8: Python code for convergence analysis and utilization plot