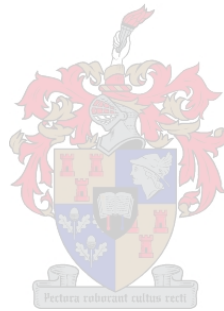


# A clustering approach towards the identification of suitable waiting areas for drivers of transportation network companies

by

Monica Rolo



Thesis presented in fulfilment of the requirements for the degree of  
**Master of Engineering (Industrial Engineering)**  
in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof JH van Vuuren

Co-supervisor: Mr GS Nel

March 2020



---

# Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2020





---

# Abstract

Technological innovation has transformed transportation into a more personal and customer driven experience. Traditional public transport, *e.g.* trains and busses, is limited to pre-defined routes that are fixed. Consequently, riders (*i.e.* passengers) have to travel along one of these routes regardless of their destination. Technology-driven companies, such as Uber, Bolt and Lyft, are so-called *transportation network companies* (TNCs) that provide riders with the option of selecting any pick-up and drop-off location. The nearest available driver (to the rider's pick-up location) is then assigned to the rider, thereafter the driver proceeds to pick up the rider who is then dropped off at their destination. The customer-driven approach of TNCs necessitates that the time elapsed between when a rider requests a ride and when they are picked up should be minimised. The rider's waiting time could be reduced by identifying suitable areas for drivers to wait between ride-requests.

In this thesis, clustering is adopted as a technique towards identifying waiting areas with the aim of reducing passenger waiting time. After an extensive review of the literature of popular clustering algorithms, *K*-means clustering is deemed best suited for the problem considered. The *K*-means algorithm is employed by clustering historical pick-up locations experienced by a well-known TNC in New York City in order to determine suitable waiting areas. Thereafter, the identified waiting areas are applied in the context of a developed agent-based model which represents the working of TNCs. The aim is to demonstrate the effectiveness of the areas identified by the clustering algorithm. The developed model simulates the movement of drivers together with the interactions between drivers and riders. The agent-based simulation model is subjected to the process of verification and validation, including parameter variation and a face validation by a subject matter expert.

Experimentation of eleven scenarios are simulated using the developed model. Two of the scenarios contain parking areas (*i.e.* openly accessible areas for drivers to park their vehicles) each employing a different strategy in respect of the driver's selection of parking areas. The other nine scenarios include the waiting areas discovered *via* clustering, where each scenario includes different parameter combinations for the clustering method employed. Thereafter, the scenarios are evaluated by means of statistical comparisons in order to determine if the differences between the outputs of the scenarios are statistically significant. Improvements in respect of the average and maximum waiting times as well as the average assignment times were identified as statistically significant.



---

# Opsomming

Tegnologiese innovasie het vervoer in 'n meer persoonlike en klantgedrewe ervaring omskep. Tradisionele openbare vervoer, byvoorbeeld treine en busse, is beperk tot voorafbepaalde roetes wat vas is. Gevolglik moet ruiters (d.w.s. passasiers) op een van hierdie roetes reis ongeag hul bestemming. Tegnologie-gedrewe ondernemings, soos Uber, Bolt en Lyft, is sogenaamde vervoernetwerkondernemings wat aan die ruiters die opsie bied om enige afhaal- en aflaaiplek te kies. Die naaste beskikbare bestuurder (na die plek van die ruiter) word dan aan die ruiter toegewys, waarna die bestuurder voortgaan om die ruiter op te tel wat dan by hul bestemming afgelaai word. Die klante-gedrewe benadering van vervoernetwerkondernemings noodsaak dat die tyd wat verloop tussen die tydstip wanneer 'n ruiter 'n rit vra en wanneer hulle opgetel word, tot die minimum beperk word. Die wagtyd van die ruiter kan verminder word deur geskikte gebiede te identifiseer vir bestuurders om tussen ritversoeke te wag.

In hierdie tesis word groepering gebruik as 'n tegniek om wagareas te identifiseer met die doel om die wagtyd van passasiers te verminder. Na 'n uitgebreide oorsig van die literatuur van gewilde groepering-algoritmes, word  $K$ -gemiddelde groepering as die beste geskik beskou vir die probleem wat oorweeg word. Die algoritme van  $K$ -gemiddelde groepering word gebruik om historiese oplaaiplekke van 'n bekende vervoernetwerkonderneming in New York te groepeer om geskikte wagareas te bepaal. Daarna word die geïdentifiseerde wagareas toegepas in die konteks van 'n ontwikkelde agentgebaseerde model wat die werking van vervoernetwerkondernemings verteenwoordig. Die doel is om die doeltreffendheid van die gebiede wat deur die groepering-algoritme geïdentifiseer is, te demonstreer. Die ontwikkelde model simuleer die beweging van bestuurders tesame met die wisselwerking tussen bestuurders en ruiters. Die agentgebaseerde simulasiemodel word onderwerp aan die proses van verifikasie en validering, insluitend parametervariasie en 'n gesigvalidering deur 'n vakkundige.

Eksploimentering van elf scenario's word gesimuleer met behulp van die ontwikkelde model. Twee van die scenario's bevat toeganklike gebiede vir bestuurders om hul voertuie te parkeer wat elkeen 'n ander strategie gebruik ten opsigte van die keuse van die bestuurder van parkeerareas. Die ander nege scenario's bevat die wagareas wat deur groepering ontdek is, waar elke scenario verskillende parameterkombinasies bevat vir die gebruik van die groepering algoritme. Daarna word die scenario's aan die hand van statistiese toetse geëvalueer om te bepaal of die verskille tussen die uitsette van die scenario's statisties beduidend is. Verbeteringe in die gemiddelde en maksimum wagtydperk asook die gemiddelde toekennings tyd is as statisties beduidend geïdentifiseer.



---

# Acknowledgements

The author wishes to acknowledge the following people and institutions for their various contributions towards the completion of this work:

- Stephan Nel, for the guidance that he provided throughout this year. His words of encouragement pushed me to strive for excellence. He taught me how to think like a true researcher and how to write scientifically. I am grateful for not only his commitment to my work but also for the friendship that developed over the course of this year.
- Prof Jan van Vuuren for welcoming me into SUnORE. Prof has been an inspiration to me over these two years and I feel honoured to have known him.
- My parents for being with me through all the late nights and for the endless amount of support and love. They both encouraged me every step of the way. I would not have been able to complete my postgraduate studies without them.
- My colleagues at *Stellenbosch Unit for Operations Research and Engineering* (SUnORE), for their friendship over these two years. Their willingness to help others is something I have never experienced in my life.



---

# Table of Contents

<b>Abstract</b>	<b>iii</b>
<b>Opsomming</b>	<b>v</b>
<b>Acknowledgements</b>	<b>vii</b>
<b>List of Acronyms</b>	<b>xv</b>
<b>List of Figures</b>	<b>xvii</b>
<b>List of Tables</b>	<b>xxi</b>
<b>List of Algorithms</b>	<b>xxiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Problem description . . . . .	5
1.3 Thesis objectives . . . . .	5
1.4 Thesis scope . . . . .	6
1.5 Thesis organisation . . . . .	6
<b>2 Machine Learning</b>	<b>9</b>
2.1 History of machine leaning . . . . .	9
2.2 Machine learning paradigms . . . . .	10
2.2.1 Supervised learning . . . . .	10
2.2.2 Reinforcement learning . . . . .	11
2.2.3 Unsupervised learning . . . . .	12
2.3 Clustering . . . . .	12
2.3.1 Distance measures . . . . .	12
2.3.2 Partitional clustering . . . . .	13

2.3.3	Hierarchical clustering . . . . .	18
2.3.4	Density-based clustering . . . . .	25
2.4	Chapter summary . . . . .	32
<b>3</b>	<b>Computer Simulation Modelling</b>	<b>33</b>
3.1	Simulation modelling overview . . . . .	33
3.2	Simulation modelling concepts . . . . .	34
3.3	Types of simulation models . . . . .	35
3.4	Simulation modelling paradigms . . . . .	35
3.4.1	Discrete event modelling . . . . .	36
3.4.2	System dynamics modelling . . . . .	36
3.4.3	Dynamic systems modelling . . . . .	37
3.4.4	Agent-based modelling . . . . .	38
3.5	Steps in a simulation study . . . . .	40
3.6	Verification and validation . . . . .	43
3.6.1	Verification . . . . .	43
3.6.2	Validation . . . . .	44
3.7	Advantages and disadvantages of simulation . . . . .	45
3.8	Review of simulation modelling in the context of TNCs . . . . .	45
3.9	Chapter summary . . . . .	46
<b>4</b>	<b>An Agent-Based Model of TNCs</b>	<b>47</b>
4.1	AnyLogic simulation modelling environment . . . . .	48
4.2	Simulating the movement of drivers . . . . .	48
4.3	Parking area selection . . . . .	49
4.4	Clustered waiting area selection . . . . .	50
4.4.1	Selection of input parameters . . . . .	51
4.4.2	Output analysis and verification . . . . .	52
4.5	The simulation experiment graphical user interface . . . . .	55
4.5.1	The main environment . . . . .	61
4.5.2	Functionality assessment of the GUI . . . . .	63
4.6	The simulation model . . . . .	66
4.6.1	Driver agent . . . . .	67
4.6.2	Rider agent . . . . .	70
4.6.3	Parking area agent . . . . .	72
4.6.4	Clustered waiting area agent . . . . .	73



Table of Contents	xi
4.6.5 Verification of the TNC simulation model . . . . .	74
4.6.6 Validation of the TNC simulation model . . . . .	75
4.7 Chapter summary . . . . .	79
<b>5 Experimentation and Results</b>	<b>81</b>
5.1 Experimental design . . . . .	81
5.1.1 Determining the number of drivers . . . . .	82
5.1.2 Specifications of the simulation model . . . . .	83
5.1.3 The simulation warm-up period . . . . .	84
5.1.4 Methods of statistical analysis . . . . .	86
5.2 Computational results . . . . .	89
5.3 Statistical evaluation . . . . .	91
5.4 Chapter summary . . . . .	93
<b>6 Summary and Conclusions</b>	<b>95</b>
6.1 Thesis summary . . . . .	95
6.2 Appraisal of thesis contributions . . . . .	96
<b>7 Future Work</b>	<b>99</b>
7.1 Simulation modelling suggestions . . . . .	99
7.2 Solution methodology suggestions . . . . .	101
<b>References</b>	<b>103</b>







---

## List of Acronyms

**AI:** Artificial intelligence

**ANOVA:** Analysis of variance

**AWT:** Assignment waiting time

**BIRCH:** Balanced iterative reducing and clustering using hierarchies

**CLARA:** Clustering for large applications

**CLINK:** Complete-link hierarchical algorithm

**CURE:** Clustering using representatives

**DBSCAN:** Density-based spacial clustering of applications with noise

**DENCLUE:** Density-based clustering

**DIANA:** Divisive analysis

**GIS:** Graphic Information System

**GUI:** Graphical user interface

**NYC:** New York City

**OPTICS:** Ordering points to identify clustering structures

**PAM:** Partitioning around medoids

**PMI:** Performance measure indicator

**SC:** Silhouette coefficient

**SLINK:** Single-link hierarchical algorithm

**SSE:** Sum of Squared errors

**TNC:** Transportation Network Company

**TWT:** Total Waiting Time



---

## List of Figures

1.1	Disruption threat of digital innovation . . . . .	2
1.2	Evolution of TNCs . . . . .	3
1.3	Growth of TNCs in NYC . . . . .	4
2.1	History of machine learning . . . . .	11
2.2	Reinforcement learning agents response to the environment . . . . .	12
2.3	Different distance measures on a Cartesian plane . . . . .	13
2.4	$K$ -means algorithm demonstrated on the Iris data set . . . . .	15
2.5	Elbow method applied to Iris data set . . . . .	16
2.6	Silhouette method applied to Iris data set . . . . .	17
2.7	Gap statistic method applied to Iris data set . . . . .	18
2.8	Dendrogram of Iris data set . . . . .	19
2.9	Clusters of Iris data set formed by the hierarchical clustering algorithm . . . . .	19
2.10	Proximity measures used in agglomerative hierarchical clustering . . . . .	20
2.11	Single-link method . . . . .	21
2.12	Complete-link method . . . . .	22
2.13	Average link method . . . . .	22
2.14	Ward's method . . . . .	23
2.15	Centroid method . . . . .	23
2.16	Median method . . . . .	24
2.17	Clustering non-spherical data structures . . . . .	25
2.18	DBSCAN relationships . . . . .	26
2.19	The $k$ -dist graph used to determine $\epsilon$ . . . . .	27
2.20	Clusters of varying densities . . . . .	28
2.21	Distances for OPTICS . . . . .	29
2.22	Reachability plot . . . . .	29
3.1	Abstraction levels of the different types of simulation modelling . . . . .	36

3.2	Example of discrete event model . . . . .	37
3.3	Example of system dynamics model . . . . .	37
3.4	Example of dynamic systems model . . . . .	38
3.5	A depiction of agents interacting among themselves . . . . .	39
3.6	Statechart example . . . . .	40
3.7	Transition triggers . . . . .	40
3.8	Steps in a typical simulation study . . . . .	41
3.9	Verification and validation process . . . . .	44
4.2	Parking area selection . . . . .	51
4.3	Clustering initialisation . . . . .	53
4.4	Snapping cluster centres to parking areas . . . . .	53
4.5	Verification of cluster type . . . . .	54
4.6	Variables declared in the Simulation environment . . . . .	55
4.7	Selection of waiting area type on the GUI . . . . .	55
4.8	Screenshot of the initial view of the GUI . . . . .	56
4.9	Selection of percentage on the GUI . . . . .	57
4.10	Selection of the time period of historical data on the GUI . . . . .	57
4.11	Screenshot of GUI with clustered waiting area and clustering type selected . . . . .	58
4.12	Screenshot of GUI with the selection of Clustered waiting area . . . . .	59
4.13	Screenshot of GUI with the selection of Parking area . . . . .	60
4.14	Declarations inside Main agent corresponding to the five components . . . . .	61
4.15	Screenshot of Main agent . . . . .	62
4.16	GUI functionality assessment of the number of drivers selection . . . . .	64
4.17	GUI functionality assessment of the waiting area type selection . . . . .	65
4.18	GUI functionality assessments of the cluster type selection . . . . .	66
4.19	Statechart of driver agent . . . . .	67
4.20	Components of driver agent . . . . .	67
4.21	Inner workings of the rider agent . . . . .	71
4.22	Components used in the generation of the rider agents . . . . .	71
4.23	Inner workings of the parking area agent . . . . .	72
4.24	Hotspot generation components of the simulation . . . . .	73
4.25	Components of the clustered waiting area agent . . . . .	74
4.26	Graphs produced for 60 driver agents during validation . . . . .	76
4.27	Graphs produced for 120 driver agents during validation . . . . .	76
4.28	Graphs produced for 180 driver agents during validation . . . . .	77



---

4.29	Graphs produced for Strategy 1 during validation . . . . .	78
4.30	Graphs produced for Strategy 2 during validation . . . . .	78
5.1	UberX growth in NYC . . . . .	83
5.2	Warm-up of period of the simulation experiments . . . . .	86
5.3	Average TWT box plots . . . . .	89
5.4	Maximum TWT box plots . . . . .	90
5.5	Average AWT box plots . . . . .	90
5.6	Maximum AWT box plots . . . . .	91
7.1	Batching driver-rider pairing strategy . . . . .	101



---

## List of Tables

2.1	A sample of the Iris data set . . . . .	14
2.2	Interpretation of SC . . . . .	17
2.3	Lance-Williams parameters . . . . .	20
4.1	A sample of the data set employed . . . . .	52
4.2	Experiments conducted for verification of cluster type . . . . .	53
5.1	Distribution of number of hours worked by Uber drivers . . . . .	82
5.2	Determining the number of hours worked per day by Uber drivers . . . . .	82
5.3	Summary of the scenarios for which experiments are conducted . . . . .	84
5.4	ANOVA and Bartlett's test results for Scenario 1 to Scenario 4 . . . . .	91
5.5	ANOVA and Bartlett's test results for Scenario 5 to Scenario 8 . . . . .	92
5.6	ANOVA and Bartlett's test results for Scenario 6 to Scenario 11 . . . . .	92
5.7	Differences between the average TWT of the Scenarios . . . . .	92
5.8	Differences between the maximum TWT of the Scenarios . . . . .	93
5.9	Differences between the average AWT of the Scenarios . . . . .	93



---

## List of Algorithms

2.1	<i>K</i> -Means clustering algorithm . . . . .	14
2.2	Agglomerative clustering algorithm . . . . .	19
2.3	Divisive clustering algorithm . . . . .	24
2.4	DBSCAN clustering algorithm . . . . .	27



---



---

## CHAPTER 1

---

# Introduction

### Contents

1.1	Background . . . . .	1
1.2	Problem description . . . . .	5
1.3	Thesis objectives . . . . .	5
1.4	Thesis scope . . . . .	6
1.5	Thesis organisation . . . . .	6

## 1.1 Background

Improvements in technology are rapidly altering the manner according to which individuals conduct their day-to-day lives. Mankind finds itself deep into the so-called *Digital Age*, also known as the Information Age, which started around 1975 and is predominantly characterised by the sheer extent to which an abundance of data can be gathered and transferred rapidly — as exemplified by the growth of information-based industries [21]. The Digital Age has brought about change in many industries such as commerce, communication and transportation, the latter of which is the focal point.

According to Deloitte [31], changes brought about by the Digital Age, which include smart devices, real-time planning, open traffic data and social customer service, will result in disruptive trends for transportation and smart mobility services. The first trend relates to the user-centred nature of mobility services which enable riders (*i.e.* passengers or users of mobility services) to exert greater control, increasing the personalisation of public transport. Smartphones and innovative applications afford people the ability to request a lift service, together with securing the cost, in advance. This innovation results in fewer cars being present on the roads and secures more jobs in the transportation industry. Furthermore, mobility services focus on meeting the demand of the riders by adapting and personalising the routes according to rider demands instead of only providing fixed routes. The capabilities associated with intelligent and integrated transportation networks with respect to measuring performance, demand and the condition of the vehicle is another notable trend. These systems work in real-time in order to predict and avoid problems and manage the vehicle's capacity. The changes concerning payment methods are another noteworthy trend. Tickets are digitised and payments require little to no human involvement. The advancement of different technologies facilitates the trend of automation and the improvement of safety which result in quality of life improvements for many people

around the globe. The disruption threat that digital innovation imposes on different industries is illustrated graphically in Figure 1.1. The industries depicted with green dots are applicable to the transportation industry. Industries that are subject to a high disruption threat level in conjunction with a high level of digital innovation — such as marketing, media and retail — are experiencing impediments in respect of well-established business (operational) practices, *i.e.* the manner according to which flights and accommodation are reserved as well as the promptitude thereof. In the public transport industry, digital innovation enables and facilitates drivers of public transport to provide a safer and more efficient service. Furthermore, the introduction of autonomous (self-driving) cars in the arguably near-future together with the increasing connected nature of vehicles are bound to have an unequivocal impact on the transportation industry.

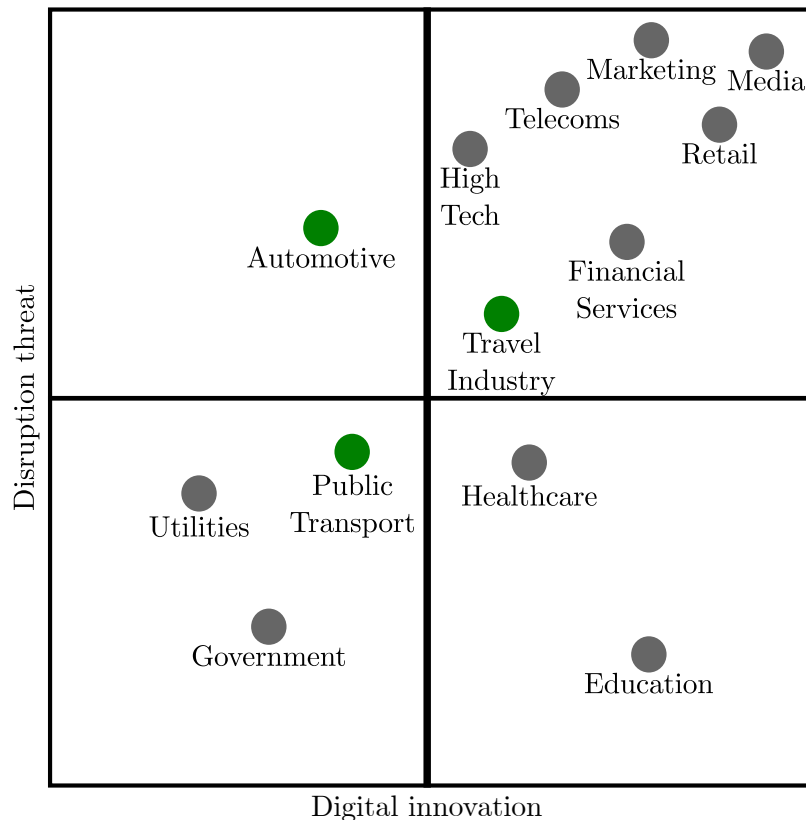


FIGURE 1.1: The disruption threat that digital innovation imposes on different industries [31].

*Transportation Network Companies* (TNCs), also known as mobility service providers, are organisations that use web- and mobile-based applications to pair drivers and riders so as to provide individuals with a private ride service [17]. These organisations epitomise the trend of affording greater control to the user, *i.e.* the rider. Consequently, the perception amongst the general public and key entities within the transportation industry is undergoing a significant change. As alluded to earlier, many of these services are rather simple — a rider is required to indicate their intended destination using an application on their smartphone, thereafter the estimated time of arrival, together with the estimated cost, is displayed on the application and a nearby driver is assigned. Some organisations provide the added functionality of displaying the driver's details on the application once the driver is assigned to the requested ride, such as the driver's number plate, name, car model and cellphone number. This provides the rider with greater transparency and peace of mind during the ride service and ultimately enables greater accessibility. After arrival and subsequent pick-up, the rider is then transported to the desired destination.



TNCs have evolved over the past two decades — Figure 1.2 depicts the chronological evolution. This evolution commenced in the late 1990 with Carsharing 1.0 where a service was provided allowing vehicles to be picked up by customers from a certain location and then dropped off at the same location. This service is known as *station-based* carsharing and is provided by companies such as Zipcar, Hertz On Demand and City Carshare. Thereafter, Carsharing 2.0 developed, which is regarded as *one-to-many* carsharing. Companies, such as DriveNow, car2go and Scoot, provide a service that allows for vehicles to be picked-up by customers and dropped off at different designated areas so as to improve the convenience of the service. Companies such as RelayRides and Getaround then started to provide a peer-to-peer carsharing service which enables individuals to rent out their vehicles to others when they are not used, officially establishing Carsharing 3.0. Currently, there are companies, such as Uber and Lyft, that provide ride-hailing services which enable drivers to be requested in real-time by using a purpose-built application on mobile devices. A service that emanates from ride-hailing, known as shared ride-hailing, is provided by companies such as Lyft line and UberPool, and allows riders travelling along similar routes to share a ride and its cost. Another service that stems from ride-hailing is microtransit services where riders travelling along similar routes are picked up in van-sized (*i.e.* medium sized) vehicles. Companies that provide this service, such as Via and Chariot, either provide flexible routing according to rider needs or have fixed routes along which the vehicles travel [17].

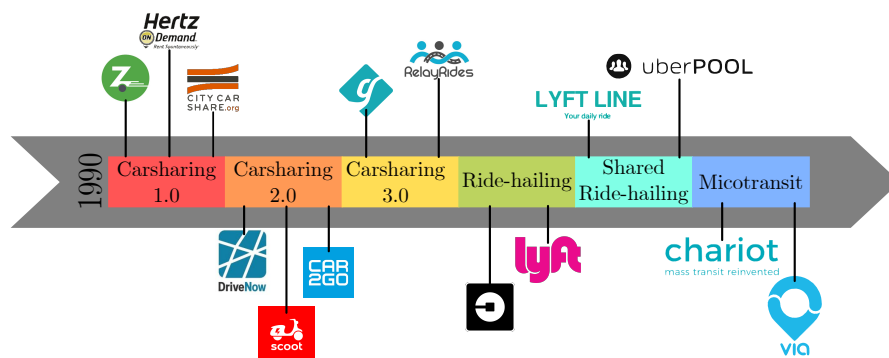


FIGURE 1.2: The evolution of TNCs over the past two decades [17].

The widely-known yellow taxi plays an important role in the transportation network of *New York City* (NYC). Yellow taxis are the only vehicles permitted to pick-up riders anywhere in NYC *via* street-hailing and prearranged ride-requests. At the start of 2015, yellow taxi's were completing approximately 459 000 trips per day which decreased to about 285 000 trips per day by the end of 2018. The decline in yellow taxis is largely attributable to the introduction and growth of TNCs. The growth of two prominent and influential TNCs, namely: Uber and Taxify, is contrasted graphically with the decline of yellow taxi trips per day in Figure 1.3.

In 2017, Clewlow and Mishra [17] conducted a survey so as to gain insight into the adoption, utilisation and early impacts of TNCs on the transportation industry. The survey was carried out in seven metropolitan areas in the United States, namely: Boston, Chicago, Los Angeles, NYC, San Francisco, Seattle and Washington DC. The results of the survey concluded that individuals in these areas prefer to use the services provided by TNCs as opposed to traditional public transport, mainly ascribed to the slow nature of public transport and the limited number

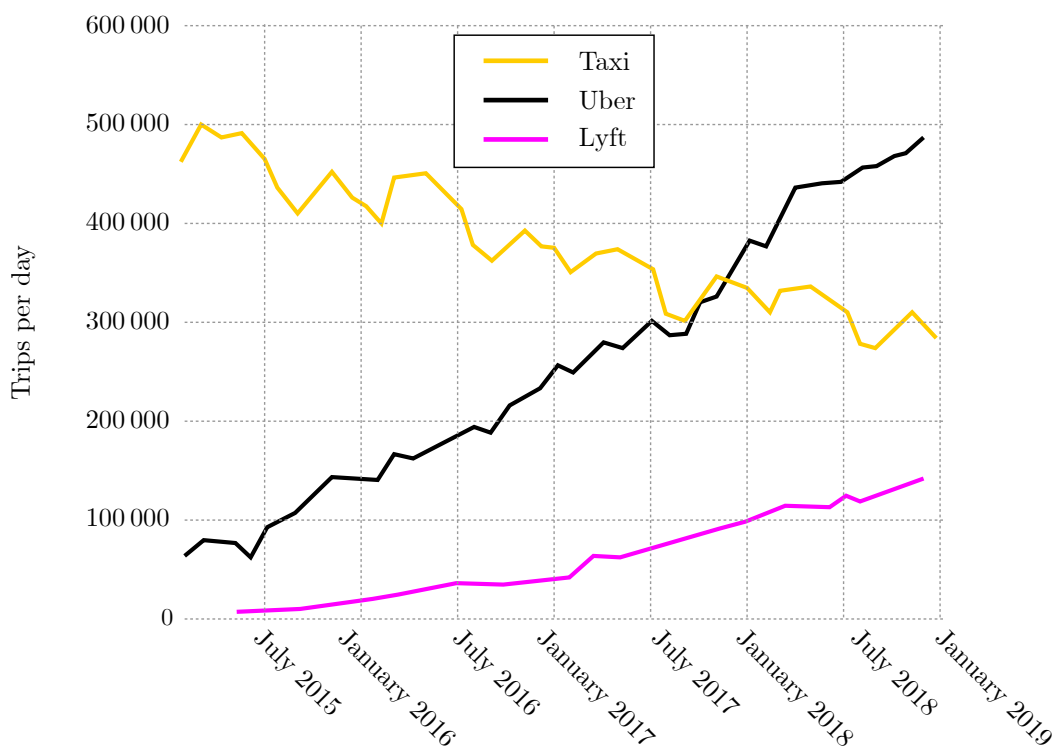


FIGURE 1.3: The growth of TNCs over a three and a half-year period and the resultant decline in yellow taxi trips per day [17].

of stopping locations. The traditional advantages associated with the use of public transport, *i.e.* the low cost and convenience pertaining to parking considerations are both advantages also associated with TNC services [80].

In order to ensure the continued growth and successful operation of TNCs, it is essential that drivers are provided with some form of decision support pertaining to effective trips. Upon requesting a ride-request from a rider, the driver receives two locations, *i.e.* the rider's pick-up and drop-off location. A driver's typical considerations when accepting a ride-request include the route from the current location to the pick-up location, the route from the rider's pick-up location to the rider's drop-off location and, lastly, the location where the driver must wait for the next ride-request upon completion of the current ride. These factors can have a significant impact on the customer service level — the longer the rider waits, the greater the level of dissatisfaction.

The customer-driven approach of TNCs necessitates that the time elapsed between when a rider requests a ride and when they are picked up should be minimised. Currently, in practice, the driver has access to routing software which provides information pertaining to the fastest (or shortest) route from the current location to the rider's pick-up location or drop-off location. The driver is, however, uninformed with respect to waiting locations between ride-requests. TNC can provide information to the drivers regarding hotspots, *i.e.* areas with historically high demand, however, this approach could result in drivers not being paired with riders in outlying areas or drivers travelling extensive distances to reach these riders. This problem can potentially be addressed by assigning drivers to waiting areas identified by means of a technique within the realm of unsupervised learning called *clustering*. Clustering is a technique which is able to group data based on the intrinsic characteristics and structure of the data. This technique can be employed to cluster pick-up locations in real-time so as to determine suitable waiting areas. This approach considers riders in outlying areas whilst taking into account riders in areas classified

as hotspots. To the best of the author's knowledge, the proposed approach towards choosing suitable waiting areas for TNC drivers by means of clustering presents possible yet meaningful benefits in respect of both the TNC and its customers in the form of reduced travel and waiting times.

## 1.2 Problem description

The problem considered in this thesis is to investigate to what extent clustering can be employed to identify suitable waiting areas for drivers of TNCs. The waiting areas are to be identified by means of clustering historical pick-up locations in real-time. The data to be clustered consists of the information regarding the pick-ups (*i.e.* pick-up location and time) of a well-known TNC during April 2014. Subsequently, the identified waiting areas are to be implemented in a detailed agent-based simulation model of a TNC where various clustering configurations (*i.e.* scenarios) are to be compared. The waiting areas identified by employing a clustering technique are also to be compared with openly accessible parking areas available to the driver.

## 1.3 Thesis objectives

The following eight objectives are pursued in this thesis:

- I To *conduct* a thorough survey of the literature related to:
  - (a) machine learning in general,
  - (b) clustering algorithms in particular, and
  - (c) simulation principles and guidelines, with a specific focus on agent-based simulation modelling.
- II To *create* a suitable agent-based simulation model for use as a benchmark for evaluating the effectiveness of different waiting area strategies within the context of TNCs. The simulation model should be formulated according to the research conducted in Objective I(c).
- III To *verify* and *validate* the working of the simulation model designed in Objective II.
- IV To *evaluate* different clustering algorithms in the literature according to the research conducted in Objective I(b) and *identify* the clustering algorithm best suited for the problem under consideration.
- V To *implement* the clustering algorithm deemed applicable in pursuit of Objective IV so as to cluster historical pick-up locations in order to identify suitable waiting areas.
- VI To *evaluate* the effectiveness of the clustered waiting areas of Objective V by means of the developed simulation model in pursuit of Objective II
- VII To *compare* the effectiveness of the waiting areas of Objective V statistically using the results obtained in pursuit of Objective VI.
- VIII To *propose* possible suggestions for future work in the context of TNCs as expansions of the work contained in this thesis.

## 1.4 Thesis scope

Due to the complexity of TNCs, the following assumptions are made:

**Clustering algorithms:** Only the clustering algorithms that are deemed popular in the literature are to be considered as a solution methodology to the problem considered in this thesis.

**Driver behaviour:** It is assumed that a driver accepts all ride-requests and, furthermore, that the drivers are to be active throughout the simulation run time, *i.e.* drivers do not enter or exit the system. Therefore, the number of drivers throughout the simulation is assumed to remain constant. This is a result of imperfect information regarding the fluctuation of the number of drivers throughout the day.

**Trip cancellations:** Ascribed to the fact that the data set only includes the riders that were picked up by a driver, it is assumed that no riders will cancel their trips before a driver arrives at the pick-up location. The riders, therefore, only leave the simulation when they have been dropped off by a driver.

**Cost to driver:** Since the purpose of this project is to reduce the time that riders wait to be picked up by a driver, the cost to the driver will not be taken into account. One can, however, infer cost implications based on reduction in driver travel times.

## 1.5 Thesis organisation

Excluding this introductory chapter, this thesis comprises six chapters. Chapter 2 presents a survey of the literature pertaining to machine learning in general with a focus on unsupervised learning — more specifically, clustering. This chapter provides a detailed overview of the clustering algorithms relevant to this study by discussing the working of the algorithms together with the associated advantages and disadvantages.

In Chapter 3, insight into computer simulation modelling is provided by describing the modelling technique, together with the various salient aspects pertaining to computer simulation modelling. The different types of simulation models are discussed together with the various modelling paradigms. Furthermore, the steps pertinent in a simulation model are presented, two of which are discussed in greater detail, *i.e.* verification and validation, along with the advantages and disadvantages associated with computer simulation modelling. The chapter then closes with a review of computer simulation modelling in the context of TNCs with respect to agent-based modelling.

The aim of Chapter 4 is to present the developed agent-based simulation model by providing a detailed description of the operation of the model together with the components that constitute the model. The chapter opens with a motivation and description pertaining to the simulation software chosen for the development of the model. A discussion on the movement of drivers of TNCs is then provided followed by the methods used to assign drivers to waiting areas. Furthermore, the graphical user interface is presented and its components verified. The chapter continues with a detailed description of the simulation model describing each agent in the model individually. Upon completion of the discussion of the model components, the verification and validation of the model is described.

In Chapter 5, the experimental design, which includes statistical analysis, is discussed. The description of the experimental design comprises the calculation of a suitable number of drivers in a simulation run, a discussion on the scenarios tested, the establishment of the simulation warm-

---

up period and the methods of statistical analysis to be applied. Thereafter, the computational results are analysed in full. This chapter concludes with a statistical analysis performed on the results obtained through experimentation.

A summary of the thesis contents is provided in Chapter 6 together with a discussion on the contributions made. Finally, in Chapter 7, possible avenues for follow-up work are presented which build on the work presented in this thesis.



---



---

## CHAPTER 2

---

# Machine Learning

### Contents

2.1	History of machine leaning . . . . .	9
2.2	Machine learning paradigms . . . . .	10
	2.2.1 <i>Supervised learning</i> . . . . .	10
	2.2.2 <i>Reinforcement learning</i> . . . . .	11
	2.2.3 <i>Unsupervised learning</i> . . . . .	12
2.3	Clustering . . . . .	12
	2.3.1 <i>Distance measures</i> . . . . .	12
	2.3.2 <i>Partitional clustering</i> . . . . .	13
	2.3.3 <i>Hierarchical clustering</i> . . . . .	18
	2.3.4 <i>Density-based clustering</i> . . . . .	25
2.4	Chapter summary . . . . .	32

This chapter aims to provide insight into the field of machine learning with a focus on an unsupervised learning technique known as clustering. In §2.1, a brief history of machine learning is presented. Thereafter, the three main paradigms of machine learning are discussed in §2.2, namely: Supervised learning, reinforcement learning and unsupervised learning. In §2.3, a description of clustering is provided which includes a discussion on popular distance measures which is followed by a detailed discussion on the most prominent types of clustering algorithms, namely: Partitional clustering, hierarchical clustering (both of which are distance-based algorithms) and density-based clustering. Lastly, the contents of this chapter are summarised in §2.4.

### 2.1 History of machine learning

Machine learning originated in the 20<sup>th</sup> century which coincided with the birth of *artificial intelligence* (AI). The official arrival of notations pertaining to AI can be attributed to a test developed by Alan Turing [79]. In 1950, Turing proposed the question “Can machines think?” resulting in the development of the Turing Test [101], which was employed to determine if a computer can imitate a human sufficiently so that it could possibly be mistaken as a human. This was followed by the first checkers-playing program developed by Arthur Samuel [77] in 1952, who later, in 1955, developed the first checkers-playing program which incorporated *learning* methods. The term AI was then “coined” in 1956 by John McCarthy who was the co-organiser of the seminal Dartmouth conference — a two month AI-focussed conference which was attended

by ten influential and prominent role players within machine learning [60]. Thereafter, in 1957, the *perceptron* was invented by Frank Rosenblatt [73] which is the simplest form of an *artificial neural network*.

Machine learning continued to evolve with the development of the *nearest neighbour algorithm* in 1967 by Clover and Hart [18]. This algorithm is a machine learning technique able to perform classification. Thereafter, in 1973, a report was published by James Lighthill, known as the Lighthill report [56]. The report aimed to assist the scientific research council in Britain with necessary decisions regarding funding work related to AI. Lighthill discredited the ability of AI to solve real-world complex problems. Consequently, a so-called “AI winter” was initiated — a term used to describe a great decline in funding towards AI projects [36]. Terrence Sejnowski [81] developed NETtalk in 1985 which was an artificial neural network that could convert English text to speech. This neural network consisted of 309 neurons arranged in three layers, an input layer which received words, a hidden layer and an output layer which produced the sounds.

In the late 20<sup>th</sup> century, machine learning enabled machines to play some board games at a human level. Gerald Tesauro developed an artificial neural network called TD-Cammon [98] in 1992 with the ability to play backgammon almost as proficiently as experts in the game. Thereafter, Deep Blue was developed at IBM Research [13] which defeated world champion chess player Garry Kasparov in 1997. Furthermore, IBM also developed a machine named Watson in 2007 [26] which was able to participate in a game of Jeopardy!. In 2011, Watson outperformed two of the highest ranked players in a two-game match. In 2015, AlphaGo [88] was developed using *reinforcement learning* to play the notoriously difficult game Go. AlphaGo went on to defeat the European champion Fan Hui and after further development AlphaGo defeated Lee Sedol — an eighteen-time world champion.

It is clear that the field of machine learning has evolved into a powerful utility that can be used to solve markedly difficult problems. In Figure 2.1, the profound accomplishments of machine learning are abundantly clear. Today, machine learning is at the forefront of many breakthroughs, showing little to no signs of stagnation. Research in this promising field is therefore well warranted.

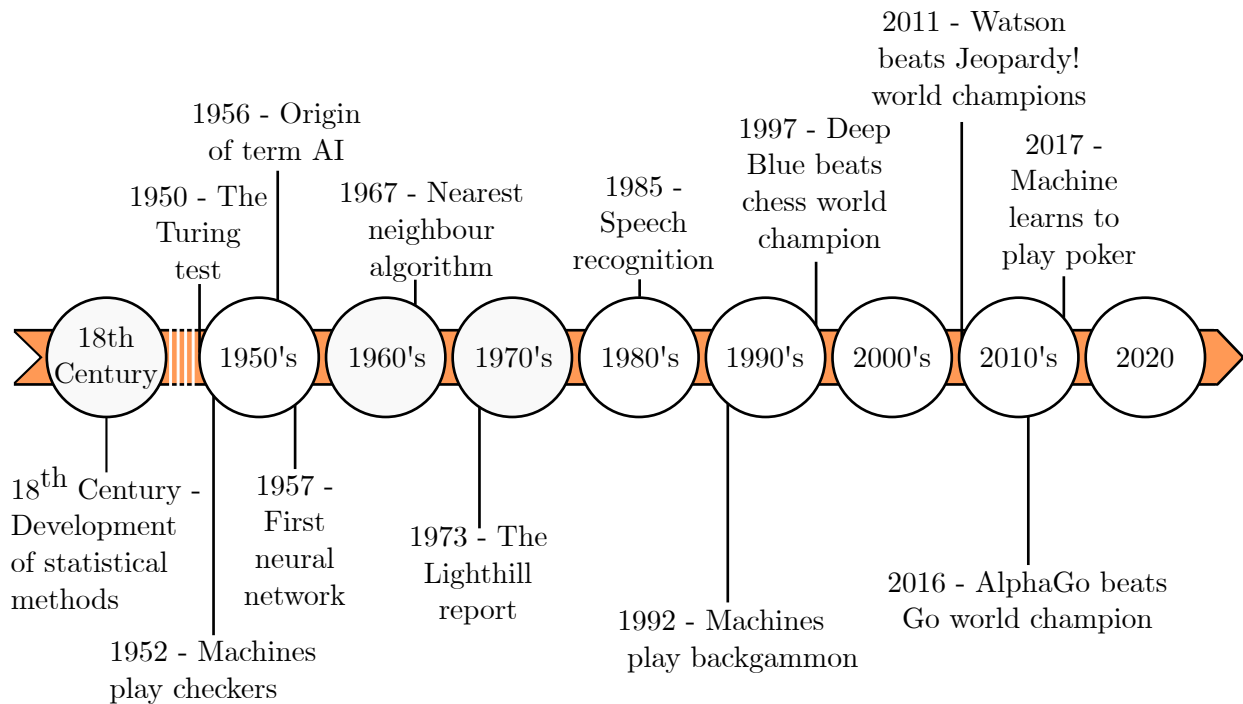
## 2.2 Machine learning paradigms

Machine learning can be categorised according to the learning paradigm employed. A computer program is said to learn from experience  $E$  in task  $T$  with performance measure  $P$  if the performance  $P$ , improves with experience  $E$  [63]. Accordingly, there are three popular paradigms of machine learning, namely: *Supervised learning*, *reinforcement learning* and *unsupervised learning* [8].

### 2.2.1 Supervised learning

Supervised learning is a method of machine learning which includes the investigation of input-output pairs  $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$  where  $y_i$  is determined by an unknown function  $y_i = f(x_i)$ . The training set is used in order to determine a function  $h$ , known as the hypothesis, which describes this input-output behaviour and is an approximation of the true function  $f$ . A test set is then employed to test the accuracy of the hypothesis. The model is provided with the  $x$ -values of this test set and the accuracy of the hypothesis is measured by evaluating whether the correct  $y$ -values are determined. The ability of a model to predict the output of instances not in the training set is known as *generalisation*. There are two different types of supervised learning



FIGURE 2.1: *History of machine learning* [75, 99].

approaches, namely: Regression and classification. The output of a regression is continuous, for example, temperature or distance, on the other hand, the output of a classification model is a finite set of values or classes, for example, the classification of a picture as a dog, cat or bear or the classification of facial expression as happy or sad [76]. Popular models used for supervised learning include decision trees, linear regression, nearest neighbour models, support vector machines, naive bayes and artificial neural networks [24].

### 2.2.2 Reinforcement learning

In reinforcement learning, an agent learns to optimise its actions based on its environment. The agent, *i.e.* an object that performs an action, receives a penalty if its action was incorrect and a reward if the action was correct with respect to some objective. The aim of the agent is then to select a sequence of actions that maximise the cumulative reward.

An agent is able to interpret its environment through its sensors and responds to the environment through its actuators, for example, a human agent uses its sensors, such as its eyes, ears and nose, to perceive its environment, on the other hand, it uses its actuators, such as its legs, arms and hands, to perform an action. An agent can only act on what it has already perceived which is known as the agent's percepts sequence. A rational agent can be defined as an agent that performs an action in order to achieve the best result. This agent will therefore choose the correct action in response to its environment, *i.e.* maximise the reward and minimise the penalty. The agent determines the correct action by evaluating the consequences this action has on its state. If this outcome is desirable then the action is performed. The desirability of the outcome can be measured using different performance measures. A rational agent can then be redefined as an agent that performs actions in order to maximise its performance measure [76]. The interaction between an agent and its environment is illustrated graphically in Figure 2.2.

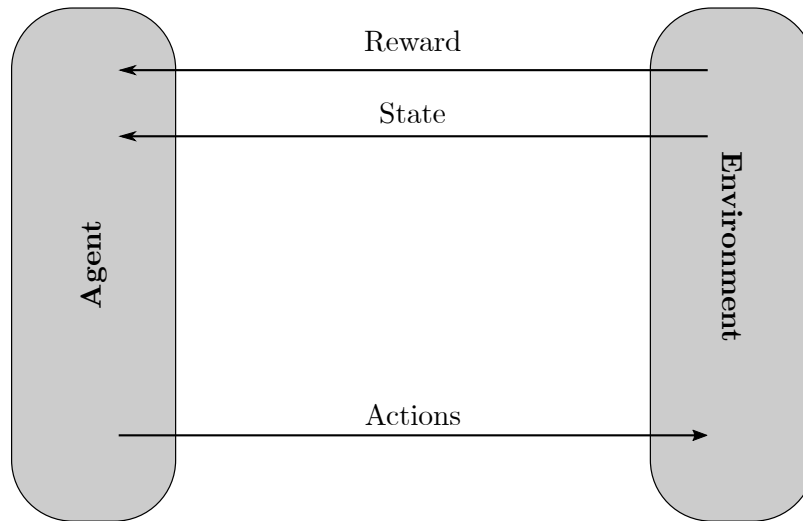


FIGURE 2.2: An agent uses its sensors to perceive the environment and uses its actuators to perform actions [75, 99].

### 2.2.3 Unsupervised learning

Unsupervised learning is defined as a method of machine learning where the model detects and forms patterns in the input data, however, no feedback is received on the detected patterns. Typically, no output value is associated with this paradigm of machine learning — the main aim is to discover insight into the data under consideration [76]. A popular method of unsupervised learning is clustering, which forms the basis of the work conducted in this thesis.

## 2.3 Clustering

The main aim when performing clustering is to identify (or uncover) the inherent structure and grouping of a data set based on the similarity of the data's features [44, 71]. Clustering has several application areas such as multimedia, social networks and biology [1]. Given the nature of the problem considered in this thesis and the corresponding data available, *i.e.* the input data only comprise latitude and longitude coordinates — no further information regarding the best suited waiting areas is provided, clustering is a suitable method. The notion of performing clustering on such data and using the obtained insight to identify suitable waiting areas warrant investigation — especially so considering the novelty of the approach. Upon a review of various sources in the literature pertaining to clustering algorithms it is evident that the most popular clustering methods include *partitional clustering*, *hierarchical clustering*, *density-based clustering* and *grid-based clustering* [1, 6, 30, 39]. Grid-based clustering is, however, not addressed in this project which can be ascribed to the fact that this class of clustering is more suited to multi-dimensional data sets (*i.e.* comprising many dimensions) — the data set considered in this project comprises only two dimensions, *i.e.* latitude and longitude. The following text describes these algorithms in more detail as well as provides insight into the distance measures employed in the operation of these algorithms.

### 2.3.1 Distance measures

Distance measures are calculated to find the distance between two data points or the distance between a data point and the cluster centre. The selection of distance measures is an essential

component of most clustering techniques in order to determine the similarity between the data points. Popular distance measures include *Manhattan distance*, *Euclidean distance* and *Chebyshev distance*, which are forms of the Minkowski metric [89]. The Minkowski metric can be regarded as the generalisation of the normal distance between two  $l$ -dimensional data points  $\mathbf{x} = (x_1, \dots, x_l)$  and  $\mathbf{y} = (y_1, \dots, y_l)$  in Euclidean space, expressed as

$$D(\mathbf{x}, \mathbf{y}) = \left( \sum_{i=1}^l |x_i - y_i|^r \right)^{1/r},$$

where  $r = 1$  represents Manhattan distance and  $r = 2$  represents Euclidean distance. Consider the Cartesian plane (comprising the  $x$  and  $y$  axes) depicted in Figure 2.3, if the distance between the two data points, *i.e.*  $A$  and  $B$ , is desired, the Euclidean distance corresponds to the straight-line distance between the data points, *i.e.* the hypotenuse  $c$  of the right triangle, whereas the Manhattan distance is the distance  $a$  plus the distance  $b$ . The Manhattan distance represents the manner according to which one would traverse around buildings in a city [10]. The Euclidean distance, on the other hand, is the most common distance metric employed towards calculating the distance between two data points [53]. Furthermore, the Chebyshev distance can be described as the value that the Minkowski metric tends to as  $r$  approaches infinity [89]. Accordingly, this provides the maximum distance along any dimensions, for example, in Figure 2.3, the Chebyshev distance is defined as  $\max\{a, b\}$  which would be distance  $b$ .

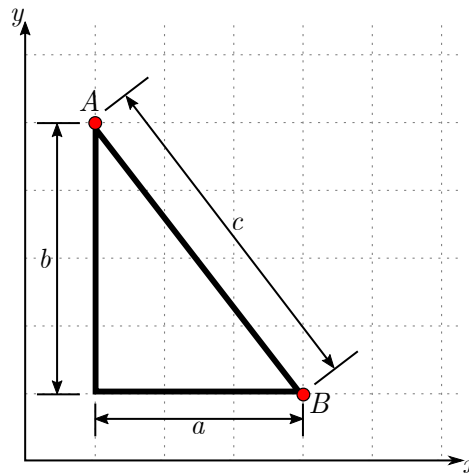


FIGURE 2.3: A Cartesian plane for illustrating the different distance measures.

### 2.3.2 Partitional clustering

Partitional clustering algorithms divide data into clusters by defining some function (corresponding to the clusters) and iteratively improving the quality of the clusters [1]. The main advantage of partitional clustering is its computationally efficient nature. These algorithms can, however, be sensitive to outliers and can exhibit poor convergence to global optima. This often depends on the initial centroids chosen. To avoid this undesired effect, many repetitions of the algorithm can be performed each with different initial centroids so as to determine the global optimum. The results are also sensitive to the number of clusters chosen, both of which are specified before the execution of the algorithm. Furthermore, these algorithms are unable non-spherical clusters [1, 39, 43].

A popular partitioning clustering algorithm is the  $K$ -means algorithm, first proposed by Macqueen in 1967 [59]. The  $K$ -means algorithm is a method of grouping unlabelled data into clusters by using each clusters' centre or centroids. The algorithm begins with selecting the desired number of centroids  $K$  (which is fixed throughout the algorithm) and selecting  $K$  points as centroids. Each data point is then allocated to the nearest centroid according to the chosen distance measure. After each data point has been assigned to a cluster, the centroids' positions are calculated as the mean value of each cluster. The data points are then re-assigned to the nearest centroid and the centroids are subsequently recalculated. This is iterated until the centroids remain the same or until some other convergence criterion is met. A proof of the convergence of the  $K$ -means algorithm can be found in [86]. Each iteration of the  $K$ -means algorithm has a time complexity of  $O(knt)$  where  $k$  is the number of clusters,  $n$  is the number of data points and  $t$  is the time elapsed when calculating the distance between two data points. The pseudocode description of a basic version of the  $K$ -means algorithm is presented in Algorithm 2.1.

---

**Algorithm 2.1:** Basic  $K$ -Means clustering algorithm [1]

---

```

1 Select  $K$  ;
2 Initialise  $K$  points as centroids;
3 repeat
4   | Allocate each data point to its nearest centroid to form  $K$  clusters;
5   | Change the centroids to be the mean of each cluster;
6 until the convergence criterion has been met ;

```

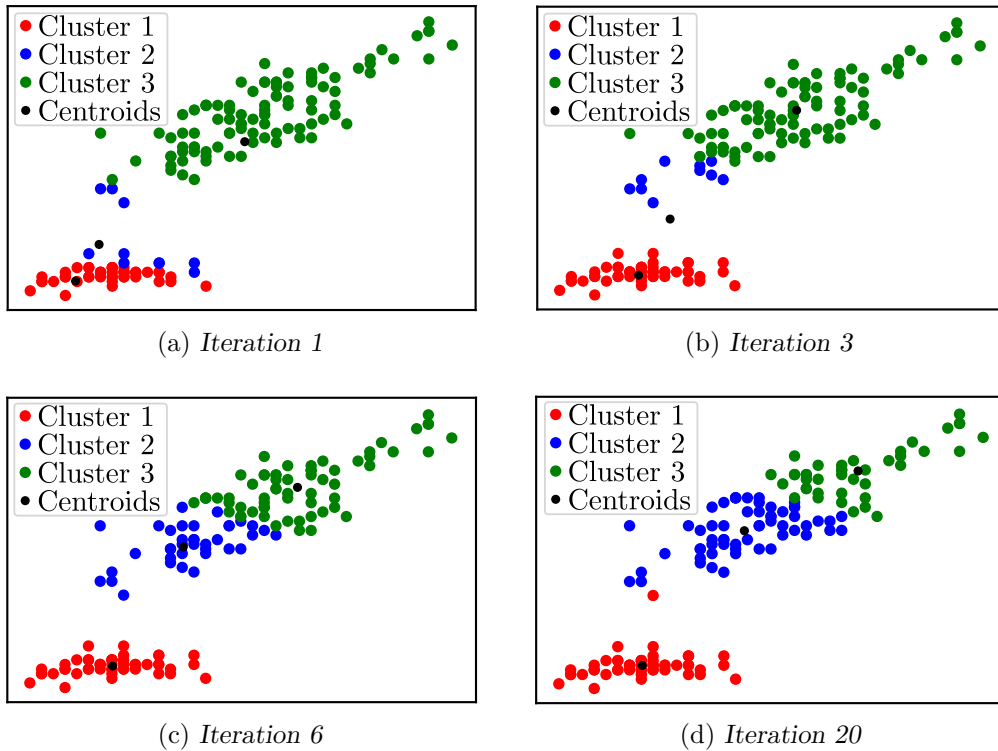
---

Several iterations of the  $K$ -means algorithm is shown in Figure 2.4. Here the  $K$ -means algorithm was performed in respect of the Iris data set using the sepal length and petal length attributes which can be found at [103]. Furthermore, a sample of the Iris data set is shown in Table 2.1. In this table,  $q$  indicates the data point number (or index), whereas sepal length, sepal width, petal length and petal width are the four attributes and the class indicates the type of Iris plant. In the first iteration, shown in Figure 2.4(a), the centroids are selected as random points. Figures 2.4(b) and 2.4(c) illustrate the movement of the centroids after being assigned to the mean value of the respective clusters together with the reassignment of data points to the centroids, forming new clusters. The final clustering is illustrated in Figure 2.4(d) after twenty iterations.

TABLE 2.1: A sample of the Iris data set in its original format.

$q$	Sepal length	Sepal width	Petal length	Petal width	Class
1	5.1	3.5	1.4	0.2	1
2	4.9	3.0	1.4	0.2	0
3	4.7	3.2	1.3	0.2	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
148	4.6	3.1	1.5	0.2	1
149	5.0	3.6	1.4	0.2	1
150	5.4	3.9	1.7	0.4	0

The objective of the  $K$ -means algorithm is to minimise an error function (*i.e.* *objective function*). The most popular error function employed is the *sum of squared errors* (SSE) also known as the *within-cluster variation* [1, 39, 43]. The minimisation of the SSE results in the value of the centroids equating to the mean value of the data points in the cluster [62]. Within the given text, the formulation of SSE is now elucidated. Let  $k$  denote the number of clusters,  $C_x$  denote cluster  $x$ , where  $x = 1, 2, \dots, k$ , and  $m_{C_x}$  denote the centroid of cluster  $C_x$ . For cluster  $C_x$ , the SSE can be defined as the summation of the squared distances of the data points,  $\mathbf{x} = (x_i, \dots, x_n)$

FIGURE 2.4: The  $K$ -means algorithm demonstrated in respect of the Iris data set.

in the cluster  $C_x$  to the cluster centroid,  $m_{C_x}$ . This can be expressed mathematically as

$$SSE(C_x, m_{C_x}) = \sum_{i=1}^n D(x_i, m_{C_x})^2,$$

where  $D(\cdot)$  denotes the distance measure chosen. Therefore, the summation of the SSE for all the clusters,  $\mathbf{C} = (C_1, \dots, C_k)$ , with cluster centres  $\mathbf{m} = (m_{C_1}, \dots, m_{C_k})$  can be given by

$$SSE(\mathbf{C}, \mathbf{m}) = \sum_{x=1}^k SSE(C_x, m_{C_x}) = \sum_{x=1}^k \sum_{i=1}^n D(x_i, m_{C_x})^2.$$

The  $K$ -means algorithm consists of two stages, the initialisation stage and the iteration stage. The initialisation stage involves the selection of the initial centroids,  $m_{C_1}, \dots, m_{C_k}$ , and the number of clusters  $k$ . The initialisation of  $K$ -means is essential in the performance of the algorithm with regards to the minimisation of the error function. The initialization of centroids can be performed using several methods, the two popular methods are discussed further. The first method involves the random selection of data points in the data set as centroids, according to a uniform distribution. This is the simplest method and is widely used in literature [1]. This method, however, may not converge to the global optima. [12]. The other popular method, known as the  $K$ -means++ algorithm [3], involves the selection of one data point at random from a data set,  $\mathbf{u} = (u_1, \dots, u_n)$ , according to a uniform distribution, as the first initial centroid. The next centroid is chosen with the probability

$$p = \frac{D_c(u_i)^2}{\sum_{i=1}^n D_c(u_i)^2},$$

where  $D_c(u_i)$  is the distance from the data point  $u_i$  to the nearest centroid. This is done repeatedly until the desired number of initial centroids has been chosen which is followed by the execution of the standard  $K$ -means algorithm. This algorithm generally converges faster than selecting random initial centroids and it achieves results that almost always converge towards the global optima[3].

There are three popular methods to assist with the selection of the number of clusters  $k$ , namely: The *elbow method*, *silhouette coefficient* and the *gap statistic* [47]. The elbow method and silhouette coefficient are direct methods whereas the gap statistic method is a statistical testing method. The elbow method [49] is the most established method for determining the number of clusters in a data set. This method is based on the fact that the objective function of the  $K$ -means algorithm decreases as the value for  $k$  increases. Therefore, if an increase in the number of clusters does not significantly decrease the objective function, then the elbow point has been reached. For some data sets, however, the elbow point can not always be clearly identified [47, 51]. This procedure can be regarded as a visual method — the elbow point can be identified when the objective function is plotted against the number of clusters as shown in Figure 2.5. The elbow method is applied to the Iris data set visualised in Table 2.4, where the elbow point can be located at three clusters. After this point the line of the graph begins to flatten.

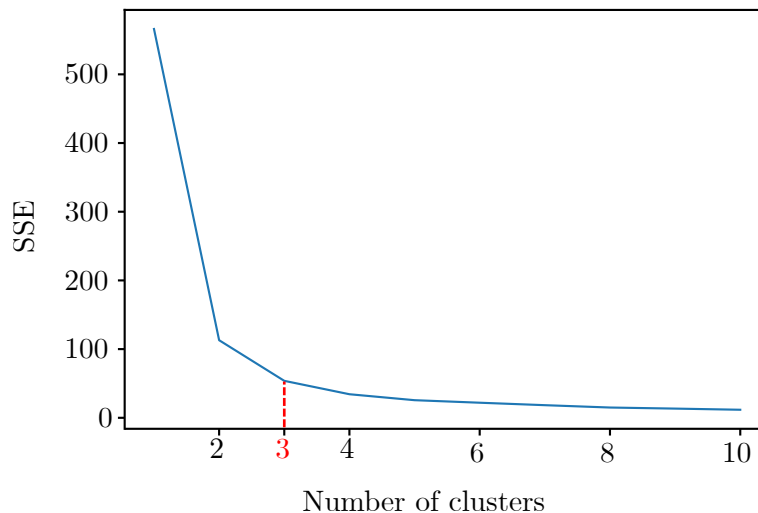


FIGURE 2.5: The elbow method applied to the Iris data set to find the number of clusters.

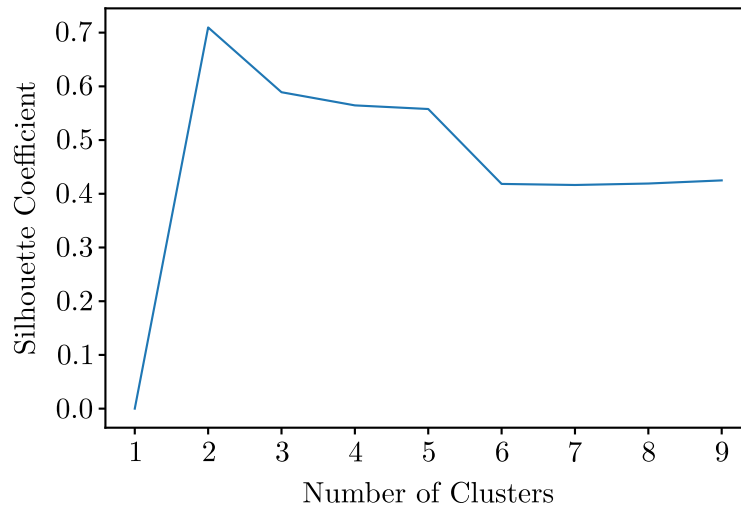
The *silhouette coefficient* ( $SC$ ) [74] can also be used to determine the number of clusters. This method returns a value between  $-1$  and  $1$  and indicates the extent to which a certain data point belongs to a cluster by using both the inter-cluster and intra-cluster distances. Given a cluster  $C_x$  containing data points  $x_1, \dots, x_n$ , for a data point  $x_i$ , the average distance between  $x_i$  and all the other data points in cluster  $C_x$  is calculated, denoted by  $a_i$ . The average of the distances between  $x_i$  and all the data points not in cluster  $C_x$  is also calculated. Therefore, let  $b_i$  denote the average distance of the cluster with the shortest average distance from  $x_i$ , *i.e.* the nearest cluster. For example, if there exist three clusters,  $C_x$ ,  $C_y$  and  $C_z$  and  $x_i$  belongs to cluster  $C_x$ , the average distances between  $x_i$  and all the data points in  $C_y$  is calculated and the same is done for the distances between  $x_i$  and all the data points in  $C_z$  (not  $C_x$  as  $x_i$  belongs to cluster  $C_x$ ). These two averages are then compared and the cluster with the shortest average distance from  $x_i$  is then the nearest cluster. The  $SC$ , of which the interpretation is summarised in Table 2.2, is given by

$$SC = \frac{\sum_{i=1}^n \frac{b_i - a_i}{\max(a_i, b_i)}}{n},$$

TABLE 2.2: *The interpretation of the SC [66, 96].*

Silhouette Coefficient	Nature of structure
0.71–1	Strong
0.51–0.7	Reasonable
0.26–0.5	Weak and may be artificial
$\leq 0.25$	None

The silhouette method was applied to the Iris data set for various number of clusters. According to this method two clusters are recommended, as illustrated in Figure 2.4.

FIGURE 2.6: *The silhouette method applied to the Iris data set to determine the number of clusters.*

The gap statistic method [100] was designed as a statistical procedure to establish the elbow method. The operation of the gap statistic method is as follows. The objective function of the  $K$ -means algorithm is calculated for  $k$  clusters, similarly to the elbow method. Let  $W_K$  be the calculated SSE for  $k$  clusters.  $B$  data sets are generated using a random uniform distribution with the same range as the original data set. Let  $W_{b,k}^*$  then equal the objective function calculated for the generated data set  $b$ , where  $b = 1, \dots, B$  for  $k$  clusters. The gap  $G(k)$  can then be expressed as

$$G(k) = \frac{1}{B} \sum_{b=1}^B \log(W_{b,k}^*) - \log(W_k).$$

The number of clusters may be chosen as the smallest  $k$  such that

$$G(k) \geq G(k+1) - \sigma_{k+1},$$

where  $\sigma_{k+1}$  is the estimated standard deviation of  $\log(W_{b,k+1}^*)$ . The gap statistic was applied to the Iris data set. According to this method three clusters are recommended which is illustrated graphically in Figure 2.7.

The iteration stage of the  $K$ -means method algorithm can be defined as the assignment of data points to the centroids based on the distance from the centroids. The algorithm proceeds to recalculate the centroids to correspond to the mean of the data points in the cluster. This stage is performed repeatedly until convergence is met — hence why it is referred to as the iteration stage.

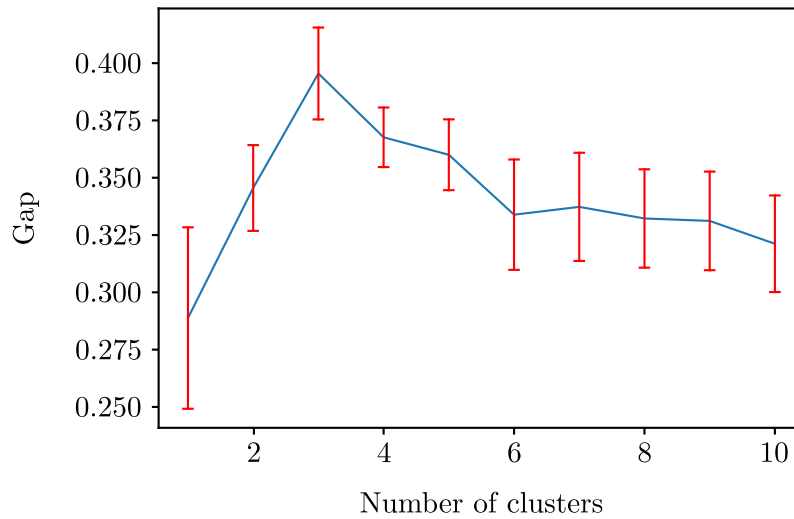


FIGURE 2.7: The gap statistic method applied to the Iris data set to determine the number of clusters.

There are a number of algorithms that build on the  $K$ -means algorithm. The first of which is *partitioning around medoids* (PAM) [48] where medoids are identified. Medoids are similar to centroids except medoids are always selected from the data set. Furthermore,  $K$ -modes [42] calculates the mode of the clusters instead of the mean. Lastly, *clustering for large applications* (CLARA) [48] selects random samples from the original data set and applies PAM to these subsets. The medoids identified from each subset are then applied to the original data set individually and the SSE is calculated. The medoids which result in the smallest SSE are then selected.

### 2.3.3 Hierarchical clustering

In the hierarchical clustering algorithm, the number of clusters is not required to be preselected, unlike in  $K$ -means [43]. Hierarchical clustering algorithms provide a nested series of partitions which consist of a large cluster at the top including all data points and small clusters containing individual data points at the bottom [93]. This is provided in the form a dendrogram, which is a tree diagram where the clusters are repeatedly split into smaller subsets. The number of clusters is determined by observing the location at which the vertical lines on the dendrogram are the longest and “cutting” the dendrogram at this point by drawing a horizontal line. The number of cluster may then be defined as the number of vertical lines this horizontal line crosses [28]. The hierarchical algorithm was performed on the Iris data set using the sepal length and the petal length, which produces the dendrogram shown in Figure 2.8. The resulting clustering is then shown in Figure 2.9. The dendrogram can be formed by either using a “bottom-up” approach, known as agglomerative clustering, or by using a “top-down” approach, known as a divisive clustering [38, 43].

Advantages pertaining to hierarchical clustering include the algorithm’s ability to manage a wide range of data granularity together with the ability to accommodate several distance measures. The dendrogram produced by the hierarchical algorithm provides supplementary information related to the clusters and is reasonably easy to read and interpret [6, 78]. The hierarchical algorithm may also be used to cluster a data set with a non-spherical structure [107]. The disadvantages associated with hierarchical clustering include the fact that clusters are not revisited once they have been made, therefore once a data point is assigned to a cluster, it cannot be undone and clusters cannot be improved (if improvement is possible) [6, 30, 78].



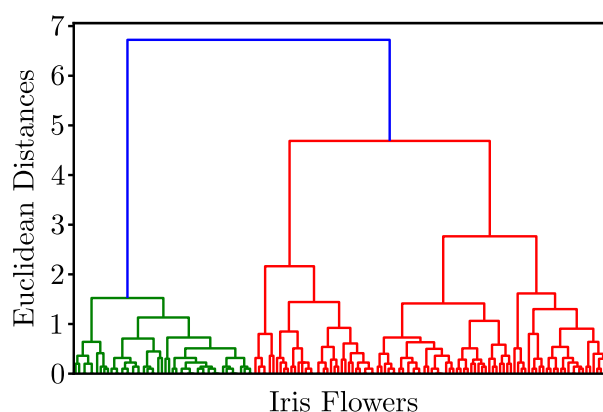


FIGURE 2.8: The dendrogram created when the hierarchical algorithm is performed on the Iris data set.

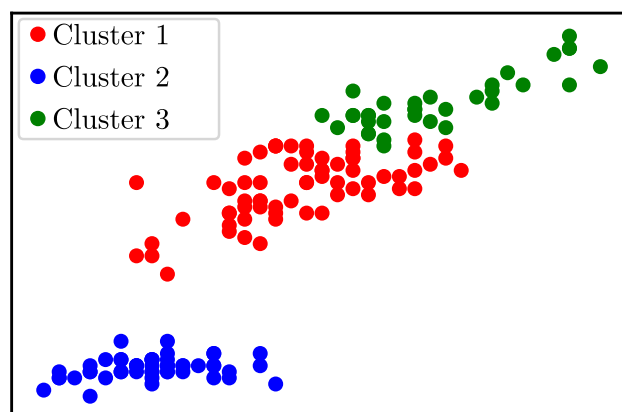


FIGURE 2.9: The clusters formed when the hierarchical clustering algorithm is performed on the Iris data set.

Agglomerative clustering commences with each data point becoming a cluster, therefore the number of clusters is equal to the number of data points. Thereafter, the closest clusters are merged. This is repeated until only a single cluster remains. Agglomerative clustering is the most common hierarchical clustering technique with a time complexity of  $O(n^3)$ , where  $n$  is the number of data points being clustered [93]. The pseudocode description of a basic version of the agglomerative clustering algorithm is presented in Algorithm 2.2.

---

**Algorithm 2.2:** Basic agglomerative clustering algorithm [1]

---

- 1 Let each data point be a cluster;
  - 2 **repeat**
  - 3     Compute distance between clusters;
  - 4     Merge the closest clusters;
  - 5 **until** a single cluster is left ;
- 

The merging of these clusters can be accomplished using several proximity methods. According to Murtagh [65] and Olson [69], these methods can be categorised into two groups, namely: Graph methods where the inter-cluster distances are calculated using the distance between the data points in the clusters and geometric distance methods, where the distances between

clusters are determined by calculating the distance between the cluster centres. Popular graph methods include single-link, complete-link and average-link method and popular geometric distance methods include centroid, median and Ward's methods [1, 69]. These methods are summarised in Figure 2.10.

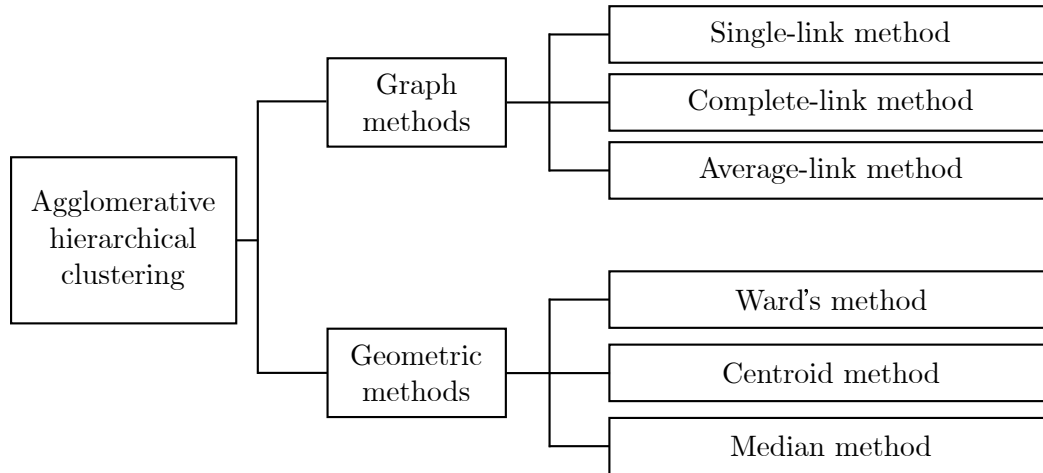


FIGURE 2.10: The proximity measures used in agglomerative hierarchical clustering [30].

In 1967, a formula was proposed by Lance and Williams [54] that provides the distance between clusters  $C_x$  and  $C_u$  where  $C_u$  is a union of clusters  $C_y$  and  $C_z$ , i.e.  $C_y \cup C_z$ . This may be expressed as

$$D(C_x, C_u) = \alpha_y D(C_x, C_y) + \alpha_z D(C_x, C_z) + \beta D(C_y, C_z) + \gamma |D(C_x, C_y) - D(C_x, C_z)|,$$

where  $D(\cdot)$  is the distance between two clusters whereas parameters  $\alpha_y$ ,  $\alpha_z$ ,  $\beta$  and  $\gamma$  are chosen according to the proximity measure specified. The parameters used in the Lance-Williams formula for each of the proximity measures are shown in Table 2.3.

TABLE 2.3: Values for parameters used in the Lance-William formula for the different proximity measures where  $n_x$ ,  $n_y$  and  $n_z$  are the number of data points in cluster  $C_x$ ,  $C_y$  and  $C_z$ , respectively [54].

Method	$\alpha_y$	$\alpha_z$	$\beta$	$\gamma$
Single-link	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete-link	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
Average-link	$\frac{n_y}{n_y+n_z}$	$\frac{n_z}{n_y+n_z}$	0	0
Ward's Method	$\frac{n_y+n_z}{n_x+n_y+n_z}$	$\frac{n_x+n_z}{n_x+n_y+n_z}$	$-\frac{n_x}{n_x+n_y+n_z}$	0
Centroid	$\frac{n_y}{n_y+n_z}$	$\frac{n_z}{n_y+n_z}$	$-\frac{n_y n_z}{(n_y+n_z)^2}$	0
Median	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0

The single-link method is one of the simplest proximity methods which was first introduced by Florek *et al.* in 1951 [27] and then again in 1957 by McQuitty [61] and Sneath [91], independently. According to the single-link method, the distance between any two clusters is the minimum distance between any pair of data points such that one point belongs to each cluster. According

to the Lance-William method, the distance between  $C_x$  and  $C_u$  may be defined as

$$\begin{aligned} D(C_x, C_u) &= \frac{1}{2}D(C_x, C_y) + \frac{1}{2}D(C_x, C_z) - \frac{1}{2}|D(C_x, C_y) - D(C_x, C_z)| \\ &= \min \{D(C_x, C_y), D(C_x, C_z)\}. \end{aligned}$$

This may also be written as

$$D(C_x, C_u) = \min_{i \in C_x, j \in C_u} d(i, j),$$

where  $d(i, j)$  is the distance between data point  $i$  and  $j$ . An illustration of this method is shown in Figure 2.11, where the data points marked with a cross  $\times$  belong to cluster  $C_x$  and the data points marked with a circle  $\bullet$  belong to cluster  $C_u$ .

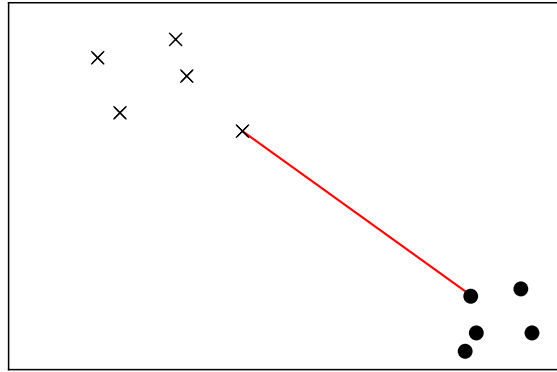


FIGURE 2.11: *Demonstration of the single-link method.*

With respect to the complete-link method [45], the distance between two clusters is defined as the largest distance that exists between any pair of data points such that one data point belongs to each cluster. According to the Lance-William method, the distance between clusters  $C_x$  and  $C_u$  may be expressed as

$$\begin{aligned} D(C_x, C_u) &= \frac{1}{2}D(C_x, C_y) + \frac{1}{2}D(C_x, C_z) + \frac{1}{2}|D(C_x, C_y) - D(C_x, C_z)| \\ &= \max \{D(C_x, C_y), D(C_x, C_z)\}, \end{aligned}$$

which may also be defined as

$$D(C_x, C_u) = \max_{i \in C_x, j \in C_u} d(i, j).$$

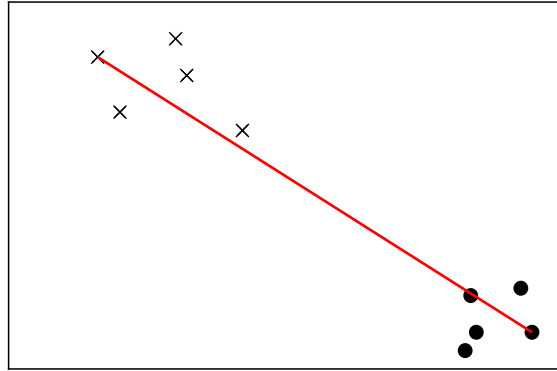
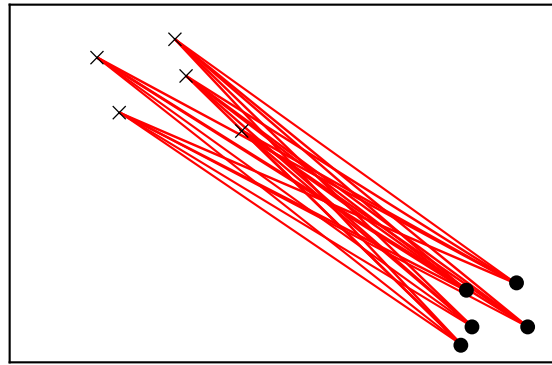
The complete-link method is illustrated in Figure 2.12, where the data points are indicated as previously stated (*i.e.* data points marked with an  $\times$  belong to cluster  $C_x$  and the data points marked with a  $\bullet$  belong to cluster  $C_u$ ).

The average-link method [45], illustrated in Figure 2.13, calculates the average distance of all the possible pairs of data points between two clusters in which one data point is from each cluster. The distance between clusters  $C_x$  and  $C_u$  may be expressed using the Lance-William method as follows

$$D(C_x, C_u) = \frac{n_y}{n_y + n_z}D(C_x, C_y) + \frac{n_z}{n_y + n_z}D(C_x, C_z).$$

Therefore, if  $n_x$  data points are present in cluster  $C_x$  and  $n_u$  data points are present in cluster  $C_u$  then the distance between the two clusters may be expressed as

$$D(C_x, C_u) = \frac{1}{n_x n_u} \sum_{i=1}^{n_x} \sum_{j=1}^{n_u} d(i, j).$$

FIGURE 2.12: *Demonstration of the complete-link method.*FIGURE 2.13: *Demonstration of the average link method.*

Ward's method, introduced by Ward Jr. [105], defines the distance between two clusters as by the increase in the SSE upon merging the clusters. If data point  $x_i$  is contained in the merged cluster  $C_x \cup C_u$ , the change in the SSE,  $\Delta(C_x, C_u)$ , can be expressed as

$$\Delta(C_x, C_u) = \sum_{i \in C_x \cup C_u} d(x_i, m_{C_x \cup C_u})^2 - \sum_{i \in C_x} d(x_i, m_{C_x})^2 - \sum_{i \in C_u} d(x_i, m_{C_u})^2,$$

where  $m_{C_x}$  and  $m_{C_u}$  are the centres of clusters  $C_x$  and  $C_u$ , respectively. This is known as the cost associated with merging clusters  $C_x$  and  $C_u$ , *i.e.* the merging cost. In agglomerative hierarchical clustering, the SSE is initially zero as each data point is a cluster. The SSE then increases after every merging procedure. Ward's method attempts to minimise this growth by merging clusters with the smallest merging cost [19, 83]. The increase in the SSE when clusters are merged is illustrated in Figure 2.14. Figure 2.14(a) and Figure 2.14(b) are graphical representations of the SSE when the clusters are separated and merged, respectively.

Once again, the distance between clusters  $C_x$  and  $C_u$  can be defined, using the Lance-William method, as

$$D(C_x, C_u) = \frac{n_x + n_y}{n_x + n_y + n_z} D(C_x, C_y) + \frac{n_x + n_z}{n_x + n_y + n_z} D(C_x, C_z) - \frac{n_x}{n_x + n_y + n_z} D(C_y, C_z).$$

The centroid method [1, 106], illustrated in Figure 2.15, uses the distance between the centre of clusters where the centre is calculated as the mean of all the data points in that cluster. According to the the Lance-William formula, the distance between clusters  $C_x$  and  $C_u$  may be

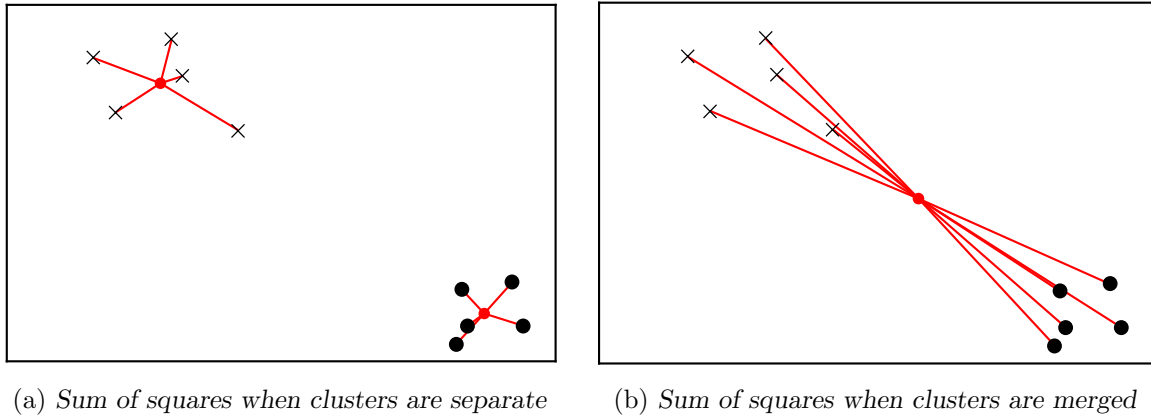


FIGURE 2.14: Demonstration of Ward's method.

expressed as

$$D(C_x, C_u) = \frac{n_y}{n_y + n_z} D(C_x, C_y) + \frac{n_z}{n_y + n_z} D(C_x, C_z) - \frac{n_y n_z}{(n_y + n_z)^2} D(C_y, C_z).$$

Therefore, the distance between the two clusters  $C_x$  and  $C_u$  may also be written as

$$D(C_x, C_u) = d(m_{C_x}, m_{C_u}).$$

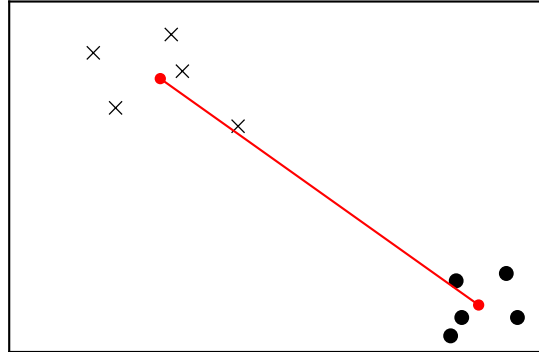


FIGURE 2.15: Demonstration of the centroid method.

Similarly, the median method [69] also uses the distance between the cluster centres, however, the cluster centres are equate to the average of the cluster centres belonging to the merged clusters. For example, if cluster  $C_y$ , with cluster centre  $m_{C_y}$ , and cluster  $C_z$ , with cluster centre  $m_{C_z}$ , were merged to make  $C_u$ , the cluster centre of  $C_u$  would therefore be

$$m_{C_u} = \frac{m_{C_y} + m_{C_z}}{2}.$$

The distance between cluster  $C_u$  and cluster  $C_x$  is then given by

$$D(C_x, C_u) = d(m_{C_x}, m_{C_u}).$$

According to the Lance-William formula, the centroid method defines the distance between clusters  $C_x$  and  $C_u$  as

$$D(C_x, C_u) = \frac{1}{2} D(C_x, C_y) + \frac{1}{4} D(C_x, C_z) - \frac{1}{4} D(C_y, C_z).$$

The median method is illustrated in Figure 2.16 where the data points shown with a cross  $\times$  form cluster  $C_y$  and those shown with a triangle  $\blacktriangle$  form cluster  $C_z$ . These two clusters were merged to form  $C_u$  which is circled in green and the centre is shown with a green dot. Lastly, cluster  $C_x$  is circled in blue and the centre is indicated with a blue dot.

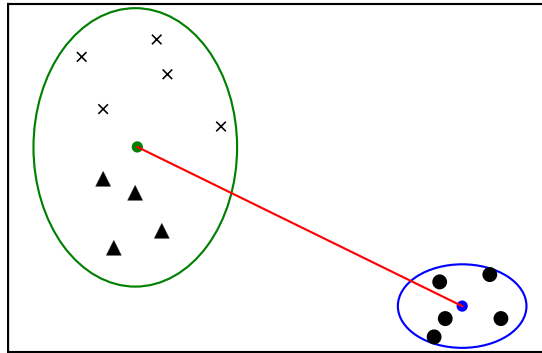


FIGURE 2.16: *Demonstration of the median method.*

Several algorithms exist which build on the basic agglomerative hierarchical clustering algorithm, namely: *Single-link hierarchical algorithm* (SLINK) [87], *complete-link hierarchical algorithm* (CLINK) [20] (both of which were developed in an attempt to lower the time complexity of the algorithm) *balanced iterative reducing and clustering using hierarchies* (BIRCH) [111] and *clustering using representatives* (CURE) [35].

Divisive clustering is initiated with one large cluster containing all the data points, unlike agglomerative hierarchical clustering algorithm. After each iteration, the cluster is split until each cluster only contains one data point [93]. For moderate- to large-sized data sets it is not practically viable to evaluate all possible ways to split the cluster — there are  $2^{n_i-1}$  possible ways to split a cluster comprising  $n_i$  number of data points. Therefore, divisive clustering is seldom used ascribed to its computationally expensive nature [30, 106]. If an exhaustive search is performed of all the possible splits, this algorithm has a time complexity of  $O(2^n)$ , however heuristics are often employed in conjunction with the algorithm in an attempt to reduce its complexity. Despite the time complexity, the divisive algorithm has been used in by [9] for the partitioning of documents by assembling these documents into a sparse matrix so as to increase the efficiency of the algorithm. An algorithm known as divisive analysis (DIANA) [48] builds on the basic divisive hierarchical clustering algorithm. The pseudocode description of the divisive clustering algorithm is presented in Algorithm 2.3.

---

**Algorithm 2.3:** Basic divisive clustering algorithm [1].

---

- 1 *Create a single cluster containing all the data points;*
  - 2 **repeat**
  - 3     For all the possible clusters, find the split procedure that creates two clusters with the largest dissimilarity;
  - 4     Split the cluster;
  - 5 **until** *Until each data point belongs to its own cluster ;*
-

### 2.3.4 Density-based clustering

Density-based clusters can be defined as dense regions which are separated by scattered regions [52]. This clustering algorithm identifies clusters that have a non-spherical shape — a common occurrence in the context of spacial data ascribed to geographic obstacles such as mountains and rivers. Density-based clustering can be applied in problems such as in a geo-marketing investigation with the aim of clustering homes so as to determine high, medium and low household income regions or in crime analysis in order to cluster different types of crimes or to find so-called crime hotspots [1]. Density-based algorithms, most of which have a time complexity of  $O(n \log(n))$ , create clusters by allowing a cluster to increase in size if the density of neighbouring clusters exceed a certain threshold [14]. A data set was generated to observe the ability of density-based algorithms to cluster data sets containing clusters of non-spherical shapes. This is illustrated in Figure 2.17(a) and Figure 2.17(b) where this generated data set is clustered using density-based clustering and  $K$ -means clustering, respectively. It is evident that density-based clustering results in clusters that are non-spherical and the data points which are closer together, *i.e.* more dense regions, are assigned to the same cluster, whereas  $K$ -means attempts to create spherical clusters and therefore incorrectly clusters the data. Popular density based clustering algorithms include *density-based spatial clustering of applications with noise* (DBSCAN) [25], *ordering points to identify clustering structure* (OPTICS) [2] and *density-based clustering* (DENCLUE) [40].

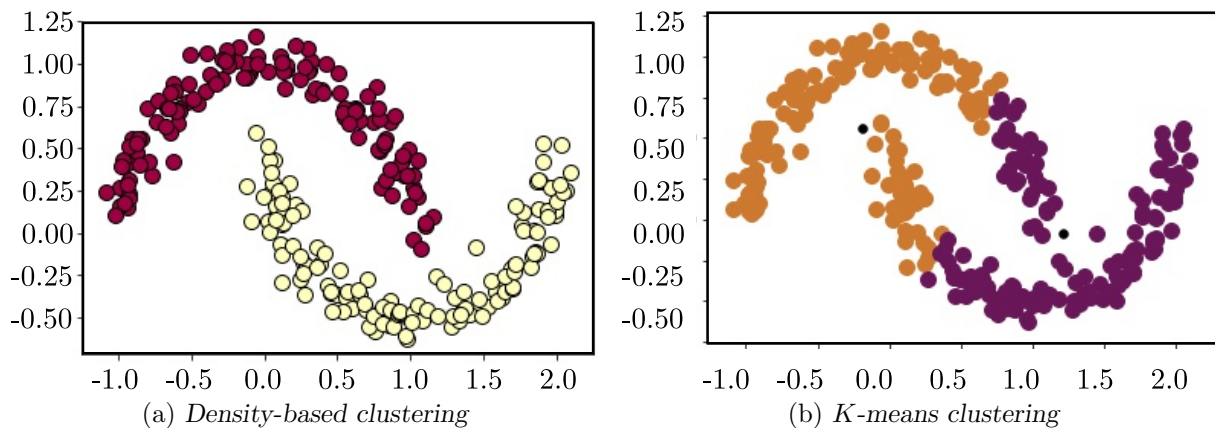


FIGURE 2.17: Demonstration of the density-based clustering algorithm's ability to cluster non-spherical data structures

The DBSCAN algorithm was proposed by Ester *et al.* [25] in 1996. The algorithm identifies clusters as regions of a high density, whereas low density regions are seen as noise or outliers. The algorithm is especially effective when applied to large data sets containing noise. Two input parameters are required in order to perform the DBSCAN algorithm, namely: Maximum neighbourhood radius  $\epsilon$ , which must be greater than zero, and MinPts, which denotes the minimum number of data points required. If data points  $p$  and  $q$  both belong to data set  $D$ , the neighbourhood of radius  $\epsilon$  of data point  $p$  is given by as

$$N_\epsilon(p) = \{q \in D : d(p, q) \leq \epsilon\}, \quad (2.1)$$

where  $d(\cdot)$  is a distance function. DBSCAN categorises data points into three different types, namely: Core points, boarder points and noise points. If a data point  $p$  has at least MinPts data points in its neighbourhood radius of  $\epsilon$  then data point  $p$  is considered to be a core point. A border point is a point which belongs to a cluster but whose neighbourhood is not dense, *i.e.* its

neighbourhood of radius  $\epsilon$  does not contain MinPts data points. Lastly, a noise point is defined as a data point that does not belong to a cluster.

Furthermore, DBSCAN distinguishes between the types of relationships that exist between data points, as shown in Figure 2.18. Point  $q$  is directly density-reachable from core point  $p$  if  $q$  is in the neighbourhood  $\epsilon$  of  $p$ , *i.e.*  $p \in N_\epsilon(p)$ , and if  $|N_\epsilon(p)| \geq \text{MinPts}$ , where  $|N_\epsilon(p)|$  indicates the number of data points in  $N_\epsilon(p)$ . Therefore, if data point  $q$  is within core point  $p$ 's neighbourhood radius of  $\epsilon$  then data point  $q$  is directly density-reachable from core point  $p$ , illustrated in Figure 2.18(a). If data point  $q$  is directly density-reachable from core point  $p$ , then core point  $p$  is only directly density-reachable from  $q$  if  $q$  is also a core point. Data point  $q$  is density-reachable from core point  $p$  if there exists a chain of data points  $\{x_1, \dots, x_n\}$  such that  $x_1 = p$ ,  $x_n = q$  and  $x_{i+1}$ , for  $i \in \{1, \dots, n-1\}$ , is directly density reachable from  $x_i$  with respect to the user input of  $\epsilon$  and MinPts. In Figure 2.18(b), data point  $p$  is density-reachable from data point  $q$  as there is a chain of data points (*i.e.*  $p$ ,  $r$  and  $q$ ) such that  $q$  is directly density-reachable from  $r$  and  $r$  is directly density-reachable from  $p$ . Two data points  $p$  and  $q$  are density-connected as there exists a data point  $r$  such that both data points  $p$  and  $q$  are density-reachable from  $r$  with respect to the user input of  $\epsilon$  and MinPts as illustrated is shown in Figure 2.18(c).

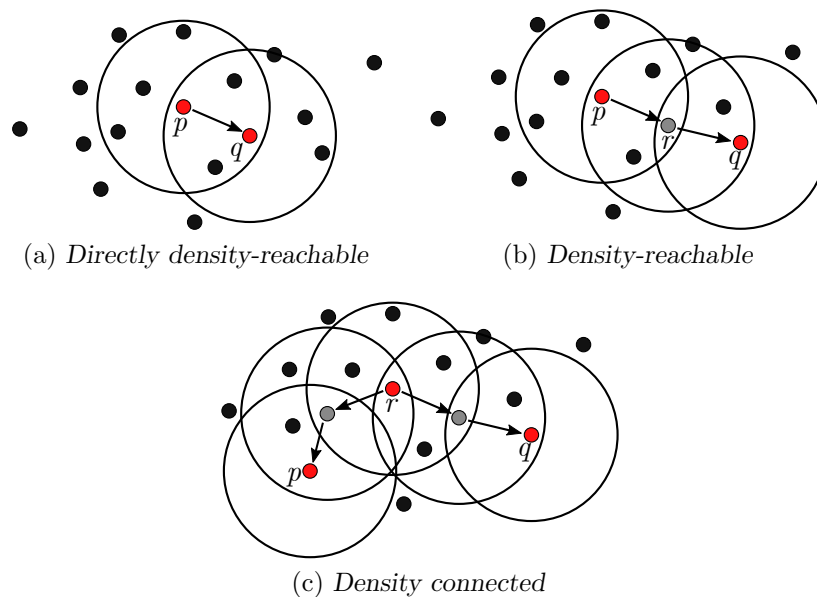


FIGURE 2.18: Relationships between data points distinguished by the DBSCAN algorithm [25].

Cluster  $C$  then formed in data set  $D$  based on two conditions. Firstly, if for all  $p$  and  $q$  in data set  $D$ , if  $p$  is assigned to cluster  $C$  and  $q$  is directly density-reachable from  $p$  then  $q$  is also in cluster  $C$ . Furthermore, if for all  $p$  and  $q$  in cluster  $C$ ,  $p$  is density connected to  $q$  with respect to the user input of  $\epsilon$  and MinPts.

The input values  $\epsilon$  and MinPts are fixed during the algorithm's operation. Therefore, the selection of these values is an important matter due to the sensitivity of the algorithm to these inputs. There is a heuristic which can be used to determine the value of  $\epsilon$ . The heuristic calculates the distance between all the points in the data set and their  $k^{\text{th}}$  nearest neighbours. These distances are then sorted in descending order and the result is plotted on a graph called the sorted  $k$ -dist graph, where the  $k$  is set to equal MinPts. For a two-dimensional data set it is safe to assume a value of 4 for MinPts as the  $k$ -dist graph does not change significantly for  $k > 4$ . The value for  $\epsilon$  may then be found by identifying the position where a sudden change in the gradient occurs in the  $k$ -dist plot, *i.e.* the bend. This data point is known as the *threshold point*. The data points



to the left of the threshold point are classified as noise and all data points to the right of the threshold are assigned to some cluster. Figure 2.19 is a  $k$ -dist plot where  $k=4$  and the horizontal dotted line indicates the value of  $\epsilon$  for a certain data set.

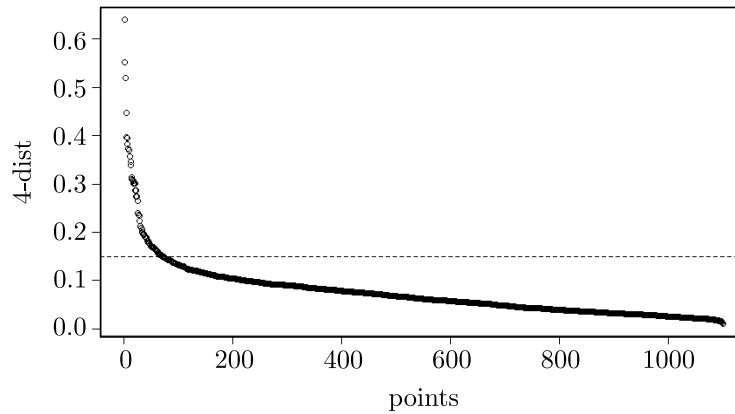


FIGURE 2.19: The  $k$ -dist graph used to determine  $\epsilon$ .

Due to the user input of  $\text{MinPts}$  and  $\epsilon$  being fixed, DBSCAN does not work effectively when a data set's clusters vary greatly. Furthermore, a considerable amount of memory is required when working with large data sets. Advantages pertaining to the DBSCAN algorithm include that the number of clusters is not required as an input, it is resistant to noise and it can detect clusters that are of non-spherical shape [15, 85, 90]. A pseudocode description of the DBSCAN algorithm is presented in Algorithm 2.4 [39].

---

**Algorithm 2.4:** DBSCAN clustering algorithm.

---

```

1 Select  $\epsilon$  and  $\text{MinPts}$ ;
2 Mark all data points as unvisited;
3 repeat
4   Select unvisited data point  $p$  randomly;
5   Mark data point  $p$  as visited;
6   if the  $\epsilon$  neighbourhood of data point  $p$  has at least  $\text{MinPts}$  data points then
7     Create a cluster  $C$  and add data point  $p$  to cluster  $C$ ;
8     Let  $N$  be all the data points in the  $\epsilon$  neighbourhood of data point  $p$ ;
9     for all data points  $q$  in  $N$  do
10      if data point  $q$  is unvisited then
11        Mark data point  $q$  as visited;
12        if the  $\epsilon$  neighbourhood of data point  $p$  has at least  $\text{MinPts}$  data points then
13          Add all these data points to  $N$ ;
14        if data point  $q$  does not belong to a cluster then
15          Add data point  $q$  to cluster  $C$ ;
16      Return  $C$ ;
17   else
18     Mark data point  $p$  as noise;
19 until no data point remains unvisited ;

```

---

OPTICS, proposed by Ankerst *et al.* [2] in 1999, is an extension of the DBSCAN algorithm. This algorithm is able to identify clusters in a data set with varying densities and, therefore, is not defined by fixed density parameters. When using fixed density parameters, such as in DBSCAN, to cluster the data set shown in Figure 2.20, depending on the user input of MinPts and  $\epsilon$ , the algorithm may either identify clusters A, B and C or, alternatively, identify clusters  $C_1$ ,  $C_2$  and  $C_3$ , whereas A and B would be classified as noise.

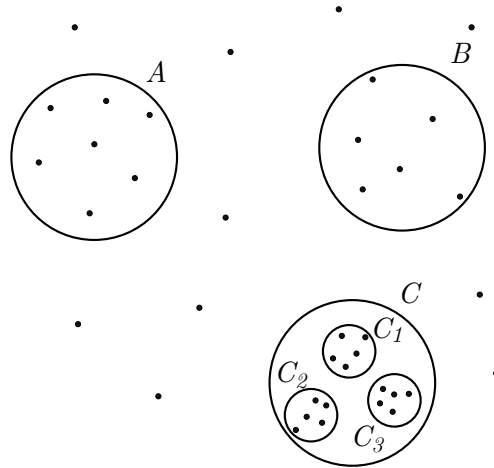


FIGURE 2.20: The clusters identified when using different fixed density parameters.

When the OPTICS algorithm is performed, one explicit data clustering outcome is not produced, instead the algorithm produces a cluster-ordering based on the density structure of the data points. This is built on the observation that for a constant value MinPts, a density based cluster  $C_1$  formed with a low value of  $\epsilon$  (*i.e.* small neighbourhood radius) will be a subset of cluster  $C_2$  which has a larger  $\epsilon$  and is therefore less dense. The DBSCAN algorithm can then be extended to cluster the same data set for different values of  $\epsilon$  simultaneously, however, this needs to be done in a certain order. The clusters that have the highest density will be completed first by evaluating the data points that are density-reachable according to the lowest  $\epsilon$ -value. The DBSCAN algorithm is used to evaluate  $\epsilon_i$ , where  $0 \leq \epsilon_i \leq \epsilon$  and  $\epsilon$  is the largest distance that may be considered when forming clusters, *i.e.* the generating distance.

The cluster-ordering output of OPTICS is a list containing order in which data points are evaluated. This list contains the core-distance and the reachability-distance for each data point. The core-distance of data point  $p$  can be defined as the smallest value  $\epsilon'$  such that the  $\epsilon'$ -neighbourhood of  $p$  has at least MinPts, *i.e.* the smallest distance  $\epsilon'$  such that  $p$  is a core point. If data point  $p$  is not a core point with respect to  $\epsilon$ , then the core-distance is undefined. The reachability-distance of data point  $p$  is the minimum distance  $\epsilon'$  from data point  $p$  to a core point  $q$  such that data point  $p$  is directly density-reachable from core point  $q$ . The reachability-distance cannot be smaller than the core-distance of data point  $p$  — if this occurs then no data point is directly density-reachable from core point  $q$  attributed to the fact that  $q$  will not be a core point. In this situation, the reachability-distance to data point  $p$  from core point  $q$  equates to the core-distance of  $q$ , however, if this distance is larger than  $\epsilon$  then the reachability-distance is set to undefined. The reachability-distance of data point  $p$  is dependent on the core point from which it is calculated. These two concepts are illustrated graphically in Figure 2.21. In Figure 2.21(a),  $\epsilon$  is the generating distance and  $\epsilon'$  is the core-distance for core point  $p$ . In Figure 2.21(b), the reachability-distance to data point  $q_1$  from core point  $p$  is the distance from data point  $p_1$  to core point  $p$ , whereas the reachability-distance to data point  $q_1$  from core point  $p$  is  $\epsilon'$  as the distance from  $q_1$  to  $p$  is greater than  $\epsilon'$ .

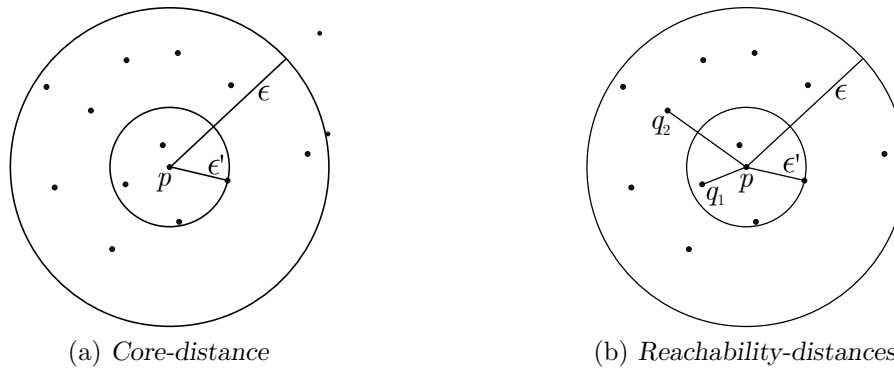
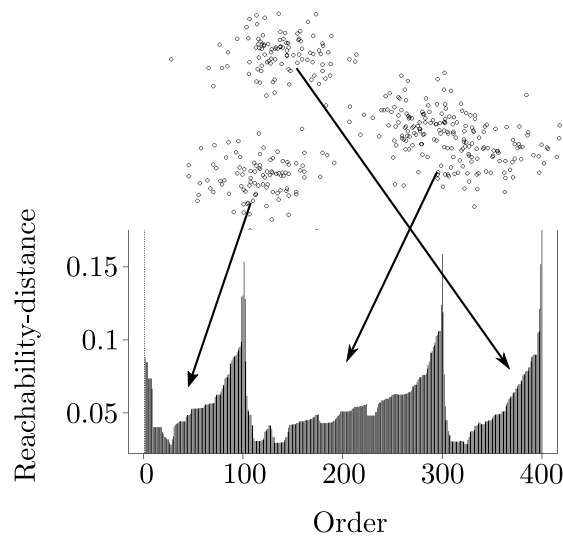
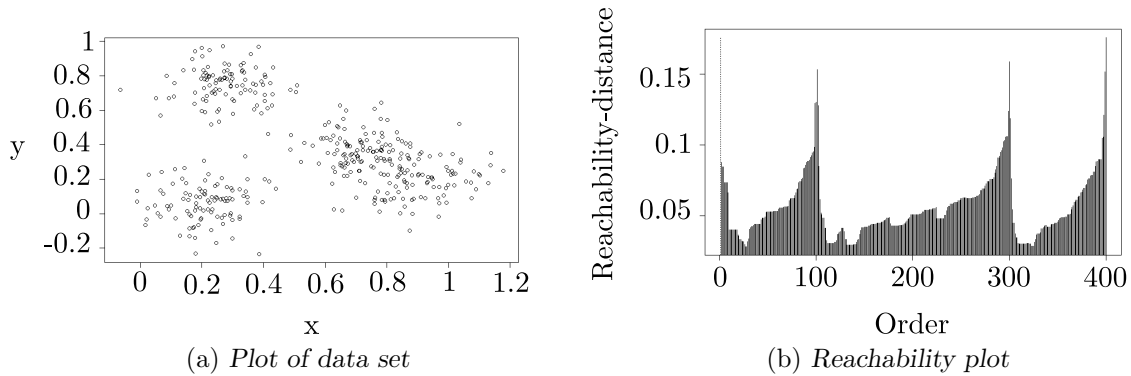


FIGURE 2.21: Core-distance and reachability-distance for the OPTICS algorithm.

The data sets ordering may be represented graphically in a reachability plot which assists in understand the clustering structure. In this plot, the reachability-distances of each data point is illustrated, as in Figure 2.22. A plot of a 2-dimensional data set is illustrated in Figure 2.22(a) and the corresponding reachability plot is shown in Figure 2.22(b) where the reachability-distance is plotted against the data points in their clustering order. The three “dips” in the plot indicate the three clusters as illustrated in Figure 2.22(c).



(c) Reachability plot with plot of data set

FIGURE 2.22: Reachability plot showing the data set ordering provided by the OPTICS algorithm.

DENCLUE, proposed by Hinneburg *et al.* [40] in 1998, is based on the idea that a data point can be modelled using a so-called influence function which is continuous, symmetric and differentiable and provides an indication of the impact of a data point on its neighbourhood. These influence functions can be expressed using various functions, such as the square wave function where the influence of data point  $y$  on data point  $x$  is given by

$$f_{square}^y(x) = f_{square}(x, y) = \begin{cases} 0, & \text{if } d(x, y) > \sigma, \\ 1, & \text{otherwise} \end{cases}$$

and the Gaussian function

$$f_{Gauss}^y(x) = f_{Gauss}(x, y) = e^{-\frac{d(x, y)^2}{2\sigma^2}},$$

where input parameter  $\sigma$  is a threshold parameter. The overall density function is then calculated as the sum of the influence function applied to all the data points. Therefore, the overall density function of data set  $D = x_1, \dots, x_N$  is given by

$$f^D(x) = \sum_{i=1}^N f^{x_i}(x).$$

Thereafter, clusters are formed by identifying density-attractors which are the local maxima of the overall density function and can be found using a hill-climbing procedure. Hill-climbing follows the gradient of the overall density function until the local maximum is found. The gradient of a density function  $f^D(x)$  can be expressed as

$$\Delta f^D(x) = \sum_{i=1}^N (x_i - x) f^{x_i}(x).$$

Hill-climbing may be performed using the method of steepest ascent [30]. In order to understand steepest ascent, an understanding of directional derivatives is required. A directional derivative [94] is an extension of a partial derivative but instead of moving in the direction of the axis movement is in the direction of a unit vector  $\vec{u}$ . The partial derivative of function  $f(x, y)$  with respect to  $x$  may be defined as

$$f_x = \frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h, y) - f(x, y)}{h},$$

and the partial derivative with respect to  $y$  may be defined as

$$f_y = \frac{\partial f}{\partial y} = \lim_{h \rightarrow 0} \frac{f(x, y+h) - f(x, y)}{h}.$$

These partial derivatives represent rates of change of  $f$  in the  $x$  and  $y$  direction, respectively. For a directional derivative, the change is made in the direction of a vector  $\vec{u} = \langle a, b \rangle$  given by

$$D_{\vec{u}}f(x, y) = \lim_{h \rightarrow 0} \frac{f(x+ha, y+hb) - f(x, y)}{h}.$$

Therefore, if  $\vec{u} = \langle 1, 0 \rangle$  then

$$\begin{aligned} D_{\vec{u}}f(x, y) &= \lim_{h \rightarrow 0} \frac{f(x+h(1), y-h(0)) - f(x, y)}{h}, \\ &= f_x \end{aligned}$$

and similarly, if  $\vec{u} = \langle 0, 1 \rangle$  then  $D_{\vec{u}}f(x, y) = f_y$ . If  $g(h) = f(x_0 + ha, y_0 + hb)$ , then

$$\begin{aligned} g'(0) &= \lim_{h \rightarrow 0} \frac{g(h) - g(0)}{h} \\ &= \lim_{h \rightarrow 0} \frac{f(x_0 + ha, y_0 + hb) - f(x_0, y_0)}{h} \\ &= D_{\vec{u}}f(x_0, y_0). \end{aligned}$$

It is also possible to write  $g(h) = f(x, y)$  where  $x = x_0 + h$  and  $y = y_0 + h$ . Using the chain rule  $g'(h)$  may then be defined as

$$\begin{aligned} g'(h) &= \frac{\partial f}{\partial x} \frac{dx}{dh} + \frac{\partial f}{\partial y} \frac{dy}{dh} \\ &= f_x(x, y)a + f_y(x, y)b \end{aligned}$$

If  $h = 0$ ,  $x = x_0$  and  $y = y_0$  then  $g'(0) = f_x(x_0, y_0)a + f_y(x_0, y_0)b$ . Therefore,  $D_{\vec{u}}f(x_0, y_0) = f_x(x_0, y_0)a + f_y(x_0, y_0)b$ . This can also be described as the dot product of two vectors expressed as

$$\begin{aligned} D_{\vec{u}}f(x_0, y_0) &= \langle f_x(x_0, y_0), f_y(x_0, y_0) \rangle \cdot \langle a, b \rangle \\ &= \langle f_x(x_0, y_0), f_y(x_0, y_0) \rangle \cdot \vec{u}. \end{aligned}$$

The vector  $\langle f_x(x_0, y_0), f_y(x_0, y_0) \rangle$  is the gradient of  $f$ , *i.e.*  $\Delta f$ . The directional derivative of  $f$  in the direction of unit vector  $\vec{u}$  can be written as  $D_{\vec{u}}f(x, y) = \Delta f(x, y) \cdot \vec{u}$ . Since the dot product between two vectors  $\vec{x}$  and  $\vec{y}$  can be defined as  $\vec{x} \cdot \vec{y} = \|\vec{x}\|\|\vec{y}\|\cos\theta$  where  $\theta$  is the angle between the two vectors,  $D_{\vec{u}}f(x, y)$  is given by

$$D_{\vec{u}}f(x, y) = \|\Delta f(x, y)\|\|\vec{u}\|\cos\theta,$$

however, since  $\vec{u}$  is a unit vector (having a magnitude of 1) this expression can be simplified to

$$D_{\vec{u}}f(x, y) = \|\Delta f(x, y)\|\cos\theta.$$

When using the method of steepest ascent, the largest directional derivative  $D_{\vec{u}}f(x, y)$  must be determined in order to find the greatest increase in  $f(x, y)$ . Since  $\cos\theta$  ranges between -1 and 1, the largest value of  $D_{\vec{u}}f(x, y)$  is obtained when  $\cos\theta = 1$  (*i.e.*  $\theta = 0$ ). Therefore, the direction of  $\vec{u}$  is the same as that of the gradient  $\Delta f(x, y)$ . As a result, the direction of unit vector  $\vec{u}$  is required to be defined such that  $\theta$  is almost zero. This can be done by setting vector  $\vec{u}$  equal to the gradient  $\Delta f(x, y)$ , however, since  $\vec{u}$  is a unit vector it is then divided by the magnitude of the gradient  $\|\Delta f(x, y)\|$ . Therefore,

$$\vec{u} = \frac{\Delta f(x, y)}{\|\Delta f(x, y)\|},$$

which indicates the direction of steepest ascent. The hill climbing method then moves in this direction until a local maximum is found.

A data point  $y \in F^d$  is density-attracted to a density-attractor  $x$  if data points  $x$  and  $y$  points are connected through a path of data points having a high-density. Expressed mathematically, data point  $y \in F^d$  is density-attracted to a density-attractor  $x$  if there exists a path of points  $x^k$  with  $d(x^k, x) \leq \epsilon$  such that

$$x_i = x_{i-1} + \delta \frac{\Delta f_B^D(x_{i-1})}{\|\Delta f_B^D(x_{i-1})\|}, \quad i = 1, 2, \dots, k,$$

where  $x_0 = y$  and step size  $\delta$  is a small positive number which controls the speed of convergence. Therefore, data point  $y$  is density-attracted to a density-attractor  $x$  if it is possible to start at  $y$  and move in the direction of the gradient of the density function until a data point  $x^k$  is reached which is situated within the  $\epsilon$ -radius of  $x$ .

The DENCLUE algorithm is able to find arbitrary shapes and exhibits resistance to noise. Furthermore, the algorithm is able to cluster high-dimensional data sets. It also has a strong mathematical base and forms clusters faster than DBSCAN — approaching a factor of 45. The algorithm is, however, sensitive to its input parameters [90].

## 2.4 Chapter summary

This chapter provides a brief overview of machine learning with a specific focus on clustering, an unsupervised learning technique. A brief history of machine learning is provided in §2.1 which is then followed with a discussion on popular divisions of machine learning in §2.2. The focus of the chapter then shifts to clustering in §2.3 with a review of the literature pertaining to popular clustering methods, namely: Partitional clustering, hierarchical clustering and density-based clustering. Partitional clustering was discussed in §2.3.2 and in §2.3.3, hierarchical clustering was presented, which can be classified into two types, namely: Agglomerative and divisive. The last clustering method, *i.e.* Density-based clustering, was discussed in §2.3.4. Three popular density-based algorithms, *i.e.* DBSCAN, OPTICS and DENCLUE, were discussed in greater detail.

---



---

## CHAPTER 3

---

# Computer Simulation Modelling

### Contents

3.1	Simulation modelling overview . . . . .	33
3.2	Simulation modelling concepts . . . . .	34
3.3	Types of simulation models . . . . .	35
3.4	Simulation modelling paradigms . . . . .	35
3.4.1	<i>Discrete event modelling</i> . . . . .	36
3.4.2	<i>System dynamics modelling</i> . . . . .	36
3.4.3	<i>Dynamic systems modelling</i> . . . . .	37
3.4.4	<i>Agent-based modelling</i> . . . . .	38
3.5	Steps in a simulation study . . . . .	40
3.6	Verification and validation . . . . .	43
3.6.1	<i>Verification</i> . . . . .	43
3.6.2	<i>Validation</i> . . . . .	44
3.7	Advantages and disadvantages of simulation . . . . .	45
3.8	Review of simulation modelling in the context of TNCs . . . . .	45
3.9	Chapter summary . . . . .	46

In this chapter, the domain of simulation modelling is described and the most salient elements constituting a simulation model are introduced. An overview of simulation modelling as well as the fundamental concepts that form part of every simulation model are discussed in §3.1 and §3.2, respectively. Furthermore, in §3.3, the various types of simulation models are discussed which is followed by an introduction into the four main paradigms of simulation in §3.4. The twelve-step methodology employed to guide a simulation study is discussed in §3.5, thereafter, in §3.6, the verification and validation of a simulation model are described in greater detail. Some of the key advantages and disadvantages pertaining to the employment of simulation as a modelling tool are provided in §3.7. Penultimately, in §3.8, the use of simulation within the context of TNCs is discussed which focusses on pertinent work in the literature. Lastly, the contents of this chapter are summarised in §3.9.

### 3.1 Simulation modelling overview

The science of simulation modelling is typically employed in cases where real-world problems are to be solved, however physical (real-world) experimentation is not practically and/or financially

viable. Real-world problems are typically represented by some processes (or facilities) which is, according to common convention in simulation literature, referred to as a system. Simulation modelling enables and facilitates the understanding of the system structure, the experimentation of the system under different conditions (defined by different scenarios) and the optimisation of the system in a hypothetical context, *i.e.* without incurring operational or financial difficulties. After a suitable solution or outcome is obtained from the modelling approach — according to some pre-defined set of criteria — it can be considered for implementation in the real world [34, 55].

Simulation is a prolific and effective methodology employed in many scientific domains, for example, operations research, robotics and computer science [55]. Simulation can be defined as an imitation of the operation of a real-world system, be it an existing system or a concept demonstrator, in the form of a computerised model so as to study the behaviour of the system over time and provide insight into possible improvements that can be made in respect of its design and functional working [4, 72].

A simulation study is mainly characterised by its so-called *level of abstraction*. The extent (or degree) to which the exact detail and characteristics of the system under investigation are modelled is reflected by the level of abstraction. A low abstraction corresponds to a high level of detail — typical low abstraction simulation studies include computer hardware, automotive control systems, pedestrian movement and traffic micro models. A high abstraction, on the other hand, corresponds to a low level of detail for which an aggregation is typically used to model this type of system. Typical examples of simulation studies characterised by a high level of abstraction are economical dynamics (*e.g.* market and competition), population dynamics and ecosystems [34].

## 3.2 Simulation modelling concepts

There are several salient components that are common in all simulation modelling approaches. These include (but are not limited to) *system*, *entities*, *attributes*, *model*, *system state*, *system state variables*, *activities*, *resources* and *events*.

A system can be defined as the combination of a set of objects, *i.e.* entities, such as people or vehicles, that perform certain actions and interact in order to achieve some goal. The system state variables is a collection which represents the information necessary to describe the operation of the system at any point in time. This collection of variables provides a “snapshot” description of the system which is typically referred to as the system state [55].

Entities are objects possessing unique characteristics (*i.e.* attributes), for example consider a person as an entity, its attributes can be age, gender and height. A model is a representation of the system and is instrumental towards conducting meaningful analysis of said system. The model typically only includes aspects of the system relevant to the problem under consideration [4].

An activity is a process for which the duration is known prior to its execution in the simulation run — when the activity is initiated, its completion time can be determined, *i.e.* the termination of the activity can be scheduled. This duration can be a constant time period, a value sampled from some statistical distribution, a value retrieved from a file input, the output of a pre-defined function or it can be dependent on the system state. A delay is the result of a combination of conditions and, unlike an activity, its duration is indefinite. If an entity is queueing for a resource, the time spent in the queue is dependent on the events under consideration [4].



An event can be defined as an instance that alters the state of the system. Events may be either internal (*i.e.* endogenous) or external (*i.e.* exogenous). Internal events occur within the system whereas external events are those which occur outside the system but within the environment and have an impact on the system. For example, if a bank system was being modelled, the initiation of servicing a customer would be an internal event and a customer arrival would be an external event [4].

### 3.3 Types of simulation models

According to Law [55], simulation models can be classified along three dimensions namely: *Static versus dynamic*, *deterministic versus stochastic* and *continuous versus discrete*.

A static simulation model, also referred to as a *Monte Carlo* simulation model, represents a system at a certain point in time or a system in which time does not play a telling role. The results of a static model are functionally dependent on the user input. For example, the rolling of a die can result in equally likely outcomes of either 1, 2, 3, 4, 5 or 6, independent of time. A dynamic model, on the other hand, represents systems that change and evolve over time such as the growth of a population over some time period.

In deterministic simulation models, the input parameters are known values — a set of input parameters will result in the same output when replicated. For example, the return on an investment that is subject to a fixed interest rate. Stochastic models, on the other hand, contain random values — various possible outputs originate from the same set of input parameters. Therefore, the output generated by stochastic models should be considered as statistical estimates of the real-world system being modelled. A stochastic model can, for example, include the modelling of inter-arrival or service times of a bank by sampling from an exponential distribution.

A system can be modelled as a discrete simulation model if changes to the system only occur at points in time that are countable, whereas a system in which the state variables continually change over time can be modelled as continuous simulation models. For example, the arrival of customers at a mall at specific points in time is considered as a discrete model whereas the flow of water in a dam is considered as a continuous model.

### 3.4 Simulation modelling paradigms

There are four salient paradigms constituting simulation modelling, namely: *Agent-based modelling*, *discrete event modelling*, *system dynamics modelling* and *dynamic systems modelling*. The applicability of each paradigm depends on the level of abstraction at which the modelling ought to transpire. Agent-based modelling can be applied to systems at various levels of abstraction whereas system dynamics modelling is most appropriate when applied to systems requiring a high level of abstraction. Discrete event modelling, on the other hand, are typically applied to systems requiring abstraction levels that range from low to medium. Lastly, dynamic systems modelling is typically applied to systems necessitating a low level of abstraction. System dynamics modelling and dynamic systems modelling are mostly used in cases for which continuous processes are involved whereas agent-based modelling and discrete event modelling perform favourably when applied to cases where the working of the system is characterised by discrete processes. A graphical illustration of the four paradigms of simulation modelling is presented in Figure 3.1.

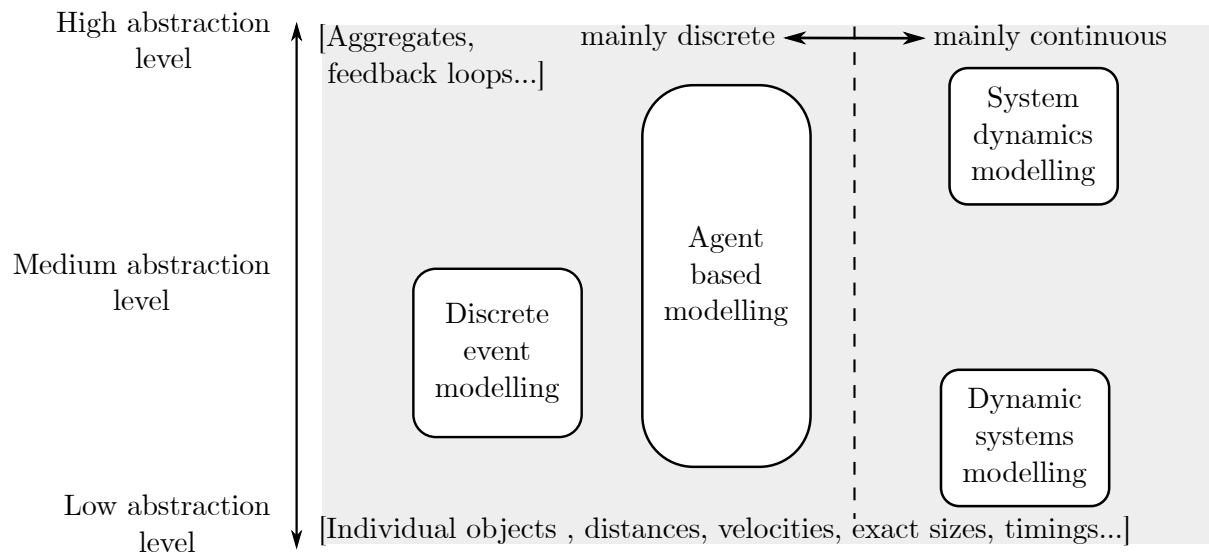


FIGURE 3.1: Abstraction levels of the different types of simulation modelling, namely: Agent-based modelling, discrete event modelling, system dynamics modelling and dynamic systems modelling [11].

### 3.4.1 Discrete event modelling

Discrete event modelling was introduced by Geoffrey Gordon, an IBM engineer, in 1961 [32]. This method regards the system as a process, mainly discrete, or a series of actions carried out by the agents. Entities in discrete event modelling, for example, can include (but are not limited to) clients, patients and products. Entities respond to events deemed instantaneous occurrences which alter the state of the system. Resources, on the other hand, are system entities which provide some service, for example, staff, workers, equipment and transport. The movement of entities are described using a flow chart, as illustrated graphically in Figure 3.2, according to which the customers are the entities and the tellers are the resources in a model of a bank kiosk [11].

### 3.4.2 System dynamics modelling

System dynamics modelling was developed by Jay Forrester at MIT in the 1950's [29] and typically includes discrete quantities of items such as people, products and events. System dynamics modelling can be defined as the modelling of complex organisational and social systems made up of an accumulation of entities using stock, flows, feedback loops and delays in order to provide an overview of the changes in a system over time [41, 97]. System dynamics models are mainly used to simulate continuous processes, represented by three aspects, they are stocks (*e.g.* materials, people or money), the flow between stocks and the calculation of the stock quantity. The system is described using several interacting balancing and reinforcing loops and delays. Balancing loops comprise a mixture of changes that have an increasing and/or decreasing effect whereas in the case of a reinforcing loop the changes in the loop are either all increasing or all decreasing. In Figure 3.3, an illustrative example of a system dynamics model is presented which models the adoption of new ideas and products. This is also known as the well-known *Bass Diffusion model*. The stocks in this model include the potential adopters and the adopters whereas the adoption rate describes the flow between these two stocks. The adoption rate naturally depends on advertising and word of mouth, as illustrated in the figure. Two balancing loops and one reinforcing loop are also included in the figure [11].

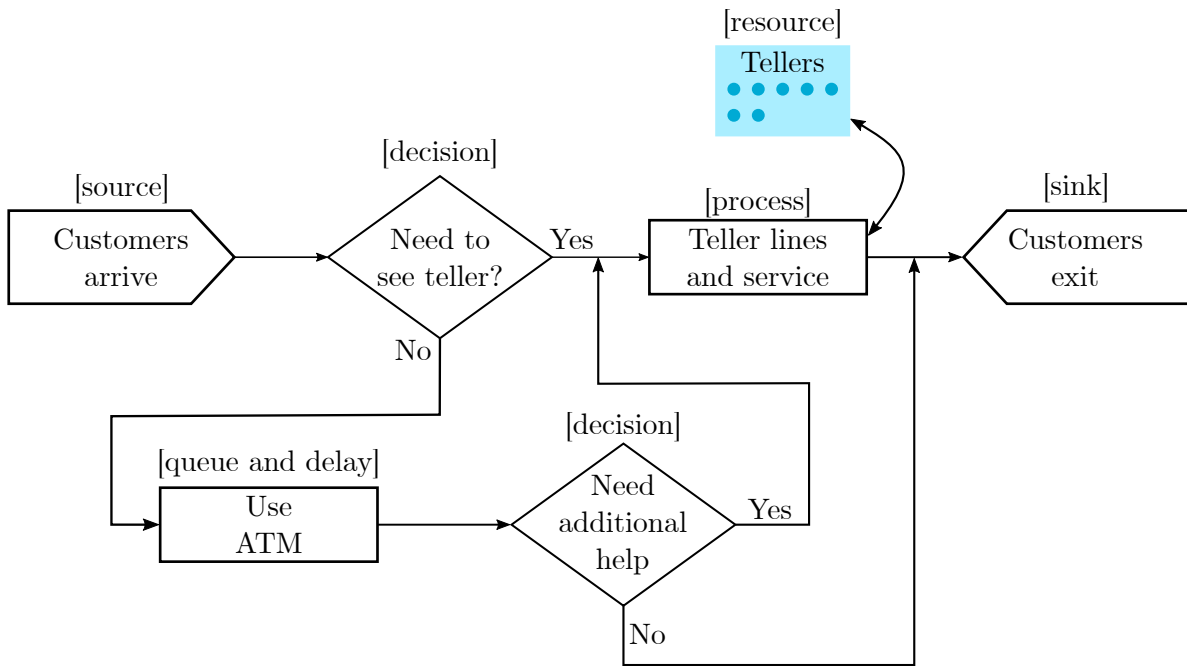


FIGURE 3.2: A discrete event model of a bank kiosk [11].

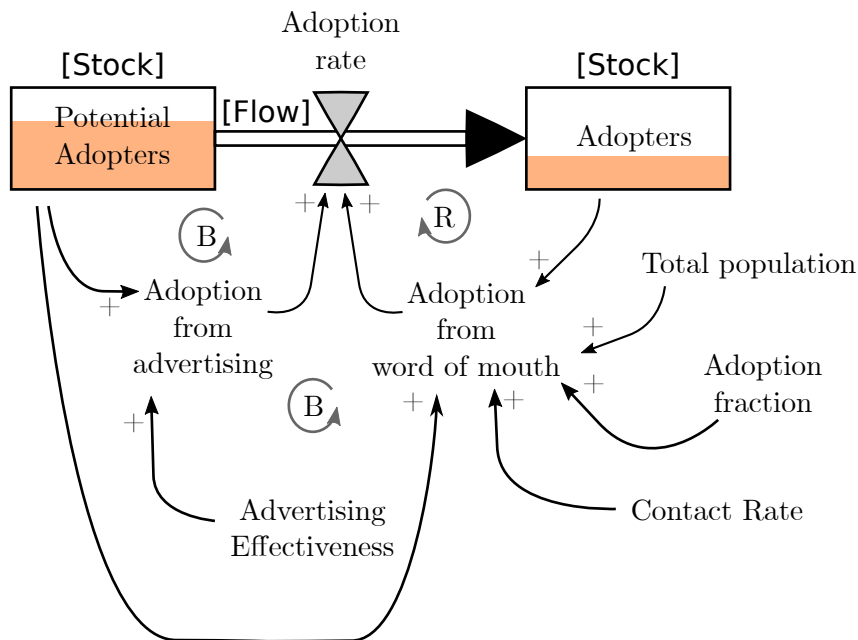


FIGURE 3.3: A system dynamics model in respect of the adoption of new ideas and products [11].

### 3.4.3 Dynamic systems modelling

Dynamic systems modelling is considered as the predecessor of system dynamics modelling and forms part of the design process in technical engineering disciplines such as mechanical, electrical and chemical engineering [11]. In Figure 3.4, an illustrative example of the graphical coding language used in the modelling of a system dynamics model is presented which simulates a

bouncing ball. The system dynamics model of a ball bouncing on a surface where a loss in energy of the ball results in the time interval between collisions with the surface decreasing. The mathematical model of a dynamic system model comprises multiple state variables and algebraic differential equations relating to the variables. Unlike in system dynamics modelling, these variables are continuous and are not aggregates of entities. These values also often include a physics-based counterpart such as velocity, acceleration and pressure. Dynamic systems enable greater complexity (functionality and mathematically) when compared with system dynamics and can be used to solve any system dynamics problem more effectively [11].

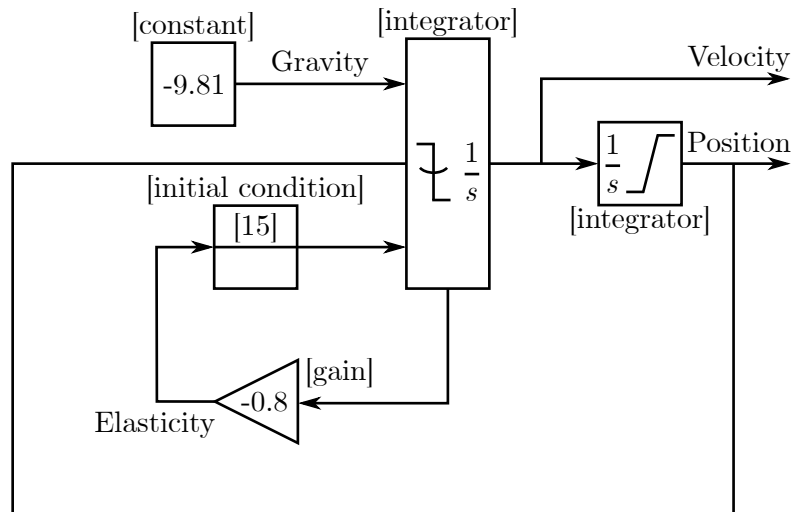


FIGURE 3.4: A dynamic systems model of a physics-based bouncing ball [11].

### 3.4.4 Agent-based modelling

Agent-based modelling is a recent approach to simulation modelling when compared with both system dynamics and discrete event modelling. Agent-based models generally require greater computational resources and, as a result, has experienced greater interest as advances in computer hardware have increased. Instead of considering the system as a whole, this method employs a “ground-up” approach by modelling the behaviour of and interaction between the objects (or entities) in the system. The patterns and behaviours embedded within the system are therefore not explicitly modelled (*i.e.* hard coded) but they are developed implicitly by means of the simulated interactions between the objects [34]. Some pertinent examples of agent-based models developed include the spread of diseases, the adaptive immune system and the purchasing behaviours of customers [58].

An agent-based model comprises three main elements, namely: A set of agents each with their own attributes and behaviours, the relationships and interactions between agents, and the environment of the agent which include the interaction between the agent and its environment. The main attributes of an agent is its autonomous behaviour. An agent should be able to act or react to situations without external influence or guidance. Furthermore, an agent should be self-contained, *i.e.* the agent should be an identifiable individual with a boundary. Agents also have various states that change with time. The state of an agent describes its current situation and its behaviour. Agents should also have a social property and interact with other agents. The attributes of an agent may be static or dynamic. Static attributes do not change during the simulation, such as the agent’s name. Dynamic attributes, on the other hand, change during the simulation, such as the agent’s neighbours and the agent’s memory. An agent’s location

can be deduced by using the environment in which the agent is modelled, *i.e.* by means of the *graphic information system* (GIS) location of the agent [58]. In Figure 3.5, a graphical illustration of the interaction between agents, directly and indirectly, together with the interaction with the environment is presented. The agent *statechart* describes the agent's behaviour and is also illustrated in this figure.

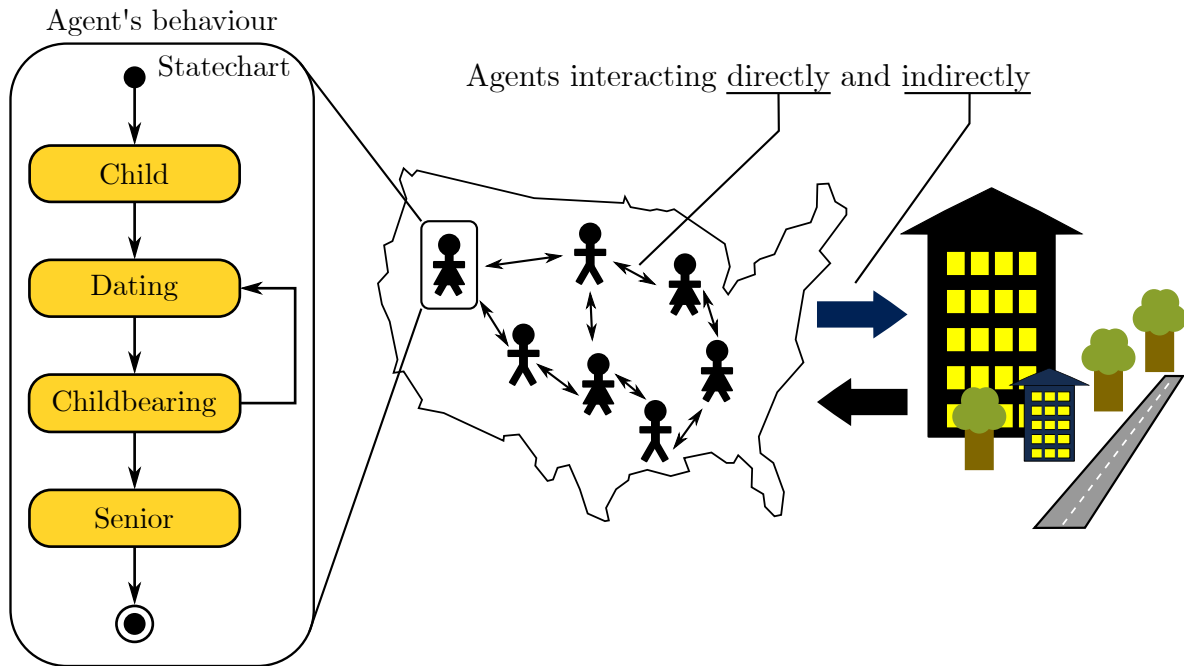


FIGURE 3.5: A depiction of agents, individually controlled by a statechart enabling them to interact among themselves directly and indirectly (*i.e.* through the environment) [11].

The definition of an agent's behaviour forms an integral part of agent-based modelling. The behaviour of the agents may be described using a statechart which is a graphical method of defining behaviours that occur at certain points in time or during certain events. A statechart comprises states, where each state dictates the agent's behaviour, *i.e.* the actions and the transitions that move an agent between states. An agent can only be in one state at a time. The different components of a statechart are depicted graphically in Figure 3.6. The statechart entry point is used to indicate the initial state of the statechart, *i.e.* an indication of the state the agent assumes at the initiation of the simulation run. In reference to Figure 3.6, the agent will enter State A when the simulation is executed, as indicated by the statechart entry point. There are two kinds of states, namely: A simple state and a composite state. A composite state is one that contains other states that are unlike a simple state which does not contain any substates. A history state only exists within composite states and provides a reference to the state that was last entered within that composite state. The final state pointer indicates the termination of the statechart. Once an agent enters the final state it can not transition to any other state within the statechart. A branching state enables the agent to move to a selection of states depending on which transition is triggered. The agent in State B can go to either State D or to State C. A branching state can also be used to merge more than one transition into a single state. Lastly, the initial state pointer indicates the initial state of a composite state. When an agent enters the composite state, the initial state pointer indicates which state the agent should enter. In Figure 3.6, when the agent transitions from State A to the composite state it will then immediately enter into State B.

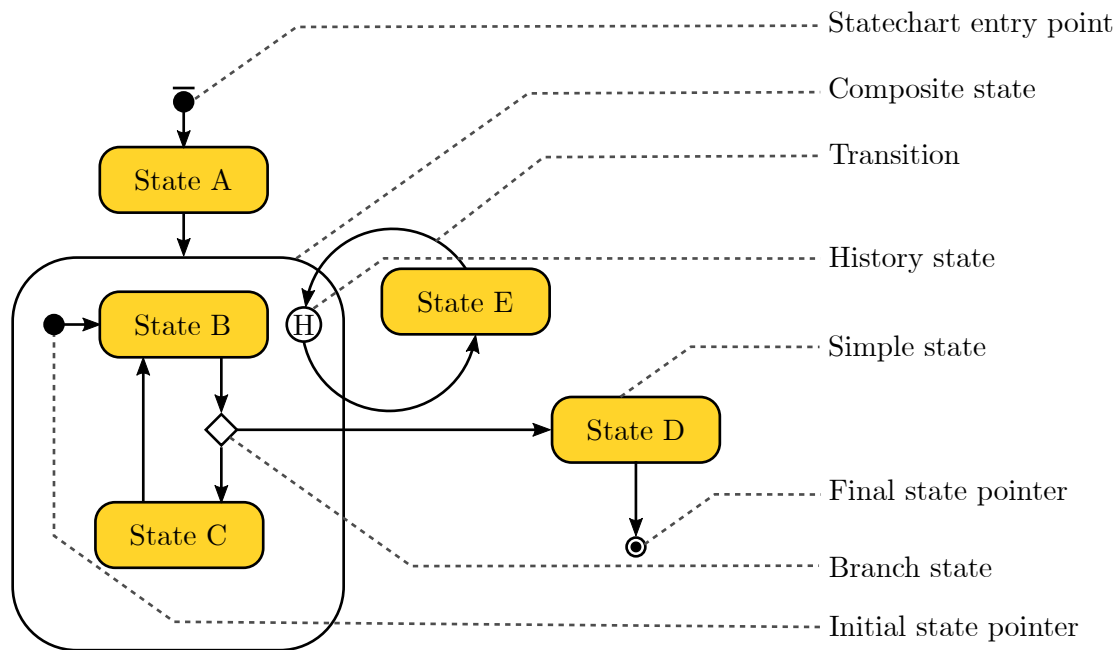


FIGURE 3.6: Statechart containing states A to E with transitions between states [11].

As mentioned previously, a transition moves an agent from one state to another upon its initiation. A transition can be initiated (or activated) by using a *timeout*, *rate*, *condition*, *message* or *arrival* triggers. A timeout trigger, illustrated graphically in Figure 3.7(a), indicates that once an agent has entered State A, it remains in State A for a specified time period before transitioning to State B. A rate trigger, illustrated in Figure 3.7(b), is similar to a timeout trigger, however instead of specifying a constant time interval, some probability distribution (*e.g.* exponential) is used to sample random values from. In Figure 3.7(c), a condition trigger is illustrated graphically. Accordingly, after an agent enters State E, it only moves to State F when a specified condition is true. A message trigger, illustrated in Figure 3.7(d), permits an agent to move from State E to State F when the agent receives a message. This transition can be set to trigger on any message or only on a specific message. Lastly, according to an arrival trigger, as illustrated graphically in Figure 3.7(e), an agent transitions from State G to State H once it has arrived at its destination.

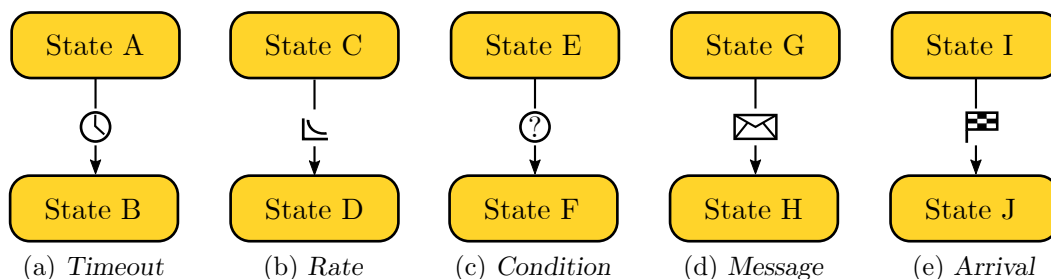


FIGURE 3.7: The different types of transition triggers.

### 3.5 Steps in a simulation study

The seminal work by Banks [4], Law [55] and Shannon [84] addresses the steps that should be present in all simulation studies. These steps are illustrated in Figure 3.8 and are discussed

below.

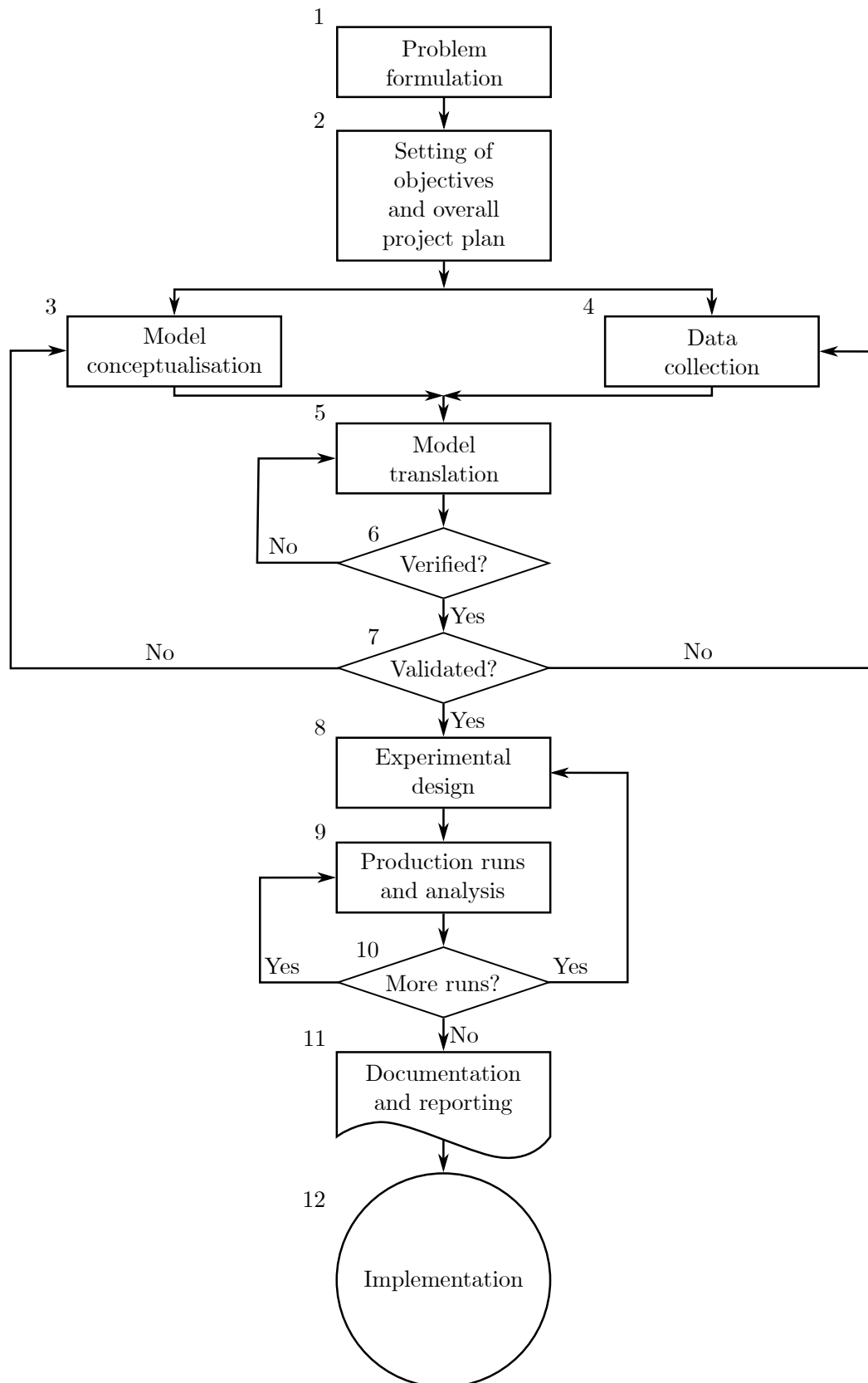


FIGURE 3.8: The twelve steps constituting a typical simulation study [4].

1. *Problem formulation:* At the start of a simulation study, a problem statement should be formulated. A crucial aspect of the problem formulation is to ensure that both the client and the simulation analyst understand and agree with the problem formulation. As the study progresses, the problem may be reformulated — attributable to the attainment of an improved understanding of the problem under investigation.
2. *Project planning:* The project plan details the scope of the project and the objectives to be achieved. These objectives specify the questions that must be answered at the end of the study. The project plan should also describe the different scenarios that are to be investigated during the study and in addition, address key considerations with respect to planning hardware and software requirements, the number of people involved, the cost, the proposed timeline of the study and the desired outputs after each stage.
3. *Model conceptualisation:* In this step, a conceptual model is constructed to abstract the real-world system. The conceptual model comprises mathematical and logical relationships that exist between the components and structure of the system. Often, the best method is to start with a simple model and add complexity as the modelling process progresses. The model complexity, however, should not exceed the level that is sufficient for achieving the objectives of the study for which the model is built — this unnecessarily increases the computational, financial and operational burden associated with the model-building procedure. The client should be involved during the model conceptualisation step in order to improve the quality of the model and, furthermore, enhance the confidence that certain stakeholders have in the constructed model.
4. *Data collection:* This step occurs concurrently with the model conceptualisation step (as indicated in Figure 3.8). There is a continual interplay between the model conceptualisation and the required input data as an increase in model complexity could result in a change in the data required for model development. The objectives of the study also largely dictate the required data. Data availability is an important consideration during model conceptualisation.
5. *Model translation:* This step involves the selection of a suitable programming language or simulation software package which will be used to convert the conceptual model, developed in Step 3, into a computer-recognisable format.
6. *Model verification:* Verification involves the assessment of the constructed computer model in order to establish whether the model is performing correctly. Verification should be performed continually throughout the development of the computer model to ensure that each new addition to the model operates as expected and contributes to the successful working of the model. Debugging functionality included in most simulation software suites can be used to assist in the verification process by identifying and addressing errors in the model.
7. *Model validation:* Validation is employed so as to assess if the model is an accurate representation of the real-world system. This is accomplished by comparing the developed model with the real-world system by observing the similarities and differences between the two. If there are any significant differences, the model is then revised. This is repeated until the accuracy of the model is acceptable.
8. *Experimental design:* During this step, decisions pertaining to the key considerations of the system's design should be investigated. For each scenario under investigation, the number of simulation runs (or replications), the length of each simulation run and the initialisation conditions are to be specified.



9. *Production runs and analysis:* The various simulation runs are executed according to the key decisions made in the previous step. The results of these runs are then recorded and analysed in order to gain insight into the working of the system under different circumstances, *i.e.* to analyse the performance of the simulated system. Typical methods include the calculation of confidence intervals for measuring the performance of a particular system design or to decide on the best simulated system according to certain performance measures.
10. *More production runs:* Based on the analysis performed in the previous step, a decision is made whether more simulation runs are necessary or if additional experiments should be designed and performed.
11. *Documentation and reporting:* Documentation of both the simulation model and the implementation of the simulation, software together with the progress of the study and simulation results, must be performed especially if the developed simulation model is to be used in follow-up work, either by the same or different analyst, in order to get a better understanding of the working of the model. The documentation assists an analyst who would want to modify the model and enhances the confidence in the operation of the model. Furthermore, it allows for insight into the procedure followed and the decisions made during the study. A clear and concise documentation of the final report should be constructed which includes all scenarios investigated, the criteria of comparison, the experimental results and recommended solutions to the problem.
12. *Implementation:* The analyst acts as a reporter and provides recommendation to the client (or other stakeholders) which forms part of the documentation procedure performed in the previous step. Decision-making pertaining to the final implementation of the model is therefore improved. The success of the implementation is increased if the analyst has meticulously followed the previous steps and if the client has been involved throughout the study.

## 3.6 Verification and validation

During simulation modelling, the real-world system is first *abstracted* into a conceptual model. The conceptual model is then developed and coded into a computerised, functionally correct model. It is essential that the developed model is an accurate representation of the real-world system. The process of verification and validation ensures that the real-world system is best emulated. This process is characterised by a highly iterative nature. If pertinent differences between the developed model and the real-world system are identified, appropriate changes should be implemented so as to ensure it represents the conceptual model or changes should be made to the conceptual model to represent the real-world system. Once the appropriate changes have been made, the verification and validation processes, illustrated in Figure 3.9, are repeated [4].

### 3.6.1 Verification

According to Banks [4] and Law [55], verification is performed in order to ensure that the developed model is an accurate representation of the conceptual model. Verification is mainly concerned with debugging the model's computerised implementation. The model is "disassembled" into its constituent sub-models during development. After completion of each sub-model, they

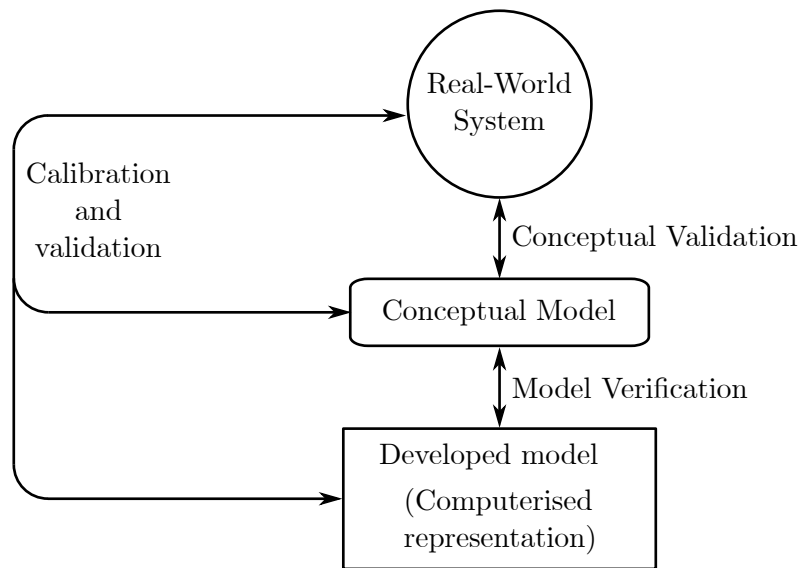


FIGURE 3.9: A flow diagram illustrating the verification and validation process when developing a simulation model [5].

must be debugged so as to ensure that it is free from errors before developing the next sub-model. This process is iterated until the model accurately represents the system. The development of the model commences with the development of a less-detailed model and then adding complexity as opposed to attempting to develop a complex model from the start. Another method used to verify a simulation model is a so-called *trace*. A trace is employed by displaying the system state, *i.e.* system state variables, agent indices and certain statistical counters in the console and conducting a comparative study with logical expectations. The model's animated output can also be used to ensure that the entities of the simulation are performing as expected and that there are no actions violating logical principles. Lastly, an interactive debugger is recommended ascribed to its ability to continually check for and identify errors during a simulation run.

### 3.6.2 Validation

Validation is a method for testing if the accuracy of the conceptual model represents the real-world system with respect to achieving the objectives of the study. Banks [4] describes several techniques that can be used for validation. The first technique is face validation, according to which the output of the simulation is observed and proffered to subject matter experts. Another technique relates to performing a sensitivity analysis where the input of the model is varied in order to observe if the input-output relationship corroborates the real-world system. Extreme condition testing can also be performed during which the behaviour of the system is analysed when its inputs are subjected to extreme values. For example, if the arrival rate of a model of a queueing system is set to a significantly high value, an increase in the number of individuals in the queues should be observed together with an increase in the time each individual spends in the system. The conceptual model assumptions must also be validated. Two types of model assumptions exist, namely: Structural assumptions and data assumptions. Structural assumptions relate to the operation of the real-world system which can be validated by observing the real-world system and discussing these assumptions with individuals who interact with the real-world system. Data assumptions can also be validated by consulting with those knowledgeable about the real-world system or by collecting data from the real-world system by, *i.e.* conducting a survey or by means of empirical analysis of the real-world system.

### 3.7 Advantages and disadvantages of simulation

Studies performed by Banks [4] and Law [55] summarise the advantages and disadvantages pertaining to simulation modelling. A fundamental advantage of simulation is the modelling of complex real-world systems that often cannot be modelled analytically. Furthermore, simulation enables the ability to observe the system and, more importantly, the effects of proposed changes to the system, for example, changes to the hardware, labour, layouts and policies, without allocating resources and interrupting the real-world system. The results produced by a correctly developed simulation model are deemed reliable as the simulation model has been subjected to rigorous development during which it was tested, verified and validated. Simulation software suites allow for the compression and expansion of time during a simulation run. The compression of time allows for the study of systems with a long time period, such as the study of population growth over many years, in a relatively short time. The expansion of time, on the other hand, allows for the observation of the details of system operations and dynamics which occur rapidly in the real world. The ability to alter the speed of time enables and improves the investigation and understanding of the nuances and general working in the real-world system.

During simulation development, a better understanding of the system evolves and insight gained into the interaction of various entities and their importance with respect to the system. The understanding of the components existing within the complex system facilitates problem diagnosis which is further enhanced by the creation of a visual representation of the real-world system while operations are running — a feature of the majority of simulation packages. Simulation also enables the testing and understanding of systems that do not exist in the real world enabling prescriptive analytics before implementation by testing the impacts of various scenarios and, therefore, reducing the risk of implementing a new or altered system in the real world which might not be fully operational or suitable to the environment.

Disadvantages of simulation, on the other hand, can include additional training and expertise required to develop a model which accurately represents the real-world system. Since two analysts' experience in building models will typically differ, two models of the same system which was developed separately will seldom be identical. However, many simulation packages include access to models of various systems which only require input values in order to operate, allowing inexperienced users to test the working of systems. If the simulation results are based on random inputs, several independent runs of the simulation should be performed for each set of input parameters in order to develop estimates of the model's characteristics. The results must then be compared to the real-world system in order to confirm that the model is operating as expected. Simulation software packages often provide tools to analyse the outputs of developed models so as to assist with the interpretation of results. Furthermore, the development of an accurate simulation model is time-consuming and the simulation software necessary to develop such a model is often very expensive, however, as computer hardware improves, the time spent testing various scenarios decreases. The possibility also exists for simulation to be applied unnecessarily, for example, developing a simulation model when an analytical solution exists. Lastly, if the model does not accurately represent the real-world system then the results provide no value.

### 3.8 Review of simulation modelling in the context of TNCs

Simulation modelling has been used in the literature to describe the operation of TNCs. A agent-based model has been developed by Shaheen [82] in which the area used to model the operation of TNCs is modelled using spacial cells and the road network is used to determine

whether a driver may move between cells. The driver movement between rides is modelled as a random movement between cells. Nourinejad and Roorda [67] used agent-based simulation to model dynamic ridesharing (*i.e.* where a ride is shared by riders whose routes overlap either fully or partially). This system was modelled using a simplified representation instead of modelling real-world road infrastructure. Lokhandwala and Cai [57] used agent-based simulation to study the differences between dynamic ridesharing using traditional taxis and using shared autonomous vehicles. Both of these switch focus on when a driver is active (*i.e.* they have been requested by a rider or their vehicle is occupied by a rider) and have little or no mention on driver activity when waiting to be requested. A dynamic ridesharing model was also developed by Birschoff *et al.* [7] where after a ride is completed, the driver remains at the drop-off location until requested again. A popular topic of discussion in the literature, similar to TNCs, is the taxi industry. The traditional yellow taxis are the only vehicles in NYC that are allowed to pick riders up anywhere in NYC by means of street-hailing (*i.e.* picking-up a rider from the street who has not reserved a ride) and by means of pre-arranged ride-requests. Drivers of TNCs, however, may only pick riders up by means of pre-arranged ride-requests. Simulation models have been developed by Cheng and Nguyen [16], Grau and Romeu [33] as well as Kim *et al.* [50], among others, to study certain behaviours of this system.

### 3.9 Chapter summary

Simulation is a widely employed method in solving complex real-world problems with minimal resources whilst accommodating and facilitating the attainment of an understanding into the system under consideration together with the visual representation of proposed solutions. This chapter provided a review of the literature regarding simulation modelling. In §3.1, a brief overview of simulation modelling was provided which was followed, in §3.2, by a discussion on the intrinsic simulation modelling concepts *i.e.* system, entities, attributes, model, system state, system state variables, activities, resources and events. Furthermore, the three dimensions that can be used to classify simulation models, *i.e.* static *versus* dynamic, deterministic *versus* stochastic and continuous *versus* discrete, were discussed in §3.3. The focus in the chapter then shifted to the four paradigms of simulation models in §3.4, namely: Agent-based modelling, discrete event modelling, system dynamics modelling and discrete event modelling. In §3.5, the steps that should be present in a simulation study were identified and discussed which was followed, in §3.6, by the processes employed to verify and validate a simulation model. Some advantages and disadvantages of simulation modelling were discussed in §3.7 and, lastly, an overview of simulation modelling in respect of TNCs was provided in §3.8.

---



---

## CHAPTER 4

---

# An Agent-Based Model of TNCs








### Contents

4.1	AnyLogic simulation modelling environment . . . . .	48
4.2	Simulating the movement of drivers . . . . .	48
4.3	Parking area selection . . . . .	49
4.4	Clustered waiting area selection . . . . .	50
	4.4.1 <i>Selection of input parameters</i> . . . . .	51
	4.4.2 <i>Output analysis and verification</i> . . . . .	52
4.5	The simulation experiment graphical user interface . . . . .	55
	4.5.1 <i>The main environment</i> . . . . .	61
	4.5.2 <i>Functionality assessment of the GUI</i> . . . . .	63
4.6	The simulation model . . . . .	66
	4.6.1 <i>Driver agent</i> . . . . .	67
	4.6.2 <i>Rider agent</i> . . . . .	70
	4.6.3 <i>Parking area agent</i> . . . . .	72
	4.6.4 <i>Clustered waiting area agent</i> . . . . .	73
	4.6.5 <i>Verification of the TNC simulation model</i> . . . . .	74
	4.6.6 <i>Validation of the TNC simulation model</i> . . . . .	75
4.7	Chapter summary . . . . .	79

An agent-based model of the movement of drivers and the interactions between riders and drivers of TNCs is designed and developed in this thesis. In this chapter, an in-depth discussion on the operation and composition of the model is provided. In §4.1, the AnyLogic simulation environment is described together with the most salient components forming part of in the development of computer simulation models in AnyLogic. Thereafter, background information of the movement of drivers of TNCs is provided in §4.2. The focus then shifts to a description of the selection of parking areas and clustered waiting areas in §4.3 and §4.4, respectively. In §4.5, the graphical user interface, which can be used to conduct experiments and analyse various scenarios, is described and the functionality of its various components are verified. Thereafter, in §4.6, the simulation model is described in detail which is accompanied by the verification and validation thereof. The contents of the chapter are then summarised in §4.7.

## 4.1 AnyLogic simulation modelling environment

An agent-based model of drivers and riders of TNCs has been designed and developed in the AnyLogic Personal Learning Edition 8.5.0 software suite which is developed by *The AnyLogic Company*. AnyLogic is a multi-method modelling environment in which models can be developed in three modelling paradigms, namely: Agent-based modelling, discrete event modelling and system dynamics modelling. The combination of these three paradigms can be used to simulate complex systems at various levels of abstraction. Furthermore, AnyLogic includes functional features such as advanced animation and visualisation — consequently, model outputs can be presented using a visually appropriate and understandable medium. The models developed in AnyLogic can also be enhanced by implementing extensions in the form of Java Code. AnyLogic is selected as an appropriate and powerful simulation software environment in which the system under investigation (*i.e.* TNCs) is modelled. This decision is further supported by the GIS capabilities offered by AnyLogic which includes advanced routing and the capability to import shapefiles — a necessity given the nature of the problem at hand. Databases can assist in the creation and parametrisation of agents and their behaviour — an important feature ascribed to the fact that historical data are considered in this thesis. Agents can also be added and removed as time progresses, therefore riders can be added to the simulation environment upon request.

In order to develop the model in the AnyLogic environment, several simulation components are utilised. AnyLogic illustrates these components by means of icons. The first component describes the agent type, *i.e.* object class, and is illustrated using the  icon. Each agent in an agent-based model corresponds to a certain agent type. An agent may also have parameters associated with it and are indicated in AnyLogic by the  icon. Parameters represent characteristics of the agent which usually do not change during a simulation run. An agent may also have variables and are indicated in AnyLogic by the  icon. Variables store results of the simulation and are used to model agent characteristics that change over time. A collection contains a group of objects, *i.e.* elements, and is indicated by the  icon. AnyLogic permits the user to define functions which are indicated by the  icon. Functions can either return a value or perform an action and are especially useful when multiple replications of the same functional task are executed during the simulation run. Users may also add events which are denoted visually by the  icon. Events schedule some type of action during the simulation run. Lastly, AnyLogic facilitates the addition of schedules, illustrated by the  icon, enabling the change of values at specified time instances.

## 4.2 Simulating the movement of drivers

The model developed in this thesis aims to simulate the movement of drivers and the interactions (*i.e.* pick-up and drop-off) between drivers and riders. The general operation of this process is illustrated graphically in Figure 4.1. Once a rider requests a ride, the nearest available driver is identified and assigned to the ride-request. Some TNCs provide the option of selecting a larger vehicle or a upper-class vehicle, however an assumption of homogeneity is made — *i.e.* all vehicles are assumed to be the same. An available driver can be either present at a waiting area (waiting for a ride-request) or driving towards a waiting area after completing a ride. Waiting areas can include well-known (established) hotspots, *i.e.* areas known for having a high pick-up concentration (based on historical data). In this model, the waiting areas are restricted to known parking areas. Once all the available drivers are identified, the driver closest to the rider, *i.e.* the driver having the shortest driving distance to the rider, is selected and the ride-request is confirmed. If there are no available drivers, the model re-checks for available drivers every second until one is found. Expectedly, the waiting time experienced by a rider is therefore affected by the availability of drivers.

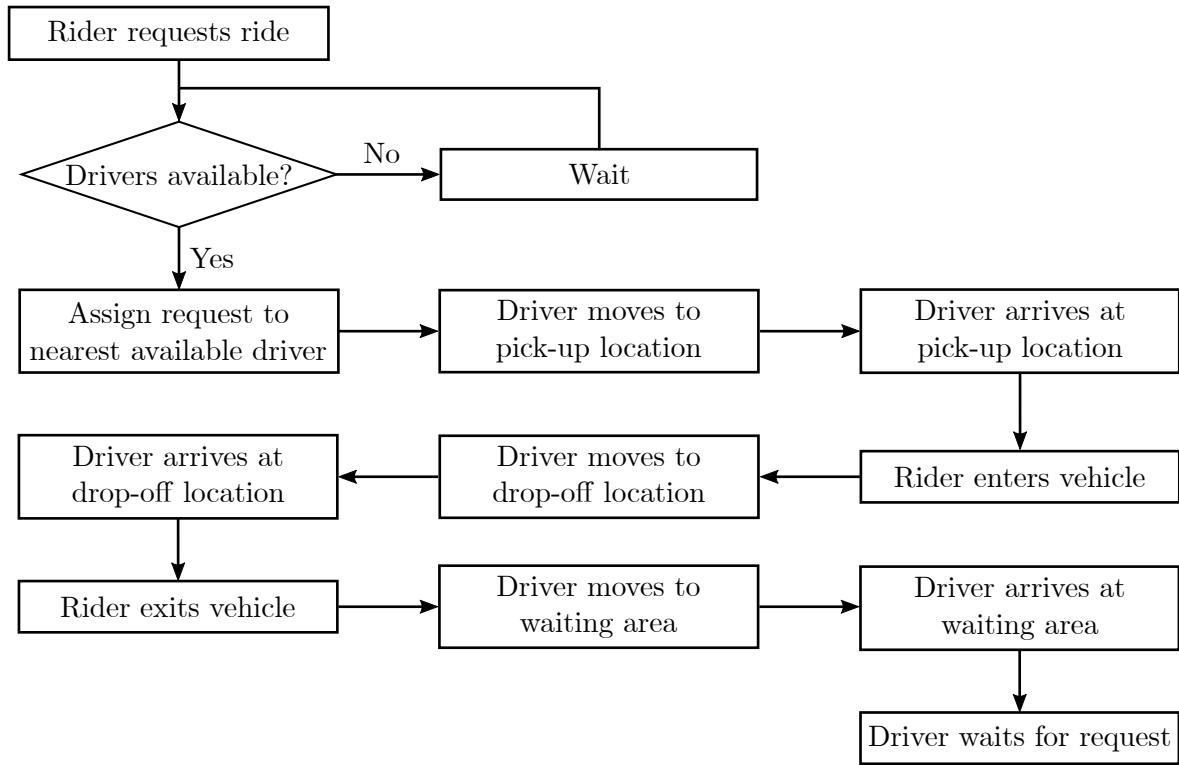


FIGURE 4.1: Flowchart of the general operation of a TNC service.

Once a driver is assigned to a ride-request, the driver is no longer available to accept ride-requests and movement towards the rider is initiated. When the driver arrives at the pick-up location, the rider enters the vehicle and the driver then travels to the drop-off location. Upon arrival, the rider exits the vehicle and the driver moves to a parking area (*i.e.* an open area where individuals can park their vehicles) so as to wait for the next ride-request. In the developed model, there are two types of areas where drivers may wait, the first of which is referred to as parking areas whereas the second type is referred to as the clustered waiting areas which are the areas identified using clustering. The time elapsed whilst travelling some route is determined using Bing Maps API which ensures realistic travel times, ascribed to the fact that traffic is taken into consideration.

### 4.3 Parking area selection

Parking areas are areas accessible by drivers to park their vehicles whilst waiting for ride-requests which reflects the real-world situation pertaining to parking area selection (*i.e.* without the effect of clustering historical pick-up locations) by simulating the driver's decision regarding the selection of a parking area. Drivers select a parking area using an appropriate selection strategy. The selection strategy considered is based on two aspects, *i.e.* experience and distance, as defined by Cheng and Nguyen [16]. The attractiveness of parking area  $j$ , denoted by  $a_j$ , is given by

$$a_j = \frac{c_j}{r_j(p)^2}, \quad (4.1)$$

where  $c_j$  denotes the number of times a driver has been historically requested from parking area  $j$  and  $r_j$  denotes the travelling distance between parking area  $j$  and the driver's current location  $p$ .



The attractiveness measure of (4.1) ensures that a parking area is more attractive if the driver has experienced many ride-requests from that parking area and if it is closer to the driver. The distance to the parking area is squared — a larger distance corresponds to an increase in both travel time and travel cost. The driver therefore moves towards parking area  $j$  with probability  $p_j$  which is given by

$$p_j = \frac{a_j}{\sum_j a_j}. \quad (4.2)$$

Uber Technologies Incorporated [102] identified certain areas in NYC as so-called hotspots. Hotspots can be defined as areas with a historically high demand during certain periods throughout the day. These hotspots are incorporated by adjusting the attractiveness of a parking area in this hotspot to

$$a_j = \frac{c_j^2}{r_j(p)^2}. \quad (4.3)$$

A driver, however, is only able to move to parking areas in its neighbourhood or surrounding neighbourhoods. If a driver has experienced no pick-up requests from its current neighbourhood or from surrounding neighbourhoods, the driver then moves to the nearest parking area. For example, in Figure 4.2(a), there are three parking areas known to the driver, *i.e.* A, B and C. In the case of parking area A, the values of  $c_A$  and  $r_A$  are 0 requests and 5km, respectively. Since parking area A is not a hotspot, the attractiveness  $a_A$  can be calculated using (4.1), resulting in  $a_A = 0$ . Similarly, in the case of parking area B, its corresponding values are  $c_B = 8$  requests and  $r_B = 10$ km, which results in  $a_B = 0.08$ . Lastly, in the case of parking area C the corresponding values are  $c_C = 1$  request and  $r_C = 2$ km, which results in  $a_C = 0.25$ . Finally, the probabilities of the driver travelling to either parking area A, B or C can be calculated using the expression in (4.2). Accordingly, the resulting probabilities of the driver travelling to either parking area A, B or C equates to 0, 0.24 and 0.76, respectively. Even though the driver has experienced a high number of pick-up requests from parking area B, the distance between the driver and parking area B is relatively large. Consequently, there is a higher probability that the driver will select parking area C even though the number of pick-up requests experienced is lower. However, if parking area B is a known hotspot, as illustrated in Figure 4.2(b), the attraction of parking area B is then calculated using (4.3) and results in an attraction of  $a_B = 0.64$ . The probabilities of the driver selecting either parking area A, B or C can then be recalculated and equates to 0, 0.72 and 0.28, respectively. Therefore, the driver now has a higher probability of choosing parking area B in comparison with choosing parking area C — ascribed to the fact that parking area B is a hotspot.

Attributable to computational limitations, the number of parking areas was limited to one parking area per neighbourhood. NYC was divided into 229 neighbourhoods according to a shapefile obtained from NYC OpenData [70]. The centroid of each neighbourhood was found and snapped to the nearest known parking area. It was found that the largest distance between a parking area and its nearest neighbour is 3.42km and the largest distance between a parking area and the border of its neighbourhood is 5.67km.

#### 4.4 Clustered waiting area selection

The clustering algorithms reviewed in §2.3, *i.e.* partitional clustering, hierarchical clustering and density-based clustering, were considered for inclusion towards cluster historical pick-up requests. Clustering historical pick-up locations is performed with the aim of finding waiting areas for drivers of TNCs in order to decrease rider waiting time. Towards this end, clustering is performed. The centres of these clusters are then used as the waiting areas. Ideally, it is also



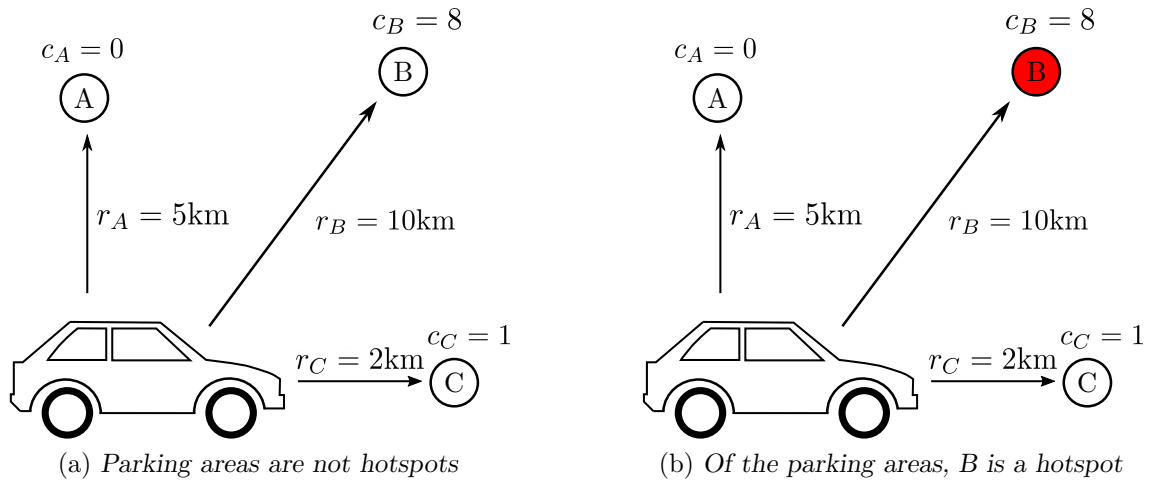


FIGURE 4.2: Illustration of the parking area selection strategy using the driver's experience.

required that these waiting areas change in real-time, *i.e.* as the demand changes.  $K$ -means clustering was chosen to form the basis of the analysis conducted in this thesis — this decision is substantiated hereafter. Firstly, this clustering algorithm takes outliers into account which is necessary to ensure that pick-up requests occurring in outlying areas are taken into account.  $K$ -means clustering is also a distance-based clustering algorithm and it creates clusters using each cluster's centres, rendering it a well-suited approach towards the problem considered in this thesis — *i.e.* cluster centres can be used as waiting areas. Furthermore, this clustering approach enables the specification of the initial centroids which facilitates real-time clustering. To the best of the author's knowledge, an extensive investigation into the employment of  $K$ -means clustering towards identifying suitable waiting areas for TNC drivers is a novel approach.

#### 4.4.1 Selection of input parameters

The  $K$ -means clustering algorithm requires several input parameters. The main inputs include an appropriate data set, the number of desired clusters and a suitable initialisation method, as discussed in §2.3.2. The manner according to which the model inputs are specified, which is illustrated graphically in Figure 4.3, is elucidated hereafter:

**Data set (to be clustered):** The pick-ups occurring in a time period before the point in time for which the waiting areas are desired are obtained from a data set containing the location, *i.e.* latitude and longitude coordinates, and pick-up times of a well-known TNC during April 2014 in NYC — this data set can be accessed from Kaggle [46]. The data set under consideration is amended so as to only include the pick-ups located within the five boroughs of NYC, namely: Manhattan, The Bronx, Queens, Brooklyn and Staten Island. A sample of the data set is presented in Table 4.1.

For example if the time period is selected as 120 minutes and the specified time is 15:00 on 1 April, 2014, then the locations of the pick-ups which occurred between 13:00 and 15:00 on 1 April, 2014 are selected to form part of the data set subjected to the clustering procedure.

**Number of clusters:** The number of clusters is selected as a percentage of the number of pick-up locations present in the selected data set. An upper bound is set on the number of

TABLE 4.1: A sample of the data set employed. The pick-up times and locations of riders during April 2014 is indicated.

Pick-up date and time	Pick-up latitude	Pick-up longitude
2014/04/01 09:00:00 AM	40.7841	-73.9542
2014/04/01 09:02:00 AM	40.7426	-73.9963
⋮	⋮	⋮
2014/04/30 11:59:00 PM	40.7340	-74.0052
2014/04/30 11:59:00 PM	40.7302	-74.0008

clusters so as to ensure that the number of parking areas is not exceeded. If 300 parking areas exist, for example, with a cluster percentage of 10% and 4000 pick-ups then 400 clusters are to be considered, however, the number of clusters evidently exceeds the number of parking areas, therefore the number of clusters considered is readjusted to 300.

**Initialisation method.** The initialisation method can be set to  $K$ -means++, random or specified according to a user-defined array. The first two methods are discussed in §2.3.2. If the array method is selected, the user presents an array containing the latitude and longitude of the initial cluster centres — the number of rows and the number of columns correspond to the number of clusters and features (*i.e.* latitude and longitude), respectively. In the case of real-time clustering, it is preferred to use the cluster centres found during the most recent clustering procedure. This, however, cannot be performed during the first iteration of clustering or if the cluster centres found during the most recent clustering contain a different number of clusters. In the case of all other instances (*i.e.* excluding the aforementioned exceptions), an array containing the cluster centres identified during the previous clustering procedure is used as the input, otherwise  $K$ -means++ is used as the initialisation method.

#### 4.4.2 Output analysis and verification

The resulting cluster centres produced by the  $K$ -means clustering algorithm are snapped to the nearest parking area so as to ensure that it is practically possible for drivers to wait at these locations. This process is illustrated in Figure 4.4 in which a small area in Manhattan, NYC is considered. The cluster centres are illustrated using the 🌟 icon and the parking areas are illustrated using the purple polygons. In Figure 4.4(a), the locations of the cluster centres found before they are snapped to the nearest parking area as well as all the parking areas in the included area are illustrated. In Figure 4.4(b), an arrow is drawn so as to indicate the nearest parking area with respect to each cluster area. Lastly, in Figure 4.4(c), the cluster centres are snapped to the nearest parking areas.

The data set considered in this thesis (a sample of which is presented in Table 4.1) is clustered every 10 minutes, starting at 00:00 on Tuesday 2 April, 2014 and ending at 00:00 on Wednesday 3 April, 2014. This procedure is performed for clustering percentages of 10% and 20%, and clustering durations of 90 minutes and 180 minutes — therefore, four clustering experiments are considered, *i.e.* 10% and 90 minutes, 10% and 180 minutes, 20% and 90 minutes, and 20% and 180 minutes. These four clustering experiments are summarised in Table 4.2.

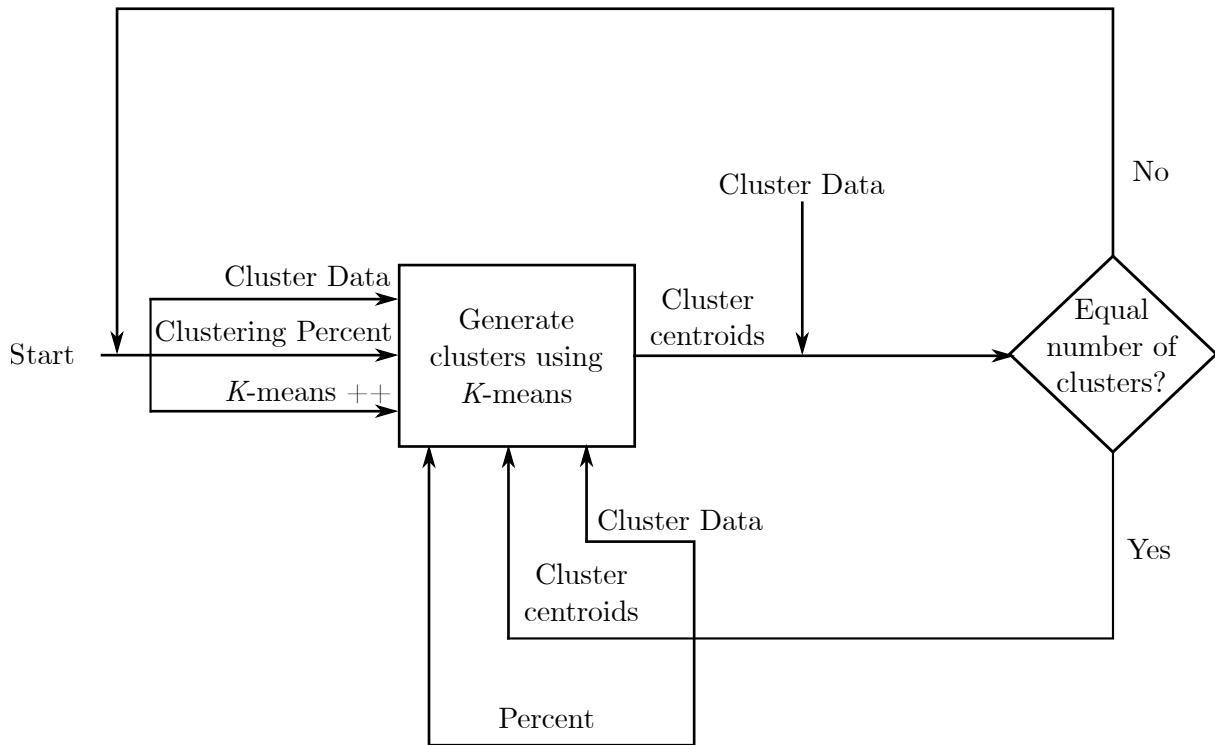
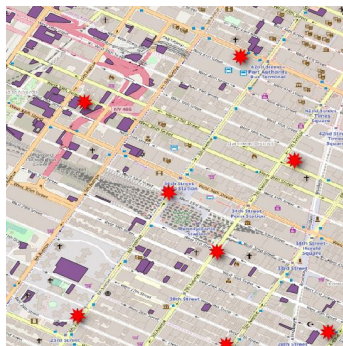
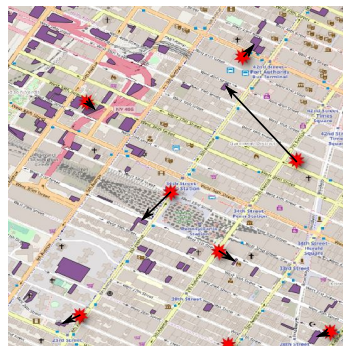


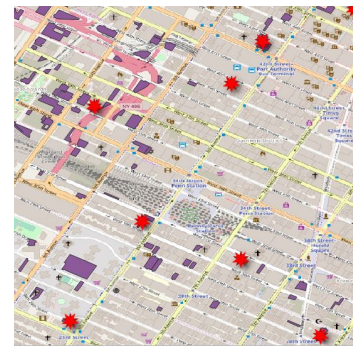
FIGURE 4.3: A graphical representation of the clustering process.



(a) Before snapping to parking area



(b) Indication of the nearest parking area



(c) After snapping to parking area

FIGURE 4.4: The process of snapping the cluster centres produced to the nearest parking area.

TABLE 4.2: The experiments conducted for verification of cluster type.

Experiment	Clustering percentage	Time period of data clustered
Cluster type 1	10%	90 minutes
Cluster type 2	10%	180 minutes
Cluster type 3	20%	90 minutes
Cluster type 4	20%	180 minutes

The number of clusters discovered for each of these experiments are illustrated in Figure 4.5. The maximum number of clusters is set to 229 in respect of all the experiments, as can be seen in each of the graphs — this decision is ascribed to the fact that there are 229 parking areas in the considered area. In Figure 4.5(a), it can be observed that the number of clusters is lower

than that of the other experiments. This is somewhat expected as the percentage and time period are set to correspond to the lowest combination of inputs. Peaks in the number of clusters can also be observed around 08:00 and 18:00, attributable to an increase in number of trips corresponding to rush-hour traffic. A large decrease can also be observed around 03:00 which is an expected outcome in the early hours of the morning. In Figure 4.5(b), a similar pattern is observed to that of Figure 4.5(a) although the number of clusters has increased. This is expected because the time period has doubled. The graph in Figure 4.5(c) exhibits a distribution that is almost identical to that of Figure 4.5(a) except that the number of clusters has doubled and, as a result, the number of clusters reaches the maximum between 08:00 and 10:00 and 16:00 and 22:00. Lastly, in Figure 4.5(d), the percentage and time period correspond to the highest combination of input values. Consequently, the only time the number of clusters is not at its maximum is between 01:00 and 06:00 coinciding with the same early-morning “dip” in pick-up requests — an observation present in the other three graphs.

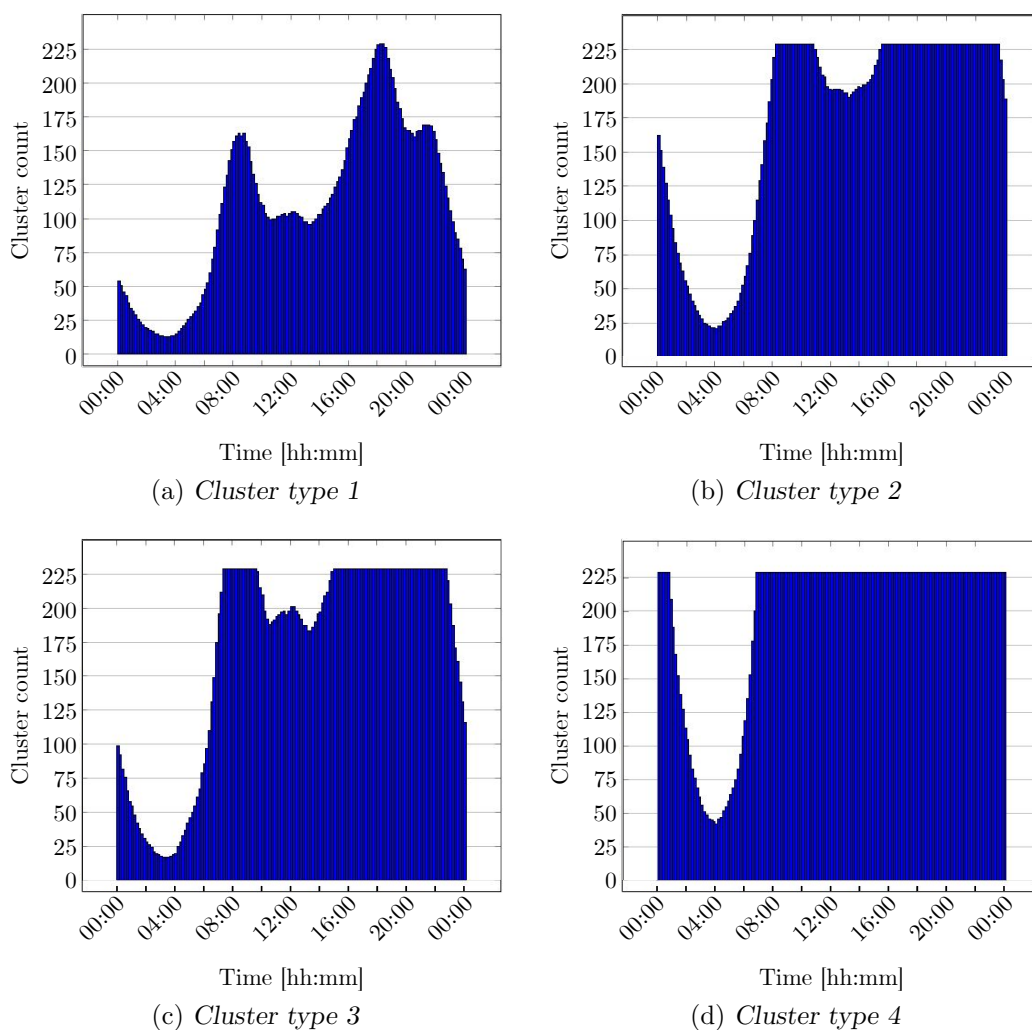



FIGURE 4.5: Number of clusters observed in the clustering experiments.

## 4.5 The simulation experiment graphical user interface

A *graphical user interface* (GUI) is developed so as to assist with carrying out of experiments. The GUI is used as part the initialisation of the experimental procedure which provides an intuitive method for the user to establish the exact specification of the experiment to be conducted. The construction of the GUI is completed in the  **Simulation** environment in AnyLogic. The variables declared so as to store the user input from the GUI are shown in Figure 4.6.

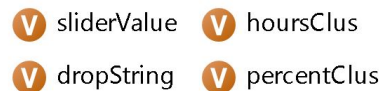







FIGURE 4.6: The variables declared in the Simulation environment.

The initial screen of the GUI is shown in Figure 4.8. The various components labelled in the figure are as follows:

- (a) **Number of drivers:** There are four agent types defined in this model, one of which is the  **drivers** agent type. The number of driver agents to form part of the experiment is set to an initial value of 750 but can be changed to any value ranging from 0 to 1500, which can be accomplished by either using the slider or the text box under the slider. These two objects work in conjunction, therefore when the slider is moved, the value in the text box changes accordingly and *vice versa*. This value is stored in  **sliderValue**.
- (b) **Waiting area type:** The two waiting areas are defined as two agent types, namely:  **parkingAreas** agent type and  **clusteredWaitingAreas** agent type, both of which are discussed in detail later in the chapter. The user can decide which waiting area type should be included in the experiment using the “combination box”. The combination box is set to initially display “Clustered Waiting Area”. If the desired experiment includes the parking areas, the user will select “Parking Area” from the combination box. Similarly if the desired experiment includes the clustered awaiting areas, the user will select “Clustered Waiting Area” from the combination box. The string value of the selected waiting area type is stored in  **dropString**. The expanded combination box is presented in Figure 4.7.

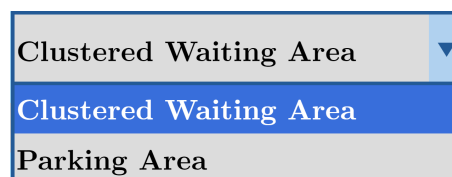


FIGURE 4.7: The expanded combination box used to select the waiting area type for the desired experiment.

- (c) **Confirm waiting area type:** Once the user has selected the number of driver agents and the waiting area type, the “Confirm Waiting Area Type” button should be selected.

If the user selects “Clustered Waiting Area” from the combination box labelled (b) and thereafter selects the “Confirm Waiting Area Type” button labelled (c), the GUI then changes accordingly and reflects the inputs necessary for the initialisation, as shown in Figure 4.12. The additional components (d)–(f), as labelled in the figure, are as follows:

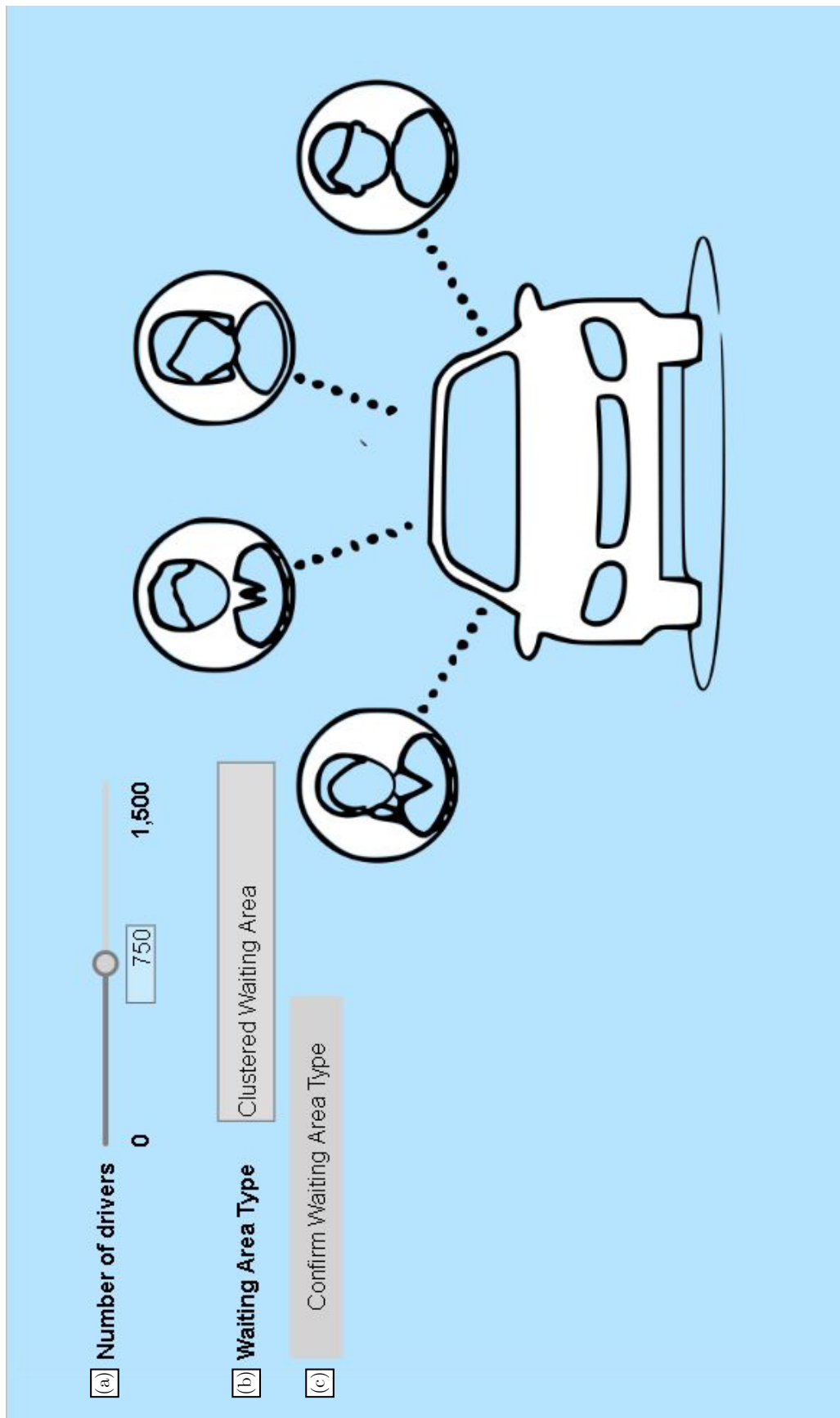


FIGURE 4.8: A screenshot of the initial view of the GUI.

- (d) **Percent:** The user can select the number of clusters as either 10%, 20% or 30% percent of the data set using the combination box. The combination box is set to initially display “10”. The selected percentage value is stored in **percentClus**. The expanded combination box is shown in Figure 4.9.

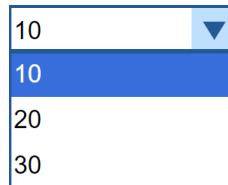


FIGURE 4.9: The expanded combination box used to select the number of clusters as a percentage of the data set.

- (e) **Hours:** The user can select the time period of historical data to cluster using the combination box. The user can select to cluster one hour and thirty minutes, three hours or four hours and thirty minutes of historical data which is displayed in the combination box as “1h30”, “3h” and “4h30”, respectively. The combination box is set to initially display “1h30”. The selected time period value is stored in **hoursClus**. The expanded combination box is shown in Figure 4.10.

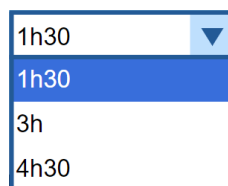


FIGURE 4.10: The expanded combination box used to select the time period of historical data to be clustered at any time.

- (f) **Confirm Cluster Type:** Once the user has selected the percent and the time period, the “Confirm Cluster Type” button should be selected.

Once the user selects the percent and hours using the combination boxes (d) and (e), respectively, which is followed by the selection of the “Confirm Cluster Type” button labelled (f), the GUI then displays a message indicating the user can now run the experiment, as shown in Figure 4.11. Another component, labelled (g), is added to the GUI which is described as:

- (g) **Run:** Once the user has completed all the necessary inputs for components (a)–(f) and is satisfied with the input, the user is prompted to run the experiment. This may be done by selecting the “Run” button. Thereafter, the main agent is displayed.

If the user selects “Parking Area” from the combination box labelled (b) and thereafter selects the “Confirm Waiting Area Type” button (c), the GUI then displays a message indicating the user can now run the experiment (as shown in Figure 4.13) since no additional inputs are necessary. Once again, a component labelled (h) is added to the GUI which is described as:

- (h) **Run:** Once the user has completed all the necessary inputs for components (a)–(c) and is satisfied with the input, the user is prompted to run the experiment. This can be performed by selecting the “Run” button. Thereafter, the main agent is displayed.



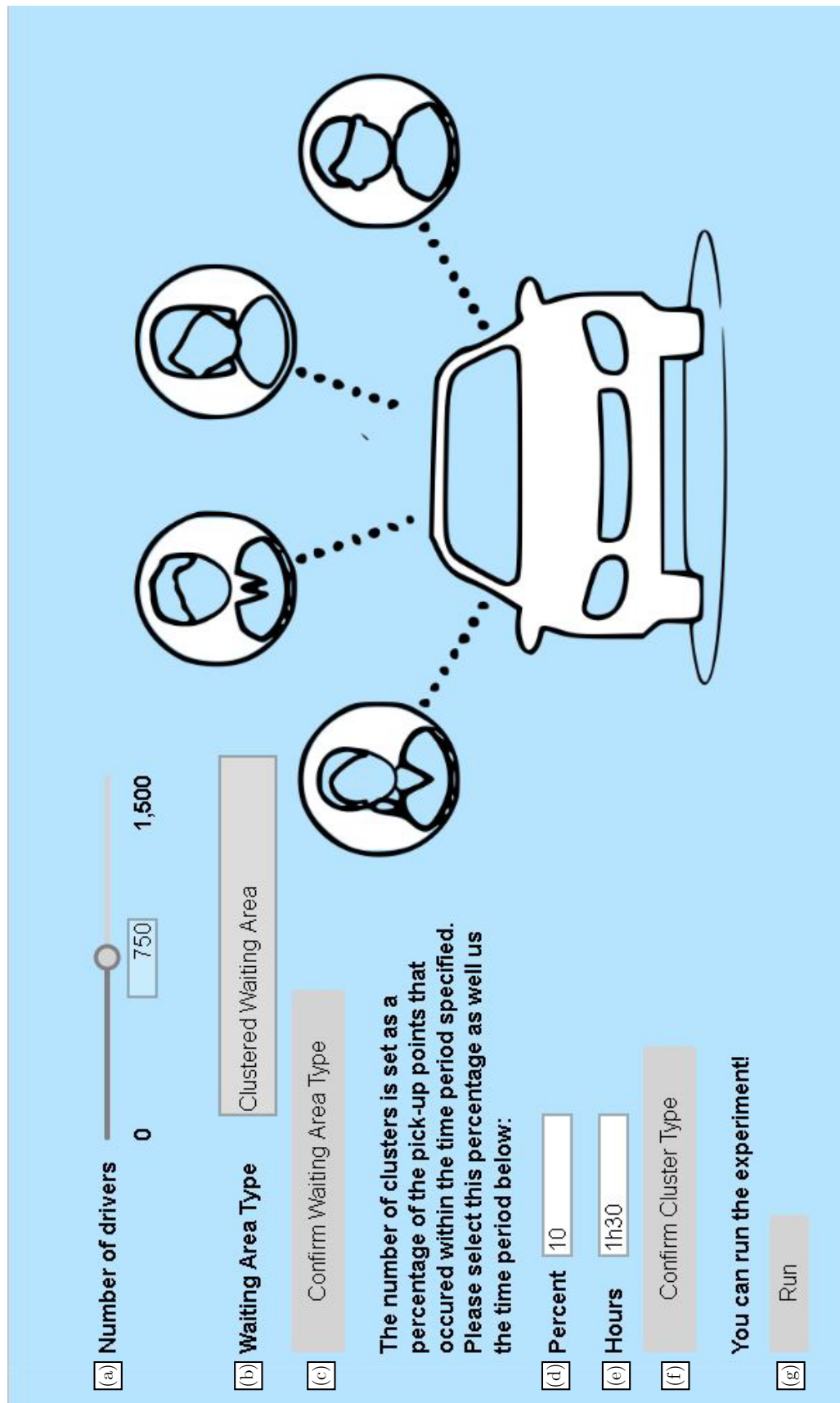


FIGURE 4.11: A screenshot of the GUI with the selection of clustered waiting areas as a waiting area type together with the selection of the percentage and hours.



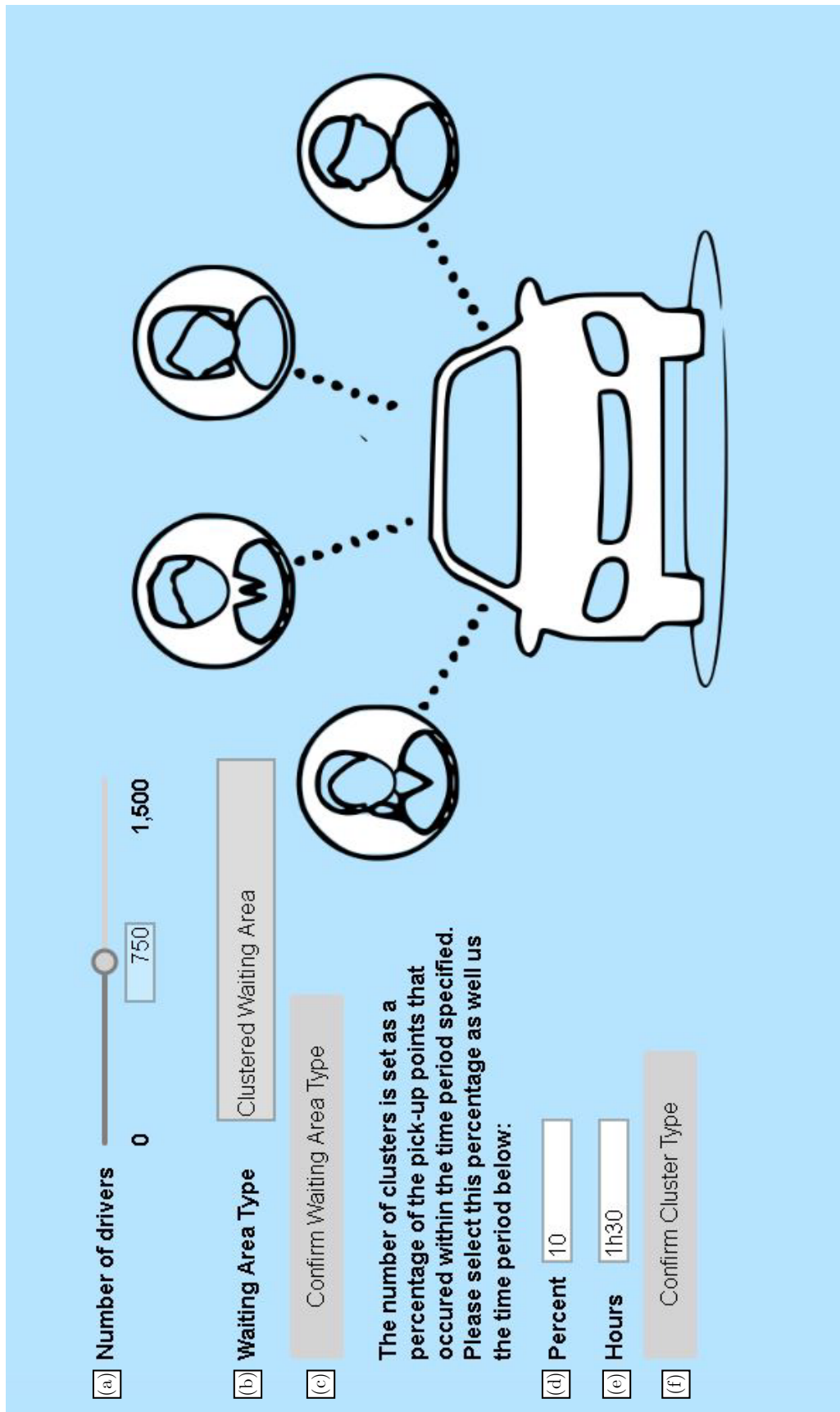


FIGURE 4.12: A screenshot of the GUI with the selection of clustered waiting area as a waiting area type.

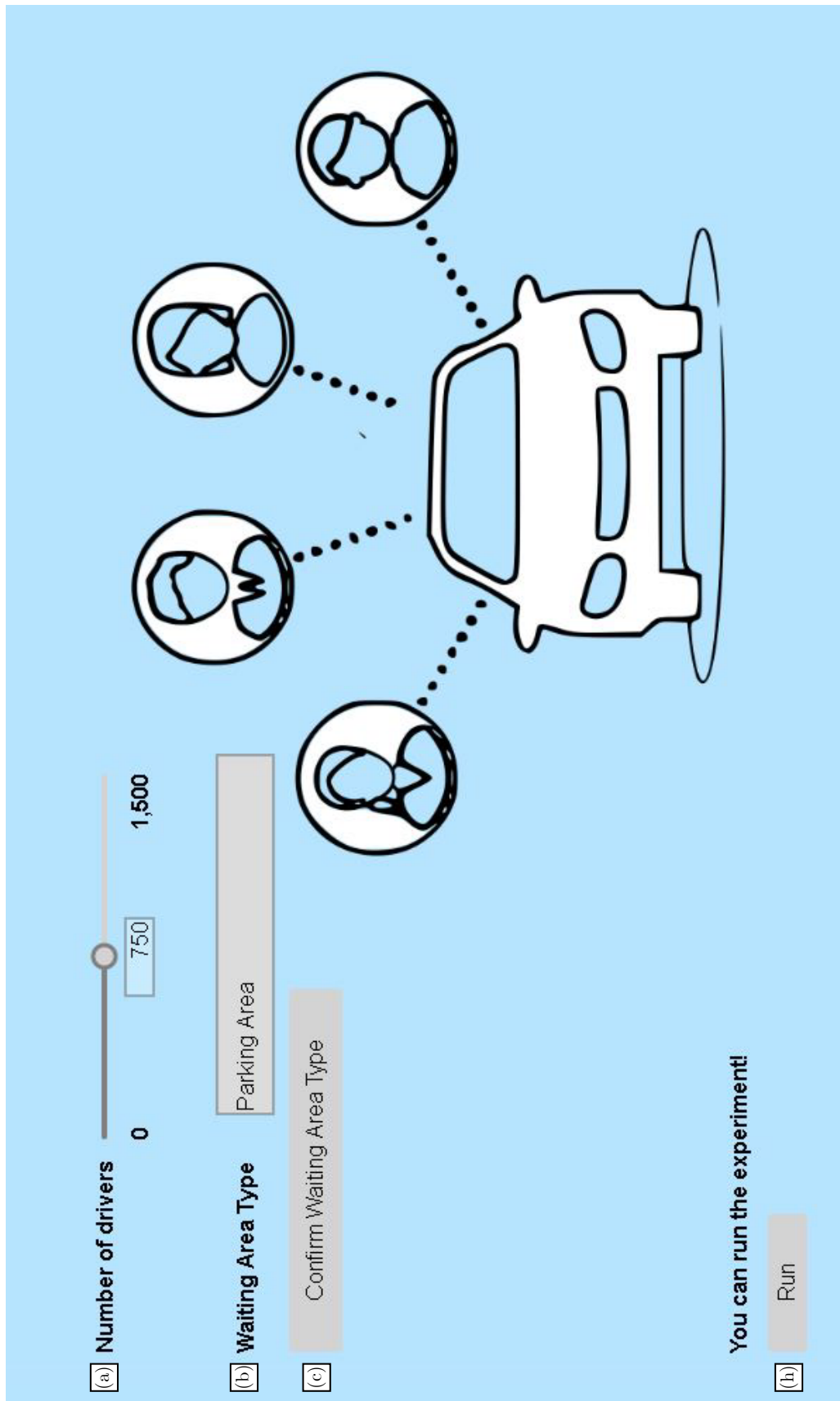











FIGURE 4.13: A screenshot of the GUI with the selection of parking area as a waiting area type.

### 4.5.1 The main environment

The simulation environment is constructed in the  **Main** agent type. This agent type is the top-agent type and contains the other agents in the model. The content of the main agent type is shown to the user during the simulation. The contents of other agent types, such as the statechart, variables and parameters, to name a few, are not shown initially, however, they can be displayed if required. There are five components in the main agent type (as shown in Figure 4.15) and the declarations in main relating to these components are shown in Figure 4.14. The variables in the **Simulation** environment (shown in Figure 4.6) are linked to parameters in main. The  **sliderValue** is linked to  **numOfDrivers**,  **dropString** to  **optionString**,  **percentClus** to  **perc** and lastly  **hoursClus** is linked to  **hours**. These components included in the main agent are as follows:


























<u>Input</u>	<u>Rider Statistics</u>		
 perc	 getStat		
 hours	 minTotal	 minAssign	 totalTimeRider
 NumOfDrivers	 averageTotal	 averageAssign	 assignTimeRider
 optionString	 maxTotal	 maxAssign	

FIGURE 4.14: Declarations inside the Main agent corresponding to the five components.

- Total waiting time:** This graph indicates the minimum, average and maximum *total waiting time* (TWT) experienced by riders. The TWT is the time elapsed from when a rider requests a ride until when they are picked-up. On the *x*-axis of the graph is the simulation time in seconds which corresponds to the time elapsed (in seconds) in the simulation day, *i.e.* if the simulation starts at 12:00 AM and 18 000 seconds have elapsed, it is therefore currently 5:00 AM in the simulation. Every time a rider is picked-up by a driver, the rider's TWT is added to  **totalTimeRider**. The  **getStat** event occurs every minute. When this event is activated, the minimum, average and maximum TWT values are calculated from  **totalTimeRider** and these values are stored in  **minTotal**,  **aveTotal** and  **maxTotal**, respectively.
- Assignment waiting time:** This graph indicates the minimum, average and maximum *assignment waiting time* (AWT) experienced by riders. The AWT is the time elapsed from when a rider requests a ride until when they are paired with a driver. On the *x*-axis of the graph is the simulation time in seconds. Every time a rider is paired with a driver, the rider's AWT is added to  **assignTimeRider**. The  **getStat** event is also used to retrieve the minimum, average and maximum AWT values from  **assignTimeRider** which are stored in  **minAssign**,  **aveAssign** and  **maxAssign**, respectively.
- Number of riders:** The graph tracks the number of rider agents currently in the system as well as the number of rider agents currently waiting to be picked-up by a driver. The number of riders in the system can be defined as riders which are either waiting to be picked-up or are being transported to their destination. Therefore, the riders enter the system when they request a ride and leave the system when they are dropped-off. Since riders request a ride as soon as they are added to the simulation, the number of drivers in the system is recorded using the AnyLogic function *riders.size()*. The process of adding and removing rider agents is discussed later in this chapter. The corresponding graph

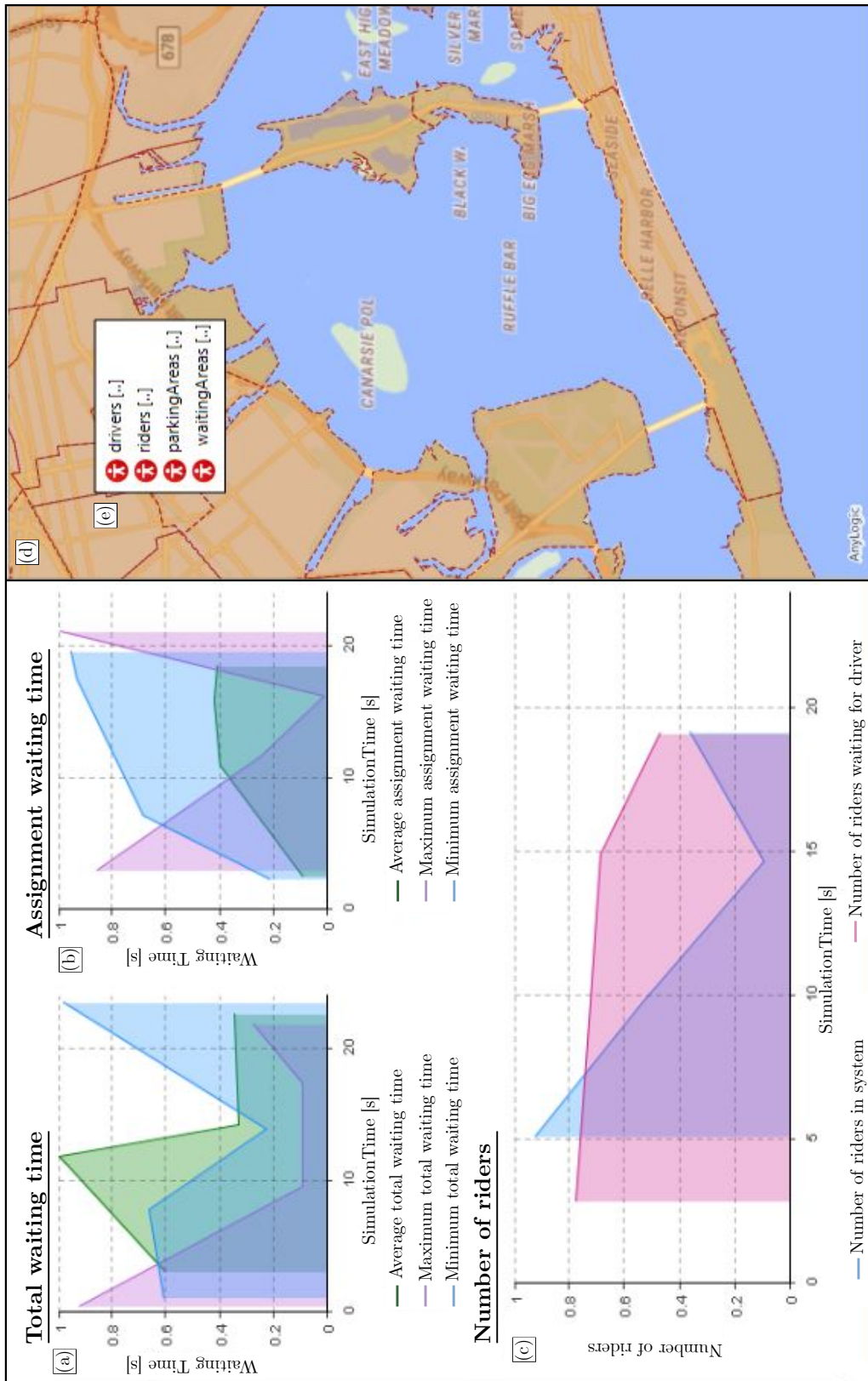






FIGURE 4.15: A screenshot of the main agent.



can be used to observe the demand pattern and reflects the availability of drivers. The simulation time is observed on the  $x$ -axis.

- (d) **GIS:** The GIS map displays the locations (*i.e.* latitude and longitude) of parking areas, riders and clustered waiting areas. It also demonstrates the movement of drivers. The GIS map enables the routing of the drivers along existing roads and a travel time to be assigned to these routes which allows for traffic to be taken into account. The polygons represent the various neighbourhoods of NYC. These polygons are not visible during a simulation run in order to simplify the map as these polygons are not necessary for the visualisation of the simulation run.
- (e) **Agent types:** The four agent types, which are  **drivers[.]** agent type,  **riders[.]** agent type,  **clusteredWaitingArea[.]** agent type and  **parkingArea[.]** agent type, are shown.

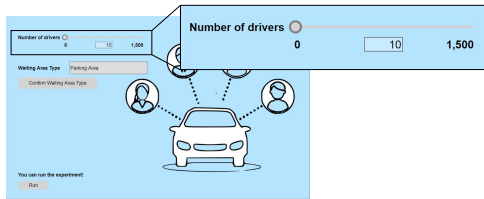
#### 4.5.2 Functionality assessment of the GUI

The functionality of the GUI is assessed by changing the inputs of the GUI (*i.e.* number of drivers, waiting area type and the cluster type) running the model and observing the various components of the model.

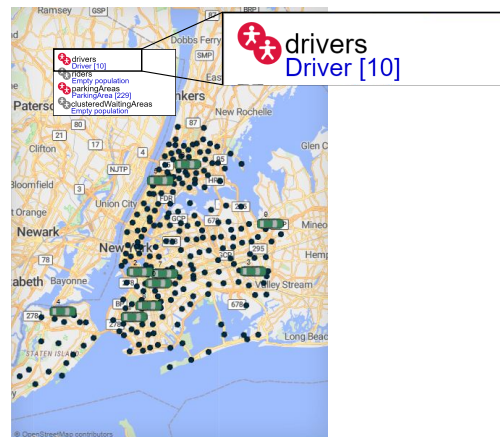
In order to test the operation of the components used to input the number of drivers, labelled (a) in Figure 4.15, the number of drivers is set to 10, 100 and 1 000. The waiting area type is set to a constant input corresponding to “Parking area”. The GUI inputs and the corresponding results of these three tests are shown in Figure 4.16. The resulting outcome is not only shown visually by the number of car icons present, but also by the agent type indicators labelled (e) in Figure 4.15. For an input of 10 drivers, shown in Figure 4.16(a), the start of the simulation run is shown in Figure 4.16(b) where 10 car icons are observed and the agent type indicates Driver[10]. Once again, for an input of 100 drivers, shown in Figure 4.16(c), the start of the simulation run is shown in Figure 4.16(d) where more car icons are observed than in Figure 4.16(b) and the agent type indicates Driver[100]. Lastly, Figure 4.16(e) presents the GUI for an input of 1 000 drivers and in the corresponding simulation run, shown in Figure 4.16(f), more car icons are observed than in Figure 4.16(d) and the agent type correctly indicates Drivers[1000].

The testing of the components used to select the parking area type is performed by changing the input of the GUI and viewing the map and the agent type indicators. The number of vehicles are set to zero so as to better visualise the waiting areas. In the simulation, parking areas are illustrated with black dots • and clustered waiting areas are illustrated with blue clouds ☁. The agent type indicators describe the waiting area which is not chosen as an empty population. The GUI inputs and the corresponding results are shown in Figure 4.17. If “Parking Area” is selected as the input, as shown in Figure 4.17(a), the start of the simulation run is observed in Figure 4.17(b). The ParkingArea[229] indicates that there are parking area agents in the simulation whereas  **clusteredWaitingAreas** is described as an empty population. Furthermore, there are only black dots, which symbolise the parking areas, and no blue clouds. Similarly, if “Clustered Waiting Area” is selected as an input, shown in Figure 4.17(c), the start of the simulation run can be observed in Figure 4.17(d). In this figure, the cluster type is set to 10% and 90 minutes — these are the default options. The ClusteredWaitingArea[55] indicates that there are clustered waiting area agents in the simulation whereas  **parkingAreas** is described as an empty population. Only blue clouds are shown in this simulation run, further verifying the GUI selection of the “Clustered Waiting Area” as a waiting area type.

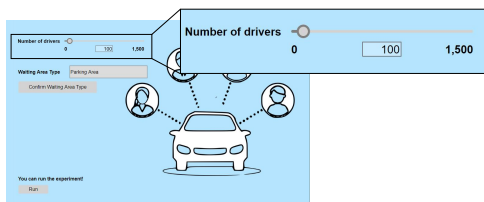




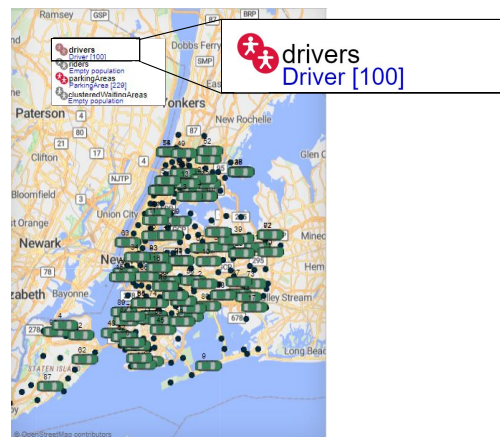
(a) A screenshot of the GUI with 10 drivers selected as an input



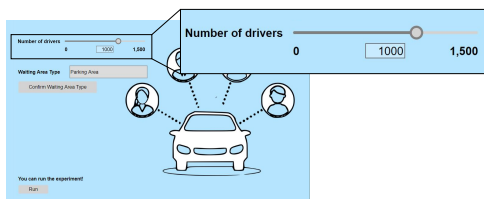
(b) Start of a simulation run with 10 drivers selected as an input



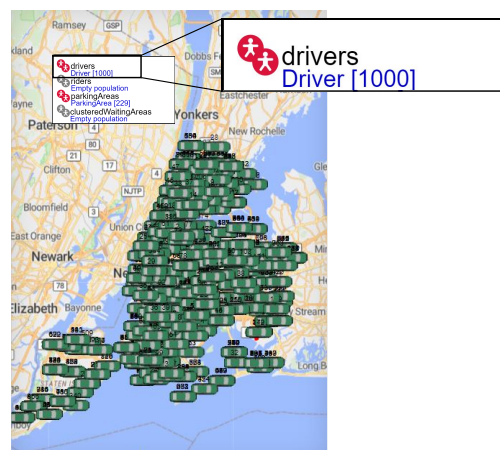
(c) A screenshot of the GUI with 100 drivers selected as an input



(d) Start of a simulation run with 100 drivers selected as an input



(e) A screenshot of the GUI with 1000 drivers selected as an input



(f) Start of a simulation run with 1000 drivers selected as an input

FIGURE 4.16: A demonstration of the functionality of the components used to input the number of drivers of the GUI.



FIGURE 4.17: A demonstration of the functionality of the components used to input the waiting area type desired.

The final functionality assessment performed relates to the testing of the GUI components used to input the cluster type. This is done by comparing two cluster types *i.e.* 10% and 90 minutes along with 20% and 180 minutes and comparing the number of clustered waiting areas to be observed in Figures 4.5(a) and 4.5(d). Accordingly, when the simulation is set to a cluster type of 10% and 90 minutes, there should be approximately 55 clustered waiting areas visible at the start of the simulation and the maximum number of clustered waiting areas observed throughout the simulation should be 244. Similarly, when the simulation is set to a cluster type of 10% and 90 minutes, there should be 229 clustered waiting areas visible at the start of the simulation and the maximum clustered waiting areas observed throughout the simulation is also 229. The inputs and corresponding results of this test are shown in Figure 4.18. Once again, the number of drivers is set to zero so as to aid the visualisation of the clustered waiting areas. In Figure 4.18(a), inputs of 10% and 90 minutes are selected as the cluster type. The resulting initiation of the simulation is shown in Figure 4.18(b). In this figure, the `WaitingArea[224]` indicates the maximum number of clusters for the time period being simulated which corresponds to Figure 4.5(a). Similarly, for a cluster type input of 20% and 180 minutes, shown in Figure 4.18(c), the resulting initiation of the simulation is shown in Figure 4.18(d). In this figure, the `WaitingArea[229]` indicates the maximum number of clusters for the simulated time period which once again corresponds to

Figure 4.5(d). Furthermore, the number of clustered waiting areas observed in Figure 4.18(d) is considerably more than that of Figure 4.18(b), which is expected.

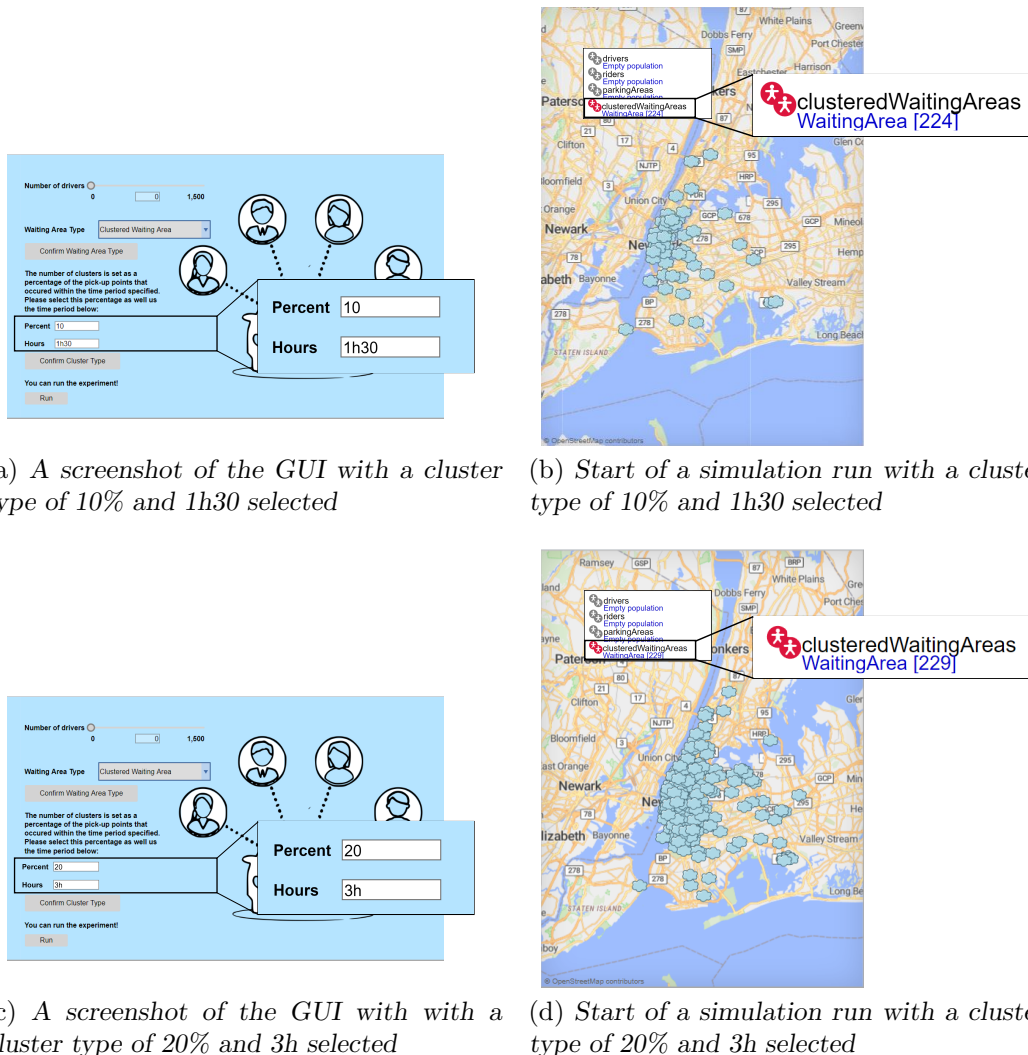



FIGURE 4.18: A demonstration of the functionality of the components used to input the cluster type desired.

## 4.6 The simulation model

In the simulation model, **+** **drivers** and **+** **riders** agent types facilitate the behaviours of the driver and rider, respectively. Furthermore, the agent types **+** **parkingAreas** and **+** **ClusteredWaitingAreas** model the behaviour of the two types of waiting areas — *i.e.* parking areas and clustered waiting areas, respectively. These four agents and the interactions between them are used to model the system described in Figure 4.1. Each of these agents and their behaviours are explained in detail hereafter.



### 4.6.1 Driver agent

The driver agent simulates the movement and behaviour of the drivers of TNCs. In the simulation, the driver agent is shown graphically using a car icon . The statechart of the driver agent is shown in Figure 4.19 and the associated components necessary to model the driver behaviour is shown in Figure 4.20.

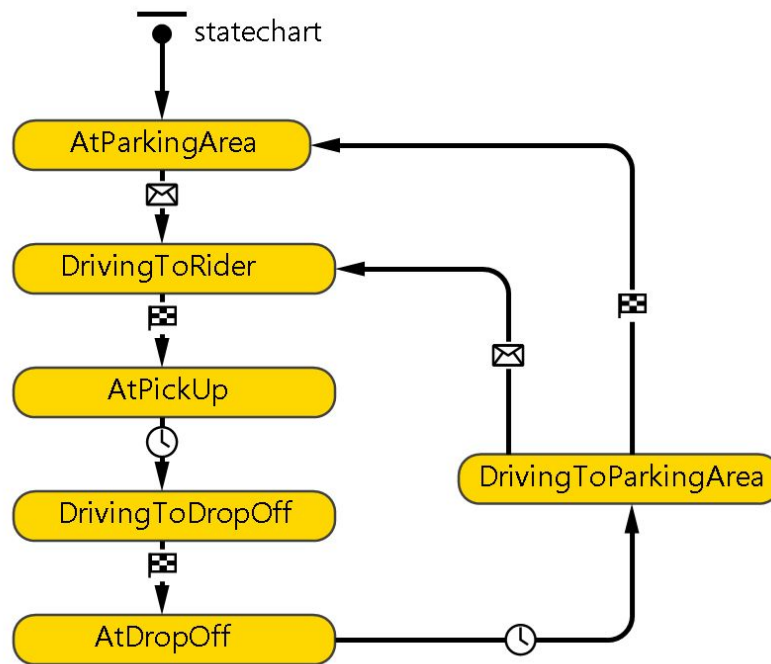


FIGURE 4.19: Screenshot of the driver agent's statechart.

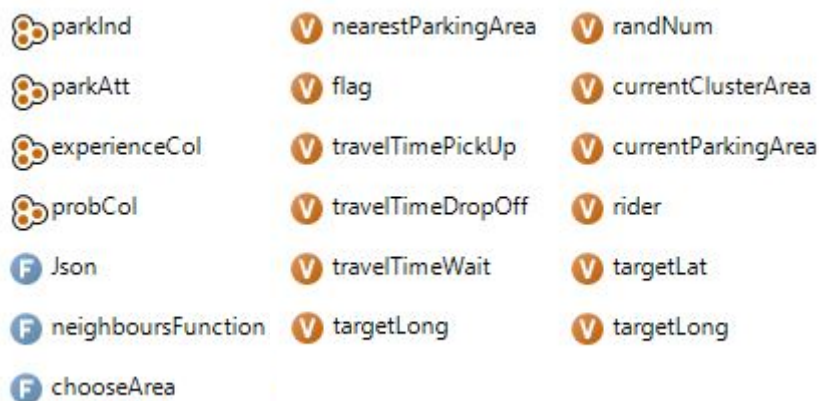
























FIGURE 4.20: Screenshot of the AnyLogic components used in conjunction with the statechart of the driver agent.

At the start of the simulation, the driver agents are assigned to waiting areas according to a uniform distribution. A driver agent enters the simulation and assumes the **AtParkingArea** state immediately. This state indicates that the agent is currently at a waiting area, *i.e.* a parking area or a clustered waiting area, waiting for a ride-request. The agent remains in the

**AtParkingArea** state until a message is received from a rider agent representing a ride-request, as indicated by the message transition trigger. If the driver agent is at a parking area (and not a clustered waiting area) upon receiving this message, the  **experienceCol** is amended so as to record that the driver agent has experienced a ride-request from the current parking area. The  **experienceCol** consists of 229 elements, one for each parking area, and the simulation begins with each of these elements equal to zero — the driver has not yet experienced any ride-requests. When a driver agent receives a ride-request, the index of the current parking area agent is retrieved and the element of  **experienceCol** corresponding to that index is incremented by 1.

The driver then transitions into the **DrivingToRider** state where the rider agent that sent the message is stored in the  **rider** variable and the rider agent's pick-up latitude and longitude are stored in  **targetLat** and  **targetLong**, respectively. The travel time from the driver agent's current location to the rider agent is then calculated using the  **Json** function. This function calculates the travel time using the Bing Maps API. This can be done by either using a origin-destination pair where the travel time from the origin to the destination is calculated, or by using an origin-waypoint-destination configuration where two travel times are calculated, *i.e.* from the origin to the waypoint and from the waypoint to the destination. If this function is called in the **DrivingToRider** state then the origin-waypoint-destination configuration is used where the origin is the driver's current location, the waypoint is the rider agent's pick-up location and the destination is the rider agent's drop-off location. This is done to limit the number of requests made to Bing Maps. If this request is successful, the  **flag** variable, which is a boolean variable, is set to **true** and the first travel time calculated, *i.e.* the travel time from the driver agent to the rider agent's pick-up location, is stored in the  **travelTimePickUp** variable and the second travel time, *i.e.* the travel time from the rider agent's pick-up location to the rider agent's drop-off location, is stored in the  **travelTimeDropOff** variable. However, if the request is not successful, the  **flag** is set to **false** and the origin-destination pair is used instead where the origin is the driver agent's location and the destination is the rider agent's pick-up location. The travel time calculated is then stored in the  **travelTimePickUp** variable. This is performed as a precautionary measure to ensure that no errors occur during a simulation run.

AnyLogic's *moveToInTime()* function is used to ensure that the driver agent reaches the rider agent in the time allocated by the  **travelTimePickUp** variable. The driver agent then moves towards the location of the rider agent that sent the message. An arrival transition is then used to transition the driver agent into the **AtPickUp** state upon arrival at the rider agent. The driver agent sends a message to the rider agent indicating the rider agent is being picked-up. A timeout transition trigger then transitions the driver agent into the **DrivingToDropOff** state to simulate the time it takes for a rider to enter the vehicle. Upon entering the **DrivingToDropOff** state, the driver processes the latitude and longitude of the drop-off location from the rider agent which is stored in  **targetLat** and  **targetLong**, respectively. The value of  **flag** is then retrieved. If  **flag** is true, the *moveToInTime()* function is used once again to ensure that the driver agent arrives at the rider agent's drop-off location according to the time allocated by the  **travelTimeDropOff** variable. If  **flag** is false, the  **Json** function is used with a simple origin-destination pair where the origin is the rider agent's pick-up location and the destination is the rider agent's drop-off location. The travel time calculated is then stored in the  **travelTimeDropOff** variable. Similar to before, the *moveToInTime()* function is used to ensure that the driver agent arrives at the rider agent's drop-off location in the time allocated by the  **travelTimeDropOff** variable.

Once again, an arrival transition trigger is employed in order to transition the driver agent into the **AtDropOff** state upon arrival at the drop-off location. When the agent arrives at the drop-off

location a message is sent to the rider agent indicating the rider agent is being dropped-off. A timeout trigger transitions the driver agent into the **DrivingToParkingArea** state to simulate the time it takes for a rider to exit the driver's vehicle. If an input of clustered waiting areas is selected on the GUI, the driver agent selects the nearest clustered waiting area agent, as discussed in §4.4, and this agent is stored in the **V currentClusterArea** variable. The latitude and longitude of this clustered waiting area agent is stored in **V targetLat** and **V targetLong**, respectively. Otherwise, if an input of parking areas is selected then the driver agent selects a parking area agent according to the selection strategy discussed in §4.3. This is performed upon execution of the **F neighboursFunction**. This function finds the nearest parking area agent to the driver and stores this parking area agent in the **V nearestParkingArea** and retrieves all the parking area agents in the surrounding neighbourhoods. It then iterates over these parking area agents and checks whether they are hotspots. If the agent is not a hotspot, the attractiveness of the area is calculated using equation (4.1) whereas if the agent is a hotspot, the attractiveness of the area is calculated using equation (4.3). The calculated attractiveness value is then added to the **parkAtt** collection and the index of the parking area agent is added to the **parkInd** collection. The sum of the **parkAtt** collection is then calculated. If the sum is greater than zero, *i.e.* the driver has received a pick-up request from at least one of these parking areas before, the collection is iterated over and the probability associated with each parking area is calculated using equation (4.2) and the probability is added to the **probCol** collection. The **F neighboursFunction** is then concluded.

A check is performed thereafter to observe if the **probCol** collection is empty. If the collection is empty, all the parking area agents have an attractiveness of zero. Therefore, the nearest parking area agent to the driver is found and stored in **V currentParkingArea** and the latitude and longitude of this parking area agent are stored in **V targetLat** and **V targetLong**, respectively. If the collection is not empty, the **F chooseArea** function is called. This function generates a random number drawn from a uniform distribution, which is stored in **V randNum**, and selects a parking area agent based on the probability calculated in the **probCol** collection and the corresponding random number generated. The selected parking area agent is then stored in **V currentParkingArea** and the latitude and longitude of this parking area agent are stored in **V targetLat** and **V targetLong**, respectively. The three collections, *i.e.* **parkInd**, **parkAtt** and **probCol**, are then cleared.

When the driver agent is in the **DrivingToParkingArea** state, the agent moves towards a waiting area. The **F Json** function is again called and the simple origin-destination pair is used where the origin is the driver agent's current location and the destination is the chosen waiting area agent's location, *i.e.* either **V currentParkingArea** or **V currentClusterArea**. The travel time retrieved is then stored in the **V travelTimeWait** variable. The `moveToInTime()` function is used once again to ensure that the driver agent arrives at the waiting area agent according to the time allocated by the **V travelTimeWait** variable. The driver agent may leave this state by means of two transitions. The first transition is triggered upon receipt of a message by a rider agent indicating a ride-request. A driver receiving a ride-request whilst driving to a waiting area is therefore simulated. When a driver agent receives this message it immediately transitions into the **DrivingToRider** state and the agent continues to move through the states as discussed previously. If the agent arrives at the waiting area without being sent a ride-request, the arrival trigger activates and the agent transitions into the **AtParkingArea** state and the agent continues to move through the states as discussed previously.

### 4.6.2 Rider agent























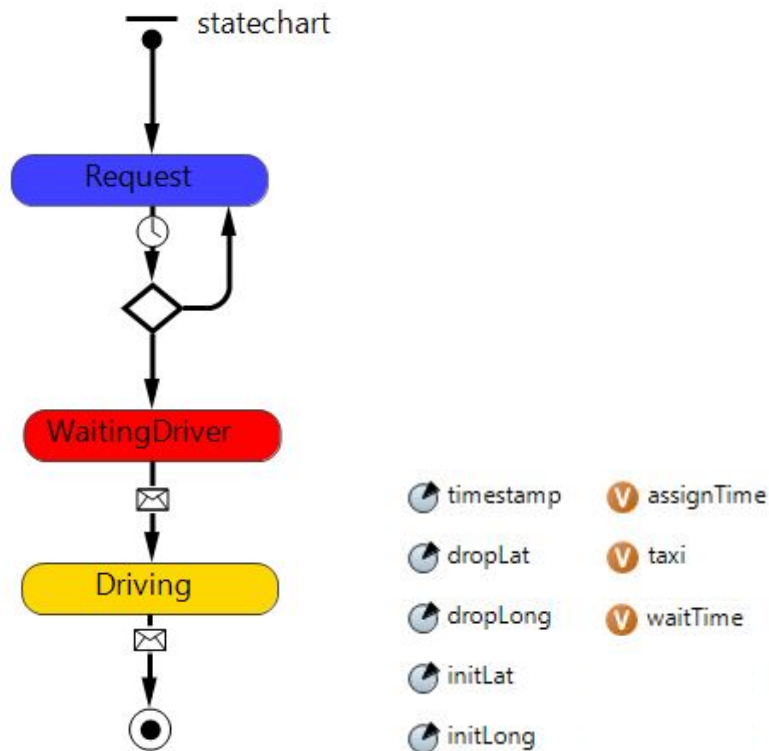
The modelling of the behaviour of riders (*i.e.* passengers) of TNC is performed utilising the rider agent. During a simulation run, the rider is shown graphically using a silhouette of a person . The rider's details are modelled according to a data set. This data set contains the location and time of pick-ups of a well-known TNC which occurred during the first week of April 2014 and is obtained from [46]. The drop-off locations, however, are not included in the obtained data set. To account for this, the drop-off locations are sampled from another data set describing the pick-ups and drop-off locations of taxi services in NYC which is obtained from [68]. This is done to ensure that the drop-off locations accurately represent potential destinations. A sample of the data set used is shown in Table 4.3.

TABLE 4.3: A sample of the data set indicating pick-up and drop-off times and locations of riders in the first week of April 2014 in NYC.

Pick-up date and time	Pick-up latitude	Pick-up longitude	Drop-off latitude	Drop-off longitude
2014/04/01 09:00:00 AM	40.7841	-73.9542	40.67358	-73.80718
2014/04/01 09:02:00 AM	40.7426	-73.9963	40.73845	-73.73162
⋮	⋮	⋮	⋮	⋮
2014/04/07 11:49:00 PM	40.7291	-73.9962	40.75862	-73.9888
2014/04/07 11:50:00 PM	40.6975	-73.9348	40.76834	-73.9867

The rider agents are generated using the components in main shown in Figure 4.22. The  **time**,  **latp**,  **lonp**,  **latd** and  **lond** collections contain the pick-up request times, pick-up latitudes, pick-up longitudes, drop-off latitudes and drop-off longitudes, respectively, of all the ride-requests which occurred in the specified time period. The  **addRider** event is a cyclic event which occurs every minute. When this event occurs, all the riders agents who made ride-requests in the previous minute are added to the simulation. The  **ind** contains the indices of the rider agents that were last generated. This is performed so as to prevent duplication of agents. The statechart of the rider agent, shown in Figure 4.21(a), and the associated components, shown in Figure 4.21(b), are necessary to model the rider behaviour.

When a rider agent is added to the simulation, its parameters are set according to the respective elements in the collections shown in Figure 4.22. The  **timestamp** parameter is retrieved from the  **time** collection, the  **dropLat** parameter is retrieved from the  **latd** collection, the  **dropLon** parameter is retrieved from the  **lond** collection, the  **initLat** parameter is retrieved from the  **latp** collection, and lastly the  **initLong** parameter is retrieved from the  **lonp** collection. A rider agent enters the simulation and immediately assumes the **Request** state, as can be seen in Figure 4.21(a). Upon entering this state, the agent silhouette is coloured blue. The nearest available driver agent is found and stored in the  **taxi** variable and the rider agent sends a message to this driver agent representing a ride-request. A driver agent is available if it is in the **AtParkingArea** state or in the **DrivingToParkingArea** state. If an available driver agent cannot be found, the rider agent re-enters the **Request** and continues searching for an available driver agent every second until one is found. Upon sending this message, the rider agent transitions to the **WaitingDriver** state and the agent's silhouette changes to a red colour. The  **assignTime**, which stores the assignment time experienced by the rider agent, is then calculated as the  **timestamp** value subtracted from the current simulation time. This value is then added to the  **assignTimeRider** collection in main. The rider agent remains in this state until a message is received by the driver agent indicating the driver has arrived at



(a) Screenshot of the rider agent's statechart.

(b) Screenshot of the AnyLogic components used in conjunction with the statechart of the rider agent.

FIGURE 4.21: Inner workings of the rider agent.

the pick-up location. The rider then transitions into the **Driving** state and the **waitTime**, which stores the total waiting time experienced by the rider agent, is then calculated as the **timestamp** value subtracted from the current simulation time. When the rider agent enters this state, it is no longer visible in the simulation as the rider is regarded as inside the driver's vehicle. The rider agent then transitions into the **FinalState**, indicated in the statechart with the **FinalState** icon, when a message is received by the driver agent indicating the agent has reached the rider agent's drop-off location. Upon entering the **FinalState**, the rider agent is removed from the simulation.

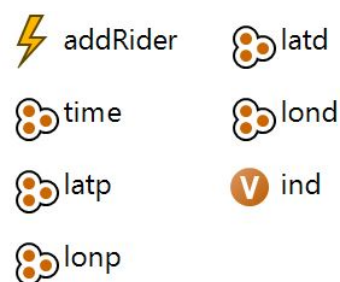


FIGURE 4.22: Screenshot of the AnyLogic components used in the main agent for the generation of rider agents.

### 4.6.3 Parking area agent

As stated previously, the parking area agent is only active in the simulation if the user selected “Parking Area” as the desired waiting area type. If the parking area agent is active, each agent is represented graphically during the simulation run by a block dot ●. A screenshot of the statechart together with the components necessary to simulate the behaviour of the parking areas are shown in Figure 4.23.

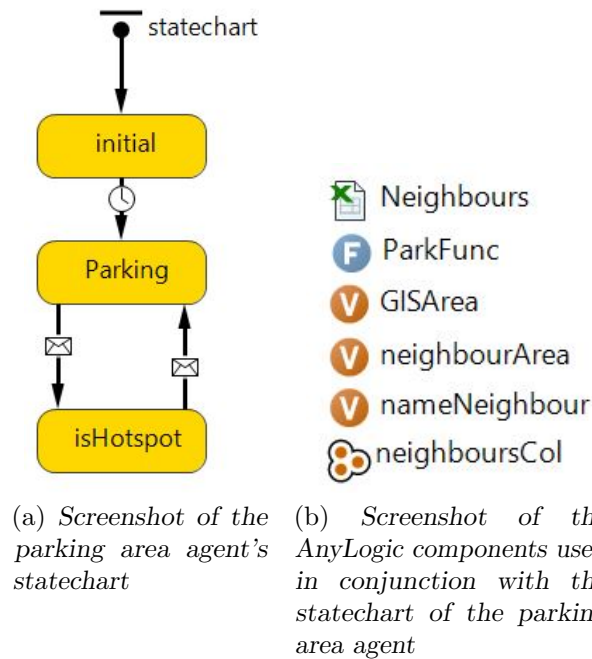




FIGURE 4.23: The inner workings of the parking area agent.

As shown in Figure 4.23(a), the parking area agent enters the simulation in the **initial** state. A condition trigger transitions the parking area agent into the **Parking** state if the **optionString** in the main agent is equal to “Parking Area”. During this transition, the neighbourhood containing the parking area agent is retrieved from the shapefile loaded on the GIS map in the main agent and stored in the **GISArea** variable. Once this is completed, the **ParkFunc** is called. This function retrieves parking area agents located in the surrounding neighbourhood by retrieving the name of these neighbourhoods from **Neighbours** and locating the GIS area from the shapefile in main. The function then identifies all the parking area agents located within these GIS areas and adds the agents to the **neighboursCol** collection. The **neighbourArea** variable and the **nameNeighbour** variable are used within the function to store the GIS area and name of the surrounding neighbours. The agent then transitions into the **isHotspot** state when a message is received specifying that this agent is a hotspot, as indicated by the message trigger. When the agent receives another message specifying that it is no longer a hotspot, the message trigger is activated and the agent transitions into the **Parking** state once again. This continues until the simulation is terminated.

The messages which activate the message triggers in the parking area agent's statechart, are sent to the parking area agents using schedules (shown in Figure 4.24) where each schedule represents an area which is classified as a hotspot at least once throughout the week. At the start of the simulation the **hotspotEvent** is triggered. This event retrieves all the parking area agents located in these areas and adds these agents to the respective collections,



*i.e.*  parking1, . . . ,  parking15. The schedules comprise boolean value where the schedule value is “on”, *i.e.* **true**, during the time interval when the area is a hotspot and the value is “off”, *i.e.* **false**, when the area is no longer classified as a hotspot. When the schedule value changes from “off” to “on”, a message is sent to the parking area agents contained in the collection corresponding to that area. Thereafter, the parking area agent transitions into the **isHotspot** state. Similarly, when the schedule value changes from “on” to “off”, a message is sent to the parking area agents in the corresponding collection which transitions the agent back into the **Parking** state.

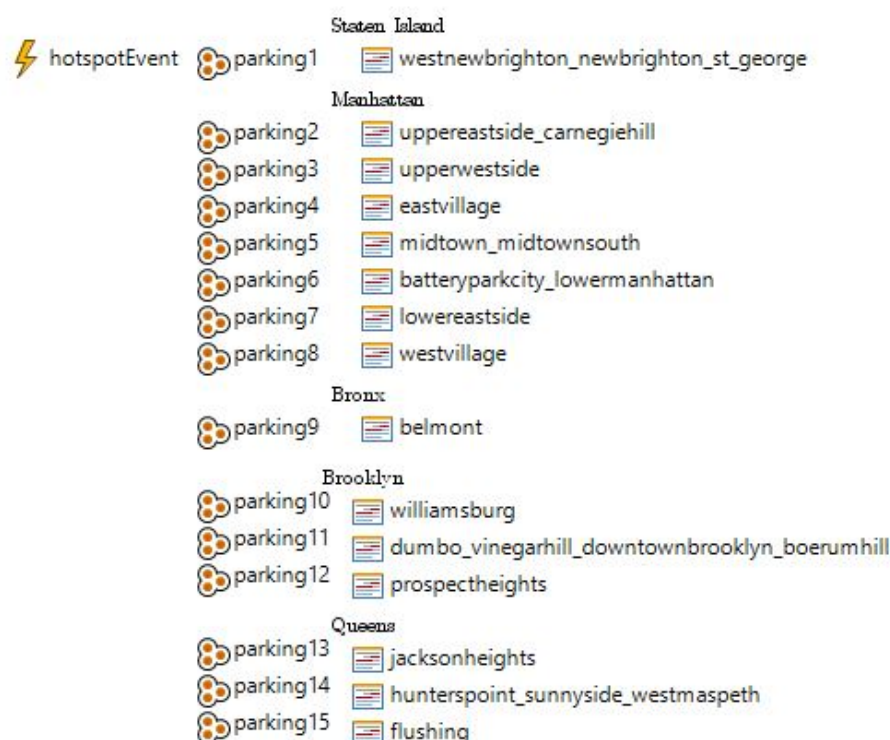







FIGURE 4.24: A screenshot of the AnyLogic components used to control the sending of messages to parking area agents regarding hotspots.

#### 4.6.4 Clustered waiting area agent

As stated previously, the clustered waiting area agents become active in the simulation when “Clustered Waiting Area” is selected as a waiting area type on the GUI. If this is true, each agent is indicated with a blue cloud . The clustered waiting area agent’s statechart only comprises one state, *i.e.* the **initial** state. The AnyLogic components used to model the agent’s behaviour are shown in Figure 4.25

When the clustered waiting area agent enters the simulation, it is located at the initial latitude and longitude coordinates stored in the  latInit and  longInit, respectively. The agent immediately assumes the **initial** state. Upon entering this state, the location of the agents are read from an Excel datasheet containing latitude and longitude coordinates of the agents for the duration of the simulation. The simulation only reads the Excel datasheet containing the locations for the cluster type selected on the GUI, therefore, if the user selected the cluster type as 10% and 1h30, the information will be read from  **locationsWait\_p10\_1h30**. The latitude and longitude coordinates contained in this Excel datasheet are stored in the  **location**

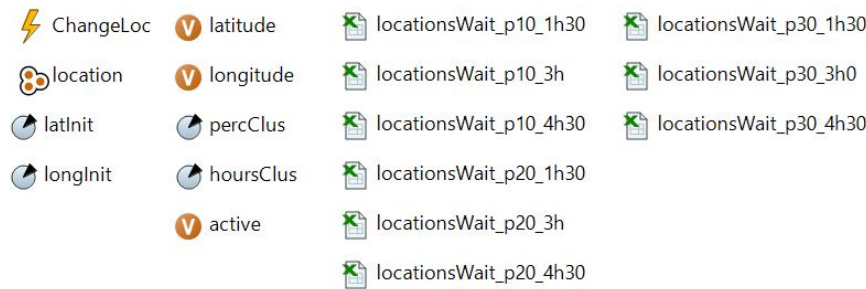


FIGURE 4.25: A screenshot of the AnyLogic components of the clustered waiting area agent.

collection. The user’s selection of the cluster type, which comprises the percentage of pick-up locations in the historical data set and the time period of historical data to be clustered, is contained in the **percClus** and **hoursClus** parameters, respectively. The **ChangeLoc** is activated every 10 minutes — this is how often the clusters change position. The latitude and longitude coordinates are then retrieved from the **location** collection and stored in the **latitude** and **longitude** variables, respectively. The clustered waiting area agent then “jumps” to this location using the AnyLogic function *jumpTo()* which moves an agent instantly to the desired location. If there are less than 229 clusters at any point in time, cells in the Excel datasheet corresponding to that agent assume a value of zero. If an agent’s **latitude** and **longitude** values are zero then the agent **active** variable is set to **false** and the agent becomes invisible, so to speak. If an agent’s **latitude** and **longitude** values are no longer 0, the agent becomes visible again and the **active** variable is set to **true**.

#### 4.6.5 Verification of the TNC simulation model

Simulation verification ensures the model performs as expected and contains no logical errors. The simulation model was continually debugged during the development of the model after the incorporation of new model components. This was accomplished by using AnyLogic’s built-in iterative run controller and debugger. Before the model can run, AnyLogic first compiles the model’s code. During model compilation, the debugger checks the model’s code to check for errors. If an error is not found then the simulation may be executed as per normal. If an error does exist, however, then the modeller is notified of this error and is given a description containing the type of error and where in the code the error can be located. Once all the errors are rectified the model can then be executed. Once the model is executed, errors can be detected during the simulation run. This may either be errors in the programming logic or Java exceptions. These Java exceptions are computational errors in the code and often involve trying to access a null pointer such as assigning a driver to a rider that does not exist in the simulation. These preventative measures assisted in ensuring the model is free from syntax and logical errors.

Further verification was performed by observing the simulation’s animation. This allows the modeller to detect any fundamental errors such as a driver agent moving in the opposite direction to the rider agents or the clustered waiting areas not changing locations every 10 minutes. Furthermore, tracing was performed throughout the simulation run. This enables the modeller to print statements to the console when an event occurs or when an agent transitions between states. This played a fundamental role in verifying the rider-passenger relationship. For example, when a driver is assigned to a rider a statement is printed indicating which rider the driver has been assigned to. Accordingly, the rider’s silhouette should change from blue to red.



### 4.6.6 Validation of the TNC simulation model

The process of validation ensures the model accurately represents the real world. This model is validated using two of the methods described in 3.6.1, namely, parameter variation and face validation.

The first parameter variation performed includes changing the number of drivers whilst keeping the number of riders and waiting area type constant. The positioning and time corresponding to the moment when a rider requests a ride are also kept constant. For these experiments, a one hour period of historical ride-requests are considered. The one hour period comprised 188 ride-requests and the waiting area type is selected as parking area. The simulation is run for a sufficient duration to ensure that all rides are completed, *i.e.* all riders that requested rides within that hour are dropped-off at their destinations. The simulation's random number generator is set to have a fixed seed so as to ensure the results are comparable. The resulting effect on the TWT, AWT and the number of riders in the system waiting to be picked-up by a driver are observed using the graphs produced during a simulation run. It is expected that an increase in the number of drivers results in a decrease in the TWT and AWT — ascribed to the increased availability of drivers to be assigned. Furthermore, an increase in drivers is expected to result in a decrease in the time elapsed before all the rides are completed.

The result of the first experiment containing 60 driver agents is illustrated in Figure 4.26. At the end of the simulation run the average TWT equates to about 5 300 seconds, as shown in the graph titled “Total waiting time” which contains the minimum, average and maximum of the TWT experienced riders throughout the simulation run. The average AWT equated to about 3 200 seconds, as observed from the graph titled “Assignment waiting time” which displays the minimum, average and maximum AWT experience by riders throughout the simulation run. Furthermore, all of the riders are either dropped-off at their destinations or being transported to their destination around 14 500 seconds into the simulation. All the riders reached their destination around 17 500 seconds into the simulation as illustrated in the graph titled “Number of riders”.

The next experiment is then performed with 120 driver agents. The average TWT at the end of this simulation run is over 2 000 seconds less than that of the one containing 60 driver agents, as shown in Figure 4.27. A similar result is observed with the average AWT which is more than six times less than of the previous experiment. The area between the curve representing the number of riders in the system and the curve representing the number of riders waiting for the driver is also larger which indicates more riders are being transported to their destinations at any point in time. Lastly, all the riders are picked-up within 10 000 seconds of the simulation run and all riders reached their destinations within 15 000 seconds.

Figure 4.28 illustrates the results generated for the last experiment performed in this parameter variation which included 180 driver agents. The average TWT at the end of this simulation run is around 2 200 seconds shorter than that of both the previous experiments performed. The average AWT is zero, *i.e.* all riders are assigned to drivers as soon as a ride is requested, which is confirmed by the curve representing the number of riders in the system with a maximum of 165 riders which is less than the number of drivers in the simulation. Furthermore, all the riders in the simulation are picked-up by drivers within 8 000 seconds and dropped-off at their destinations within 11 000 seconds, less than that of the other experiments. Lastly, the area between the two curves in the “Number of riders” graph is larger than the previous experiments.

During this parameter variation it is observed that by increasing the number of drivers, the average TWT and AWT decrease. As stated previously, this is an expected outcome which is both logical and intuitive. The average AWT equates to zero once a critical number of drivers are

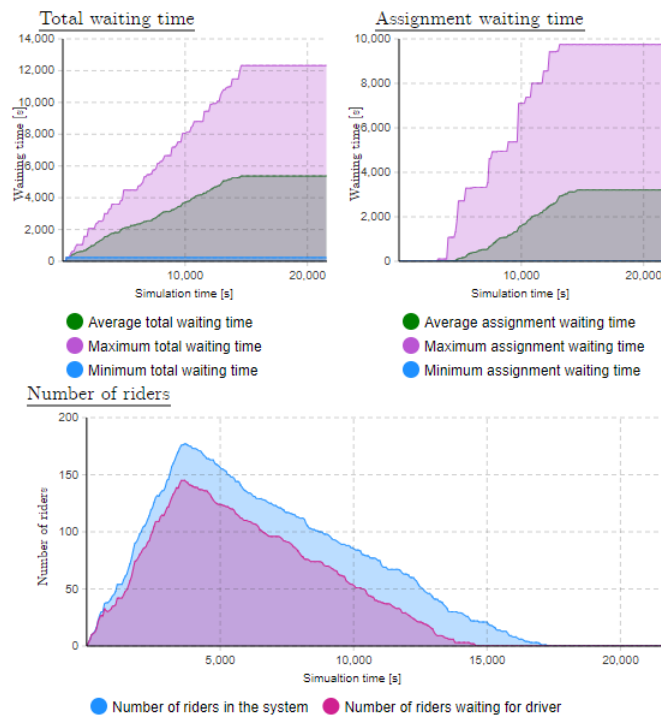


FIGURE 4.26: A screenshot of the graphs produced in a simulation run containing 60 driver agents.

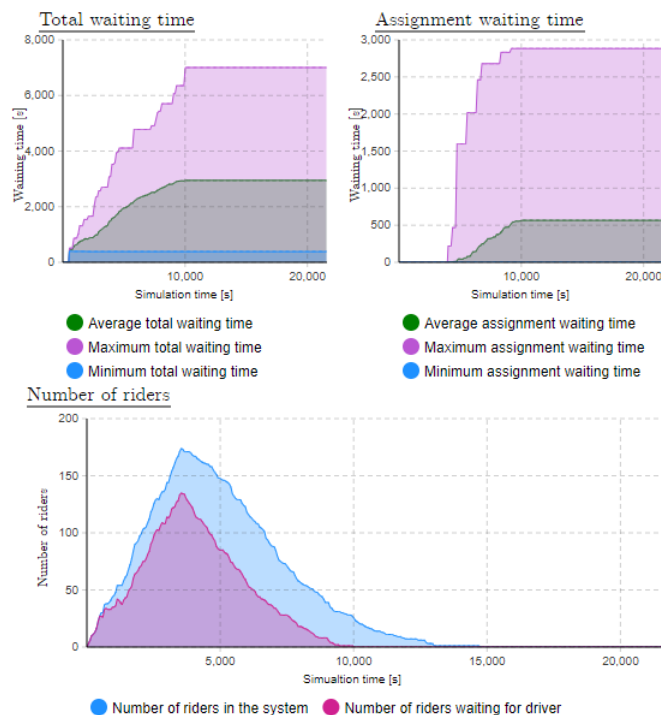


FIGURE 4.27: A screenshot of the graphs produced in a simulation run containing 120 driver agents.

present, ascribed to the fact that there are enough drivers available to be immediately assigned to riders upon ride-requests. As the number of driver agents increase, the ratio between the number of riders in the system and the number of riders waiting for drivers also increases as more drivers are available to transport riders to the destination.

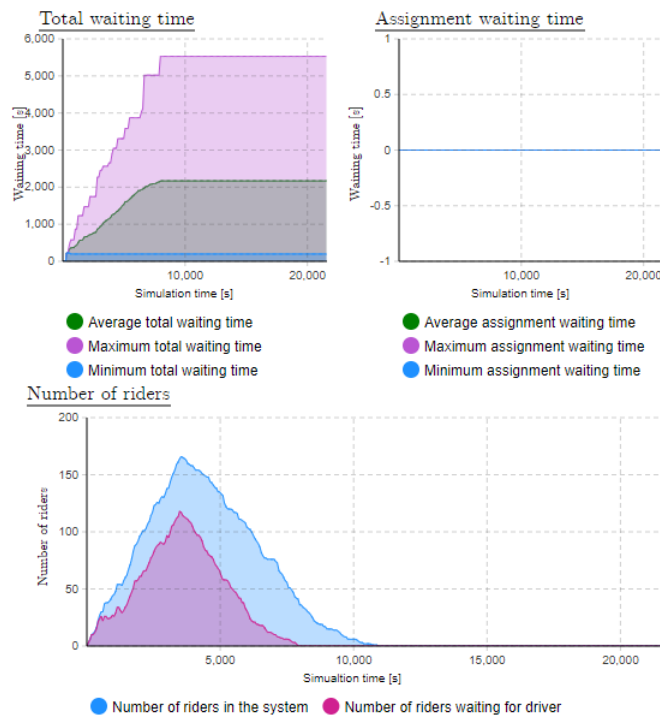


FIGURE 4.28: A screenshot of the graphs produced in a simulation run containing 180 driver agents

The next parameter variation performed evaluates the impact that different waiting strategies have on the three graphs. The first waiting strategy is to assign drivers to waiting areas with the highest number of pick-up requests within a 10km travel distance. The travel distance is selected such that it is slightly less than the average distance between the parking areas and the included pick-up locations which equates to 13.62km. In the second waiting strategy, the drivers were assigned to the waiting areas with the least pick-up requests within a 10km travel distance. The drivers in Strategy 1 should be able to access the riders sooner than the drivers in Strategy 2 and the riders should therefore have a lower average TWT. For both scenarios, there is a total of 120 drivers available and the waiting area type is set to parking area. As for the previous parameter variation experiments, the ride-requests that occurred within a one hour time period were included in the simulation and a fixed seed is used when testing both strategies.

In the first experiment, the drivers were assigned to parking areas according to Strategy 1. The results for this experiment are illustrated in Figure 4.29. At the end of the simulation run, this strategy resulted in a average TWT of 2000 and an average AWT of 250 seconds.

The results for the second experiment in which the drivers were assigned to parking areas according to Strategy 2 are shown in Figure 4.30. As expected, both the average AWT and the average TWT is longer than that of Strategy 1. The parking areas to which drivers were assigned using Strategy 1 were areas having high historical demand, therefore there is a higher probability of pick-up requests occurring near those parking areas. The drivers that were assigned according to Strategy 2 were positioned at parking areas having little demand and, as a result, had a larger distance to drive in order to arrive at pick-up locations.

The second form of validation is performed by the founder and chief financial officer of ChauffHer, Greg Wakelin [104]. ChauffHer is a start-up company in South Africa, planning to launch at the end of 2019, which provides ride-hailing service to women and children exclusively and, furthermore, their drivers are also exclusively women. This company focusses on ensuring the

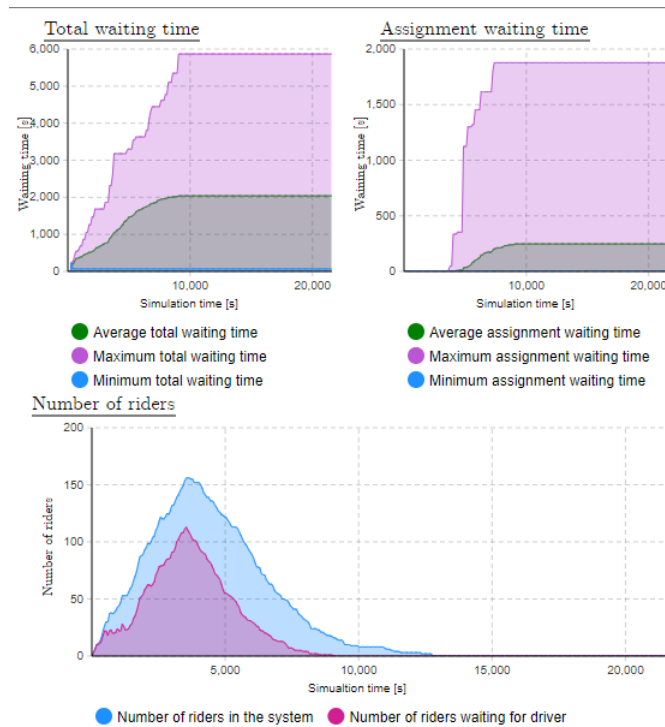


FIGURE 4.29: A screenshot of the graphs produced in a simulation run where drivers are assigned to the parking areas which experienced the highest number of pick-ups in the last hour within a 10km driving distance, *i.e.* Strategy 1.

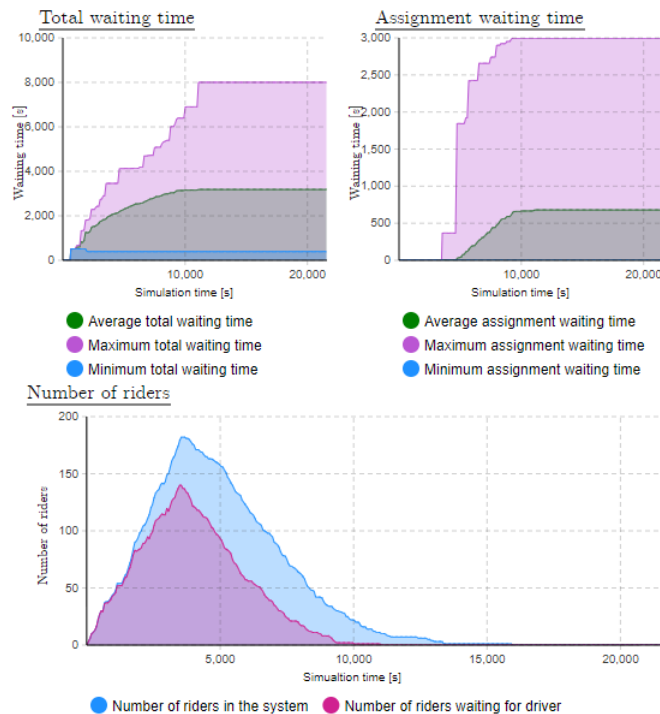


FIGURE 4.30: A screenshot of the graphs produced in a simulation run where drivers are assigned to the parking areas which experienced the lowest number of pick-ups in the last hour within a 10km driving distance, *i.e.* Strategy 2.

---

safety of the drivers as well as the safety of women and children when using the services provided by TNCs. Wakelin observed the operation of the model after the functionality of the model was demonstrated. He confirmed the potential use and benefits associated with this type of model which can not only decrease rider waiting time but also increase the rate at which drivers receive requests while decreasing driver idle time. Wakelin also stated that a platform which allows TNCs to experiment with various strategies and test scenarios could be advantageous.

## 4.7 Chapter summary

This chapter comprised a detailed description of the agent-based model developed to simulate the movement of drivers of TNCs as well as the interaction between drivers and riders. The chapter opened in §4.1 with a brief discussion of the AnyLogic modelling environment which included an explanation of the components employed during the development. The movement of drivers of TNCs was then discussed in §4.2 which was followed by a description of the parking area selection process in §4.3 and the clustered waiting area selection process in §4.4. A discussion on the development of the model was then initiated with a description on the operation of the GUI. Each component of the GUI was also verified. Furthermore, in §4.6, the various agents contained within the model, *i.e.* the driver, rider, parking area and clustered waiting area agents, and the verification and validation techniques used during and after model development were discussed.



---



---

## CHAPTER 5

---

# Experimentation and Results

### Contents

5.1	Experimental design . . . . .	81
	5.1.1 <i>Determining the number of drivers</i> . . . . .	82
	5.1.2 <i>Specifications of the simulation model</i> . . . . .	83
	5.1.3 <i>The simulation warm-up period</i> . . . . .	84
	5.1.4 <i>Methods of statistical analysis</i> . . . . .	86
5.2	Computational results . . . . .	89
5.3	Statistical evaluation . . . . .	91
5.4	Chapter summary . . . . .	93

In this chapter, the experimental design adopted in this thesis together with an analysis of the numerical results is presented. The experimental design is discussed in §5.1 which focusses on the procedure followed towards the establishment of a few salient modelling considerations, they are: The calculation of a suitable number of drivers in the simulation, the establishment of key model specifications, the calculation of the warm-up period and, lastly, an elucidation of the statistical method used for analysing the numerical results. In §5.2, the numerical results obtained by performing simulation runs with respect to eleven pre-defined scenarios are discussed and visualised using appropriate graphical illustrations. A statistical analysis of the obtained results is then performed in §5.3 so as to determine if the differences between the different scenarios are statistically significant at a specified level of confidence. Lastly, the contents of this chapter are summarised in §5.4.

### 5.1 Experimental design

In this section, the experimental design according to which the various simulation experiments are performed is described in detail. This elucidation includes the calculation of a suitable number of drivers that form part of the different experiments. In addition, the establishment of the so-called warm-up period of the simulation is described. Finally, the method employed towards analysing the output results in a statistically sound manner is discussed. The various components addressed in this section serve as a necessary pre-cursor to the analysis of the computational results thereafter.

### 5.1.1 Determining the number of drivers

Hall [37] — a subject matter expert at Uber Technologies — conducted a comprehensive analysis on the labour market for Uber drivers. In this study, two factors are influential in the calculation of the estimated average number of drivers in NYC, the first of which relates to the number of hours worked per week by Uber drivers (summarised in Table 5.1) which includes the percentage of drivers  $x$  together with the range of hours worked per week  $y$ .

TABLE 5.1: *Distribution of the number of hours worked by Uber drivers in NYC.*

Percentage of drivers	Number of hours worked per week
24%	1–15
32%	16–34
27%	35–49
17%	Over 50

The mean in respect of the different ranges indicating the hours worked per week is calculated and represents the estimated average number of hours worked per week. The average number of hours worked per week is thereafter divided by seven in order to obtain the average number of hours worked per day. Subsequently, the average number of hours worked per day is multiplied by the percentage of drivers. The resulting value is summated so as to obtain the average number of hours worked per day by the Uber drivers. The resulting values obtained (by means of the aforementioned computations) are presented in Table 5.2.

TABLE 5.2: *Determining the average number of hours worked per day by Uber drivers*

Percentage of drivers ( $x$ )	Hours worked per week ( $y$ )	Average number of hours worked per week ( $\bar{y}$ )	Average number of hours worked per day ( $\bar{y}/7$ )	Proportion of average number of hours worked ( $((\bar{y}/7) \times x)$ )
24%	1–15	8	1.14	0.27
32%	16–34	25	3.57	1.14
27%	35–49	42	6.00	1.62
17%	Over 50	50	7.14	1.21
Average hours worked per day by Uber driver				4.25

The number of drivers to form part of the simulation is deduced from basic statistics reported on in respect of the number of UberX (*i.e.* the primary service provided by Uber) drivers — a decision mainly attributable to the lack of information pertaining to the operations of TNCs. In Figure 5.1, the growth of UberX drivers in NYC since June 2012 is plotted and is used to choose a suitable number of expected drivers for the period considered in respect of the data set, *i.e.* April 2014. Based on rather simple computational analysis of the reported number of UberX drivers, it is concluded that approximately 6 200 drivers (as indicated by the red line in the figure) are considered for the proposed analysis in this thesis. The approach adopted is arguably trivial, however, considering the lack of data pertaining to this matter, one may conclude that this approach is reasonable.

The calculated average number of hours worked per day, *i.e.* 4.25, (as presented in Table 5.2) is divided by 24 so as to determine the number of 4.25-hour periods that transpire within a 24-hour day which can also be interpreted as the portion of the day that the average driver works. The number of drivers is assumed to remain constant throughout the simulation, therefore the proportion of the day for which the average Uber driver works is multiplied by the total number



of drivers, *i.e.* 6 200. This computation resulted in an expected average active number of drivers equating to 1 097.92, which is then rounded up to 1 100 drivers, for the sake of simplicity.

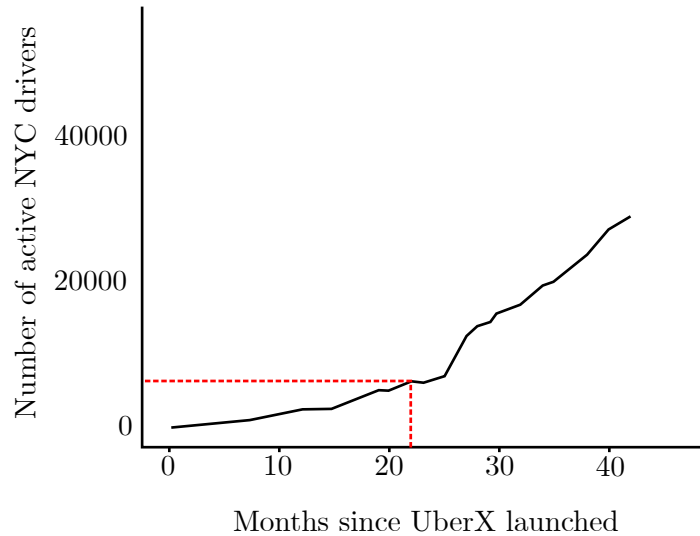


FIGURE 5.1: The growth of UberX driver partners since June 2012 [37].

### 5.1.2 Specifications of the simulation model

There is a total of eleven scenarios tested, two of which relate to the parking areas whereas the remaining nine relate to the clustered waiting areas. The first scenario pertaining to the parking areas corresponds to the driver selecting a parking area according to the strategy discussed in §4.3, whereas in the case of the second scenario, the driver is assigned to the nearest parking area. Furthermore, the drivers in the scenarios corresponding to the clustered waiting areas are assigned to the nearest clustered waiting area. Each of the clustered waiting areas differ in the cluster type, as explained in §4.4. The nine scenarios correspond to the combinations of three clustering percentages, *i.e.* 10, 20, 30 and three time periods, *i.e.* 90 minutes, 180 minutes, 270 minutes. The scenarios described above are summarised in Table 5.3 and labelled as Scenario 1, Scenario 2, ... and Scenario 11.

For each of the scenarios considered, the number of drivers is set to 1 100 and is fixed throughout the simulation and it is assumed that drivers accept every ride-request received. Furthermore, riders leave the system only upon the arrival at the destination. The agent-based simulation model developed in this thesis (described in Chapter 4) aims to model a non-terminating system, *i.e.* there is no natural event that can be used to specify the length of a run. In terminating systems, on the other hand, events occur which provide a clear indication regarding the end of a simulation run, for example, a bank which operates from 9:00 AM to 5:00 PM where the simulation would terminate at 5:00 PM or a manufacturing system where a specified number of items are to be produced. The simulation run length of a non-terminating system can be determined by time intervals, called *cycles*; the events in the system repeat themselves after each cycle. In the developed simulation model, the run length is selected as one day so as to include the fluctuations in demand which occur throughout the day.

TABLE 5.3: Summary of the eleven scenarios for which experiments are conducted.

Label	Waiting area type	Description
Scenario 1	Parking area	Drivers are assigned to nearest parking area.
Scenario 2	Parking area	Drivers select parking areas according to selection strategy.
Scenario 3	Clustered waiting area	The number of clusters is set to 10% of the data. duration of data subjected to clustering is set to 90 minutes.
Scenario 4	Clustered waiting area	The number of clusters is set to 10% of the data. The duration of data subjected to clustering is set to 180 minutes.
Scenario 5	Clustered waiting area	The number of clusters is set to 10% of the data. The duration of data subjected to clustering is set to 270 minutes.
Scenario 6	Clustered waiting area	The number of clusters is set to 20% of the data. The duration of data subjected to clustering is set to 90 minutes.
Scenario 7	Clustered waiting area	The number of clusters is set to 20% of the data. The duration of data subjected to clustering is set to 180 minutes.
Scenario 8	Clustered waiting area	The number of clusters is set to 20% of the data. The duration of data subjected to clustering is set to 270 minutes.
Scenario 9	Clustered waiting area	The number of clusters is set to 30% of the data. duration of data subjected to clustering is set to 90 minutes.
Scenario 10	Clustered waiting area	The number of clusters is set to 30% of the data. The duration of data subjected to clustering is set to 180 hours.
Scenario 11	Clustered waiting area	The number of clusters is set to 30% of the data. duration of data subjected to clustering is set to 270 minutes.

### 5.1.3 The simulation warm-up period

Upon execution of the simulation, all the drivers are at waiting areas and, as a result, the riders requesting ride-requests experience minimal waiting time. Thereafter, the drivers travel to complete ride-requests as they are requested. An increase in the number of ride-requests results in an increase in the average TWT until a steady-state is reached. Consequently, a simulation warm-up period is required so as to ensure that the results obtained are reasonable and reflective. Law [55] proposes a method for calculating the length of the warm-up period which is elucidated hereafter.

Let  $X_{ij}$  denote the waiting time observed at time point  $i$  during a simulation run with random seed  $j$ . A random seed provides an indication of the starting point for random number generation (*i.e.* a change in the random seed will result in the generation of a different set of random numbers). Suppose  $n$  independent simulation runs of length  $m$  (*i.e.*  $m$  discrete points in time) are executed, each with a different random seed, resulting in the following observations

$$\begin{array}{ccc}
X_{11}, \dots, X_{1j}, \dots, X_{1n} & & \\
\vdots & \vdots & \vdots \\
X_{i1}, \dots, X_{ij}, \dots, X_{in} & & \\
\vdots & \vdots & \vdots \\
X_{m1}, \dots, X_{mj}, \dots, X_{mn} & & 
\end{array}$$

Let  $Y_i$  denote the waiting time observed at time point  $i$ . Therefore  $E(Y_i)$  denotes the mean of  $Y_i$ , which is given by

$$\bar{Y}_i = \frac{\sum_{j=1}^n X_{ij}}{n}. \quad (5.1)$$

The steady-state mean, denoted by  $\bar{s}$ , can then be expressed as

$$\bar{s} = \lim_{i \rightarrow \infty} E(Y_i). \quad (5.2)$$

The relationship expressed mathematically in (5.2) does not, however, hold during the warm-up period  $w$ , *i.e.*  $E[\bar{Y}(w)] \neq \bar{s}$ . Consequently, observations  $[1, v]$  should be removed before the estimation of  $\bar{s}$ . Accordingly, the value for  $\bar{s}$  can then be estimated as

$$\bar{Y}(w, v) = \frac{\sum_{j=v+1}^w Y_j}{w - v},$$

instead of  $\bar{Y}(w) = \frac{\sum_{j=1}^w Y_j}{w}$ . The warm-up period can, consequently, be determined using a procedure comprising the following four steps:

1. Conduct  $n$  replications of the simulation where each simulation run is of length  $m$ . Let  $X_{ij}$  denote observation at moment  $i$  of replication  $j$  where  $i = 1, \dots, m$  and  $j = 1, \dots, n$ .
2. Determine  $\bar{Y}_1, \dots, \bar{Y}_m$  according to (5.1). The average  $\bar{Y}_i$  has a mean of  $E(\bar{Y}_i) = E(Y_i)$  and a variance  $\text{Var}(\bar{Y}_i) = \text{Var}(Y_i)/n$ .
3. If oscillations in  $\bar{Y}_1, \dots, \bar{Y}_m$  are of a high frequency, a moving average is applied in order to “smoothen” these oscillations. The moving average is given by

$$\bar{Y}_i(x) = \begin{cases} \frac{\sum_{z=-x}^x \bar{Y}_{(i+z)}}{2x+1} & \text{if } i = z + 1, \dots, m - z \\ \frac{\sum_{z=-(i-1)}^{i-1} \bar{Y}_{(i+z)}}{2i+1} & \text{if } i = 1, \dots, x, \end{cases}$$

where  $x$  is the size of the window (*i.e.* size of moving average) such that  $0 \leq x \leq m/4$ .

4. Plot  $\bar{Y}_i(x)$  for  $i = 1, \dots, m - x$ . The value for  $v$  can then be determined as the value of  $i$  for which  $\bar{Y}_i(x)$  has converged.

In order to determine a warm-up period, a total of 30 replications of the simulation run are performed, each comprising 135 minutes in respect of each experiment. The average waiting time is recorded every minute. It is found that a warm-up period of 100 minutes is sufficient, as illustrated in Figure 5.2.

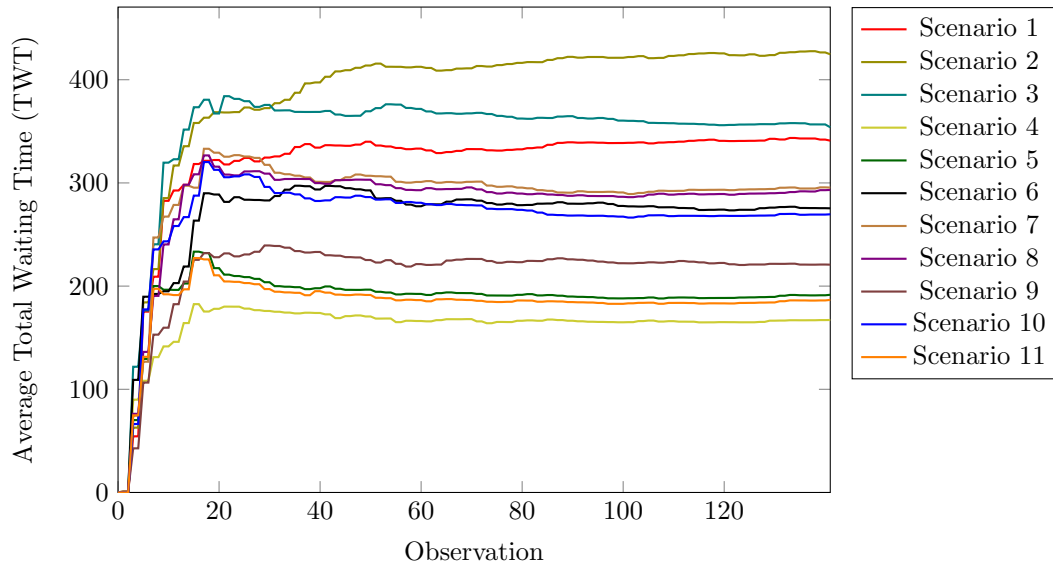


FIGURE 5.2: An indication of the warm-up period of the eleven scenarios in question with simulations consisting of 1 100 drivers.

#### 5.1.4 Methods of statistical analysis

Each scenario described in §5.1.2 forms part of a comparative study in which differences are evaluated statistically at a 5% level of significance. The main comparison focusses on the types of waiting areas, *i.e.* parking areas and clustered waiting areas. An appropriate statistical test is employed so as to determine whether statistical differences exist between the control scenario, *i.e.* Scenario 1, and the scenarios relating to clustered waiting areas *i.e.* Scenario 3, Scenario 4, ... and Scenario 11. Scenario 2 is also compared to Scenario 1 so as to observe the effects of the selection strategy. In order to evaluate whether the output data of the scenarios are statistically significant, an *analysis of variance* (ANOVA) [64] is performed. The ANOVA test identifies if at least one significant difference is present between any two means. In order to determine where exactly the significant difference lies, a suitable *post hoc* test is performed. The *post hoc* test performed is dependent on whether there is a statistically significant difference between the variances of the samples which is tested using *Bartlett's test*. If Bartlett's test identifies a statistical difference between the variances at a 5% level of significance, the *Dunnett T3* test is performed to locate the differences in the *performance measure indicators* (PMIs) of the scenarios. If the variances of the PMIs are found to be not statistically different at a 5% level of significance, the *Dunnett* test is performed.

The ANOVA test, together with the *post hoc* tests, assumes that the sample data is normally distributed. If the samples are drawn from an unknown distribution, however, according to the central limit theorem, the distribution of the sample mean would be approximately normally distributed if the sample size  $m$  is adequately large. Since the distribution of the four PMIs (*i.e.* average TWT, maximum TWT, average AWT and maximum AWT) are unknown,  $m$  is required to be large so as to ensure compliance with this basic assumption. According to Law [55], in order to determine the sample size, a sample  $x_1, \dots, x_{m_o}$ , with mean  $\bar{x}$ , is initially generated by performing  $m_o$  replications of the simulation run. The standard deviation of this generated sample is then calculated as

$$S_x = \sqrt{\frac{\sum_{i=1}^{m_o} (x_i - \bar{x})^2}{m_o - 1}}.$$

The *confidence interval* (CI) around the true mean is determined by employing the studentised range distribution [64] which is a distribution used to estimate the range of a population. Therefore, the CI half-width,  $H_0$ , at a confidence level of  $(1 - \alpha)$  can then be expressed as

$$H_0 = t_{(1-\alpha/2), (m-1)} \frac{S_x}{\sqrt{m_o}},$$

where  $t_{(1-\alpha/2), (m-1)}$  is the critical value for a two-sided error with a sum of  $\alpha$  and  $m_o - 1$  degrees of freedom. The half-width  $h$  is then evaluated. If the half-width is satisfactory then the number of replications  $m_o$  is sufficient. On the other hand, if the half-width is observed as too large, a desired half-width  $h^*$  is identified. The required number of replications  $m^*$  is then calculated as

$$m^* = m_o \left( \frac{H_0}{h^*} \right)^2.$$

Once  $m^*$  replications have been generated, the ANOVA test can be performed. The operation of the statistical tests to be employed, *i.e.* ANOVA, Bartlett's test, Dunnett T3 and *Dunnett's test*, is reviewed briefly using the following notation. Suppose  $n$  samples are being compared, each with a sample size  $m$ . Let the  $i^{th}$  sample, with mean  $\bar{x}$  and standard deviation  $s_i$ , be denoted by  $x_{i1}, \dots, x_{im}$  where  $i = 1, \dots, n$ . Lastly, let  $\bar{\bar{x}}$  denote the overall mean of all the samples.

As stated previously, the ANOVA test is performed to observe if a significant difference exists between at least two of the sample means at a  $\alpha$ -level of significance. The null-hypothesis  $H_0$  of ANOVA states that there are no significant differences between any of the sample means. Therefore, the research (*i.e.* alternative) hypothesis is that a statistically significant difference exists between at least two of the sample means. The ANOVA test evaluates the null-hypothesis by comparing the sum of squares between samples in addition to the sum of squares of observations within samples. If  $n$  samples exist, each with a sample size of  $m$  and a mean of  $\bar{x}_i$ , the sum of squares between samples  $S_b$  may be described as

$$S_b = m \sum_{i=1}^n (\bar{x}_i - \bar{\bar{x}})^2$$

The variance between samples is calculated by dividing  $S_b$  by the respective degrees of freedom  $df_b$ . Therefore the variance between samples  $MS_b$  is given by

$$MS_b = \frac{S_b}{df_b} = \frac{m}{n-1} \sum_{i=1}^n (\bar{x}_i - \bar{\bar{x}})^2.$$

Similarly, the sum of squares of observations within samples is given by

$$S_w = \sum_{i=1}^n \sum_{j=1}^m (x_{ij} - \bar{x}_i)^2,$$

thereafter, the respective variance are calculated by dividing  $S_w$  by the degrees of freedom  $df_w$ , that is

$$MS_w = \frac{S_w}{df_w} = \frac{1}{mn-1} \sum_{i=1}^n \sum_{j=1}^m (x_{ij} - \bar{x}_i)^2.$$

The test statistic is then calculated as the ratio between  $MS_b$  and  $MS_w$  (*i.e.*  $MS_b/MS_w$ ) and is compared to the critical value  $F(df_b, df_w, \alpha)$  of the  $F$ -distribution, which can be found in [64]. If it is observed that

$$S_b/S_w > F(df_b, df_w, \alpha),$$

the null-hypothesis  $H_0$  is rejected at an  $\alpha$ -level of significance. Therefore, statistically significant differences exist within the sample means with a  $(1-\alpha)$ -level of confidence. If the condition does not hold, however, then the null-hypothesis is not rejected, *i.e.* no statistically significant difference is identified between the means of the samples.

Bartlett's test [92] is used to determine if the difference between the variances of any two samples are considered statistically significant at an  $\alpha$ -level of significance. The null hypothesis  $H_0$  states that there are no statistically significant differences between the variances of any of the samples. Therefore, the research (alternative) hypothesis  $H_1$  is that a statistically significant difference exists between the variances of at least two of the samples. Two variables are required to be determined in order to perform Bartlett's test, namely: The test statistic  $\chi_o^2$  of the Chi-Squared Distribution [64] and the critical value  $\chi_{(n-1, \alpha)}^2$ , which is the same as the critical value used in the ANOVA test. The test statistic is given by

$$\chi_o^2 = \frac{(nm - n) \ln(s_i^2) - n(m - 1) \ln(s_i^2)}{1 + \frac{1}{3n(m-1)} \left( \frac{1}{m-1} - \frac{1}{nm-m} \right)}.$$

Thereafter, if

$$\chi_o^2 > \chi_{(n-1, \alpha)}^2$$

the null hypothesis is rejected, *i.e.* there is a difference between the variances of at least two of the sample means at a  $(1-\alpha)$ -level of confidence. On the other hand, if this condition does not hold, the null hypothesis is not rejected indicating that there is no statistically significant difference between the variances of the samples at a  $(1-\alpha)$ -level of confidence.

If Bartlett's test indicates a homogeneity of variances, the Dunnet test can be employed. The Dunnet test [22] is a multiple comparison method employed when the experiment contains a control sample. The null hypothesis  $H_0$  for Dunnet's test states that there is no significant differences between the means  $\bar{x}_u$  and  $\bar{x}_v$  of two samples, where  $u$  and  $v$  denote the two samples. Therefore, the research hypothesis  $H_1$  is that there is a significant difference between the two means  $\bar{x}_u$  and  $\bar{x}_v$ . The test statistic is given by  $|\bar{x}_u - \bar{x}_v|$  whereas the critical value is calculated as

$$D_c = t_{(n-1)(mn-n), \alpha} \sqrt{\frac{2MS_w}{m}},$$

where  $t_{(n-1)(mn-n), \alpha}$  is drawn from Dunnet's distribution table. If  $|\bar{x}_u - \bar{x}_v| > D_c$ , the null hypothesis is rejected at a  $(1 - \alpha)$ -level of confidence indicating that there is a statistically significant difference between the two means at an  $\alpha$ -level of significance. If this inequality does not hold, however, there exists a difference between the two means at an  $\alpha$ -level of significance. This process is repeated  $(n - 1)$ -times as every sample is compared to the control sample.

On the other hand, if Bartlett's test indicates a difference in the variances of the samples, Dunnet's T3 test [23] is employed. When comparing two samples with means of  $\bar{x}_u$  and  $\bar{x}_v$  and standard deviations  $s_u$  and  $s_v$ , the corresponding test statistic is given by

$$t_d = \frac{|\bar{x}_u - \bar{x}_v|}{\sqrt{\frac{s_u^2 + s_v^2}{m}}}.$$

The control statistic  $SMM_{\alpha,c,df_{uv}}$  is drawn from the studentised maximum modulus distribution [95] where  $c$  denotes the number of comparisons to be made and  $df_{uv}$  denotes the degrees of freedom calculated as

$$df_{uv} = \frac{m-1}{(s_u^2)^2 + (s_v^2)^2} \left( \frac{s_u^2 + s_v^2}{m} \right).$$

Once again, the null hypothesis  $H_0$  is rejected if  $t_d > SMM_{\alpha,c,df_{uv}}$ , indicating that there is a significant difference between the means of the two samples at an  $\alpha$ -level of significance, otherwise, the means of two samples do not differ at a  $(1-\alpha)$ -level of confidence.

The method of rejecting the null hypothesis at an  $\alpha$ -level of significance is known as *fixed significance level* testing [64]. The so-called *p-value* employed in fixed significance level testing is the smallest level of significance that would result in the rejection of the null hypothesis. Once the *p-value* is calculated, it is compared to a pre-defined significance level  $\alpha$  which usually assumes a value of 0.05.

## 5.2 Computational results

In this section, the simulation results corresponding to the eleven scenarios are discussed. The results are presented visually in the form of box plots indicating the means, medians and interquartile ranges of the average TWT, maximum TWT, average AWT and maximum AWT of the 30 replications performed for each scenario. The results are compared using the means of the samples.

The average TWT of each of the eleven scenarios is illustrated graphically in Figure 5.3. The average TWT of Scenarios 3–11, according to which clustered waiting areas are employed, is observed to be less than that of Scenario 1 — Scenario 8 achieves the best performance with an improvement of 35.25%, followed by Scenarios 9 and 11 which achieved improvements of 34.25% and 34.94%, respectively. Scenario 2, however, achieves the best overall performance with a reduction in average TWT of 36.82% in comparison to Scenario 1, reinforcing the impact of the selection strategy.

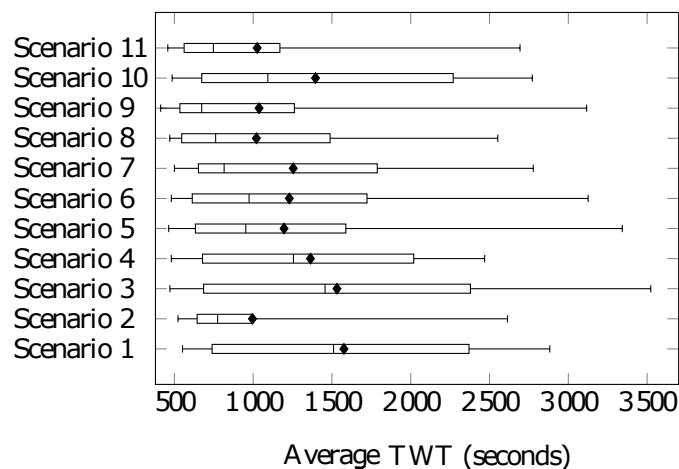


FIGURE 5.3: The average TWT for the eleven scenarios.

In Figure 5.4, the results obtained regarding the maximum TWT for the eleven scenarios are presented. A similar trend is observed according to which Scenarios 8 and 9 outperformed

Scenario 1 with respect to the means, where performance differentials correspond to 19.3% and 24.4%, respectively. Furthermore, Scenario 2 performs the best out of the eleven scenarios — a reduction of 30.11% over Scenario 1. Similarly, the average AWT for the eleven scenarios are recorded and illustrated in Figure 5.5. By observing the means of the box plot, it can be observed that Scenarios 4 to 11 (*i.e.* all the scenarios which employ clustered waiting areas, excluding Scenario 3) outperform Scenario 1, where the best performance observed corresponds to Scenarios 8 and 9, achieving improvements of 52.95% and 46.19%, respectively. The best overall performance, on the other hand, is once again achieved by Scenario 2 — a 70.4% improvement in comparison with Scenario 1 is achieved.

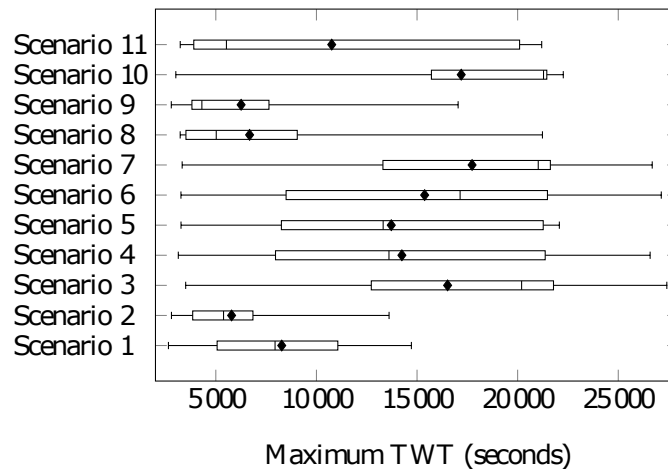


FIGURE 5.4: The maximum TWT results for the eleven scenarios.

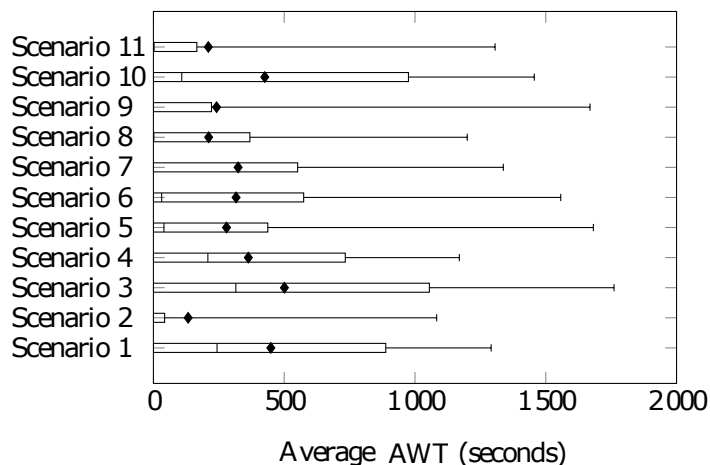


FIGURE 5.5: The average AWT results for the eleven scenarios.

The last results obtained are visually represented in Figure 5.6 and represent the maximum AWT experienced for the eleven scenarios. Scenario 1 is outperformed by all the scenarios where clustered waiting areas are employed, except Scenario 3 and Scenario 10 (*i.e.* Scenario 4–Scenario 9 and Scenario 11) — the best performance achieved is, once again, attributed to Scenario 8 with a reduction of 45%, followed by Scenario 9 and Scenario 11, achieving improvements of 38.67% and 43.72%, respectively. Furthermore, the overall best performance is once again achieved by Scenario 2, outperforming Scenario 1 by 63.72%.



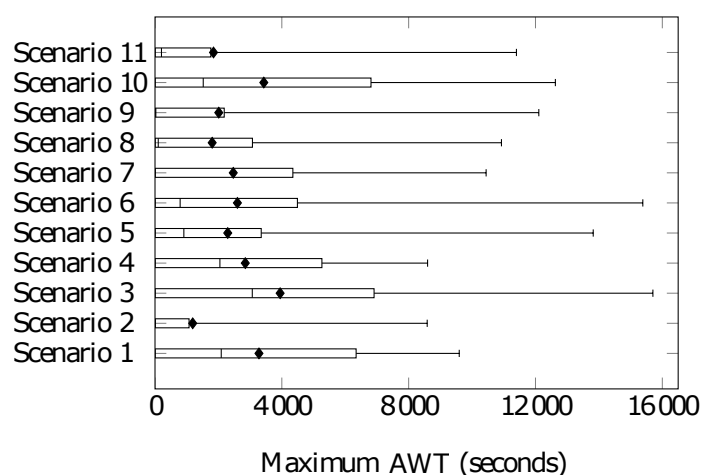


FIGURE 5.6: The maximum AWT results for the eleven scenarios.

### 5.3 Statistical evaluation

In this section, statistical tests are performed to determine if the differences observed in §5.2 are statistically significant at a 5% level of significance. The ANOVA test is first performed on the PMIs of the samples. If no significant difference is found, no further tests are performed, however, if a significant difference does exist, Bartlett's test is employed to evaluate if a significant difference exists between the variances of the samples. If the difference between the variances is identified as statistically significant, Dunnet T3 test is employed, otherwise, if there is no significant differences between the variances, Dunnet's test is performed on the samples. The results of these tests are visualised in tables which indicate whether differences exists between the PMIs of the scenarios at a 5% level of significance.

The  $p$ -values obtained from ANOVA and Bartlett's test, in respect of the PMI values of the scenarios, are presented in Figures 5.4–5.6. The ANOVA test indicates that significant differences exists between the samples for three of the four PMIs, namely: Average TWT, Maximum TWT and Average AWT. Furthermore, Bartlett's test reveals that the variances of the samples are only significantly different in respect to the maximum TWT. Therefore, Dunnet T3 test is employed in order to locate the differences with respect to the maximum TWT, whereas Dunnet's test is performed to identify the samples for which the average TWT and average AWT differ significantly.

TABLE 5.4: The mean values of all PMIs together with the  $p$ -values for ANOVA and Bartlett's test in respect of Scenarios 1–4.

PMI	Scenario mean value				$p$ -value	
	1	2	3	4	ANOVA	Bartlett's Test
Average TWT	1 576.22	995.81	1 531.91	1 363.77	$1.05 \times 10^{-2}$	$4.49 \times 10^{-1}$
Maximum TWT	8 275.93	5 784.03	16 518.51	14 247.18	$< 1 \times 10^{-4}$	$< 1 \times 10^{-4}$
Average AWT	448.95	132.84	500.82	362.93	$3.58 \times 10^{-2}$	$1.04 \times 10^{-1}$
Maximum AWT	3 278.97	1 189.53	3 946.57	2 847.50	$7.44 \times 10^{-1}$	-

According to the results obtained from Dunnet's test as shown in Table 5.7, the average TWT between Scenario 1 and Scenario 2, Scenario 8, Scenario 9 and Scenario 11 are statistically different at a 5% level of significant. The results of this test corroborate the observations in §5.2

TABLE 5.5: The mean values of all PMIs together with the  $p$ -values for ANOVA and Bartlett's test in respect of Scenarios 5–8.

PMI	Scenario mean value				$p$ -value	
	5	6	7	8	ANOVA	Bartlett's Test
Average TWT	1 196.11	1 229.35	1 254.33	1 020.58	$1.05 \times 10^{-2}$	$4.49 \times 10^{-1}$
Maximum TWT	13 719.43	15 378.50	17 735.53	6 678.33	$< 1 \times 10^{-4}$	$< 1 \times 10^{-4}$
Average AWT	279.50	316.18	323.83	211.25	$3.58 \times 10^{-2}$	$1.04 \times 10^{-1}$
Maximum AWT	2 294.17	2 598.93	2 468.20	1 803.43	$7.44 \times 10^{-1}$	-

TABLE 5.6: The mean values of all PMIs together with the  $p$ -values for ANOVA and Bartlett's test in respect of Scenarios 9–11.

PMI	Scenario mean value			$p$ -value	
	9	10	11	ANOVA	Bartlett's Test
Average TWT	1 037.60	1 395.69	1 025.55	$1.05 \times 10^{-2}$	$4.49 \times 10^{-1}$
Maximum TWT	6 256.97	17 195.87	10 760.64	$< 1 \times 10^{-4}$	$< 1 \times 10^{-4}$
Average AWT	241.55	425.73	210.08	$3.58 \times 10^{-2}$	$1.04 \times 10^{-1}$
Maximum AWT	2 011.13	3 431.57	1 845.23	$7.44 \times 10^{-1}$	-

where Scenario 2, Scenario 8, Scenario 9 and Scenario 11 achieved the best improvement with respect to the average TWT when compared with Scenario 1.

TABLE 5.7: Differences in respect of the average TWT of the eleven scenarios when compared with Scenario 1. A table entry which represents a significant difference at a 5% level of significance is indicated in red.

	Dunnet's test $p$ -value			Dunnet's test $p$ -value	
	Scenario 1	Mean		Scenario 1	Mean
Scenario 1	-	1576.22	Scenario 7	$> 9 \times 10^{-1}$	1254.33
Scenario 2	$1.96 \times 10^{-2}$	995.81	Scenario 8	$2.9 \times 10^{-2}$	1 020.58
Scenario 3	$> 9 \times 10^{-1}$	1 531.91	Scenario 9	$3.8 \times 10^{-2}$	1 037.60
Scenario 4	$> 9 \times 10^{-1}$	1 363.77	Scenario 10	$> 9 \times 10^{-1}$	1395.69
Scenario 5	$> 9 \times 10^{-1}$	1196.11	Scenario 11	$3.14 \times 10^{-2}$	1 395.69
Scenario 6	$> 9 \times 10^{-1}$	1 229.35	-	-	-

In Table 5.8 is the results obtained after performing Dunnet's test in respect of the maximum TWT is presented. Of the improvements identified in §5.2 between Scenario 1 and Scenario 2 as well as Scenario 8 and Scenario 9, only the improvement between Scenario 1 and Scenario 2 is found to be statistically significant. The other statistical differences identified between Scenario 1 and Scenario 3, Scenario 4, Scenario 5, Scenario 6 and Scenario 10 are attributable to an increase in the maximum TWT.

Lastly, the results obtained by performing Dunnet's test in respect of the average AWT are shown in Table 5.9. In the discussion of the boxplots in §5.2, the best improvements are observed between Scenario 1 and Scenario 2, Scenario 8, Scenario 9 and Scenario 11. The only difference which is found to be statistically significant, however, is between Scenario 1 and Scenario 2.

TABLE 5.8: Differences in respect of the maximum TWT of the eleven scenarios when compared with Scenario 1. A table entry which represents a significant difference at a 5% level of significance is indicated in red.

Dunnet T3 test $p$ -value			Dunnet T3 test $p$ -value		
	Scenario 1	Mean		Scenario 1	Mean
Scenario 1	-	8 275.93	Scenario 7	$< 1 \times 10^{-4}$	17 735.53
Scenario 2	$2 \times 10^{-4}$	5 784.03	Scenario 8	$> 9 \times 10^{-1}$	6 678.33
Scenario 3	$2 \times 10^{-4}$	16 518.51	Scenario 9	$> 9 \times 10^{-1}$	6 256.97
Scenario 4	$1.28 \times 10^{-2}$	14 247.18	Scenario 10	$< 1 \times 10^{-4}$	17 195.87
Scenario 5	$1.43 \times 10^{-2}$	13 719.43	Scenario 11	$> 9 \times 10^{-1}$	10 760.64
Scenario 6	$2 \times 10^{-3}$	15 378.50	-	-	

TABLE 5.9: Differences in respect of the average AWT of the eleven scenarios when compared with Scenario 1. A table entry which represents a significant difference at a 5% level of significance is indicated in red.

Dunnet's test $p$ -value			Dunnet's test $p$ -value		
	Scenario 1	Mean		Scenario 1	Mean
Scenario 1	-	448.95	Scenario 7	$> 9 \times 10^{-1}$	323.83
Scenario 2	$4.29 \times 10^{-2}$	132.84	Scenario 8	$> 9 \times 10^{-1}$	211.25
Scenario 3	$> 9 \times 10^{-1}$	500.82	Scenario 9	$> 9 \times 10^{-1}$	241.55
Scenario 4	$> 9 \times 10^{-1}$	362.93	Scenario 10	$> 9 \times 10^{-1}$	425.73
Scenario 5	$> 9 \times 10^{-1}$	279.5	Scenario 11	$> 9 \times 10^{-1}$	210.089
Scenario 6	$> 9 \times 10^{-1}$	316.18	-	-	

## 5.4 Chapter summary

This chapter provides insight into the experimental design adopted which is followed by a statistical analysis of the results obtained. In §5.1, the experimental design was discussed in detail with focus on the calculation of the number of vehicles present in the simulation, along with the specifications of the model, the calculation of the warm-up period and a discussion on the statistical methods employed. Furthermore, the computational results were presented in §5.2 followed by the statistical analysis of the results obtained in §5.3.



---



---

## CHAPTER 6

---

# Summary and Conclusions

### Contents

6.1	Thesis summary . . . . .	95
6.2	Appraisal of thesis contributions . . . . .	96

This chapter consists of two sections. Firstly, in §6.1, a chapter-by-chapter summary of the work presented in this thesis. Furthermore, an appraisal of the contributions of this thesis are highlighted in §6.2.

### 6.1 Thesis summary

In the introductory chapter of this thesis, a brief description of TNCs was presented with a specific focus on the growth and impact of these companies. The problem statement considered in this thesis was then provided, followed by the nine thesis objectives to be pursued together with the scope. The chapter concluded with a brief description of the thesis organisation.

In Chapter 2, a literature review pertaining to machine learning in general with a focus on unsupervised learning was presented in accordance with Thesis Objectives I(a) and I(b) of §1.3. The chapter opens with a brief history of machine learning followed by a discussion on the three divisions of machine learning, *i.e.* supervised learning, reinforcement learning and unsupervised learning. The chapter then continues by focussing on clustering, an unsupervised learning technique, consisting of a discussion of popular distance measures, including Manhattan distance, together with popular clustering algorithms divided into three categories, namely: Partitional clustering, hierarchical clustering and density-based clustering.

Chapter 3 contained a review of the basic concepts pertaining to computer simulation modelling in fulfilment of Thesis Objective I(c). Concepts that every simulation model is comprised of are introduced followed by a discussion on the types of simulation models, *i.e.* static *versus* dynamic, deterministic *versus* stochastic and continuous *versus* discrete. The main paradigms of simulation modelling were then presented, namely: Agent-based modelling, discrete event modelling, system dynamics modelling and dynamic systems modelling. Thereafter, the twelve-step methodology aimed to guide a simulation study was discussed, after which the verification and validation steps were discussed in more depth. Penultimately, the notable advantages and disadvantages pertaining to computer simulation modelling were described. The chapter then closed by identifying existing agent-based models of TNCs in literature.

A detailed description of the developed agent-based model of the movement of drivers and the interactions between riders and drivers of TNCs was presented in Chapter 4, in fulfilment of Thesis Objective II. The simulation modelling software used to develop the model was described including the accompanying tools. The movement of drivers of TNC was then presented followed by the strategy employed to model the manner according to which drivers select parking areas along with the clustering strategy selected to cluster historical pick-up locations in order to identify suitable clustered waiting areas in pursuit of Thesis Objectives IV and V. Thereafter, the GUI was described and the functionality of its components verified which was followed by a detailed description of the agents included in the model, namely: Driver agent, rider agent, parking area agent and clustered waiting area agent (in pursuit of Thesis Objective III).

In Chapter 5, the experimental design and the results were presented. The chapter opened with a discussion on the experimental design which included determining the number of drivers, the specifications of the model, the identification of an appropriate warm-up period and, finally, a description of the statistical methods employed. The results of the experimentation were then presented and analysed in fulfilment of Thesis Objective VI. The results of the scenarios were then compared in order to determine if statistically significant differences exist, in fulfilment of Thesis Objective VII .

## 6.2 Appraisal of thesis contributions

The main contributions of this thesis are six-fold. Each of these contributions are briefly summarised and appraised.

**Contribution 1** *The development of an agent-based simulation model within the AnyLogic software environment which models driver behaviour both during rides and between ride-request.*

The simulation model developed includes driver behaviour together with the interaction between drivers and riders. The model may be expanded upon in order for TNCs to experiment with various scenarios such as waiting strategies for drivers, driver-rider pairing strategies, waiting area location placement and the organisation of drivers. The model comprises four agents, namely: Driver, rider, parking area and clustered waiting area. The GUI allows for the adjustment of the number of drivers, together with the waiting area type and the cluster type necessary for experimentation. During a simulation run, the TWT and AWT experienced by drivers can be monitored together with the number of riders waiting for a driver and the number of drivers in the system. The model also includes a map where the movement of drivers and the interaction with riders can be observed.

**Contribution 2** *The validation and verification of the developed simulation model of TNCs.*

The verification of the model was performed with the iterative run controller and debugger within the AnyLogic Software. This ensured the model contained no logical or coding errors. Furthermore, the trace function assisted with ensuring the actions, such as picking up a rider or selection of parking areas, functioned as expected.

The model was validated by employing two methods, *i.e.* parameter variation and face validation. The first parameter variation performed involved varying the number of drivers. It was observed that an increase in the number of drivers resulted in a decrease in the TWT and the AWT, as expected. The next form of parameter variation performed observed the results obtained from two waiting strategies. Strategy 1 included placing drivers at waiting areas with a high number of historical pick-up locations within a 10km travel distance and Strategy 2, on the other hand,

drivers were placed at waiting areas having a relatively small number of pick-up requests within a 10km travel distance. As expected, it was observed that the riders in Strategy 1 experienced a shorter TWT and AWT than those in Strategy 2. The model was face validated by Wakelin [104] — a subject matter expert — who acknowledged value in the model towards decreasing rider waiting time together with driver idle time.

**Contribution 3** *A thorough review of the popular solution approaches adopted in the literature pertaining to clustering algorithms.*

In this thesis, a suitable clustering technique was selected to cluster historical pick-up locations in real-time, however, several clustering algorithms do not meet the criteria necessary, *i.e.* real-time clustering and take into account pick-ups in outlying areas. A thorough literature review was necessary in order to evaluate the various algorithms. This literature review contained a detailed description of three popular clustering techniques, namely: Partitional clustering, hierarchical clustering and density based clustering.

**Contribution 4** *A comparison of clustering approaches together with the comparison of the waiting areas identified by clustering to regular parking areas.*

Extensive experimentation was performed on eleven scenarios, two of which included parking areas where in Scenario 1 the drivers were assigned to the nearest parking area and in Scenario 2 the waiting strategy described in §4.3 was used. The other nine scenarios, on the other hand, included clustered waiting areas, each of which were identified by employing a different cluster type. Scenarios 2 – 11 were compared to Scenario 1 according to four PMIs, namely: Average TWT, maximum TWT, average AWT and total AWT. The most significant improvements found corresponded to Scenario 2, Scenario 8, Scenario 9 and Scenario 11.

**Contribution 5** *The statistical analysis towards ascertaining whether the differences that are present between the scenarios are statistically significant.*

A statistical analysis was performed in order to determine if the improvements identified in *Contribution 5* were of statistical significance at a 5% level of significance. Only three of the four PMIs were observed to include statistically significant differences between the means, which were the average TWT, the maximum TWT and the average AWT. With respect to the average TWT, the improvements observed between Scenario 1 and Scenario 2, Scenario 8, Scenario 9 and Scenario 11 were statistically significant at a 5% level of significance, whereas, according to the maximum TWT, only Scenario 2, Scenario 8 and Scenario 9 exhibited statistically significant improvements in respect of Scenario 2. Lastly, when investigating the average AWT, only Scenario 2 experienced a statistically significant improvement when compared with Scenario 1.

**Contribution 6** *The suggestion of possible ideas for future work following on the contributions made in this thesis.*

The final contribution of this thesis is presented in the next chapter, Chapter 7. These suggestions are aimed at directing the research conducted with respect to the organisation and modelling of drivers of TNCs by providing possible avenues of future work to be investigated as possible follow-up research to the contributions of this thesis.





---



---

## CHAPTER 7

---

# Future Work

### Contents

7.1	Simulation modelling suggestions . . . . .	99
7.2	Solution methodology suggestions . . . . .	101

This chapter contains suggestions for seven possible avenues of follow-up work on the contributions of this thesis. Each suggestion is first stated, followed by a brief discussion and motivation. The suggestions are divided into two sections, namely: Suggestions pertaining to the simulation model, discussed in §7.1, and suggestions with regards to the solution methodology, discussed in §7.2.

### 7.1 Simulation modelling suggestions

This section contains six suggestions for future work related to the simulation model developed in this thesis.

**Suggestion 1** *Implement the clustering algorithm within the simulation.*

The clustering algorithm is currently employed separately to the simulation model, *i.e.* beforehand. The output of the clustering is imported into the model using an Excel spreadsheet. The simulation then reads the Excel spreadsheet in order to obtain the locations of the clustered waiting areas, therefore, experimentation of a different cluster type will require a different Excel spreadsheet to be imported. Consequently, a natural extension to the model would pertain to the inclusion of the clustering in the source code of the simulation model. An Excel spreadsheet will then no longer be required for each clustering type. As a result, this increases the ease with which parameter variation can be employed with regards to the cluster type. The experimentation of various run lengths will be easily performed as the clustering process will no longer be a prerequisite to the execution of the simulation.

**Suggestion 2** *Evaluate the parking area selection methodology.*

A suggestion was proposed by Wakelin, the chief financial officer of ChaffHer, regarding the attractiveness of a parking area. The parking area selection strategy employed in this thesis was adapted from the work of Chen and Nguyen [16], as described in §4.3. Waklin proposed adapting the relationship between the attractiveness of a parking area and the distance to that area such that

$$a_j = \frac{c_j^2}{r_j(p)}.$$

Based on Wakelin's experience, he believes that drivers are more inclined to travel long distances if they are guaranteed a ride-request. It is suggested that this assumption can be compared to the selection strategy employed in this thesis and validated with real world instances (*i.e.* data).

**Suggestion 3** *Validate the simulation model by investigating different areas.*

The model developed in this thesis was applied to NYC ascribed to the fact that data sets pertaining to TNCs are publicly available by the Taxi Limousine Commission of NYC. If other data sets can be retrieved, either through industry partner or being made publicly available, the model can be further validated by testing the performance and adaptability of the model on various areas with different demand patterns. Strategies could also be applied to different areas in order to observe the strategies which perform best under certain conditions.

**Suggestion 4** *Enlarge the scope of the aspects pertaining to the modelling of the driver's behaviour.*

During the development of the simulation model, certain assumptions were made regarding driver behaviour. The first assumption pertaining to the number of drivers remaining constant throughout the simulation. This assumption was due to the lack of information available regarding driver behaviour. The number of drivers was then calculated using information available in the literature. If the information pertaining to the frequency at which drivers enter and leave the system, together with the number of drivers available at certain time periods, could be obtained (*e.g.* by means of a collaborative effort with TNC subject-matters) and subsequently employed as part of the modelling approach, the accuracy of the model could be greatly improved. Furthermore, since the data set comprised completed ride-requests, it was assumed that drivers can not reject ride-requests, however, this may occur for various reasons, for example, there could be a long travel distance between the driver and the rider or the rider could have a low rating. In order to incorporate this aspect, information pertaining to the rejection rate of drivers is required which could be obtained by consulting with a TNC. Driver behaviour could also be further sophisticated and improved by including the effects of surge pricing (*i.e.* an increase in the cost of rides in areas of high demand).

**Suggestion 5** *Perform extensive collaboration with subject matter expert towards improving modelling approach.*

This model was validated upon completion by the chief financial officer of Chauffer — a subject matter expert. In order to further enhance the capabilities of this model and increase the accuracy, it is suggested to work with a subject matter expert during development to ensure the aspects of the model reflect the real world. The subject matter expert could provide pertinent insight into possible scenarios which can be evaluated as well as the PMIs that are most valuable to the industry. This could also assist in attaining data sets that would otherwise not be available.

**Suggestion 6** *Implement various driver-rider pairing strategies.*

There are many strategies that can be employed to match drivers to riders. The strategy employed in this thesis relates to matching the nearest driver-rider pair. It could, however, be a worthwhile endeavour to evaluate the various proposed driver-rider pairing strategies. A strategy was proposed by Chewei [108], referred to as *batching*, according to which instead of observing a single request at a time, requests are collected over a certain time period. The drivers and riders are then paired so as to minimise the overall travel time of drivers and, therefore, also minimise passenger waiting time. Any riders that are not paired during this time period are then paired in the next time period. The batching driver-rider pairing strategy, which is one of many pairing strategies, is visually depicted in Figure 7.1.

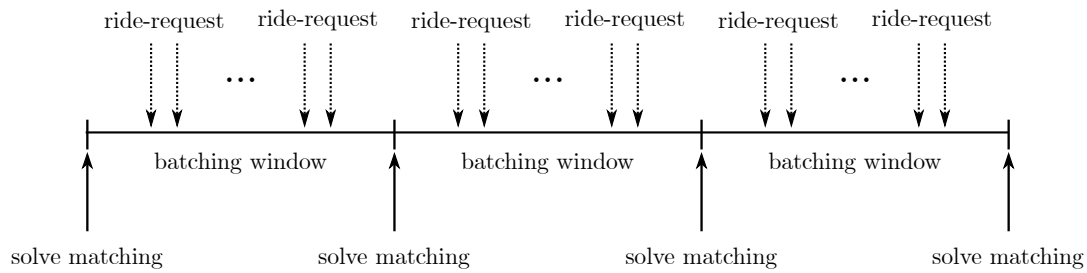


FIGURE 7.1: The batching strategy proposed by Chewei [108] towards pairing drivers and riders.

## 7.2 Solution methodology suggestions

This section contains a further two suggestions for future work related to the solution methodology employed in this thesis.

**Suggestion 7** *Explore the effectiveness of improvements of the  $K$ -means algorithm.*

Only one clustering algorithm (*i.e.*  $K$ -means) was employed when clustering the historical pick-up locations. A natural extension of the work presented in this thesis, however, would be to incorporate alternative clustering algorithms, more specifically improvements to the  $K$ -means algorithm. Zhang [109] proposed an improvement to the original  $K$ -means algorithm. The main modifications of the algorithm pertain to the cluster centre according to which each data point is assigned weights based on the distance to the cluster centre. The proposed algorithm was tested on two data sets where an improvement in clustering accuracy was observed when compared with the original  $K$ -means algorithm using random initial centre selection. Furthermore, Zhang [110] also proposed an improvement to  $K$ -means according to which the focus was on identifying the initial cluster centres. Experiments were performed on synthetic data sets where improvements were observed in clustering accuracy when compared with  $K$ -means using random initial centres.

**Suggestion 8** *Perform additional parameter variations regarding the cluster type.*

In this thesis, nine variations of cluster types were investigated. A possible suggestion for future work pertains to the investigation of other cluster types with the possible objective of minimising passenger waiting time. Further investigation can also be performed towards minimising driver idle time. This problem could possibly also evolve into a *multi-objective optimisation* problem by minimising both the number of clustered waiting areas together with the rider waiting time. TNCs often have designated parking areas for their drivers to wait at between rides, therefore, it is advantageous to observe the location of these clustered waiting areas if both the number of clustered waiting areas together with the riders' waiting time are minimised.



---

## References

- [1] AGGARWAL C & REDDY C (EDS), 2013, *Data clustering: Algorithms and applications*, CRC Press, Boca Raton (FL).
- [2] ANKERST M, BREUNIG MM, KRIEGEL HP & SANDER J, 1999, *OPTICS: Ordering points to identify the clustering structure*, Proceedings of the 2<sup>nd</sup> ACM Sigmod Record, Philadelphia (PA), pp. 49–60.
- [3] ARTHUR D & VASSILVITSKII S, 2007, *K-means++: The advantages of careful seeding*, Proceedings of the 18<sup>th</sup> Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans (LA), pp. 1027–1035.
- [4] BANKS J (ED), 1998, *Handbook of simulation: principles, methodology, advances, applications, and practice*, John Wiley & Sons, New York (NY).
- [5] BANKS J, CARSON I, NELSON BL & NICOL DM, 2010, *Discrete-event system simulation*, 5<sup>th</sup> Edition, Pearson (NJ).
- [6] BERKHIN P, 2006, *A survey of clustering data mining techniques*, pp. 25–71 in KOGAN J, NICHOLAS C & TEBoulLE M (EDS), *Grouping multidimensional data*, Springer, Berlin.
- [7] BISCHOFF J & MACIEJEWSKI M, 2016, *Autonomous taxicabs in Berlin – A spatiotemporal analysis of service performance*, Transportation Research Procedia, **19**.
- [8] BISHOP CM, 2006, *Pattern recognition and machine learning*, Springer, New York (NY).
- [9] BOLEY D, 1998, *Principal direction divisive Partitioning*, Data Mining and Knowledge Discovery, **2**, pp. 325–344.
- [10] BORA DJ & GUPTA AK, 2014, *Effect of different distance measures on the performance of K-means algorithm: An experimental study in Matlab*, International Journal of Computer Science and Information Technologies, **5(2)**, pp. 2501–2506.
- [11] BORSHCHEV A & FILIPPOV A, 2004, *From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools*, Proceedings of the 22<sup>nd</sup> International conference of the System Dynamics Society, Oxford, England.
- [12] BURKARDT J, 2009, *Weighted and continuous clustering*, lecture notes, Mathematics, CS 4414, Virginia Tech, delivered 25 September 2009.
- [13] CAMPBELL M, HOANE AJ & HSU FH, 2002, *Deep Blue*, Artificial Intelligence, **134(1-2)**, pp. 57–83.
- [14] CASSISI C, FERRO A, GIUGNO R, PIGOLA G & PULVIRENTI A, 2013, *Enhancing density-based clustering: Parameter reduction and outlier detection*, Information Systems, **38(3)**, pp. 317–330.
- [15] CHAUHAN R, BATRA P & CHAUDHARY S, 2014, *A survey on density-based clustering algorithms*, International Journal of Computer Science and Technology, **5**, pp. 169–171.

- [16] CHENG S.-F & NGUYEN TD, 2011, *Taxisim: A multiagent simulation platform for evaluating taxi fleet operations*, Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Lyon, France, pp. 14–21.
- [17] CLEWLOW RR & MISHRA GS, 2017, *Disruptive transportation: The adoption, utilization, and impacts of ride-hailing in the United States*, (Unpublished) Technical Report, Institute of Transportation Studies, University of California, Davis (CA).
- [18] COVER T & HART P, 1967, *Nearest neighbor pattern classification*, IEEE transactions on information theory, **13**(1), pp. 21–27.
- [19] DASGUPTA S, 2008, *The k-means clustering problem*.
- [20] DEFAYS D, 1977, *An efficient algorithm for a complete link method*, The Computer Journal, **20**(4), pp. 364–366.
- [21] DICTIONARY.COM, 2018, *Information age*, [Online], [Cited April 2018], Available from <http://www.dictionary.com/browse/information-age>.
- [22] DUNNETT CW, 1955, *A multiple comparison procedure for comparing several treatments with a control*, Journal of the American Statistical Association, pp. 1096–1121.
- [23] DUNNETT CW, 1980, *Pairwise multiple comparisons in the unequal variance case*, Journal of the American Statistical Association, pp. 796–800.
- [24] EREMENKO K, DE PONTEVES H & SUPERDATASCIENCE TEAM S, 2018, *Machine learning A–Z: Hands-on Python & R in data science*, [Online], [Cited May 2018], Available from <https://www.udemy.com/machinelearning/>.
- [25] ESTER M, KRIEGEL HP, SANDER J & XU X, 1996, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the the 2<sup>nd</sup> International Conference on Knowledge Discovery and Data Mining, Portland (OR), pp. 226–231.
- [26] FERRUCCI D, LEVAS A, BAGCHI S, GONDEK D & MUELLER ET, 2013, *Watson: Beyond Jeopardy!*, Artificial Intelligence, **199**, pp. 93–105.
- [27] FLOREK K, ŁUKASZEWICZ J, PERKAL J, STEINHAUS H & ZUBRZYCKI S, 1951, *Sur la liaison et la division des points d'un ensemble fini*, Colloquium Mathematicae, **2**(3-4), pp. 282–285.
- [28] FORINA M, ARMANINO C & RAGGIO V, 2002, *Clustering with dendrograms on interpretation variables*, Analytica Chimica Acta, **454**(1), pp. 13–19.
- [29] FORRESTER JW, 1958, *Industrial Dynamics. A major breakthrough for decision makers*, Harvard Business Review, **36**(4), pp. 37–66.
- [30] GAN G, 2007, *Data clustering: Theory, algorithms, and applications*, Society for Industrial and Applied Mathematic, Philadelphia (PA).
- [31] GOODALL W, FISHMAN T, DIXON S & PERRICOS C, 2015, *Transport in the Digital Age*, (Unpublished) Technical Report, Deloitte.
- [32] GORDON G, 1961, *A general purpose systems simulation program*, Proceedings of the December 12-14, 1961, Eastern Joint Computer Conference: Computers - Key to Total Systems Control, Washington, D.C. (WA), pp. 87–104.
- [33] GRAU JMS & ROMEU MAE, 2015, *Agent based modelling for simulating taxi services*, Procedia Computer Science, **52**, pp. 902–907.
- [34] GRIGORYEV I, 2018, *AnyLogic 7 in three days: A quick course in simulation modeling*, AnyLogic North America.

- [35] GUHA S, RASTOGI R & SHIM K, 2001, *Cure: An efficient clustering algorithm for large databases*, Information Systems, **26(1)**, pp. 35–58.
- [36] HALL DW & PESENTI J, 2017, *Growing the artificial intelligence industry in the UK*, Independent review for the Department for Digital, Culture, Media and Sport/Department for Business, Energy and Industrial Strategy.
- [37] HALL JV & KRUEGER AB, 2016, *An Analysis of the Labor Market for Uber’s Driver-Partners in the United States*, (Unpublished) Technical Report, National Bureau of Economic Research.
- [38] HAN J, KAMBER M & TUNG A, 2001, *Spatial clustering methods in data mining: A survey*, Data Mining and Knowledge Discovery.
- [39] HAN J, PEI J & KAMBER M, 2011, *Data mining: Concepts and techniques*, 3<sup>rd</sup> Edition, Morgan Kaufmann, Amsterdam.
- [40] HINNEBURG A & KEIM DA, 1998, *An efficient approach to clustering in large multimedia databases with noise*, Proceedings of the 4<sup>th</sup> International Conference on Knowledge Discovery and Data Mining, New York (NY), pp. 58–65.
- [41] HOMER JB & HIRSCH GB, 2006, *System dynamics modeling for public health: Background and opportunities*, American Journal of Public Health, **96(3)**, pp. 452–458.
- [42] HUANG Z, 1998, *Extensions to the k-Means algorithm for clustering large data sets with categorical values*, Data Mining and Knowledge Discovery, **2(3)**, pp. 283–304.
- [43] JAIN A, MURTY M & FLYNN P, 1999, *Data clustering: A review*, ACM Computing Surveys, **31(3)**, pp. 264–323.
- [44] JAIN AK, 2010, *Data clustering: 50 years beyond K-means*, Pattern Recognition Letters, **31(8)**, pp. 651–666.
- [45] JOHNSON SC, 1967, *Hierarchical clustering schemes*, Psychometrika, **32(3)**, pp. 241–254.
- [46] KAGGLE, 2019, *Uber Pickups in New York City*, [Online], [Cited March 2019], Available from <https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city>.
- [47] KASSAMBARA A, 2017, *Practical guide to cluster analysis in R: Unsupervised machine learning*, Statistical Tools for High-Throughput Data Analysis.
- [48] KAUFMAN L, 1990, *Finding groups in data : An introduction to cluster analysis*, John Wiley & Sons, New York (NY).
- [49] KETCHEN DJ & SHOOK CL, 1996, *The application of cluster analysis in strategic management research: An analysis and critique*, Strategic Management Journal, **17(6)**, pp. 441–458.
- [50] KIM H, YANG I & CHOI K, 2011, *An agent-based simulation model for analyzing the impact of asymmetric passenger demand on taxi service*, Korean Society of Civil Engineers Journal of Civil Engineering, **15(1)**, pp. 187–195.
- [51] KODINARIYA TM & MAKWANA PR, 2013, *Review on determining number of clusters in K-means clustering*, International Journal of Advance Research in Computer Science and Management Science, **1(6)**, pp. 90–95.
- [52] KRIEGEL HP, KRÖGER P, SANDER J & ZIMEK A, 2011, *Density-based clustering*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, **1(3)**, pp. 231–240.
- [53] KUMAR V, 2000, *An introduction to cluster analysis for data mining*, lecture notes, Data Mining CSci 8980, University of Minnesota, delivered 2 September 2000.



- [54] LANCE GN & WILLIAMS WT, 1967, *A general theory of classificatory sorting strategies: 1. Hierarchical systems*, The Computer Journal, **9(4)**, pp. 373–380.
- [55] LAW AM, 2015, *Simulation modeling and analysis*, McGraw-Hill, New York(NY).
- [56] LIGHTHILL J, SUTHERLAND N, NEEDHAM R & LONGUET-HIGGINS H, 1973, *Artificial intelligence: A paper symposium*, Science Research Council, London.
- [57] LOKHANDWALA M & CAI H, 2018, *Dynamic ride sharing using traditional taxis and shared autonomous taxis: A case study of NYC*, Transportation Research Part C: Emerging Technologies, **97**, pp. 45–60.
- [58] MACAL CM & NORTH MJ, 2010, *Tutorial on agent-based modelling and simulation*, Journal of Simulation, **4(3)**, pp. 151–162.
- [59] MACQUEEN J, 1967, *Some methods for classification and analysis of multivariate observations*, Proceedings of the 5<sup>th</sup> Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297.
- [60] MCCARTHY J, MINSKY ML, ROCHESTER N & SHANNON CE, 2006, *A proposal for the dartmouth summer research project on artificial intelligence*, AI Magazine, **27(4)**, pp. 12.
- [61] MCQUITTY LL, 1957, *Elementary linkage analysis for isolating orthogonal and oblique types and typal relevancies*, Educational and Psychological Measurement, **17(2)**, pp. 207–229.
- [62] MIRKIN BG, 2005, *Clustering for data mining : A data recovery approach*, Chapman & Hall/CRC, Boca Raton (FL).
- [63] MITCHELL TM, 1997, *Machine Learning*, McGraw-Hill, New York(NY).
- [64] MONTGOMERY DC & RUNGER GC, 2014, *Applied statistics and probability for engineers*, 6<sup>th</sup> Edition, John Wiley & Sons.
- [65] MURTAGH F, 1983, *A survey of recent advances in hierarchical clustering algorithms*, The Computer Journal, **26(4)**, pp. 354–359.
- [66] NG RT & HAN J, 1994, *Efficient and Effective Clustering Methods for Spatial Data Mining*, Proceedings of the 20<sup>th</sup> International Conference on Very Large Data Bases, San Francisco (CA), pp. 144–155.
- [67] NOURINEJAD M & ROORDA MJ, 2016, *Agent based model for dynamic ridesharing*, Transportation Research Part C, **64**, pp. 117–132.
- [68] NYC TAXI & LIMOUSINE COMMISSION, 2019, *Neighborhood Tabulation Areas*, [Online], [Cited November 2019], Available from <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>.
- [69] OLSON CF, 1995, *Parallel algorithms for hierarchical clustering*, Parallel Computing, **21(8)**, pp. 1313–1325.
- [70] OPENDATA N, 2019, *Neighborhood tabulation areas*, [Online], [Cited August 2019], Available from <https://data.cityofnewyork.us/City-Government/Neighborhood-Tabulation-Areas/cpf4-rkhq>.
- [71] PATERLINI S & KRINK T, 2006, *Differential evolution and particle swarm optimisation in partitionial clustering*, Computational Statistics and Data Analysis, **50(5)**, pp. 1220–1247.
- [72] PEGDEN CD, SADOWSKI RP & SHANNON RE, 1995, *Introduction to Simulation Using SIMAN*.
- [73] ROSENBLATT F, 1957, *The perceptron, a perceiving and recognizing automaton Project Para*, Report 85-60-1.



- [74] ROUSSEEUW PJ, 1987, *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*, Journal of computational and applied mathematics, **20**, pp. 53–65.
- [75] ROYAL SOCIETY, 2017, *Machine learning: The power and promise of computers that learn by example*.
- [76] RUSSELL SJ & NORVIG P, 2010, *Artificial intelligence: A modern approach*, 3<sup>rd</sup> Edition, Pearson Education Limited.
- [77] SAMUEL AL, 1959, *Some studies in machine learning using the game of checkers*, IBM Journal of Research and Development, **3(3)**, pp. 210–229.
- [78] SANTINI M, 2016, *Advantages and disadvantages of k-means and Hierarchical clustering (Unsupervised Learning)*, lecture notes, Machine Learning for Language Technology ML4LT, Uppsala University, delivered 12 December 2016.
- [79] SAYGIN AP, CICEKLI I & AKMAN V, 2000, *Turing test: 50 years later*, Minds and Machines, **10(4)**, pp. 463–518.
- [80] SCHALLER B, 2017, *Unsustainable? The growth of app-based ride services and traffic, travel and the future of New York City*.
- [81] SEJNOWSKI T, 1987, *Net talk: A parallel network that learns to read aloud*, Complex Systems, **1**, pp. 145–168.
- [82] SHAHEEN JA, “Simulating the Ridesharing Economy: The Individual Agent Metro-Washington Area Ridesharing Model”, in: *Complex Adaptive Systems*, Springer, 2019, pp. 143–168.
- [83] SHALIZI C, 2009, *Distances between Clustering, Hierarchical Clustering*, lectures notes, Data Mining 36-350, Carnegie Mellon University, delivered 18 September 2009.
- [84] SHANNON RE, 1998, *Introduction to the art and science of simulation*, Proceedings of the 30<sup>th</sup> Conference on Winter Simulation, Washington, D.C., USA, pp. 7–14.
- [85] SHARMA N, BAJPAI A & LITORIYA R, 2012, *Comparison the various clustering algorithms of weka tools*, International Journal of Emerging Technology and Advanced Engineering, **2(5)**, pp. 73–80.
- [86] SHOKRI S & ISMAIL M, 1984, *K-means-type algorithms: A generalized convergence theorem and characterization of local optimality*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **(1)**, pp. 81–87.
- [87] SIBSON R, 1973, *SLINK: An optimally efficient algorithm for the single-link cluster method*, The Computer Journal, **16(1)**, pp. 30–34.
- [88] SILVER D, SCHRITTWIESER J, SIMONYAN K, ANTONOGLIOU I, HUANG A, GUEZ A, HUBERT T, BAKER L, LAI M & BOLTON A, 2017, *Mastering the game of Go without human knowledge*, Nature, **550(7676)**, pp. 354.
- [89] SINGH A, YADAV A & RANA A, 2013, *K-means with three different distance metrics*, International Journal of Computer Applications, **67(10)**.
- [90] SISODIA D, SINGH L, SISODIA S & SAXENA K, 2012, *Clustering techniques: A brief survey of different clustering algorithms*, International Journal of Latest Trends in Engineering and Technology (IJLTET), **1(3)**, pp. 82–87.
- [91] SNEATH PH, 1957, *The application of computers to taxonomy*, Microbiology, **17(1)**, pp. 201–226.
- [92] SNEDECOR GW & COCHRAN WG, 1989, *Statistical Methods*, Iowa state University press, Ames, Iowa (IA).

- [93] STEINBACH M, KARYPIS G & KUMAR V, 2000, *A comparison of document clustering techniques*, Proceedings of the International KDD Workshop in Text Mining.
- [94] STEWART J, 2012, *Calculus*, 7<sup>th</sup> Edition, Cengage Learning.
- [95] STOLINE MR & URY HK, 1979, *Tables of the studentized maximum modulus distribution and an application to multiple comparisons among means*, Technometrics, **21(1)**, pp. 87–93.
- [96] STRUYF A, HUBERT M & ROUSSEEUW P, 1997, *Clustering in an object-oriented environment*, Journal of Statistical Software, **1(4)**, pp. 1–30.
- [97] TANG V & VIJAY S, 2001, *System dynamics: Origins, development, and future prospects of a method*, Research Seminar in Engineering Systems.
- [98] TESAURO G, 1992, *Practical issues in temporal difference learning*, Machine Learning, **8(3)**, pp. 257–277.
- [99] THOTA S, 2016, *The evolution of machine learning*, [Online], [Cited December 2018], Available from <http://www.smdi.com/evolution-machine-learning>.
- [100] TIBSHIRANI R, WALTHER G & HASTIE T, 2001, *Estimating the number of clusters in a data set via the gap statistic*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), **63(2)**, pp. 411–423.
- [101] TURING AM, 1950, *Computing machinery and intelligence*, Mind, **59**, pp. 433–60.
- [102] UBER TECHNOLOGIES INC., 2019, *Driving in New York City*, [Online], [Cited September 2019], Available from <https://www.uber.com/drive/new-york/>.
- [103] UCI, 1988, *Iris data set*, [Online], [Cited June 2018], Available from <https://archive.ics.uci.edu/ml/datasets/iris>.
- [104] WAKELIN G, 2019, CFO and founder at *ChaufHer*, [Personal Communication], Contactable at [greg@chaufher.com](mailto:greg@chaufher.com).
- [105] WARD JR JH, 1963, *Hierarchical grouping to optimize an objective function*, Journal of the American Statistical Association, **58(301)**, pp. 236–244.
- [106] WILKS DS, 2011, *Chapter 15 - Cluster Analysis*, pp. 603–616 in WILKS DS (ED), *Statistical Methods in the Atmospheric Sciences*, Academic Press.
- [107] XU D & TIAN Y, 2015, *A Comprehensive Survey of Clustering Algorithms*, Annals of Data Science, **2(2)**, pp. 165–193.
- [108] YAN C, ZHU H, KOROLKO N & WOODARD D, 2019, *Dynamic pricing and matching in ride-hailing platforms*, Naval Research Logistics.
- [109] ZHANG C & FANG Z, 2013, *An improved K-means clustering algorithm*, Journal of Information and Computational Science, **10**, pp. 193–199.
- [110] ZHANG C & XIA S, 2009, *K-means clustering algorithm with improved initial center*, Proceedings of the 2009 2<sup>nd</sup> International Workshop on Knowledge Discovery and Data Mining, Moscow, Russia.
- [111] ZHANG T, RAMAKRISHNAN R & LIVNY M, 1996, *BIRCH: An efficient data clustering method for very large databases*, Proceedings of the 2<sup>th</sup> ACM Sigmod Record, pp. 103–114.