Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Profit Driven Drone Scheduling in Last Mile Food Delivery Networks

Author:
Andrew Pearcy

Supervisor:
Prof. William Knottenbelt

Second Marker:
Prof. Peter Harrison

June 19, 2019

Submitted in partial fulfillment of the requirements for the MEng Joint Mathematics and Computing of Imperial College London

Abstract

Autonomous delivery has the potential to dramatically cut the costs of courier services. Significant research has been undertaken regarding autonomous vehicles, but recent developments in drone technology offer an alternative delivery solution. Multiple organisations are investigating the potential of drones for package delivery. Significant media attention has been given to Amazon Prime Air, but little research has been undertaken to explore alternative use cases.

Meal delivery is one avenue where drones could significantly reduce lead times. The intrinsic value associated with delivering food faster suggests that drone delivery could be an ideal solution. The emergence of Deliveroo and Just Eat have demonstrated the consumer demand for food delivery services. A drone's ability to overcome existing infrastructure constraints and deliver food faster than any ground based courier makes it the perfect solution to the meal delivery problem.

This project explores the use of drones in a multi-source node delivery network, where orders can be scheduled from any source node to any delivery location within that nodes service radius. We present a visual simulation of the drone delivery network in an urban environment and assess the performance of multiple time-value profit-driven scheduling algorithms. We determine that the reallocation abilities of global schedulers to cope with demand in localised delivery models are only beneficial in highly dynamic scenarios. Finally, we deduce that drone transit time is a key factor in minimising lead times in distributed meal courier delivery.

Acknowledgements

This project could not have been completed without the help of those around. I would like to thank the following people for their support and encouragement:

- My supervisor Professor William Knottenbelt for giving me the opportunity to pursue a state of the art, fascinating and relevant topic. Without your encouragement this project would have been half of what it is.
- The Simulation team at Ocado Technology from whom I learnt everything I know about constructing Simulations.
- My housemates who, on numerous occasions, have stayed up late into the night to bounce ideas around.
- My girlfriend, for dealing with my stress whenever I hit a road block and was ready to give up.
- The Imperial College Canoe Club, which has allowed me to step away from my academic work, clear my mind and come back with new ideas.
- My parents, for pushing me to go to university in the first place. Without them I would not be here.

Contents

List of Figures

Part I

Foundational Knowledge

Chapter 1

Introduction

Since 2016, the number of parcel deliveries has grown at approximately 40% year on year(46). Much of this growth can be contributed to the acceleration of e-commerce. In the US alone, e-commerce sales have risen from 5.1% of total retail sales in 2007 to 13% of retail sales in 2017, increasing from $136 billion to $453 billion(24). However, this enthusiasm for e-commerce has led to consumers wanting their products ever faster. First came Amazon Primes next day delivery, but consumers who need their products even faster can pay a premium for Same-Day delivery. Next emerged Amazon Prime Now, which allows customers to have their packages delivered within an hour. Amazon Prime's guaranteed next day delivery epitomises the current consumer mentality. The ever-increasing consumer demand to receive packages faster has led to a bottleneck in the last-mile delivery logistics. A survey carried out by McKinsey noted how younger consumers are more inclined to choose same-day or instant delivery(29), but that the price of delivery is still the over-arching decision factor for consumers. On the supplier end, these last mile delivery costs often reach, or even exceed, 50% of total parcel delivery costs(29). Therefore, there is a need for suppliers to reduce their costs, whilst satisfying the consumer demand to receive their packages faster and faster.

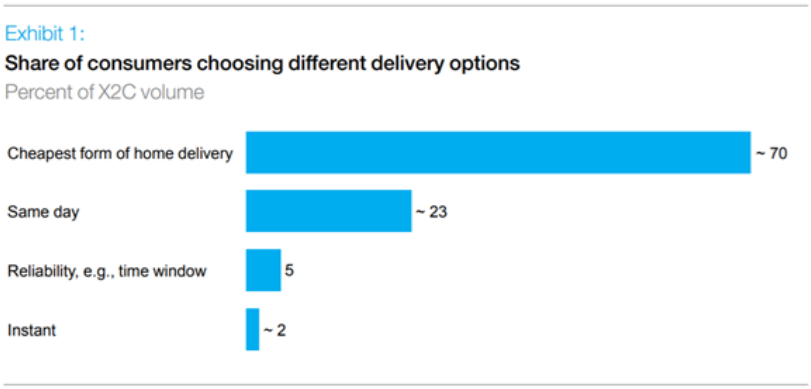

Figure 1.1: Proportion of consumers opting for different delivery options (29)

The rapid increase in package delivery has resulted in a surge of delivery vehicles on the roads, further congesting already jammed streets. Moreover, the additional drivers incur higher labour costs for the companies, resulting in higher delivery costs. Research has been undertaken to try and implement ground-based autonomous home delivery(27); however, the developments in drone technology have offer an alternative solution. Amazon Prime Air was first advertised in 2013, with the promise of being able to deliver packages within 30 minutes(10). Moreover, their reports claim they have made significant progress in collision avoidance and noise reduction. Amazon Prime Air was able to successfully deliver its first package on the 7th December 2016 in Cambridge, UK. With this knowledge, it is apparent that the control and route planning of delivery drones is not an issue. What we will be investigating is the scheduling of drone deliveries based upon a time discounted cost model. This time discounted cost model aims to fulfil the consumers time requirements whilst maximising both profit and drone utilisation for the supplier. However, we must consider what circumstances warrant a time discounted model. What product does a consumer desire which they are willing to pay more for to receive faster? Amazon Prime Air is attempting to fulfil this need by delivering goods to its customers in 30 minutes or less, but what other products fall into this small time span window. One example might be medical supplies. In these instances, delivery of crucial medical supplies is time critical and it makes sense that the consumer would want to pay more for the products to arrive as fast as possible. Another industry is food delivery. With the heightened interests in food start ups such as Just Eat, Deliveroo and UberEats, the Food Delivery Sector is rapidly expanding.

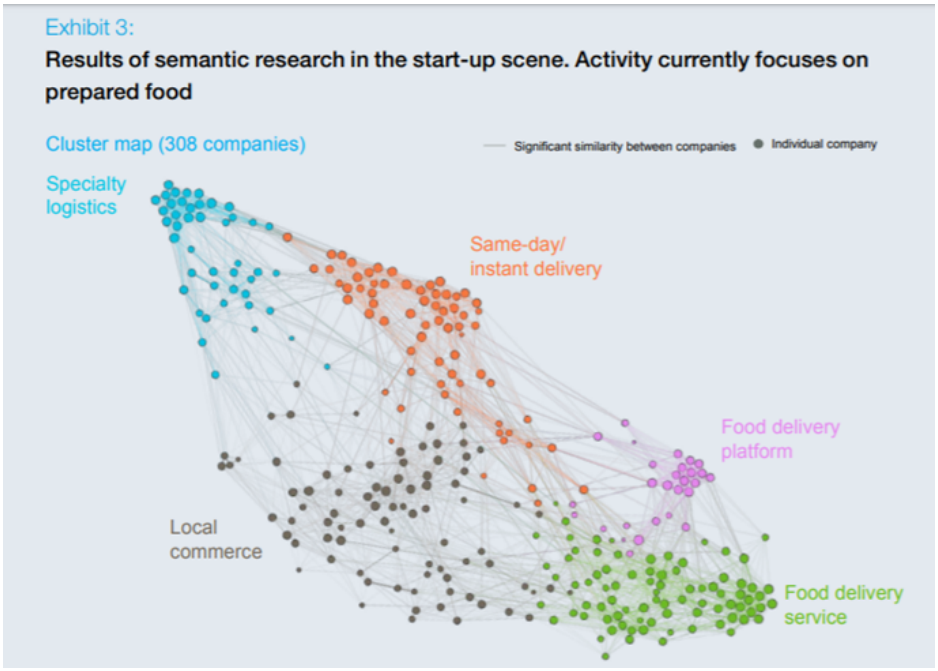

Figure 1.2: Spread of Delivery Start-Ups by sector (29)

As seen in the figure below, Food delivery, Food delivery platform, and Same day/instant

delivery start-ups make up the majority of delivery start-ups. The Food Delivery sector has seen an explosion of interest, with their being 83 start-ups whose median year of founding was 2012.

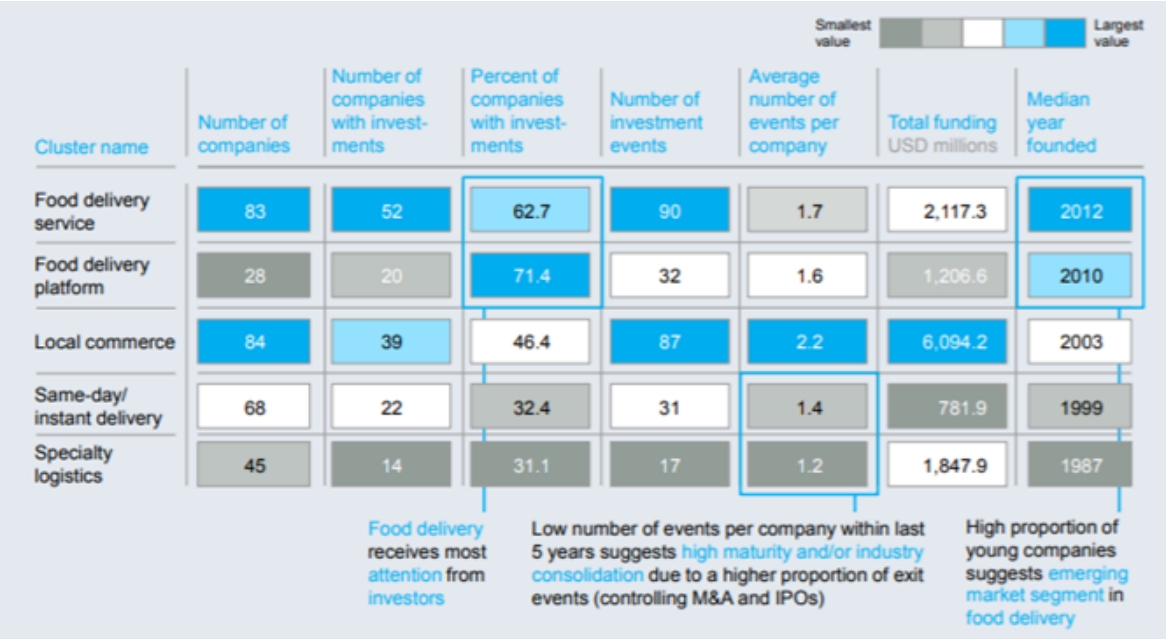

Figure 1.3: Analysis of Delivery Start-Ups (29)

1.1 Contributions

This project expands upon the work of previous Imperial student(6), who proposed the Least Lost Value (LLV) scheduling algorithm, and compared it against against First Come First Served (FCFC) and Shortest Job First (SJF) in a single source node model. We implement our own Unity based model to simulate and visualise a multi-source node drone delivery network.

We present 4 key contributions:

- An extendable Drone Delivery Network architecture and visualisation tool in Unity (Chapters 4 and 6).
- An extension of Least Lost Value to multi-source node networks and the implementation of a dynamic programming inspired Just In Time scheduler (Chapter 5).
- An analysis of time-valued profit driven localised scheduling over global scheduling in a Domino's Pizza delivery model (Chapter 7).
- An analysis of time-valued profit driven schedulers in a Just Eat delivery model at different geographical scales with different numbers of source nodes and drone couriers (Chapter 8).

Chapter 2

Background

Here we provide an overview of some of the recent technological developments and research into the last mile delivery sector. We focus on a few specific companies, their implementations, use cases, and developments. This will give us a strong understanding of the current capabilities of drones and what assumptions we can make for our simulation. We will also cover some of the regulations currently hindering the execution of drone delivery networks, and perform an analysis of what research has already been undertaken. Finally, we will give an overview of scheduling algorithms used in the implementation and assessment.

2.1 Last Mile Delivery

2.1.1 Autonomous Vehicles - Ocado

Figure 2.1: Ocado Autonomous Delivery Van(27)

There have been multiple companies investing in autonomous vehicle technologies, from Tesla to Google, but few have ventured into the autonomous delivery sector. One of the few forays into this sector is Ocado's driverless delivery vehicle, developed in partnership with Oxbotica(27). Ocado's attempts to cut costs on last mile delivery entailed a test route through the backstreets of Woolwich London in June 2017, with the company hoping to commercial launch this system by the end of 2019.

Ocado is an interesting case study when it comes to delivery scheduling. Unlike Amazon, Ocado has its own internal system which handles orders from beginning to end. Most companies delegate delivery to a courier service, but Ocado constructed their own in-house system. Moreover, they have been operating their internal delivery system since 2002, giving them years of experience on last mile delivery scheduling(1). They were the first to introduce one hour delivery slots, years ahead of Amazon Prime Now, and have become one of the market leaders in large scale simulations. All of their Customer Fulfilment Centres were extensively modelled and simulated before any warehouse foundations were laid. Their latest warehouse design entails autonomous robots moving along a grid, retrieving totes of groceries to be picked by individuals(23). It is this simulation, which Ocado wrote from the ground up, we gained inspiration from. This was due to time spent working in Ocados Simulation Algorithm Development team, where a summer was spent trying to optimise the warehouse throughput via alternative robot scheduling algorithms.

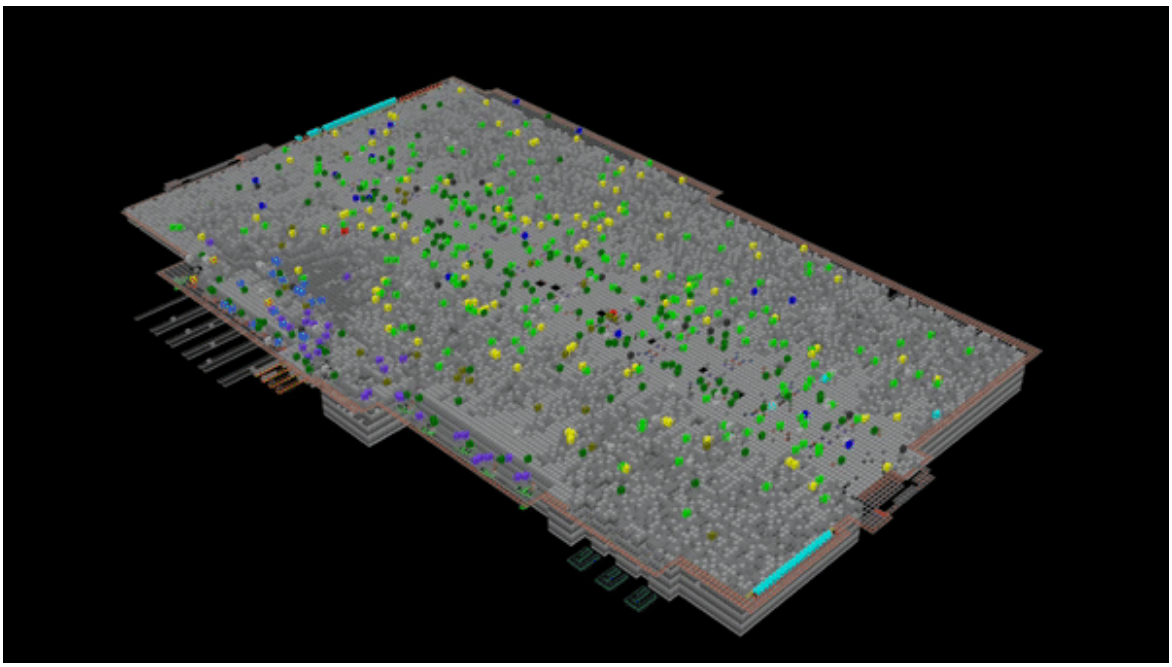

Figure 2.2: Simulation of Ocado's new robotic warehouses(23)

2.1.2 Amazon Prime Air

Amazon Prime Air is perhaps the most developed autonomous drone solution. Since it was first announced in 2013, there has not been a lot of information officially released about the project. In December 2016, Amazon Prime Air successfully carried out its first test flight and delivered a package in Cambridge, UK. Reports have stated that engineers have been working on automated collision avoidance with both static and dynamic obstacles, allowing for the drones to avoid birds(43). Whilst implementation and testing has been limited by the Civil Aviation Authority, Amazon has been able to take advantage of relaxed aviation regulations in and around Cambridge.

The company states that first iterations of the service will be available to those in rural areas, with sufficient space for a drone to land. For those living in urban environments, additional challenges must be overcome before the service can be fully implemented. This is due to the current landing mechanism employed by Amazon Prime Air. At the moment, experiments have only been undertaken where the consumer places a landing mat in their back garden to act as a homing beacon for the drones, and to signal a safe space to land.(30)

The most recent Amazon Prime Air drone model measures 91.4cm in diameter, has 8 motors and an estimated flight time of 30 minutes(31). They are said to have a top speed of 80km/h, giving them a range of 16km. The drones will be able to carry packages weighing up to 2.3kg which incorporates 86% of products currently available on Amazons marketplace(26).

Whilst there are not many accurate studies of the potential cost savings that drone delivery could introduce, there are a few studies which give us an idea of the running costs. Drones have been employed to transport medicines and drug samples throughout Lesotho where the road infrastructure is poor. The creators of the system estimated that their 6.2 miles trip cost only 24cents. Compare this with the cost of Amazon Prime Now $7.99 and it gives an indicator of the potential gain drone delivery systems could bring(47). Having spoken to a logistics manager at one of the Amazon Fulfilment Centres in the UK, I have discovered that the Amazon Prime service can be a huge expense to the business. To such an extent, that warehouse managers have been known to hire vans and personally deliver prime packages to ensure delivery before Christmas, costing in excess of £1000 for a single delivery.

2.2 Time Sensitive Deliveries

Ocado is the prime example of a company which is able to schedule its deliveries to within 1 hour intervals thanks to its end to end control of its entire logistics system. It has had huge success in its last mile delivery service, which allows it the full flexibility to cater to its customers‘ needs. Amazon on the other hand, is trying to gain

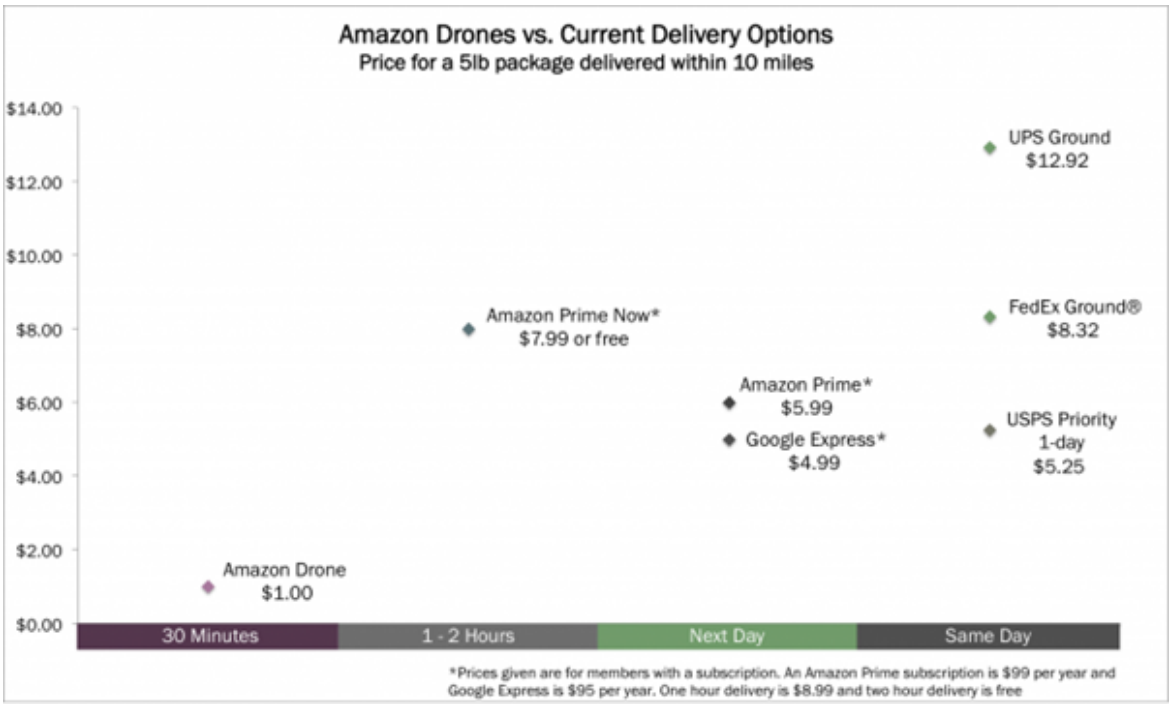

Figure 2.3: Cost analysis of different delivery methods

greater control over its last mile delivery service. It is seeking to decrease the lead time from making an order, to delivery, preying on the customer demand to receive their products faster. However, neither of these two companies directly operate with time sensitive deliveries.

With time sensitive deliveries we are referencing products which must arrive within specified time constraints otherwise their value is diminished. This is quite a small category of products; however the two notable examples are medicines and food delivery.

2.2.1 DHL Parcelcopter

DHL was the first parcel delivery service to successfully integrate drones into its delivery chain(17). In 2016, they were able to complete a three-month trial in which their drones made autonomous deliveries to customers in the Bavarian Alps. DHL's first commercial use of its Parcelcopter was in late 2014, when it delivered supplies to the North Sea island of Juist. The Parcelcopter 2.0 was used to deliver medicines and other urgent goods over the open sea. It successfully completed numerous 12km flights over the open sea at an altitude of 50m, and at speeds of up to 40mph. It made regular express deliveries for a pharmacy on the island, and made weekend deliveries of supplies when ferries and flights were unavailable. This allowed customers to receive their prescriptions and receive urgent medication much faster than conventional delivery methods.

Figure 2.4: DHL's Parcelcopter 3.0(17)

This demonstrates that there is a need for medical supplies to be delivered in a timely fashion in hard to reach areas. DHL acknowledges that this is one of the use cases for their drone delivery system, but they increased its versatility in 2016, when they developed the Parcelcopter 3.0. Instead of a quadcopter design, the Parcelcopter 3.0 is a tilt-wing aircraft with a payload of up to 2kg(17). This latest iteration is fully autonomous and can operate at a height of 1200m. It was trialled extensively in the Bavarian Alps between the towns of Reit Im Winkl to Winklmoosalm, a distance of 8km. It successfully completed consumer parcel delivery in under 8 minutes, a trip that ordinarily takes 30 minutes by road(19). Alongside this most recent iteration of the parcelcopter, DHL was able to implement a fully automated parcel loading and unloading system known as the Parcelcopter Skyport(40). This innovation is what enabled the delivery service to be used by private customers.

More recently in 2018, DHL announced that their 4th generation of the Parcelcopter had been successful in delivering medical supplies to remote areas of East Africa. The Parcelcopter 4.0 managed to complete more than 180 journeys, travelling over 2200km, and completing a 60km journey from the mainland to an island in the middle of Lake Victoria(18). It has an increased payload of up to 6kg, and a range of 100km when fully loaded(13). This provides further evidence that drone delivery services can add great value to time sensitive goods.

Whilst DHL has proved that it is possible to add value to both medical deliveries and private consumer deliveries in remote hard to reach areas, they are still working on carrying out field tests in urban areas. Their ultimate aim to provide same-day drone delivery services to towns and cities.

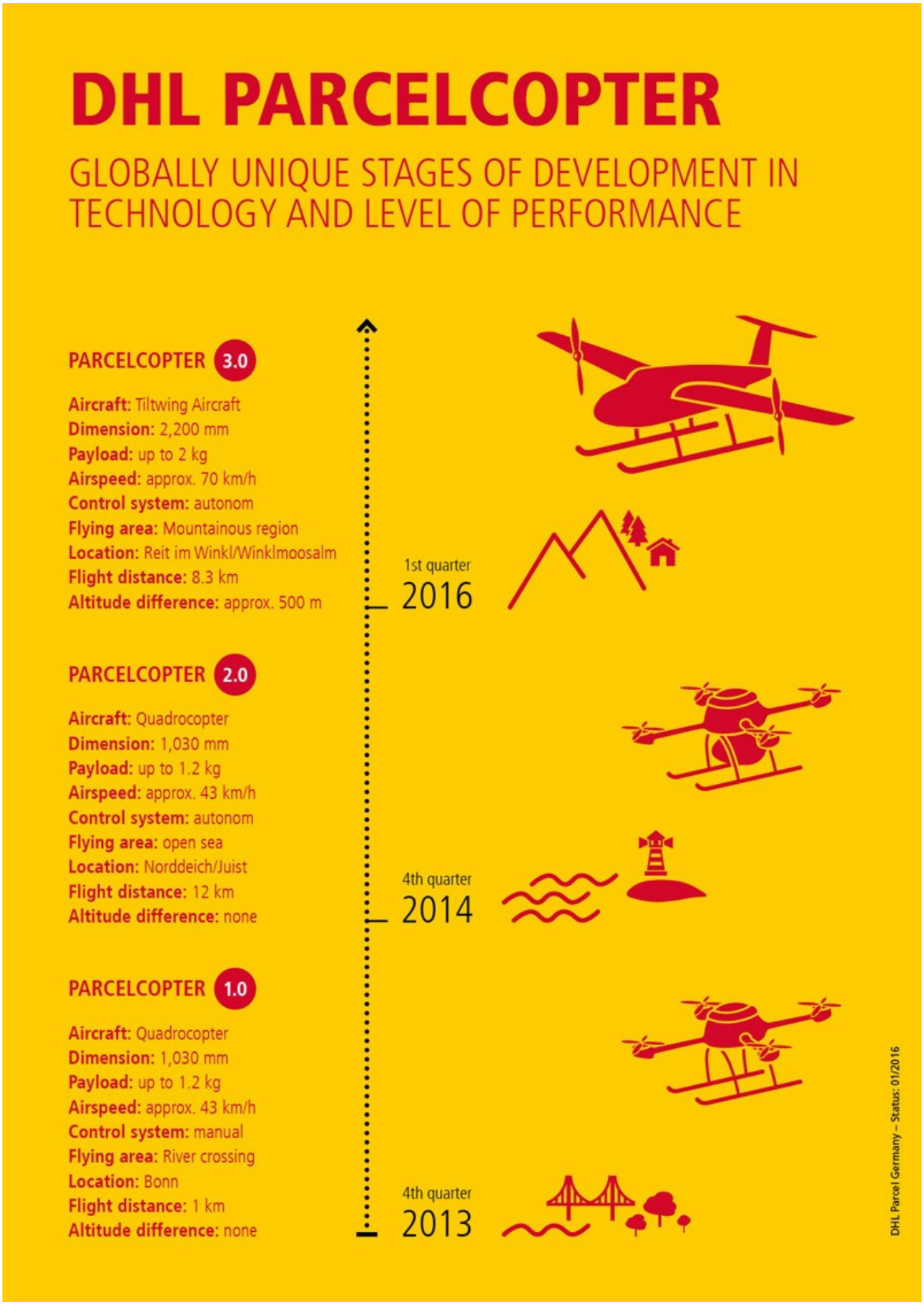

Figure 2.5: DHL Pareclcopter generations(17)

2.2.2 Matternet

Figure 2.6: Matternet drone landing on a Mercedes Vitaro van(14)

DHL are not the only company who have found value in utilising drones for medical deliveries. A start-up called Matternet has successfully implemented a drone delivery network in Lesotho(47). Matternet prototyped their drone network in Maseru, the capital of Lesotho, where paved roads are almost non-existent(28). Their research showed that the entire 140km capital could be connected by a drone network costing just $900,000(28). Compared to the cost of building 2km of road at $1,000,000, this offers an alternative way to establish a delivery network infrastructure.

Matternet used their drones for delivering blood samples throughout Maseru. Blood samples acted as the perfect cargo for drones since they are small, light, valuable and time sensitive(47). Whilst these tests are promising, it is important to note that the drones were able to follow regular paths in Maseru and did not have to deal with polluted airspace and high-rise buildings. However, since their initial tests, Matternet has also signed a deal to deliver lab samples between two hospitals in Lugano(25), which suggests possible developments towards the use of drones in urban areas.

In more recent years, Matternet has attempted to implement drones in more congested urban environments. In 2017, they piloted a hybrid van-drone delivery service throughout Zurich for 3 weeks. Their aim being to assess the efficiency gains from incorporating the two methods(11). In this instance though, drones were used to shuttle parcels from warehouses to vans, allowing delivery drivers to fulfil on demand orders without having to return to the warehouse. This could reduce the number of drivers and vehicles needed for delivery services, if delivery vans could be continuously resupplied with packages whilst en route.

Perhaps the most ingenious part of this system is the ability to convert the roof of a van into a landing pad. Matternet has been able to work alongside Daimler to create a fully autonomous landing pad on the roof of Mercedes Vitaro vans(14). This development will allow drones to deliver packages without having to worry about landing in an unknown environment.

2.2.3 Flitery

Flirtey was the first company to successfully carry out food delivery trials(41). In 2016, Flirtey was able to successfully deliver the first pizza by drone to a customer in Auckland, New Zealand. Flirtey has been working alongside Domino's Pizza to try and develop a more cost effective way of delivering food in a timely manner. They were also the first company to deliver food by drone in the United States. This time they worked alongside convenience store 7-Eleven, to deliver doughnuts, coffee, a chicken sandwich, candy and a slurpee to a customer in Nevada(45).

Figure 2.7: Flitery Drone lowering its delivery on a winch(9)

2.2.4 Uber Express

Most recently, Uber announced it is beginning research into a drone delivery service for its UberEats division, known as UberExpress. They hired an operations manager in October 2018, with plans to launch functional drones in 2019, with a launch into multiple markets aimed at 2021(8). Just this past week, they announced they had partnered with McDonald's to launch their pilot program in San Diego this summer(33). Considering that Uber has invested heavily into autonomous cars(16), it is no surprise that they are researching autonomous drone delivery. Both avenues would allow Uber to cut its costs by removing the need for drivers.

2.3 Drone Considerations

2.3.1 Obstacles

There are not many constraints on drone routing, since they are not restricted to road infrastructure. However, there are a few obstacles which must be taken into account.

Firstly, No Fly Zones (NFZ) must be accounted for. Generally, NFZ's are often dictated by the military. They prevent civilian aircraft from entering airspace where military training is taking place, or High Intensity Radio Transmission Areas, (HIRTA's): where such signals can interfere with electronics(2). Other NFZ's which must be considered are restricted areas: these usually cover sensitive regions such as prisons and nuclear facilities. There are also controlled airspace's which are most notably found around airports, due to the heavy air-traffic volumes(37). Finally, there are registered Prohibited Areas, which in the case of the UK, are listed in the UK Integrated Aeronautical Information Package(38). This document details all classifications of airspace in and around the UK. Whilst the regulations surrounding drones are still to be formalised with regards to their use in a commercial capacity, it is reasonable to assume that drones will have to conform to some defined NFZ's.

It is worth noting that, in an urban setting, drones would have to deal with routing themselves in and around high-rise buildings. This would not be an issue if drones are programmed cruise at altitudes above high-rise buildings. However, in the case of London this would result in drones having to operate above 310m (The Shard), and in New York above 546m (One World Trade Center). Clearly, this is still to be decided, but if drones are permitted to be routed below these levels, then we would have to ensure to avoid static obstacles such as tall buildings. One option would be to simply set a small No Fly Zone around any building which passes through the drones operating altitude. However, drone technology has developed to the point where even consumer drones have collision avoidance built in(12).

The other obstacles to worry about are: other drones, birds, and potentially passenger aircraft. It is unlikely that drones in delivery scenarios would have to contend with commercial aircraft, since they should be avoiding flight paths due to to NFZ's. However, in urban areas, helicopters could become an issue. The airspace in which drones and helicopters operate is likely to overlap. Therefore, drones would have to be fitted with some form of Detect and Avoid (DAA) technology, in order to successfully avoid dynamic objects. Amazon has already filed a patent for such a technology using multispectral sensors(39).

Taking these considerations into account, any drone delivery network must be sophisticated enough to schedule optimal routes to avoiding static obstacles, whilst also reactive enough to avoid dynamic obstacles. However, the purpose of this project is not to devise a means of routing drones. This has already been researched(7).

2.3.2 Regulations

Currently, the commercial use of drones is severely restricted prohibited in the UK by the Civil Aviation Authority (CAA). The CAA requires that direct permission be obtained before any commercial use of drones may be carried out. Furthermore, drones cannot be flown in excess of 400 feet (122m), without the CAA‘s permission, and the operator must maintain direct unaided visual contact with the aircraft(4). The CAA defines Small Unmanned Aircraft to be those with a weight under 20kg. Using the Amazon Prime drone as an example, it is safe to assume that small delivery drones would fall within this category. Whilst Amazon has not disclosed the weight of its prototype drone, its weight can be estimated at 3.8kg, which would allow it to carry a payload of up to 2.3kg(31). Since no official guidance on the regulations concerning the commercial use of drones has been published, we will adhere to the current regulations, with the exception of maintaining line of sight.

2.3.3 Package Delivery

There are a few different ways for a drone to perform the delivery of its package. The most obvious is landing to allow the customer to retrieve their package, as demonstrated in the Amazon Prime trials(30). However, as Flitery demonstrates(9), drones could be supplied with a winch to lower the packages down to customers. This removes the need for a clear landing zone. Alternatively, Matternet‘s approach of using the roof of cars as landing pads(14) may be the most realistic in an urban environment. One other speculative approach is to supply packages with parachutes. Whilst this allows the drone to leave faster, it causes other issues, such as ensuring the package lands in the correct place. Moreover, the CAA currently bans drones from dropping any articles which could endanger a person or property(4).

2.4 Path Planning

In order to implement a sufficiently realistic simulation of a drone delivery network, it is necessary to devise a means of navigating drones to and from their destinations. In conventional path planning, constraints are usually imposed by the existing road infrastructure. In the case of drones, such infrastructure isn‘t necessary, and you should be able to path a drone to its destination directly, as the crow flies. However, as outlined above, one consideration is No Fly Zones. Therefore, we must ensure that we path our drones around such obstacles.

2.4.1 Autonomous Air Traffic Control (AATC)

Autonomous Air Traffic Control (AATC) was a collaborative project undertaken between Imperial students, Microsoft and Altitude Angel. The aim was to navigate drones from start to finish, whilst avoiding obstacle, and taking the shortest path possible(7).

The project resulted in a two-tiered system, with a global layer and a reactive layer(7). The global layer stores a static representation of the world, allowing for the calculation of an optimal route from source to destination. AATC uses a modified version of the A* algorithm, known as the Theta* algorithm, for route planning. It is an any-angle path-finding algorithm which propagates information along grid edges(15). Whilst the execution time of theta* tends to be worse than the A* algorithm, the theta* algorithm is more successful in finding the shortest route. This is a primary concern for drones since their range is limited by their battery life. Therefore, the theta* algorithm is the underlying route planning algorithm which AATC uses.

The reactive layer of AATC is what allows it to avoid obstacles and dynamic No Fly Zones. It achieves this through the use of an Artificial Potential Field (APF), which attracts or repels the drone(7). The destination is assigned a low potential, and obstacles are assigned a high potential. This means the drone has to navigate to the point of lowest potential. However, it is noted that this method cannot be relied upon for full navigation since there are circumstances where a drone could become stuck: fsuch as encountering a U-shaped object. This is where the global layer is able to generate a route which avoids such obstacles(7).

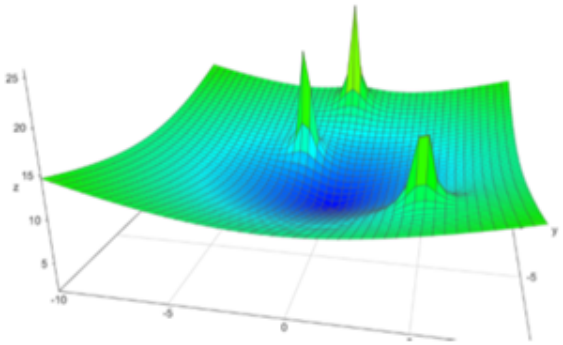

Figure 2.8: Visual Representation of Artificial Potential Field(7)

2.5 Scheduling Algorithms

In order to ensure that time sensitive deliveries are completed, we must have some way of scheduling the deliveries. Scheduling algorithms have been researched in depth with respect to CPU scheduling. However, some of these algorithms benefit from the fact that CPU tasks can be interrupted and completed in quantums. In our case, when a task is scheduled, it must be completed before the drone is available to undertake another task. There has already been research undertaken by Knotten-Belt and Balaji(44), who derived Least Lost Value as an algorithm for maximising profit per delivery. Here we outline a few general scheduling algorithms and give an overview of Least Lost Value.

2.5.1 First Come First Serve (FCFS)

This algorithm entails using a First In First Out (FIFO) queue. As drone deliveries arrive, they are added to the end of the queue. As each drone becomes available, the delivery at the head of the queue is removed and assigned. This is perhaps the most basic form of scheduling presented here, as it is only reliant on the time the order was placed. One issue with FCFS, is that shorter deliveries could be blocked behind one longer delivery, resulting in an increased average service time.

2.5.2 Shortest Job First (SJF)

In our case, Shortest Job First scheduling calculates the distance from source to destination for each delivery. The deliveries are then prioritised based upon their distance. The intuition is that the delivery which is geographically the closest would be completed first, lowering the average service time. This overcomes the pitfalls of FCFS, since shorter deliveries will not be blocked behind longer deliveries. However, it is possible for a longer delivery to be blocked indefinitely if shorter tasks keep getting assigned.

2.5.3 Priority Scheduling

Priority Scheduling is the general form of SJF. Priority scheduling orders jobs according to a defined priority metric. In the case of SJF, the priority metric is the shortest delivery time, which is equivalent to the shortest distance.

2.5.4 Least Lost Value (LLV)

Recently, Least Lost Value has been proposed as a revenue-driven scheduling algorithm for time sensitive service level agreements. It assumes that the value of a delivery follows a monotonically decreasing time value function. The Least Lost Value metric is calculated by considering the value of completing a task immediately, compared with the value of postponing it until after another task. The task which would lose the most value if postponed is then chosen(44). This should produce a schedule which allows the service provider to extract the highest value from the limited resources, given the time constraints. This algorithm has been evaluated on a single source node warehouse model, where it attained a higher average profit per delivery compared to FCFS and SJF(6). In this paper, we have extend LLV to be used in our multi-source node model.

2.6 Previous work

As the field of drone delivery has been an emergent technology in recent years, there have been several different avenues of research into drone applications. However, there are few studies which assess the potential of fully autonomous drone delivery networks. This may be due to current regulations impeding the progress, or it may be due to the technicalities of actually delivering the packages to a human recipient. Nonetheless, there are a few notable papers which have studied different aspects of drone delivery networks and food delivery networks.

2.6.1 Drone Delivery Assistance

Murray and Chu(36) assess the potential added value that drones may bring to current delivery networks by of assisting drivers. They outline three different models where drones could be used to optimise the delivery network(36). The first involves using drones to deliver to all participants within range, and using a truck to arrange delivery of packages outside of drone range. The second employs an optimised assignment of customers to either a drone or a truck, allowing a truck to take the most efficient route, including covering some customers within range. The final solution proposed in this paper is the use of trucks as an aircraft carrier for drones. This enables the truck to avoid deliveries which deviate it from the most optimal route Those deliveries which are not included in the optimal route are assigned to drones, which would be launched from the truck. Whilst this paper does provide a significant insight into the integration of drones into a delivery network, its main focus is evaluating these three ‘sidekick‘ models, and determining a strategy for assigning deliveries to either a drone or driver. Moreover, it is concerned with package delivery in the Amazon style warehouse model, where all orders are shipped form a central

location. Therefore, whilst substantial inspiration can be gained from its attempts, it falls short of our multi-source model.

2.6.2 Drone Delivery From A Central Warehouse

Balaji(6) undertook a study into time valued delivery models using a fleet of drones and 2 warehouses. He proposed the LLV algorithm discussed earlier, and assessed its performance against FCFS and SJF. Balaji's work focused on the warehouse model, where drones are tied to a single source node. Whilst he did model 2 warehouses, each warehouse had a predefined fleet allocated to it. One limitation of this model is that the drones are not free to cover dynamic demand as the load on source nodes changes. This is an avenue which we explore. Balaji's project was built on Improbable's SpatialOS platform, and it was decided that this platform was too heavyweight for our needs. However, a proportion of work undertaken in his project can be adapted for our own. His implementation of Autonomous Air Traffic Control(7), is built upon and extended to our multi-source model, allowing us to focus on scheduling as opposed to route finding.

2.6.3 Analysis Of Drone Delivery Systems To Satisfy Constraints

The most extensive study found which is relevant to this project is one in which the reuse, energy consumption, time constraints and budgets of drone delivery networks are considered. ***Vehicle Routing Problems for Drone Delivery*(21)** proposes two Vehicle Routing Problems (VRPs), the first of which minimises cost subject to a delivery time limit, and the second attempts to minimise the overall delivery time limit subject to a budget(21). Moreover, it also undertakes a study into the effects of battery weight and payload on the drones energy consumption, and demonstrates that it approximately follows a linear relationship. This research will be invaluable when it comes to deciding the assumptions our model is based on.

This paper confirms the approximate linear relationship between payload and energy consumption, and also highlights the issue of recharging stations. Since our project will attempt to model a food-delivery service, we will have to consider whether to have centrally located recharging stations, or whether the source nodes (restaurants) would have some means of allowing drones to recharge. The idea of hot-swappable batteries was proposed as a solution to the recharging station issue(34), and is what we have employed in our simulations.

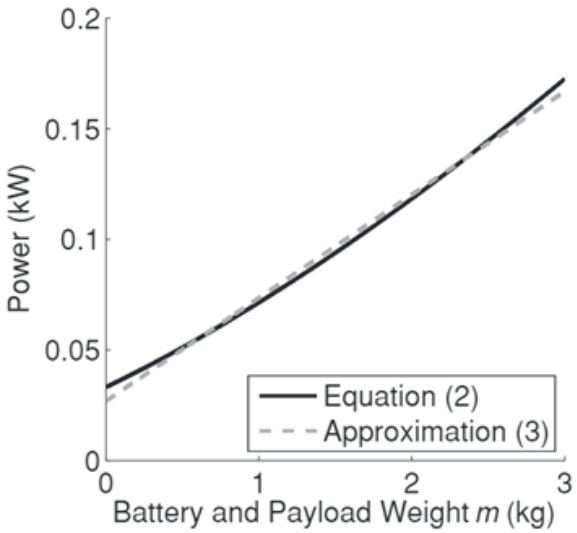

Figure 2.9: Payload vs Energy Consumption(21)

2.6.4 The Meal Delivery Routing Problem

One of the few papers relating to food delivery is a study undertaken between Georgia Institute of Technology and Grubhub(42). It formulates the Meal Delivery Routing Problem (MDRP), and performs a comprehensive investigation into solving the dynamic delivery problem which has arisen due to the likes of Deliveroo and Just Eat in the UK. It covers both the optimal route scheduling courier assignment aspects, and offline shift scheduling resulting in an in depth analysis of meal delivery networks. However, their routing relies on the use of drivers, who have hard shift times, which must be respected, and who can carry multiple orders at once. These assumptions do not carry over to our model. But we have been able to draw on their assignment strategies and scenario data to build our own models.

Finally, whilst JustEat has not published any information concerning how they schedule their courier, we were able to deduce that they use pools of drivers to cover geographical areas(32). Therefore, we have constructed our models to represent one of these pools which covers a specific area.

2.7 Development Environment

A previous drone delivery simulation, which our project builds upon(6), was written in SpatialOS. However, this was in part due to the individual‘s familiarity and preference for it. SpatialOS is designed to run massive multiplier hyper detailed

games on a distributed back-end(3). For our simulation, this is excessive. Instead, we will be making use of Unity, using its physics engine, visualisation tools, and developement environemnt to construct our simulation. We therefore do not require the performance which SpatialOS offers, as this additional performance comes at a cost of increased complexity.

Unity has a large user base and wealth of developer forums and tutorials to assist the layman in getting started. SpaitialOS, on the other hand, does not yet have as large a user-base, and whilst it is growing, it is not as well known. We hope that our simulation will provide a backbone which can be extended and adapted to simulate different circumstances. With this in mind, we will focus on having an extendable and modular architecture. This will allow future developers to plug in their components, such as alternative schedulers, pathing algorithms and physical scenarios to test their own ideas.

Chapter 3

Model Assumptions

In order to construct our model we must make a few assumptions about the world we will be simulating, and the drones we will be using.

3.1 Drone Assumptions

Firstly, we must decide on the details of our drone. Using the Amazon drone as a starting point, we know from research(31) that they have a potential top speed of 50mph, a battery life of up to 30 minutes, and a range of 16km. These estimates are in line with consumer drones, except for the top speed. Larger consumer drones have a top speed of approximately 45mph(20), and these are not designed to carry any payload. Therefore, we have chosen to give our drones a sustainable speed of 13m/s (30mph) measured when travelling horizontally through the air. We have also specified a maximum descent speed of 4m/s (9mph), and an ascent speed of 6m/s (13mph), to allow incorporation of ascent and descent times. These are in line with currently available drones(20).

Our research suggests that Amazon‘s drones can carry a payload of up to 2.3kg(26). This allows them to carry small food parcels and up to 2 Domino‘s Pizzas, assuming a pizza weighs approximately 2kg. Whilst this does mean the drones may not be appropriate for larger orders, they are perfectly adequate to serve the individual customer ordering a meal. Since the battery life of the drone is not a major feature of this study, we will model the battery life based upon the maximum range of the drone.

The CAA currently only allows drones to be used at heights of up to 122m. In accordance with these rules we have designated our drones to operate in the airspace between 50 - 100m. To minimise the probability of collisions, each drone is assigned an operating altitude. We have decided to assign these randomly, but an alternative

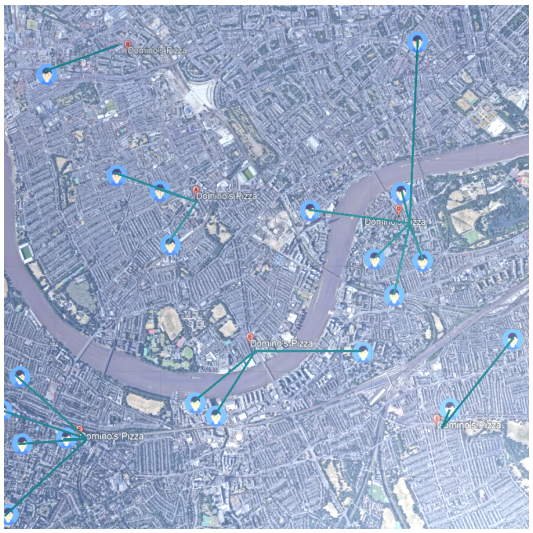

Figure 3.1: Delivery Routes for Domino's Scenario

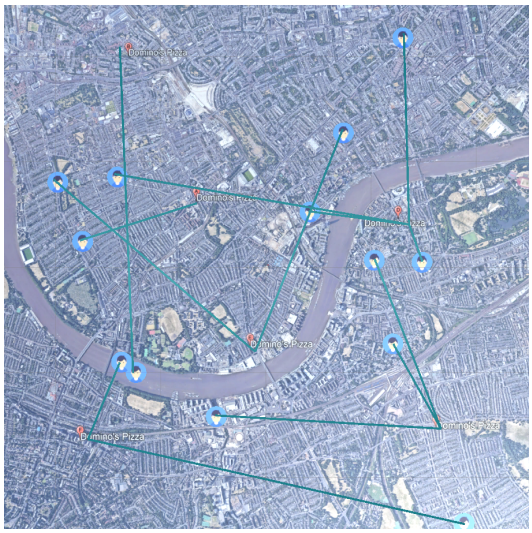

Figure 3.2: Delivery Routes for Just Eat Scenario

would be to use the airline industry standard, which assigns an altitude based upon an aircraft's heading(5).

3.2 Restaurant Assumptions

We propose two scenarios in this study. The first is the Domino's Pizza scenario. Here we assume that the delivery drones services multiple restaurants which all offer the same menu. In this scenario, an order is sent to the closest restaurant as seen in figure 3.1. This scenario is similar to the warehouse model, since each restaurant only serves the orders which are closest to it. Therefore, deliveries will only ever be within that restaurants service area. Consequently, it may make sense to station drones at each restaurant and solely serve that restaurant. We assess this scenario and how it performs under local scheduling at the restaurant level, and global scheduling, at the drone fleet level.

Our second scenario models the requirements of a food courier service such as Just Eat. In this scenario, each restaurant has a different menu, meaning an order is not sent to its closest restaurant. Instead, a delivery could be anywhere within a restaurant's service radius as seen in figure 3.2. This radius can be dictated by the restaurant, or by the organisation such as Just Eat, in order to ensure reasonable delivery times. A quick test of the Just Eat app reveals there are 245 restaurants which will deliver to my location, 235 of which are within a 3.0 mile radius. Therefore, we have chosen the restaurant service radius to be 5km. In line with the Meal Delivery Routing Problem(42), we have set our service time at both restaurant and delivery locations to be 4 minutes. However, there could be potential to decrease the restaurant service time if an Airbase(34) system was employed.

It is worth considering the landing options at restaurants. In both scenarios the restaurants area will be the most crowded airspace as drones are constantly collecting orders. Therefore, to avoid drones crashing into each when they are entering and leaving restaurants, we have decided to employ a multiple landing pad approach. We will assume that each restaurant has a choice of landing locations which a drone can use. We will arrange these locations in a grid as seen in figure 3.3 for simplicity of simulation purposes. In a real implementation, this challenge would need to be considered on an individual restaurant basis. One solution could be to use the roof of employees cars, as tested by Matternet(14). Despite the physical implementations, a drone controller would only have to be supplied with a list of landing locations to use.

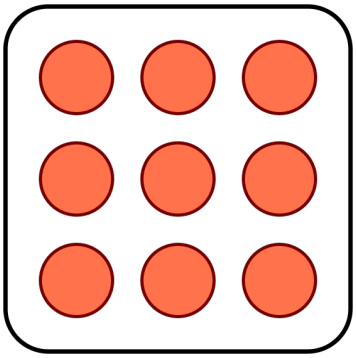

Figure 3.3: Restaurant landing location layout

3.3 Time Value Functions

In order to model the customer desire to receive their deliveries faster, we introduce a Time Value Function. This function models the idea that the delivery is worth more to the customer now rather than later, and as such it has a greater value the faster it arrives. We have chosen to model the profit from delivery as a discounted value step function which degrades by 10% every 6 minutes. Assuming there is a fixed fee for delivery, the profit is calculated as a percentage of this fee, which is dependent on the time it is delivered. For example, if the delivery was placed after 30 minutes, the customer would pay 50% of the delivery fee.

Figure 3.4: Time value function with 10% step

We will be assessing the performance of our schedulers based upon the profit they are able to achieve. Therefore, we will take into account the total number of orders completed, the average profit achieved per order, and consequently, the total profit.

Part II

Implementation

Chapter 4

Simulation Components

In order to model our food delivery network, we must first create the backbone of the simulation. This will consist of a few key components, which will be designed to be swapable, allowing the simulation to be easily extended and adapted to different scenarios.

Firstly, we must create a model of a drone which is able to path itself between waypoints. Secondly, we will require a Route finder which calculates the path a drone should take. We will also need an Order Generator which can generate orders in the real world between source nodes (restaurants) and destination nodes (customers). To start with, we will use a First Come First Serve (FCFS) scheduler to test the simulation, and we will add alternative schedulers once our base simulation is working. Finally, we will require a Fleet Controller which coordinates the deliveries. When a drone becomes available, the Fleet Controller will request the order at the front of the queue, determine the route by passing the order details to the Route Finder, and then issue the route instructions to the drone.

4.1 Drone Model

Modern drones are capable of autonomously navigating to GPS coordinates. With this in mind, our drone controller entity is only concerned with navigating between its set of waypoints.

We made use of the Unity prefab entity in order encapsulate the drone behaviour inside an object. The prefab has also allowed us to create a basic visual representation of our drone for the simulation. The behaviour of the drone is relatively simple; it needs to navigate between its list of way-points. Unity allows us to model this by updating the location of the drone with each frame, resulting in a smooth simulation. In order to gain a better understanding of how the drone‘s time is spent, and

to aid the visualisation, we implemented the status of the drone to be any of the following:

- Idle
- En Route To Pick Up Point
- Collecting Package
- Package Collected
- En Route To Delivery Location
- Delivering Package
- Package Delivered
- Returning Home

Collecting Package, Package Collected, Delivering Package and Package Delivered informs us that the drone is descending or ascending respectively. Returning Home is used to signal that the drone has not received a new order and will return to its spawn location. This allows to accurately model how much time is spent ascending, descending and travelling between source and destination. These different statuses also aided debugging in a number of cases. The only information a drone keeps track of is its current position, and current Order, which holds the list of way-points it must navigate to. All of this logic is contained within the Drone Controller.

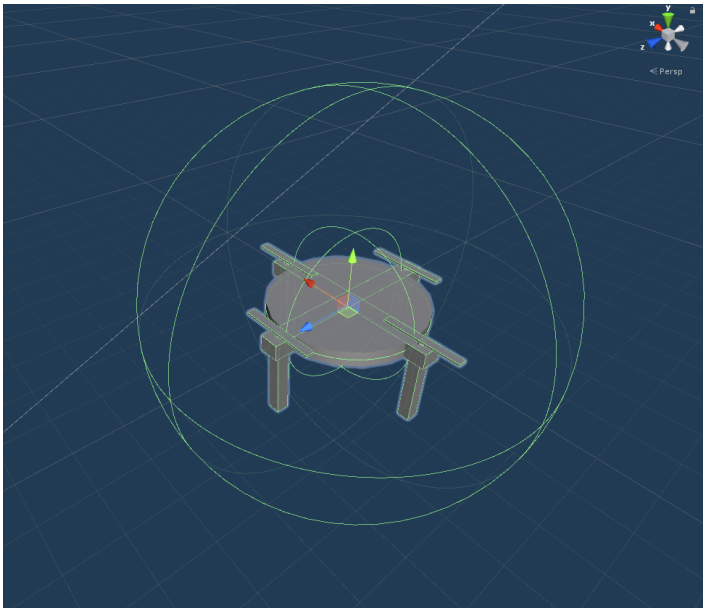

Figure 4.1: Drone prefab visualisation

4.2 Order Generation

In order to encapsulate all the details that are relevant to a specific delivery, such as the pick up point, delivery location and time the order was placed, we created an Order object to store this information. This Order object can then be passed from the Order Generator to the Fleet Controller, on to the scheduler, and finally the drone. Throughout this process, the Order object is enriched with information. It is initially created with the time the order is placed, its order ID and the pick up and delivery locations. When the Fleet Controller receives the Order, it extracts the source and destination locations and passes these to the Route Planner. Once a path is calculated, the Fleet Controller sets the way-points inside the Order object, and then passes the Order to the Scheduler for it to be queued. When the drone receives the Order, it adds information such as the pick up and delivery times as it completes the Order. At this point, the drone registers the completion of the Order with the Fleet Controller. The Fleet Controller logs the details, and assigns the drone its next Order.

The Order Generator manages the creation of these Order Objects. This is done by randomly selecting a source location, from the list of source locations in the simulation settings. A random delivery location is picked which falls outside of the No Fly Zones. We then randomly sample the inter arrival time between orders using a Poisson process. Therefore, the order generator creates the next order to be sent, and holds it until the inter arrival time has elapsed, At that point the order is sent to the Fleet Controller and the Order Generator creates a new order with a randomly sampled inter arrival time.

4.3 Route Planner

The route planner is the most one of the most complex parts of the initial simulation. A considerable amount of work is required to create a representation of the world before we can calculate routes between points.

4.3.1 World Bitmap

The first step is to create a bitmap of the world. We could choose any resolution of bitmap, with perhaps the ideal size being 1m. However, that level of granularity is only necessary for pick up and drop off locations. Path finding can be done at a lower granularity in order to avoid excessive computation time. To start with, we will cut the world into a 100x100 grid.

For the first test environment we will use a 1km square area, providing a resolution of 10m. This bitmap will be used to determine whether or not a location is

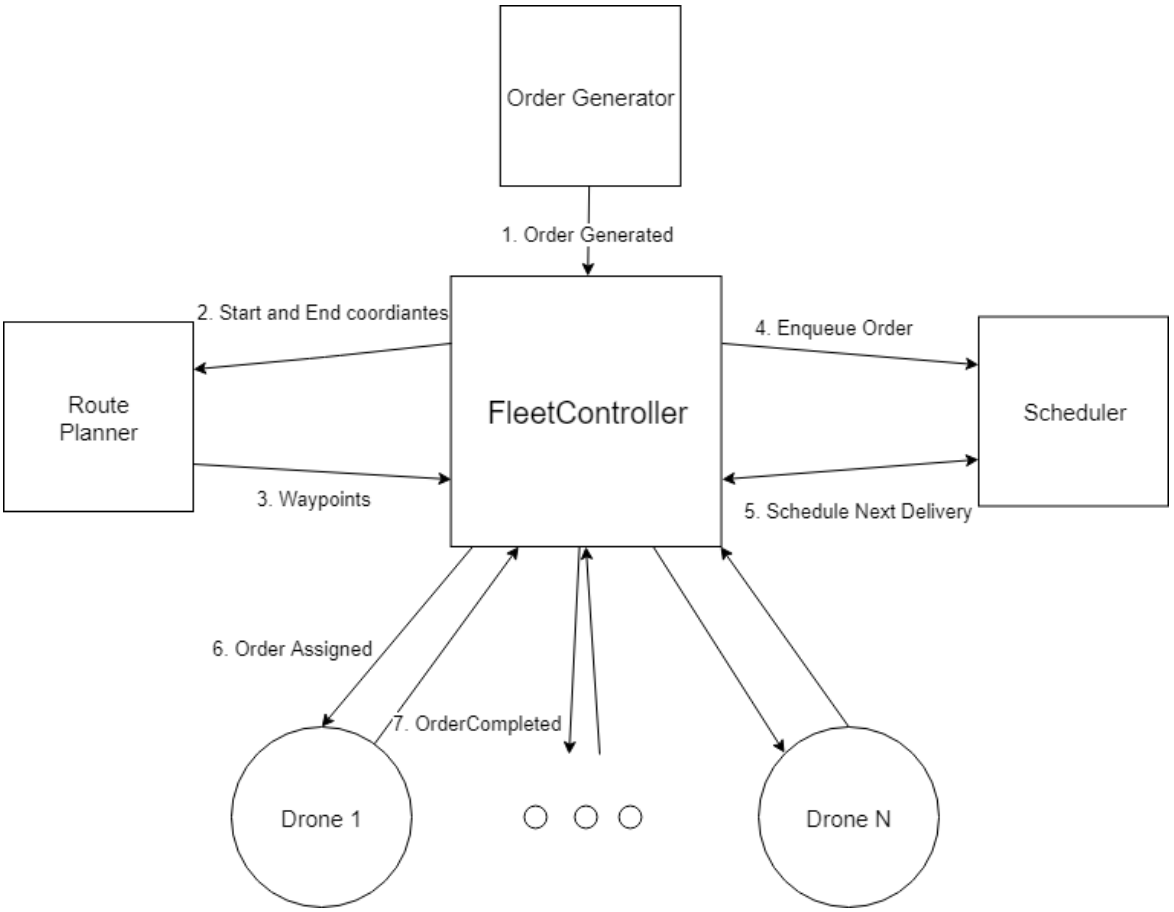

Figure 4.2: The Route an Order takes through the Simulation

within a No Fly Zones (NFZ). It would be unfeasible to manually enter grid locations as NFZ points, therefore we have taken the approach of supplying the vertexes of the NFZ's. Given the vertexes in order, we can draw the outline of the NFZ by stepping between each pair of vertexes and setting the corresponding locations in the bitmap to 1, signalling it is a NFZ.

This same approach was taken in Balaji's project(6). However, this meant that every time a coordinate was requested, it has to be tested to see whether or not it fell within the boundary drawn by the vertexes. This was achieved by iterating through all the vertexes, and checking the horizontal and vertical coordinates against those of each adjoining pairs of vertexes. We chose to take a different approach in order to avoid these computations. We decided to fill in the interior area of the NFZ's. This meant that we can do a bitmap look up in O(1) time to check whether a point falls inside a NFZ. This eliminates the need to iterate through the vertexes and will also allow us to visualise the entire No Fly Zone.

In order to fill in the interior polygon of the NFZ, we implemented a scan line algorithm to iterate through the bitmap. When it detects a boundary it will set all

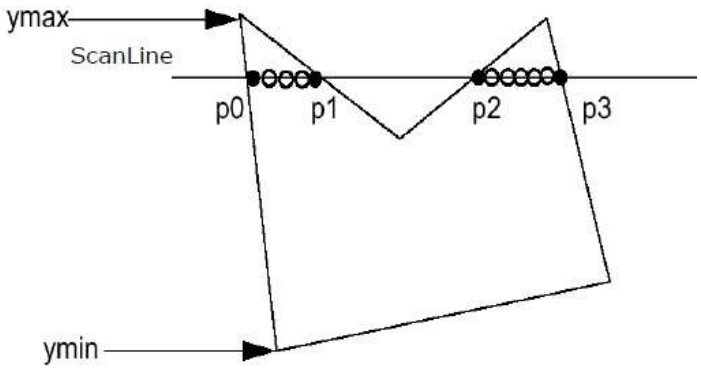

Figure 4.3: Scan line fill algorithm

interior points to 1 signalling the NFZ. We confine ourselves to the bounding box of the NFZ, and scan through line by line, filling in points as necessary. In order to ensure that our algorithm worked a few basic shaped NFZ tests were undertaken, as well as some more irregular shapes. In the case of ordinary shapes, correctness of the algorithm could be automated, but with irregular shapes, it was faster to visually inspect the results of the test.

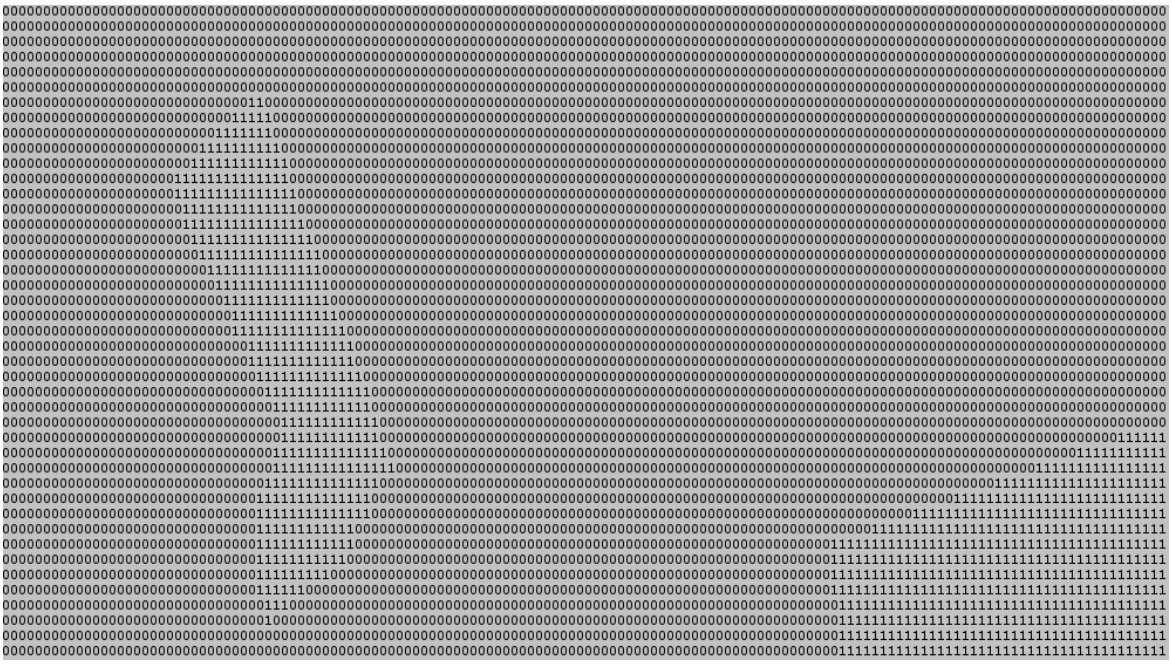

Figure 4.4: No Fly Zone Bitmap representation of Battersea park and Brompton Cemetary

4.3.2 Path Search Algorithm

Once we were confident that the bitmap held a correct representation of the world, with designated No Fly Zones, we needed to implement a path search algorithm.

Drones benefit from not being confined to current road infrastructure, and can fly directly to their destination along the fastest route, assuming no obstructions. Until this point, we had been using the most basic path finder which simply directs the drone to its destination as the crow flies. This was useful in testing the correctness of the drone instructions, and also enabled the construction of a Route Finder interface which allows us to swap in different path search algorithms as necessary.

As mentioned previously, this project builds upon previous work undertaken in Simulating Drone Delivery Networks using SpatialOS(6). As such, we have implemented the same path finding algorithms, but adapted it to use our alternative representation of the world bitmap.

The path search is comprised of a grid search, which iteratively expands outward from the source node towards the destination. The next node to be explored is based upon each explored nodes' Euclidean distance to the destination node. This grid search returns the step-wise path between neighbouring cells in the bitmap, from the source to destination. This path is then passed to the theta search algorithm which performs line of sight checks along the path, converting the neighboured grid path into a set of line of sight way-points which the drone can then follow.

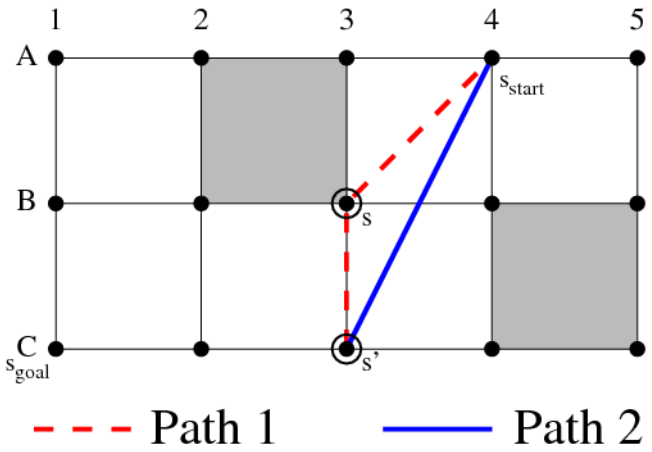

Figure 4.5: Example of Line of Sight check used to skip intermediate grid squares

Once the Theta* Search algorithm had been adapted to use our representation of the world, we tested that a drone was able to path around No Fly Zones. Since NFZ's are specified by the coordinates on the ground, we visually represent them by a cuboid the size of the bitmap resolution stretching from the ground to the maximum allowed drone altitude (100m).

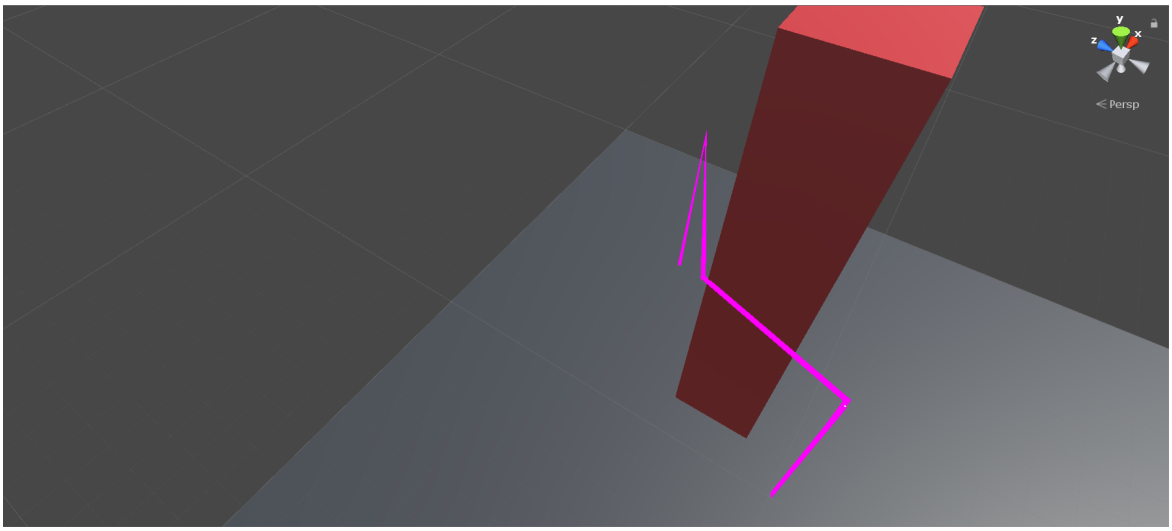

Figure 4.6: First test of a drone following way-points around a No Fly Zone

4.3.3 Global Layer

As in AATC(7), the navigation components which route a drone between two points is encapsulated within the Global Layer. This object is supplied with the list of NFZ's, and initialises the world bitmap. It then receives requests from the Fleet Controller with source and destination coordinates. These coordinates are passed to the Route Finder, which generates the list of navigation way-points.

4.3.4 Reactive Layer

AATC uses an artificial potential field to calculate the optimal direction to move relative to dynamic obstacles. It does this by calculating the distance to the nearest obstacle through the global layer. Realistically, drones would be outfitted with an array of sensors to detect when they are coming close to an obstacle. With this in mind, we have made use of Unity's built in Raycast objects to detect and avoid obstacles.

4.4 Simulation Life-cycle

The simulation initialises by setting up the Global Layer, Order Generator, and Fleet Controller. The Global Layer reads in the simulation settings, generates the world bitmap, and passes this to its Route Finder. The Order Generator receives a copy of the bitmap, to ensure it does not generate Order's within No Fly Zones, and the list of source locations. A set number of Orders are pre-generated to kick-start the simulation. The Fleet Controller instantiates the user supplied number of drones for

the simulation.

The Fleet Controller coordinates the simulation. When an Order is received, the Fleet Controller passes the start and end locations to the Global Layer which returns a list of way-points. These way-points are added to the Order, which is then added to the Scheduler. When a drone is idle, or has just completed its current order, it requests a new Order from the Fleet Controller. Within this request it sends its current location. The Fleet Controller takes the order from the head of the Scheduler, and uses the Global Layer to calculate the way-points from the drone‘s current position to the Order pick up location. The drone‘s current location is passed to the Scheduler as this information may be used to determine which is the best Order for the drone.

The lifetime of a drone follows a repeated loop. It will initially spawn at one of the source locations, register itself with the Fleet Controller and set its status to Idle. If its status is Idle, it will request a new order from the Fleet Controller. When a drone receives its next order, it will schedule its next way-point from the list of navigation way-points. The navigation way-points do not take into account the ascent and decent of the drone, they navigate it between destinations at its cruising altitude. Therefore, the drone must coordinate its ascent and descent.

Once the drone reaches its pick up destination, it will descend down to ground level, changing its status from En Route To Pick Up Point to Collecting Package. At this point the drone waits for 4 minutes to simulate the handling time. After this handling time has elapsed, the collection time is set in the Order, the drone visualisation is updated to reflect that it now has an item on board, and it begins ascending to a safe height. As it begins its ascent, it changes its status from Collecting Package to Package Collected. Once a safe height is reached, it begins following its delivery way-points with a status of En Route To Delivery Location. Its behaviour at the delivery location is the same as at the pick up location with respective status‘ Delivering Package and Package Delivered. Upon completion of the order, and once it has ascended to a safe height, it sets its status to Idle, which triggers a request for the next order.

These detailed statuses were invaluable when creating the simulation and debugging. Moreover, they have allowed us to accurately model a drone delivery network, by accounting for ascent and descent times, which occur at different speeds, as well as handling time for both collection and delivery.

Chapter 5

Scheduler Implementations

The main aim of this project is to analyse the performance of different scheduling algorithms. As already mentioned, we aim to analyse First Come First Serve (FCFS), Shortest Job First (SJF) and Least Lost Value (LLV). These have already been analysed in previous work(6), however this was only in a warehouse model. In that scenario, the drones are tied to a base and serve within a radius of that base. We will be assessing how these algorithms perform when drones are not tied to a specific base and can move freely between source nodes.

Before we outline the scheduler implementations, it is important to define a few terms. We define Wait Time to be the time from when an order is received by the scheduler to when it is picked up by a drone. We define Delivery Time to be the time taken for the drone to transport the order from source to delivery location. Finally, Drone travel time is defined as the time taken to travel from a drone‘s final delivery location to its next pick up location.

5.1 Local Scheduling

In our Domino‘s Pizza scenario we will be assessing the performance of local schedulers. By Local scheduler we mean one which follows the warehouse model. Therefore, if we use local scheduling, each restaurant will have its own scheduler and drones will not be able to move between restaurants.

5.1.1 Local First Come First Serve LFCFS

This scheduler employs a First Come First Serve Queue for each restaurant. It is meant to model our Domino‘s Pizza scenario where each location has a limited number of drivers who deliver orders once they are ready. Arguably the fairest scheduler,

it ensures every delivery is carried out. It is the easiest scheduler to implement and is often used for small scale delivery. However, when under heavy load it results in longer wait times for all deliveries.

5.1.2 Local Shortest Job First LSJF

Local Shortest Job First prioritises deliveries by their delivery distance. Since we are scheduling on a local scale, all drones we have control of are based at our location. Therefore, we need only account for the delivery distance and deliver the shortest order first in an attempt to maximise profit.

5.2 Global Scheduling

If we move to a global scheduling model, we allow the drones to dynamically move between source destinations to satisfy demand. It is possible that global scheduling may not have a significant improvement over local scheduling in the Domino‘s model due to nature of the orders. Since an order is assigned to its closest restaurant, drone travel distances are minimised, thus it may be beneficial to run a warehouse model. However, in our Just Eat model it makes little sense to employ global scheduling since orders are more dispersed.

5.2.1 First Come First Serve FCFS

First Come First Serve has been included as a control scheduler. However, it is very likely that the FCFS scheduler will perform poorly because orders will be processed at a global level. Consequently, drones may be flown from one corner of the world to the opposite corner in order to respect the FCFS ordering. Whilst FCFS can be judged as the fairest scheduler, as it will ensure that every order is completed, it is unlikely to maximise profit on a global scale.

5.2.2 Closest Job First CJF

An improvement on FCFS is Closest Job First. CJF maintains a global FCFS queue, but when a drone requests the next order, the global queue will be iterated through and the first job which is closest to the drone will be selected. It is likely that multiple jobs will be equally close since orders can only originate from a finite number of source locations. Therefore, the first one is chosen to maintain part of the FCFS ordering.

5.2.3 Shortest Job First SJF

Shortest Job First will also be implemented on a global level. The orders will be sorted by the estimated delivery time which is based upon the distance from source location to delivery location. This distance is calculated by summing the distance between each waypoint in the order‘s delivery path. It is an improvement over simply taking the euclidean distance from the pick up and delivery points, since this distance is the actual route the drone will fly. However, the estimated duration is calculated assuming that the drone travels at constant speed between the pick up and delivery locations, and does not deviate from its route to avoid obstacles.

5.2.4 Enhanced Shortest Job First ESJF

Since Shortest Job First is done on a global level, the scenario where a drone may be ordered from one side of the world to another still persists. Therefore, we have proposed Enhanced Shortest Job First which takes into account the drones position as it finishes its delivery. ESJF will iterate through the SJF queue and choose the job which has the smallest overall distance. The overall distance is calculated by adding the delivery distance of the order, following the pre-calculated waypoints, plus the distance the drone must travel from its current position to the pick up location. In order to avoid calculating a route from the drone‘s current location to the pick up location for every order in the queue, we use the Euclidean distance as an approximation. This is a compromise to save excessive computation time, but could result in a situation where the drone picks the wrong route. For example, if there is a large No Fly Zone between the drone‘s current location and the pick up location, then having to path round the NFZ may result in a greater overall distance than another order in the queue.

5.2.5 Least Lost Value LLV

At its core, Least Lost Value is a priority queue ordered by the Net Lost Value Metric.To calculate the Net Lost Value of an order, both the Potential Gain Value (the value of doing a job now), and the Potential Loss Value (the aggregated lost value of postponing all other jobs), must be calculated. To calculate these potentials we must also define an Expected Value for each job. Using the time value function $V(t)$ we defined earlier, we can determine the expected value $EV(j,t)$ of a job j at completion time t, by calculating which step it falls into. To determine a jobs completion time, we need only consider the time the order was placed and how long it will take to complete.

$$EV(j_i, t_c) = V(t_c)$$

The Potential Gain Value is the value of choosing the given order next, minus the value of waiting to execute that order, and instead executing it at time $t_c + p_i$

$$PGV(j_i) = EV(j_k, t_c) - EV(j_k, t_c + \bar{p_i})$$

The Potential Loss Value is the sum of all of the lost values for every other order which must wait to be completed. For every order except the current order we are considering, we calculate the value which is lost by postponing it as opposed to executing it now.

$$PLV(j_i) = \sum_{j_k \in J,\ k \neq i,\ k-1}^{n} (EV(j_k, t_c) - EV(j_k, t_c + p))$$

We can then calculate that Net Lost Value for each order by subtracting the Potential Gain Value from the Potential Loss Value.

$$NLV(j_i) = PLV(j_i) - PGV(j_i)$$

Least Lost Value calculates the potential gain of choosing a given order next, minus the potential loss of choosing that given order over all the others in the queue. Therefore, in order to calculate this value the algorithm must iterate through every order in the queue and calculate its potential gain, which is based on the potential of every other element in the queue. Hence the complexity of LLV is O(n^2). In order to minimise the computational load LLV is only reordered when the next order is requested by a drone.

In the original implementation of LLV, the potential gain of orders is calculated by determining which step an order will be completed in. In Balaji's warehouse scenario(6) he determined which step the order will be completed in by calculating the current wait time of the order, $currenttime - orderplacedtime$, and adding the estimated duration of the order. In his study the duration of the order was known since it is just the travel time from the warehouse to the delivery point.

However, our scenario is slightly more complicated as the estimated duration of the order changes depending on which drone completes it. This is an additional complication to the estimated duration time which must be taken into account in the Expected Value calculations. We must know the drone's current position, and estimate how long it will take to reach the pickup point, as this will affect which step the delivery will fall into. As seen in the figure 5.1 and figure 5.2 we have different expected duration times depending on whether we calculate the expected Value of completing an order now versus the expected value of postponing it.

Therefore, when it comes to calculating the Expected Duration we have two scenarios. In the first scenario we are assuming that the order will be completed next implying our Estimated Completion Time is:

$$EstimatedCompletionTime = WaitTime + DroneTravelTime + DeliveryTime$$

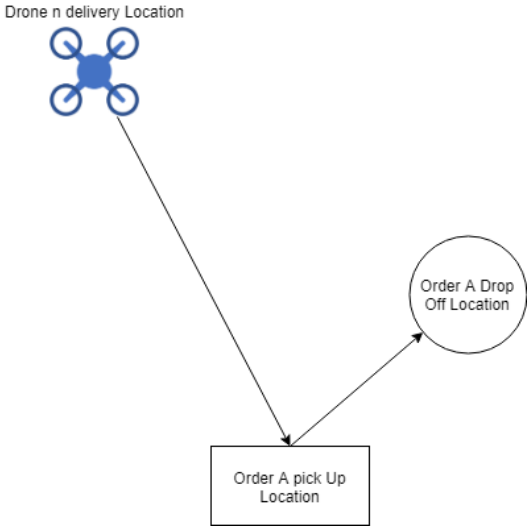

Figure 5.1: Travel Distance if Executed Immediately

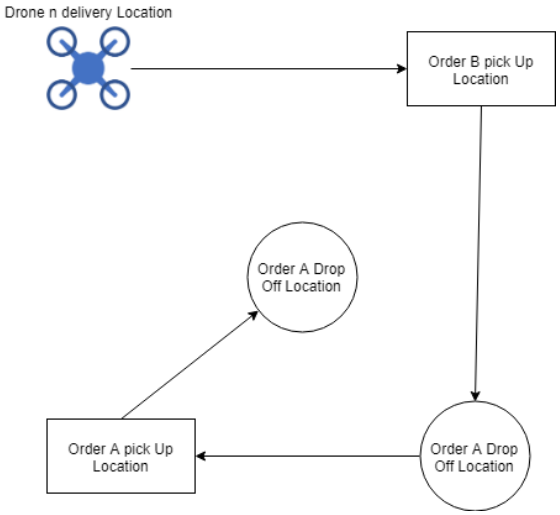

Figure 5.2: Travel Distance if postponed until after order B

Whilst in the second scenario, we are postponing order A until after order B, hence we have an Estimated Completion Time If Postponed.

$$\begin{aligned} EstimatedCompletionTimeIfPostponed = {} & WaitTime \\ & + BTravelTime \\ & + BDeliveryTime \\ & + ATravelTime \\ & + ADeliveryTime. \end{aligned}$$

Where B Travel Time is the time taken to travel from the drone‘s current position, to the pick up location for order B. Similarly, A Travel Time is the time taken to travel from the drop off location of order B, to the pick up location of order A. The current Wait Time of each order can be calculated by subtracting the order placed time from the current time.

$$WaitTime = CurrentTime - OrderPlacedTime$$

Note that we implemented LLV at both a global level, and a local level (Local Least Lost Value (LLLV), where each restaurant has its own LLV scheduler, to determine if there were significant difference.

5.2.6 Just In Time JIT

Just in Time scheduling is a dynamic programming inspired approach which we have devised. It uses the same estimated completion time calculations as LLV to evaluate the potential profit of an order, but it also calculates the order's slack time. We define slack time as the difference between the time at which the order next loses value t_b, and the estimated completion t_e time of the order.

$$SlackTime = t_b - t_e$$

Where the boundary time t_b is defined as:

$$t_b = \max_t V(t) = V(t_e)$$

Figure 5.3 demonstrates this calculation. In this example our boundary time t_b is 24 minutes, denoted by the red line. Order A, denoted by the green line, has an estimated completion time t_e of 23 minutes, which gives a slack time of 1 minute (24-23). Order B, denoted by the blue line, has an estimated completion time of 20 minutes, resulting in a slack time of 4 minutes. Therefore, order A would be chosen as the next order to be completed. The intuition behind JIT scheduling relies on postponing an order until it is about to lose value. By doing so you are making the best use of the time available and whilst minimising the expected loss.

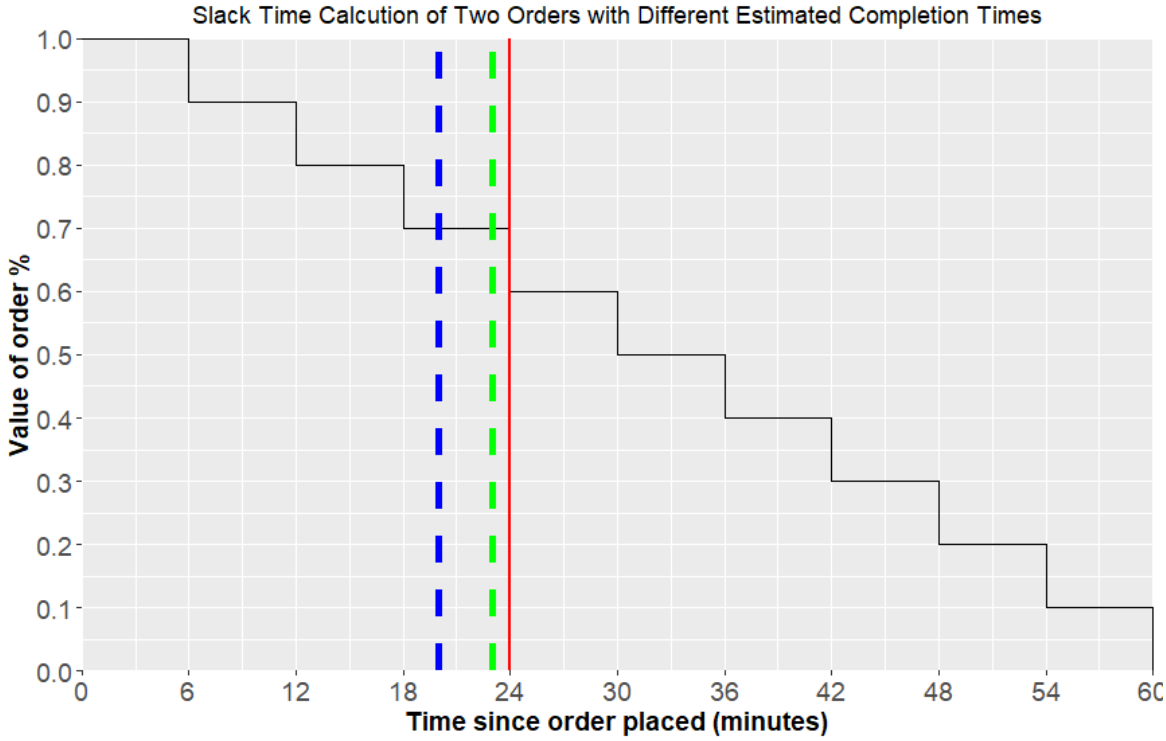

Figure 5.3: Slack time comparison for Order A green and Order B blue

Unfortunately, in our first implementation of the JIT scheduler, we did not give enough weighting to completing jobs earlier. As a result, it performed quite poorly and was unable to generate a profit which matched the other schedulers. The distribution of delivery completion times can be seen in figure 5.4. This shows how JIT does not prioritise orders with higher potential profits. Therefore, we altered our JIT scheduler to classify orders into groups based upon the potential profit of their estimated completion time. We then searched through the group with the greatest profit, and chose the order with the smallest slack time from that group.

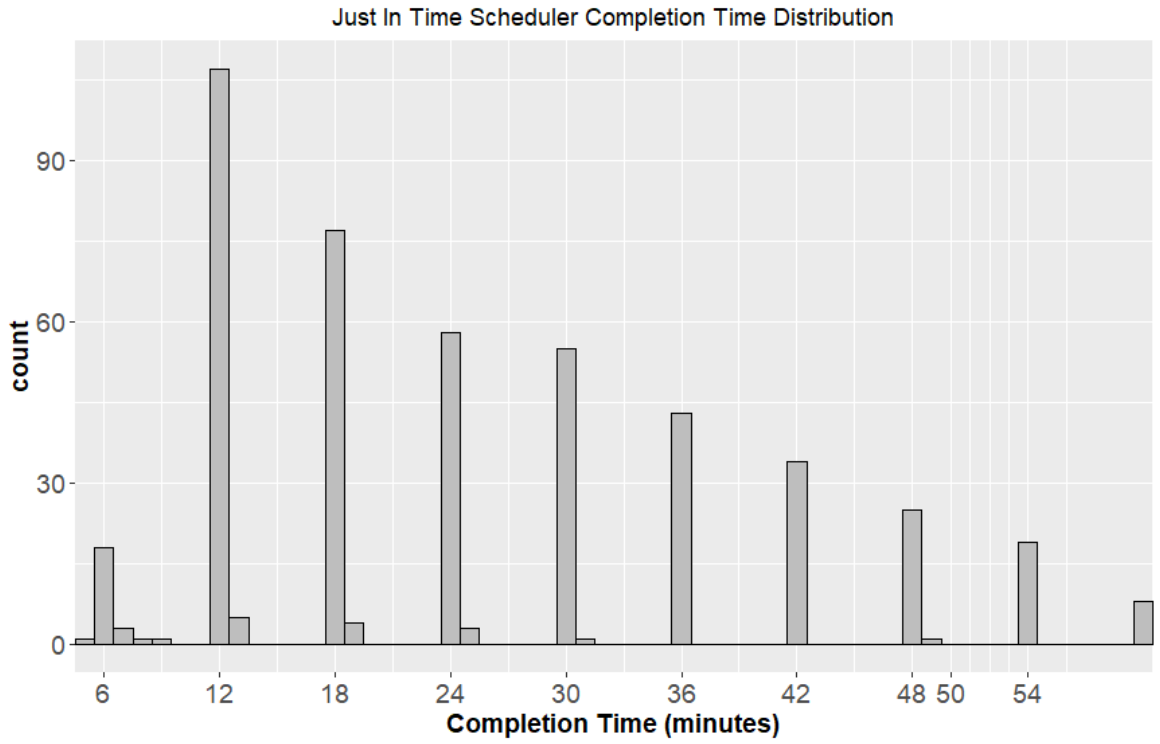

Figure 5.4: Distribution completion times for Just In Time Scheduler

Chapter 6

Visualisation

This project could have been solely implemented as a Discrete Event Simulator. This would result in a simulation time line where we add the completion time of each event and jump to each event time. However, due to the nature of our simulation being on a real world map, it was desirable to visualise the simulation. This visual tool has allowed us to create a relatable scenario and insert realistic assumptions such as restaurant locations, and No Fly Zones.

6.1 Setup

The visualisation began with moving a drone around the world and routing it between set points. This allowed us to visualise and test that the drone behaved correctly when ascending, descending and travelling between locations. Once the basic logic of the simulation had been implemented, we were able to create a snapshot of West London to be used for our scenarios. A 5kmx5km section of West London was taken from Google Earth to form a background for our simulation. This gave us a 2D image as a starting point.

6.2 Bird's Eye 2D view

With a realistic background in place, we added appropriate No Fly Zones extending from the ground to the drones' maximum operating altitude. Unity's line renderer allowed us to visualise the routes the drones were completing. The visualisation was

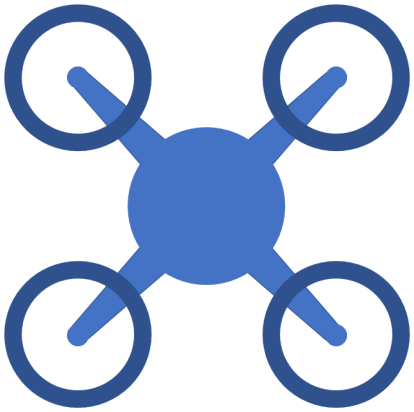

Figure 6.1: Drone not carrying food icon

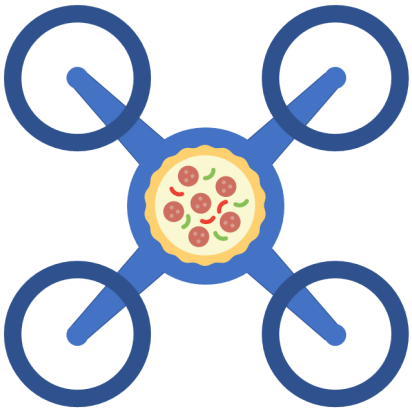

Figure 6.2: Drone carrying food icon

Figure 6.3: Restaurant Icon

Figure 6.4: Customer Icon

then enriched to include drone and customer icons, restaurant icons, and images to distinguish between drones performing deliveries, and those enroute to pick up an order. This birds eye Scene view has been our main visual tool throughout the development of the simulation. It has allowed us to quickly identify otherwise tricky to discover bugs. For example, when the list of idle drones was overruling the local scheduler forcing drones away from their home location.

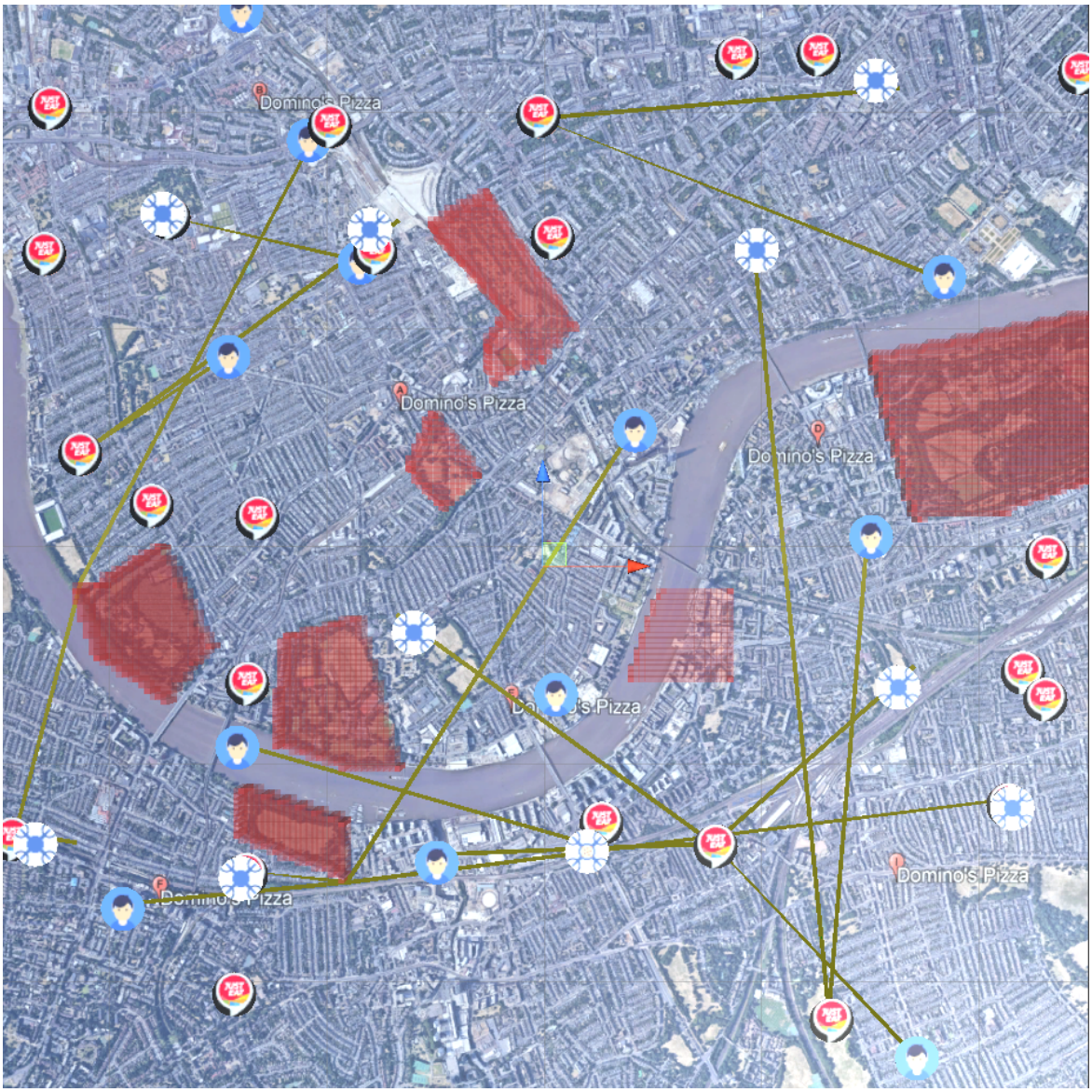

Figure 6.5: Birds eye simulation view with icons and routes

6.3 3D World Simulation

Developing the simulation in Unity has given us the opportunity to create a full 3D visualisation in addition to our 2D birds eye view. Unfortunately, the creation of an accurate 3D model would have expended significant time and effort. Since the visualisation was a tool for this project and not its main aim, the decision was made not to spend excessive time on a 3D model. However, we have been able to use a few pre-existing models to demonstrate our visualisation.

Mapbox(35) has an SDK for Unity which is designed to help you create 3D models of the world. After briefly experimenting with it, we were able to import a city simulation model. Upon integrating our simulation with the city model, we able to simulate a more realistic visualisation from the drone point of view.

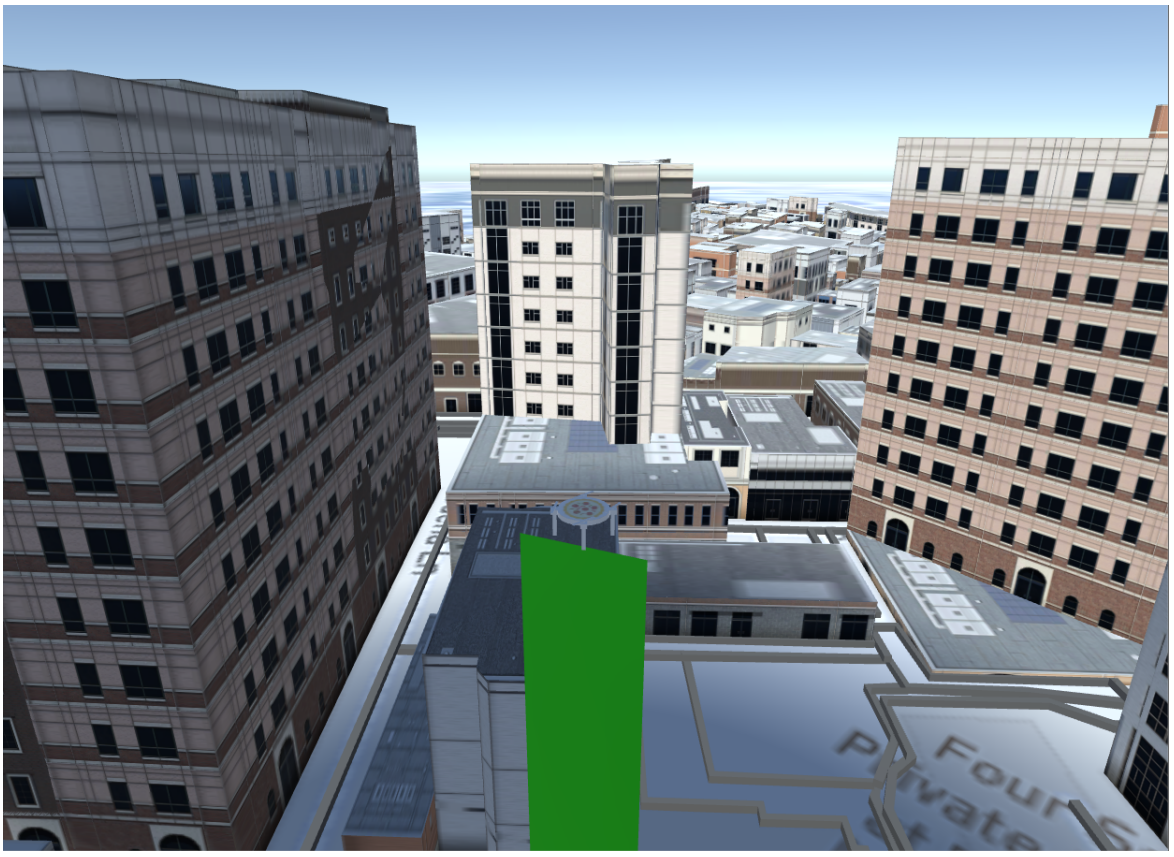

Figure 6.6: Mapbox City Visualisation

For our larger Just Eat simulation, we were able to use satellite data of London to generate a 3D model (22). This data covered a huge swathe of Greater London, which was a bit excessive for our needs, but we were able to use it to enhance the visualisation.

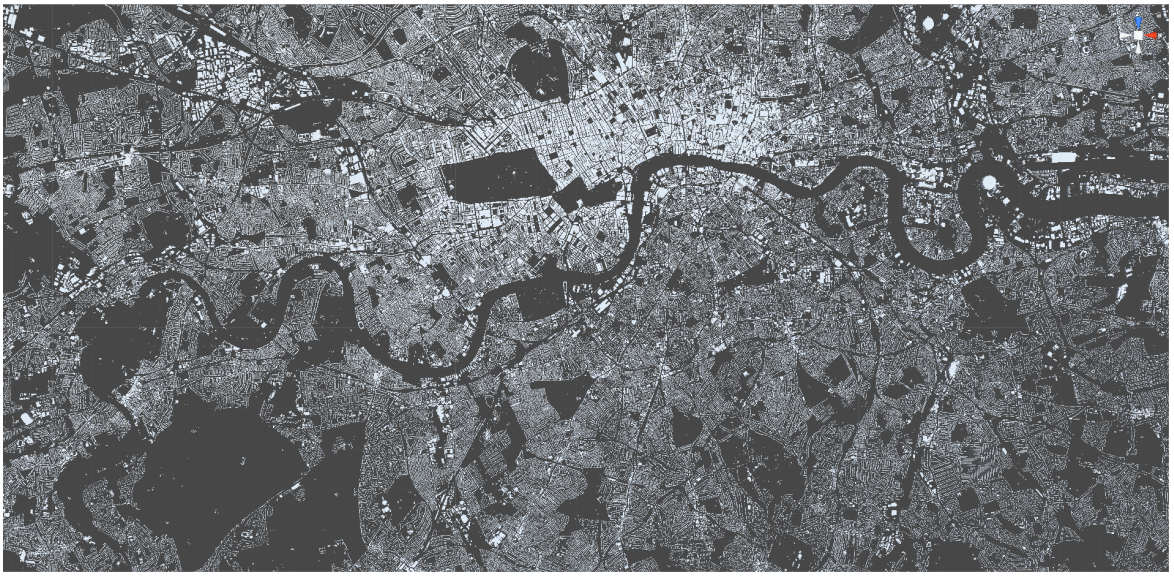

Figure 6.7: London 3D representation from satellite data

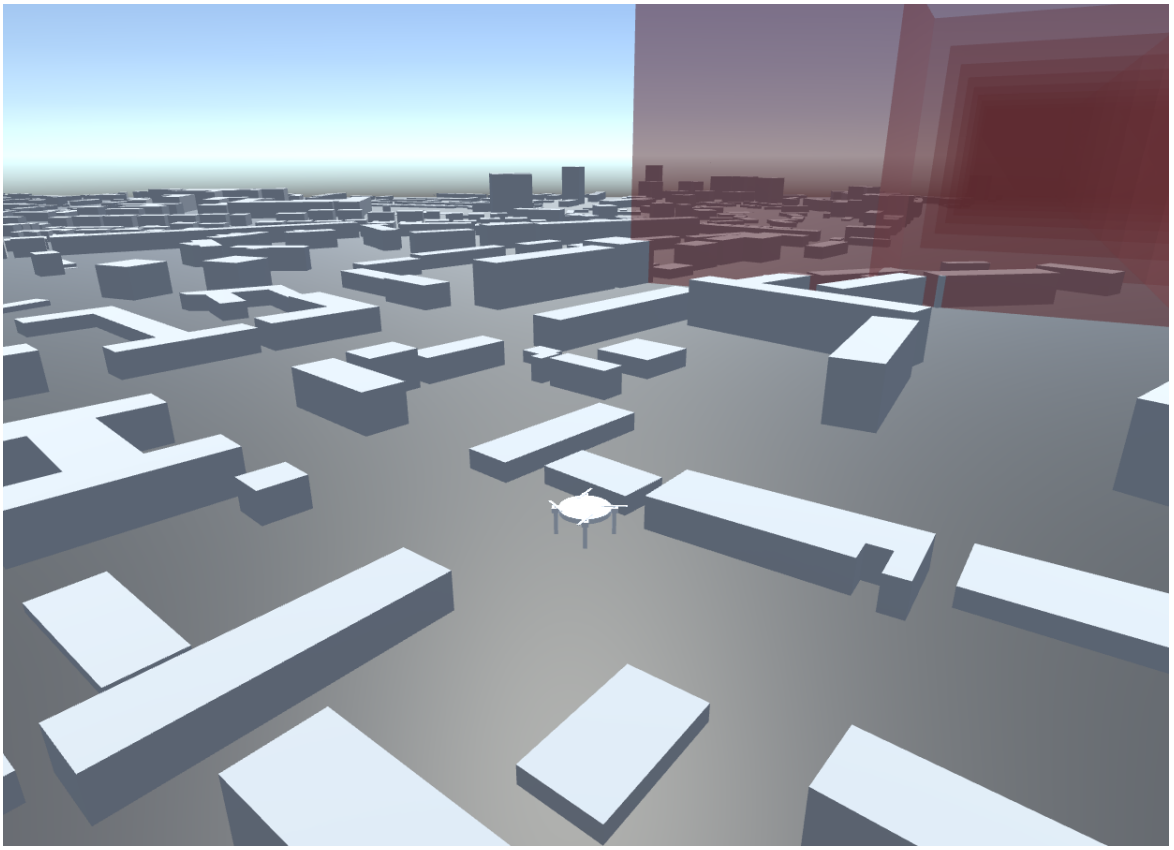

Figure 6.8: Drone first person view in London 3D model

6.4 Custom Editor

Unity allows you to write custom editors for any of the game objects you use. Adding a custom editor script for our Drone Controller allowed us to view the details about each drone as the simulation was running. As seen in figure 6.9 we can view the drones current location, status, order, way-point and destination. We made use of Unity's inspector for the Fleet Controller and Customer objects to view real time details as the simulation progressed.

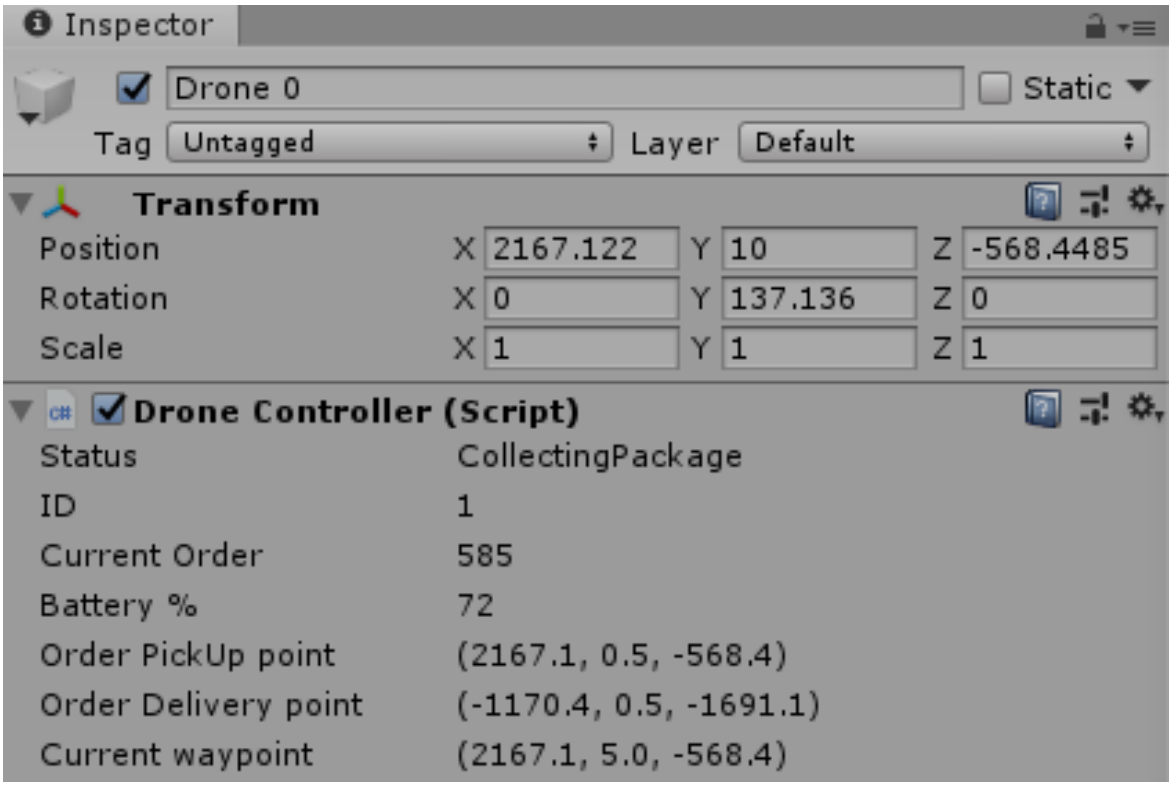

Figure 6.9: Unity Inspector showing details about each drone

Part III

Simulation Results and Evaluation

Chapter 7

Domino‘s Pizza Scenario

For our first scenario we will suppose that Domino‘s Pizza has a fleet of delivery drones. Domino‘s is a prime candidate since they have a number of locations throughout London which matches our multiple source node model. Domino‘s themselves have made forays into pizza delivery by drone(41), however there have not been many developments since. They epitomise drone delivery research when they say ”It doesn’t make sense to have a 2 tonne machine deliver a 2kg order.”(41)

7.1 Model Outline

There is one issue with the Domino‘s model. In real life, when you place an order it is sent to you closest Domino‘s Pizza restaurant. This means that each drone would only be delivering within a certain radius. Therefore, there would not be the opportunity for drones to move between source hubs to satisfy dynamic demand. This effectively takes us back to each restaurant following the warehouse model, where drones are tied to their hub. To overcome this, we have scheduled on a global basis. This means that when an order is received for a given restaurant it can be assigned to any drone, instead of just the drones which are based at that restaurant. This will allow the scheduling algorithms the freedom to move drones between restaurants in order to satisfy demand. We then compare these global scheduling algorithms against localised scheduling algorithms, where the drones are tied to one location.

To make this first scenario manageable, we will consider a 5km square region of West London from Hammersmith to Battersea, centering over Fulham. This area of London contains 6 Domino‘s Pizza restaurants which will allow us to study how well our scheduling algorithms serve several locations.

The 6 Domino‘s Pizza restaurants we used were: Battersea, Battersea bridge, Fulham, Parsons Green Putney and West Kensington. Their locations can be seen in

figure 7.1 below.

Figure 7.1: Location of theDomino's Pizza restaurants in our simulation

7.2 No Fly Zones

We will introduce No Fly Zones over public parks and the London Heliport. As such we have set No Fly Zones over Hurlingham Park, Battersea Park, EelBrook Common, Brompton Cemetary, Bishops Park, Wandsworth Park and the London Heliport which is located in Battersea. The parks were chosen because the use of drones is banned in all London parks except Richmond park. The London heliport was chosen since it is a crowded airspace for helicopters in and out of London. These No Fly Zones can be seen in figure 7.3.

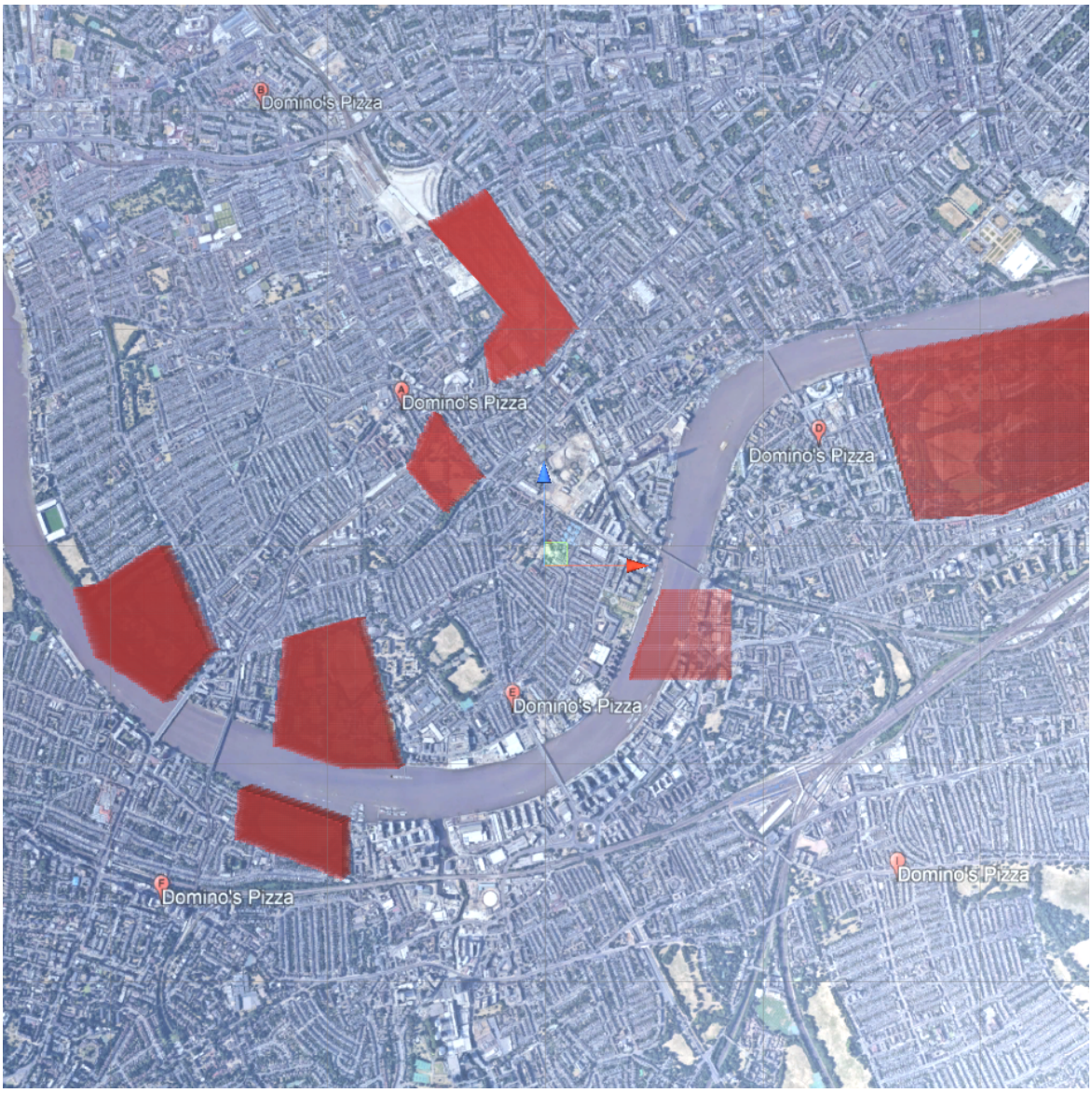

Figure 7.2: Location of No Fly Zones in our Domino‘sPizza simulation

7.3 Assumptions

For this scenario we will assume that each Domino‘s Pizza location has 2 drones, giving us a fleet of 12. For Local First Come First Serve (LFCFS), Local Shortest Job First (LSJF), and Local Least Lost Value (LLLV), we will lock the drones to their source locations. This means they will only travel to and from their designated home restaurant, delivering within that restaurants service area. In the case of the global schedulers: FCFS, SJF, ESJF, CJF, LLV, JIT; we will not lock the drones to any particular restaurant. This allows them to travel freely between locations to satisfy demand. We will run our simulation for 6 hours to model an evening shift.

Since our schedulers will not be effective when the queue is empty, we have to

ensure that there is a moderate load on the system. Therefore, we set a moderate load of an order being received on average every 25 seconds, a heavy load of an order being received every 20 seconds, and a dynamic load. The dynamic load is introduced to test how the schedulers behave when one source Location suddenly becomes very busy.

For the dynamic load, we could have just sent every second order to one node to simulate it being busier than all the other nodes. However, if this was the case in real life, you would choose to base more drones at that location in the first place, as you know it receives more orders. Therefore, to simulate a scenario where you don't know which node will be busy, we increase the load on each source location in turn. This is achieved by altering the order generator to send every second order to the same source location for a one hour period. Once the hour has elapsed, the next source location is chosen and the load increased.

Figure 7.3: Birds eye view of London Domino's Pizza simulation

7.4 Results

7.4.1 Moderate Load

Testing the system under moderate load, where an order arrived on average every 25 seconds, we found that all of the schedulers completed approximately the same number of orders, except for FCFS and SJF (Appendix figure 11.1). The top schedulers completed between 679 and 683 orders, whilst FCFS and SJF fell short with 607 and 630 respectively. For FCFS this is understandable behaviour since drones may be routed from one corner of the world to the other in order to respect the queue ordering. In the case of SJF, we are prioritising on delivery time, not taking into account the drone travel time. As a result, it will also suffer from extended pick up distances. Therefore, since these two global schedulers performed poorly in the number of orders completed, total profit, and average profit (Appendix figure 11.2), we have chosen to exclude them here. Full graphs can be found in the appendix.

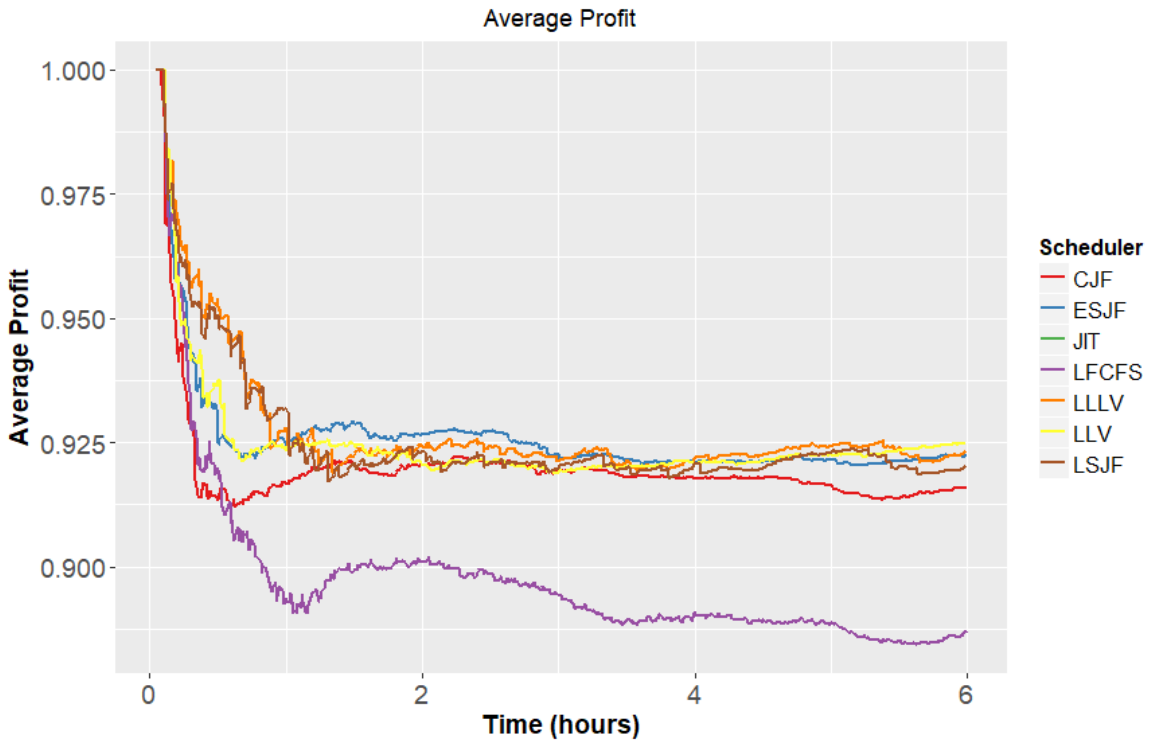

Figure 7.4: Average Profit under moderate load for Domino's Pizza scenario

The graph of average profit (figure 7.4), shows that local FCFS performs worst in this moderate load scenario. Whilst there is very little difference between the other schedulers, CJF does under perform. This could be due to the fact that it does not take into account the delivery time, which ESJF does. However, in this scenario, it is entirely possible that CJF could result in drones becoming stuck serving one source node. CJF is a global scheduler, but since each order is sent to its closest restaurant, CJF will force drones to serve that restaurant. If at any point the queue for a restaurant becomes empty, CJF will force the drone away from that restaurant. A new drone will only ever be assigned to that restaurant if a different restaurant's queue becomes empty. As a result, orders for that restaurant will have an increased wait time until a new drone is assigned.

An analysis of the queue lengths in figure 7.5 shows that no scheduler, except FCFS and SJF, ever becomes full. Moreover, it can be seen over the course of the simulation that the queue length rarely exceeds 15. This means on average, each location has approximately 2 orders in its queue. Since the local schedulers have 2 drones serving each source location, they are able to cope with demand. As such there would be no benefit gained from moving drones between source locations.

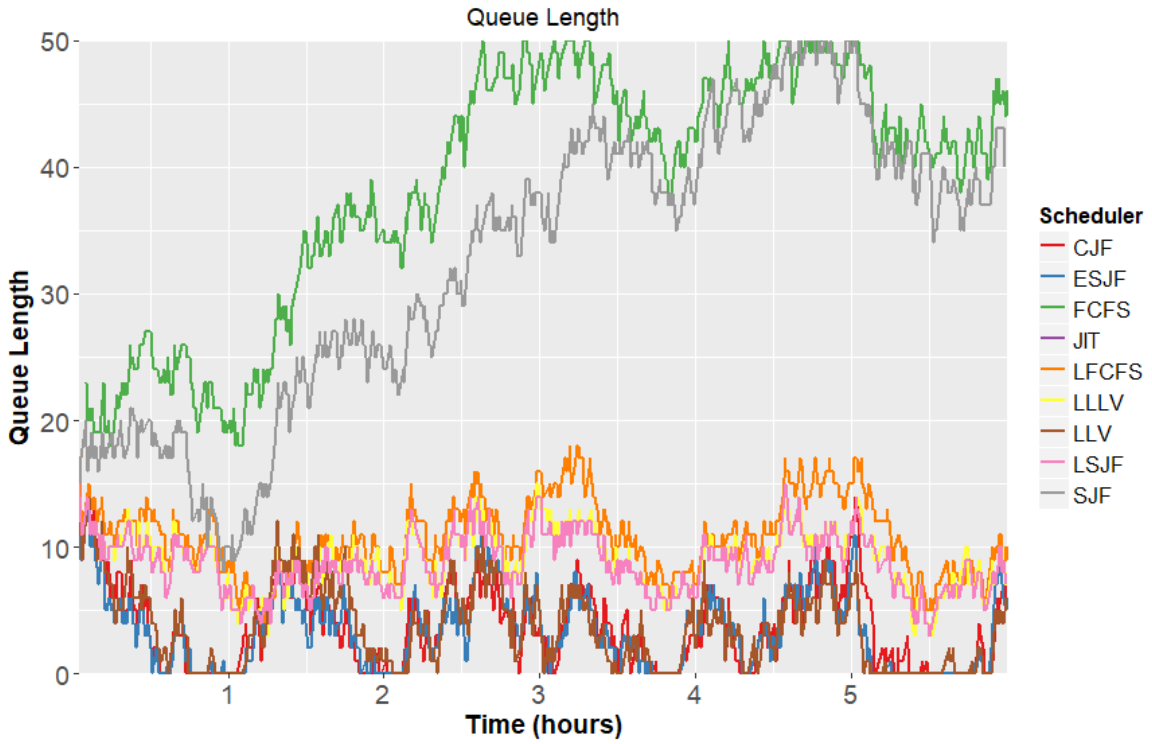

Figure 7.5: Queue Length under moderate load for Domino's Pizza scenario

7.4.2 Heavy Load

Our heavy load scenario involved decreasing the average time between orders to 20 seconds. This increases the demand on the drones, resulting in a greater influence from the schedulers. Again, we have omitted FCFS and SJF from these graphs as they completed the fewest number of orders and had the lowest average profit. Full graphs can be found in the Appendix, figures 11.3, 11.4, 11.5.

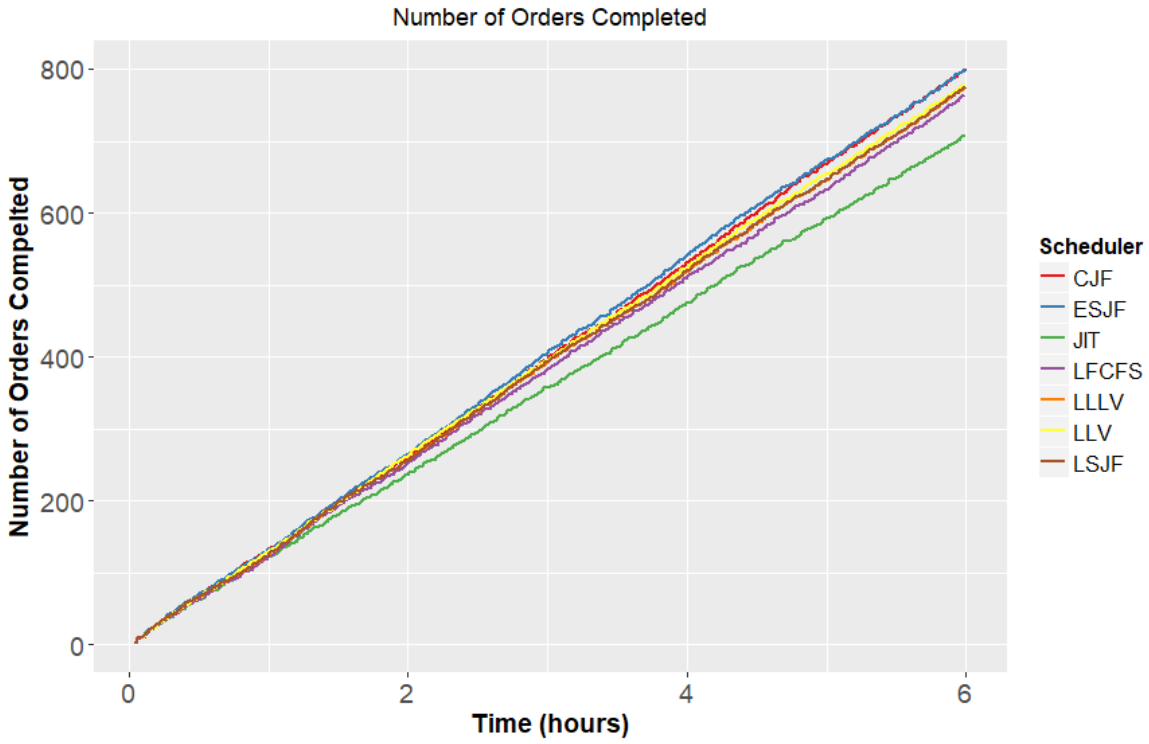

Figure 7.6: Number of Orders completed under heavy load for Domino's Pizza scenario

Figure 7.6, reveals that there is not a vast difference in number of completed orders between the majority of the schedulers. All schedulers except JIT fall Within a range of 765 - 800 orders completed, with ESJF and CJF completing the most orders. This is understandable as both schedulers are prioritising based on which jobs can be completed fastest. Moreover, with a higher demand on the restaurants, CJF is unlikely to suffer the pitfalls mentioned earlier. JIT falls behind the rest of the schedulers, only completing 708 orders. Interestingly, LLV seems to perform approximately the same irrespective of whether it is applied locally or globally.

Figure 7.7 shows the average profit per order achieved by the schedulers. JIT achieves the highest consistent average profit of 0.925. LLLV outperforms the remaining schedulers, including global LLV. LSJF performs marginally better than ESJF, supporting the argument in favour of local schedulers over their global counterparts. The local schedulers may be performing better than their global equivalents as there is no additional travel time added by moving drones between locations. This

allows all of a drone's flight time to be used for deliveries instead of moving between source locations.

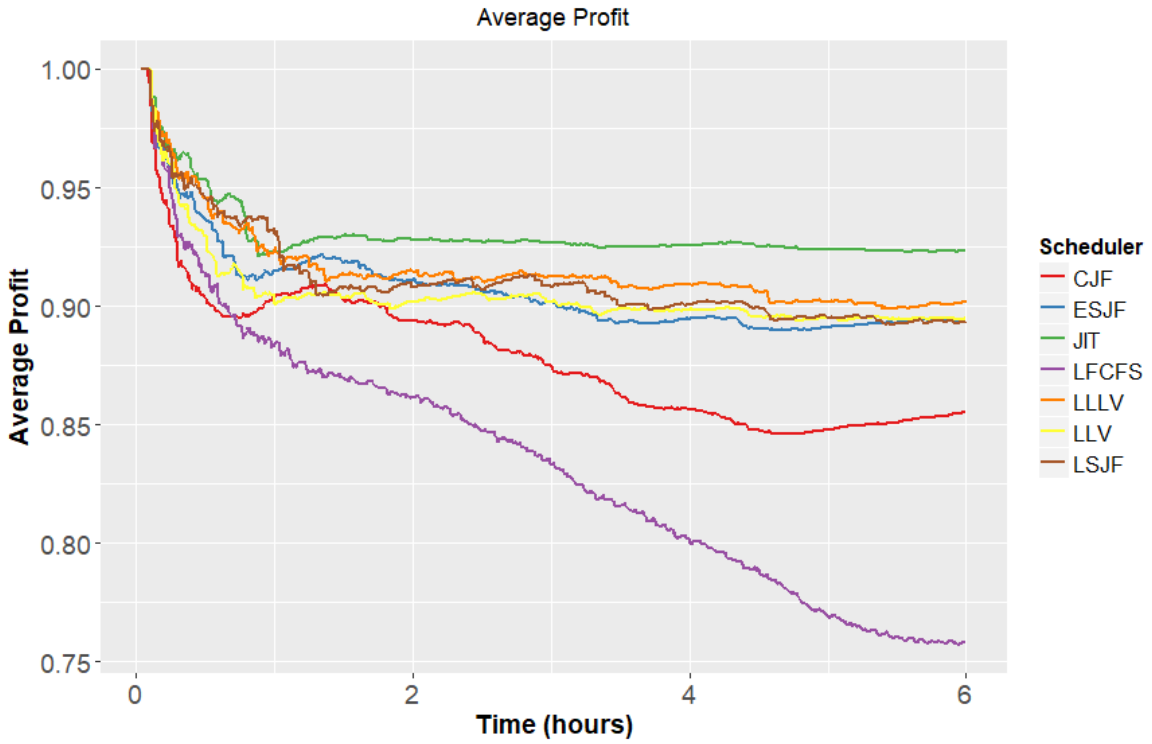

Figure 7.7: Average Profit under heavy load for Domino's Pizza scenario

7.4.3 Dynamic Load

It is interesting to consider how the schedulers cope under dynamic load. If every source locations is under heavy load, there is no reason to move drones between locations. However, if one location suddenly comes under heavy load, there is an expected gain in moving drones to that location. This flexibility is allowed by the global schedulers, and is demonstrated in our dynamic load scenario.

The local schedulers fail to complete as many orders as the global schedulers by a significant margin. Both ESJF and CJF completed 830 orders, followed closely by LLV with 817. JIT completed 763, whilst the local schedulers fell to 712 and 711 for LLLV and LSJF respectively. LFCFS performed the worst, only achieving 680 orders. These results can be seen in Appendix figure 11.6.

Figure 7.8 shows the average profit for each scheduler. A clear distinction can be seen between the global schedulers, who are able to dynamically move around the world and chase the demand, and the local schedulers which become overloaded. JIT remains the highest performing scheduler on average profit, but a clearer distinction is drawn between LLV, and both ESJF and CJF, which it outperforms. ESJF

and CJF both fall short of LLV and JIT as they do not take into account the wait time of the order. ESJF will always choose the shortest job regardless of how long it has been waiting. Consequently, it completes more jobs, but neglects those jobs which will lose profit if postponed. Both LLV and JIT take the wait time into account in order to calculate the potential profit of an order.

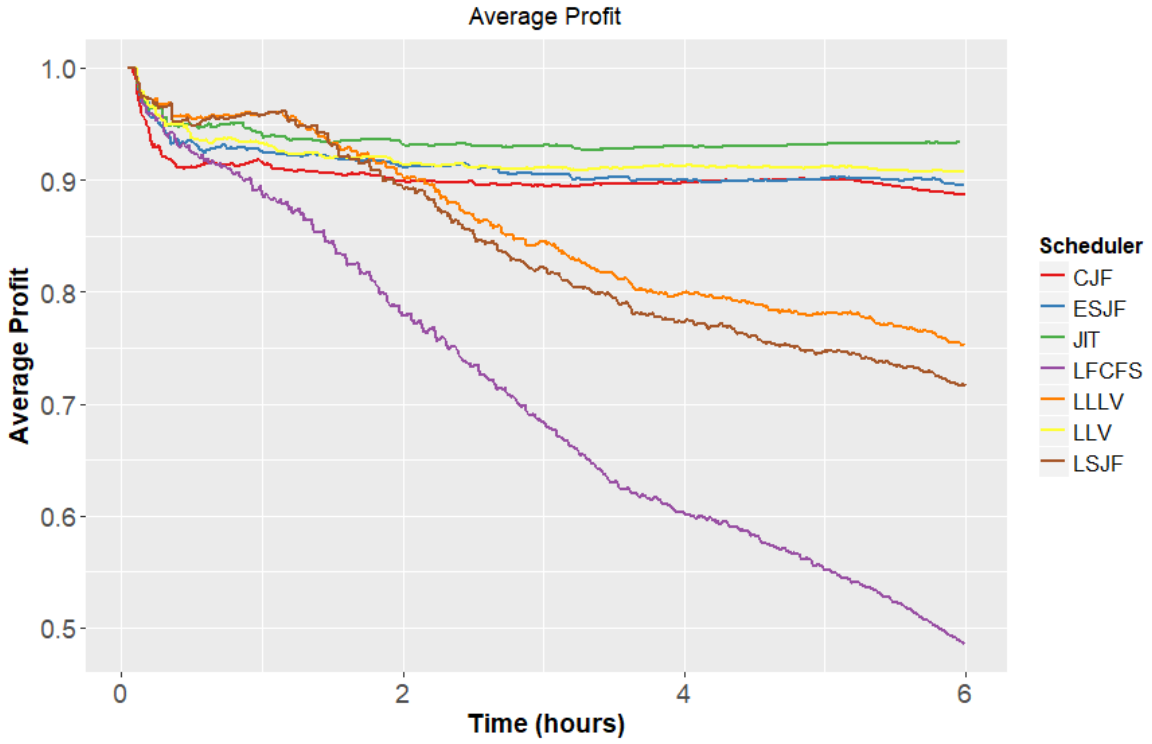

Figure 7.8: Average profit per order under dynamic load for Domino's Pizza scenario

Despite JIT having the highest average profit per order, it is unable to achieve the greatest profit as it completes far fewer orders than the other global schedulers. This can be seen in figure 7.9. It is likely that in this Domino's Pizza scenario the distances are not great enough to have a significant impact on completion times. Since each delivery is relatively short, as it is assigned to its closest restaurant, the job duration may not be long enough to have a significant effect on the profit. Indeed, the average completion time for every scheduler falls between 10-15 minutes, as seen in appendix 11.7. In different scenarios with greater delivery times, JIT's profit scheduling approach may be more distinct.

Figure 7.9: Total Profit under dynamic load for Domino's Pizza scenario

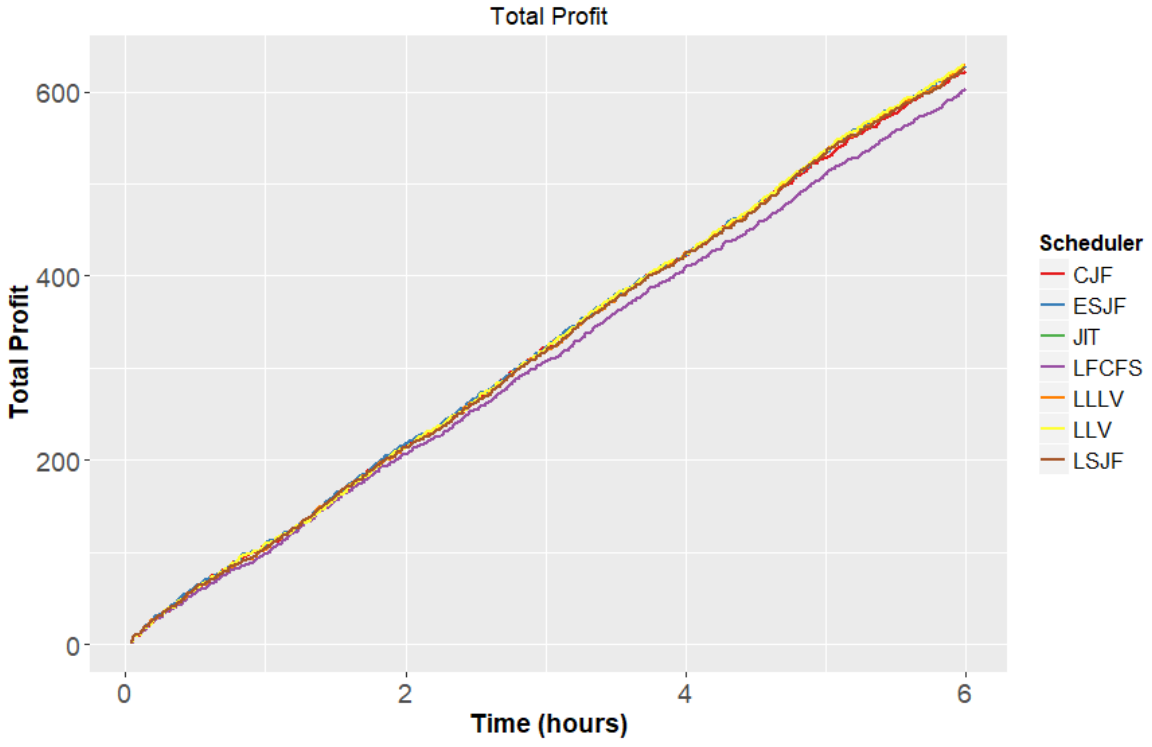

Figure 7.10: Total Profit under moderate load for Domino's Pizza scenario

7.5 Conclusions

Under moderate load there is very little difference in total profit between the schedulers (figure 7.10). In these circumstances, none of the source nodes are severely overworked. Therefore, each node is able to cope with its current demand and there is very little to be gained from the global schedulers. Due to the proximity of the nodes to each other, and the short delivery distance, the time taken for a drone to travel between source locations effectively outweighs the potential gain.

However, under heavy load a greater distinction is seen between the schedulers. JIT performs significantly fewer deliveries, but has the highest average profit per delivery, figures 7.6 and 7.7. CJF performs poorly because once a drone moves from one source location to another, it runs the risk of becoming trapped there until the load on that location decreases. This results in the drone's previous location being neglected. The remaining global schedulers, ESFJ and LLV, perform similarly to their local counterparts LSJF and LLLV. This can be attributed to the distribution of drones. If every location is experiencing heavy load then you would split your drones evenly between them. Consequently, there is little to be gained from the flexibility of global schedulers.

In the dynamic load scenario the results are very different. Here the global schedulers demonstrate their flexibility to respond to increased load, severely outperforming the local schedulers. JIT consistently achieves a high average profit, but completes fewer orders. We believe the proximity of orders in this scenario results in a very tight grouping of average profit. This is a byproduct of the short distances drones have to travel, due to the closest restaurant order allocation. Therefore, in this scenario, the greater average profit gained by JIT is not enough to overcome its lack of completed orders. The order distances are too short, resulting in every scheduler having a high average profit. If the distances were greater, we would expect the average profit of each scheduler to vary more from its counterparts.

Overall, this simulation demonstrates that under moderate load, there is little benefit scheduling at the global level. Due to the nature of orders being assigned to their closest restaurant, it is reasonable to operate each restaurant independently with a warehouse model. However, if the load on restaurants is highly dynamic, there is a significant benefit in scheduling drones at the global level.

Chapter 8

Just Eat Delivery Model

8.1 Scenario

Domino‘s Pizza is an example of a company which has multiple stores all serving the same products. However, with the emergence of food delivery companies such as Just Eat and Deliveroo, this model is changing. These companies serve multiple restaurants, each with different products. As mentioned in chapter 2, Uber Eats has begun research into its own drone delivery network. In order to model this scenario we need a few different assumptions.

In the Domino‘s Pizza model we assumed that an order would be sent to its closest restaurant. In our Just Eat model this is no longer the case. When an order is received it is for a specific restaurant and can be anywhere within that restaurant‘s service radius. The restaurant is free to set its service area, but a mentioned in section 3.2, we have set our restaurant radius to 5km. This suits our 5kmx5km scenario as each restaurant can serve the majority of the area. Moreover, it also adheres to our drone constraints, which have a maximum range of 16km.

This scenario should allow greater freedom to schedule drones to satisfy dynamic demand. They will no longer be operating within a small radius of their closest restaurant, but could be sent almost anywhere in our 5kmx5km world. This suggests there will be greater movement of drones between restaurants as their delivery location could be closer to another restaurant. In this model, it makes little sense to use local schedulers. Indeed, Just Eat does not operate a model where drivers are tied to a restaurant, they dynamically move around to satisfy demand. Therefore, we tested the following schedulers in this scenario: FCFS, SJF, ESJF, CJF, LLV and JIT.

8.2 Small Scale Simulation

8.2.1 Assumptions

For our Domino's Pizza simulation we used a fleet of 12 drones to serve our six restaurants. We will maintain these restaurants and the fleet size for this first small simulation. The reasons for doing so are to test our schedulers on a small scale model, and also to compare how the change of assumptions effects the results.

8.2.2 Results

We tested this scenario under both a moderate and heavy load, with mean inter arrival times of times 30 and 25 respectively. These inter arrival times are slightly higher than the Domino's Pizza scenario, reflecting the fact that orders are expected to take longer to complete, as the order delivery location can be anywhere in the world.

Moderate Load

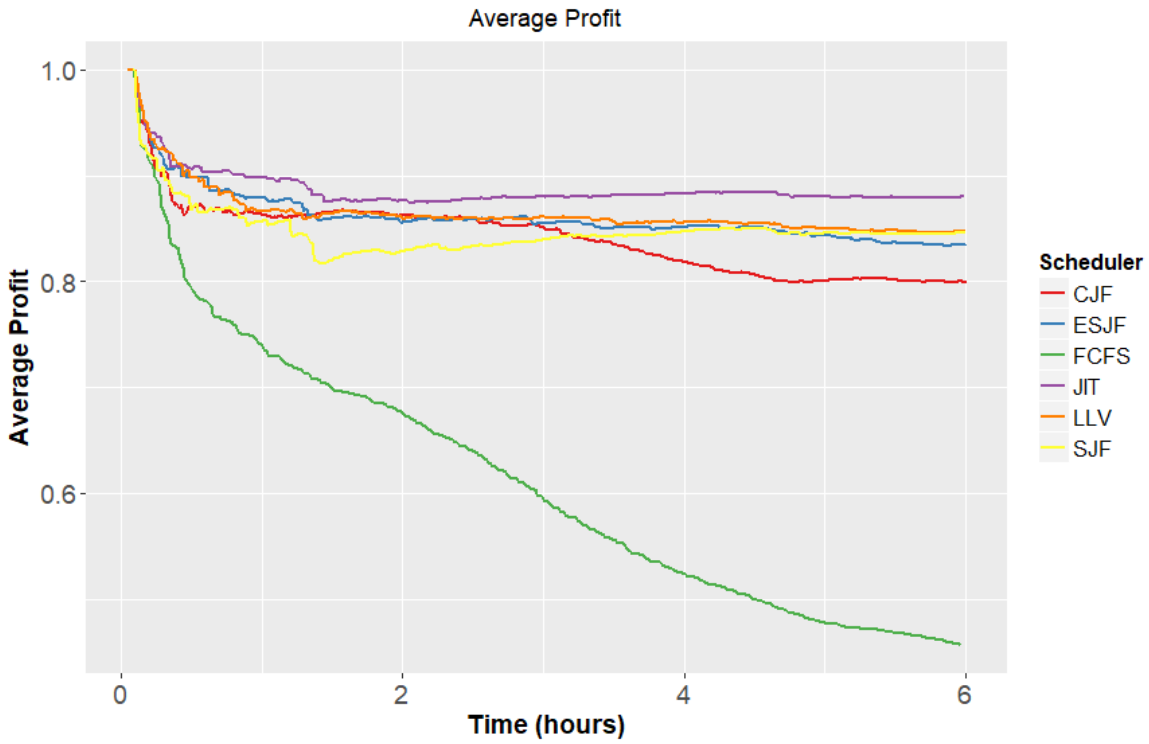

Figure 8.1: Average Profit under moderate load for Just Eat small scenario

Under moderate load the queue length remained under 25 for all schedulers except for FCFS and SJF. The number of orders completed ranged from 482 for FCFS to 527 for CJF. The ranking by number of orders completed orders was CJF first followed by LLV, ESJF, JIT, SJF and FCFS. Only 5 orders separated CJF (527), LLV(525) and ESJF(522). These results can be seen in appendix figure 11.8 The total profit follows the same hierarchy as number of orders completed (appendix figure 11.9), indicating there is not significant difference between the schedulers average profit. However, JIT maintains the highest average profit, with LLV, SJF and ESJF achieving roughly the same values. Notably, CJF under performs, as seen in figure 8.1.

Heavy Load

Figure 8.2: Number of Orders completed under heavy load for Just Eat small scenario

Under heavy load, with orders arriving on average every 25 seconds, there begins to be clear differences between the schedulers. Figure 8.2 shows the total number of orders completed by each scheduler. Despite the variation in number of orders delivered, the top four schedulers all achieve approximately the same total profit. Figure 8.3 shows a clear difference in average profit per order. JIT maintains the highest average profit per order through the simulation. LLV comes second, but performs at a similar level to ESJF. We believe SJF increases its average profit towards the end of the simulation because its queue becomes full. Therefore, it will begin rejecting orders which take too long to repeat. Consequently, there will always be short orders in the queue which SJF prioritises and completes first. This results in

the average completion time decreasing, and hence the average profit increasing as the queue fills. The graph of queue length over time can be found in appendix figure 11.10

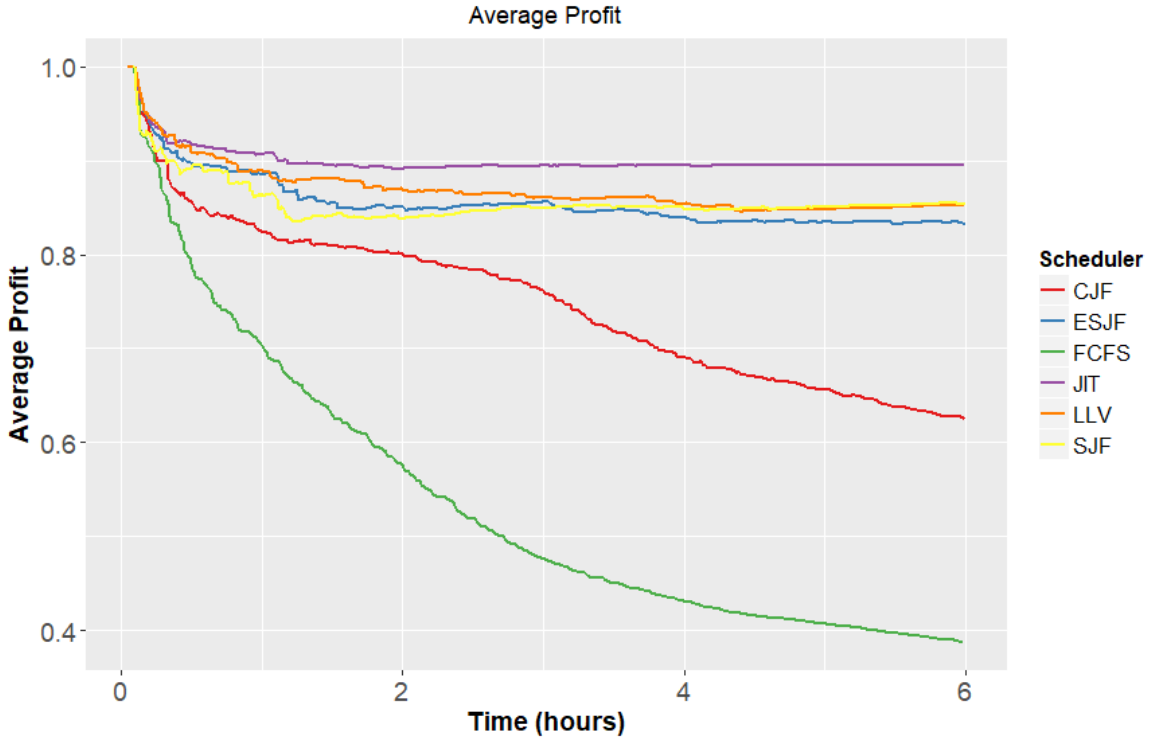

Figure 8.3: Average Profit under heavy load for Just Eat small scenario

8.2.3 Conclusions

Under heavy load it can be seen that CJF performs worse than under moderate load, figure 8.1 and figure 8.3. This comes down to CJF choosing the geographical closest pickup from its delivery location. However, as it searches for the closest job, if there are multiple jobs with the same pick up location, it will choose the first one it comes to. Therefore, CJF maintains a layer of FCFS ordering. Under heavy load, the queue length increases, resulting in the average service time of orders increasing, similarly to FCFS but not as drastically, as seen in figure 8.3.

LLV performs slightly better than ESJF in both a moderate and heavy load scenario. This can be attribute to the fact that it takes into account the wait time for an order, whilst ESJF only takes into account the delivery time and drone travel time. SJF is able to perform relatively well as its queue is almost constantly full. This is a result of prioritising shorter jobs, the longer jobs will always be postponed. As such, a shorter job will always arrive before the longer jobs are executed. This results in SJF completing a lot of short orders with a high profit, but blocks longer orders until they are evicted from the queue. The graph of queue length for the heavy load scenario

can be found in appendix figure 11.11.

JIT is able to consistently outperform the other schedulers on average profit regardless of load. In the moderate scenario, the queue is rarely full (appendix figure 11.10). However, it grows faster than LLV, and CJF because there is a priority to complete jobs with the smallest slack time. ESJF and CJF will minimises the drone travel time between destinations, whilst JIT will choose a longer travel time if it believes it can complete an order before it loses value. This is why JIT fails to complete as many orders.

Overall, it can be seen that as we change assumptions from the Domino's Pizza scenario to the Just East scenario, the increased delivery distances and effect of drone travel distances become more apparent. The schedulers complete noticeably different numbers of orders with more distinct average profit levels.

8.3 Medium Scale Simulation

8.3.1 Assumptions

The small Just Eat scenario is interesting to compare with the Domino‘s Pizza model, however, Just Eat serves hundreds of restaurants. To better model this we have run the scenario again on the same geographical area, but with 20 drones and 25 restaurants. The idea is to model one of Just Eats driver pools which serve restaurants within a designated area. The locations of the restaurants were chosen randomly, and can be seen in figure 8.7. We have adjusted the average inter arrival time for orders to 25 seconds and 20 seconds for moderate and heavy loads respectively. This is to ensure sufficient orders arrive so that our schedulers are invoked and the queues do not become empty for long periods.

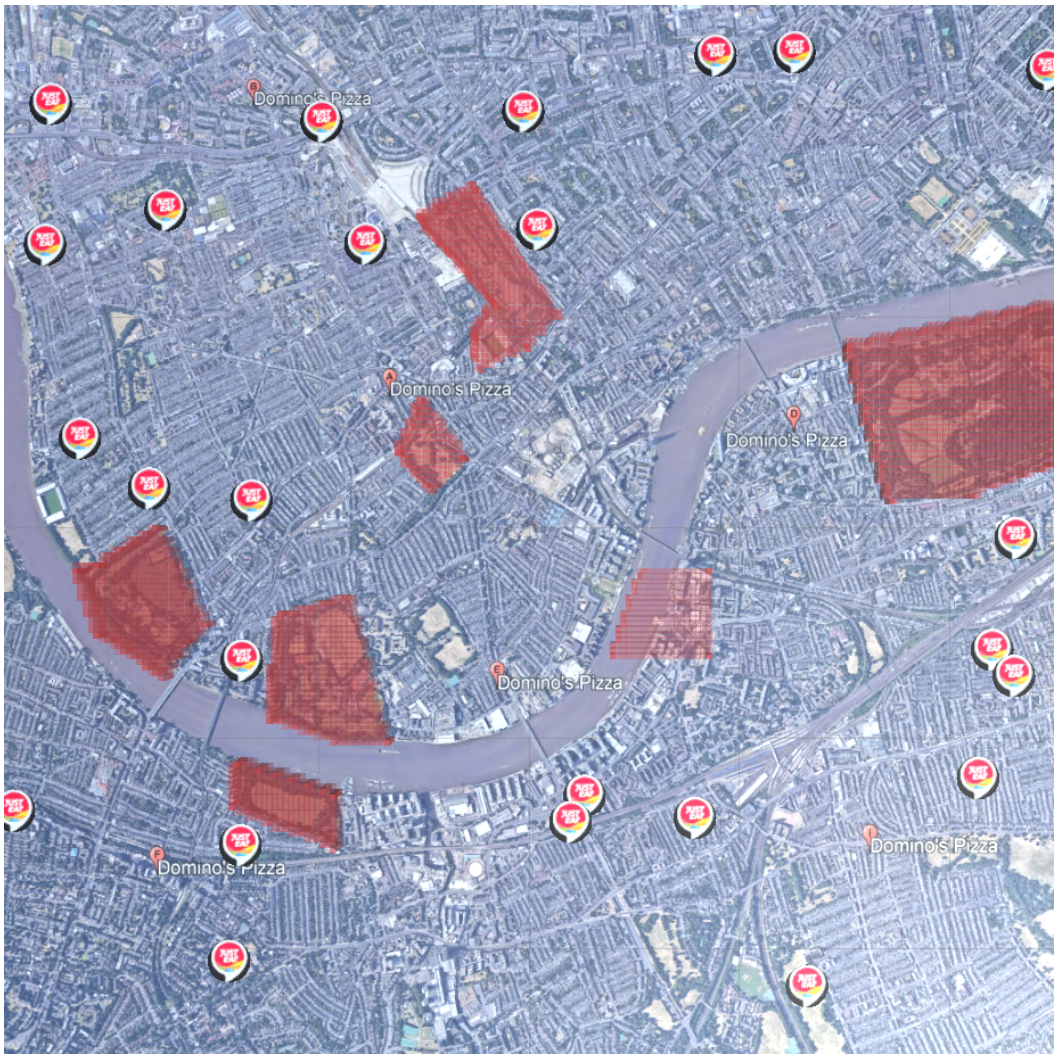

Figure 8.4: Restaurant Locations for Just Eat Medium Scale Simulation

8.3.2 Results

Moderate Load

The results under moderate load (Appendix figure 11.12) show ESJF, CJF and LLV all complete approximately the same number of orders. JIT completes noticeably fewer, in line with previous scenarios. However, by looking at the average profit in figure 8.5 we can see that CJF consistently outperforms the other schedulers. Moreover, it actually attains the greatest profit as seen in Appendix figure 11.13. This shows that CJF is able to perform better when the distances between pick up locations is smaller. CJF scheduling results in less time spent travelling between restaurants, consequently, a greater proportion of the drones' time is spent completing deliveries.

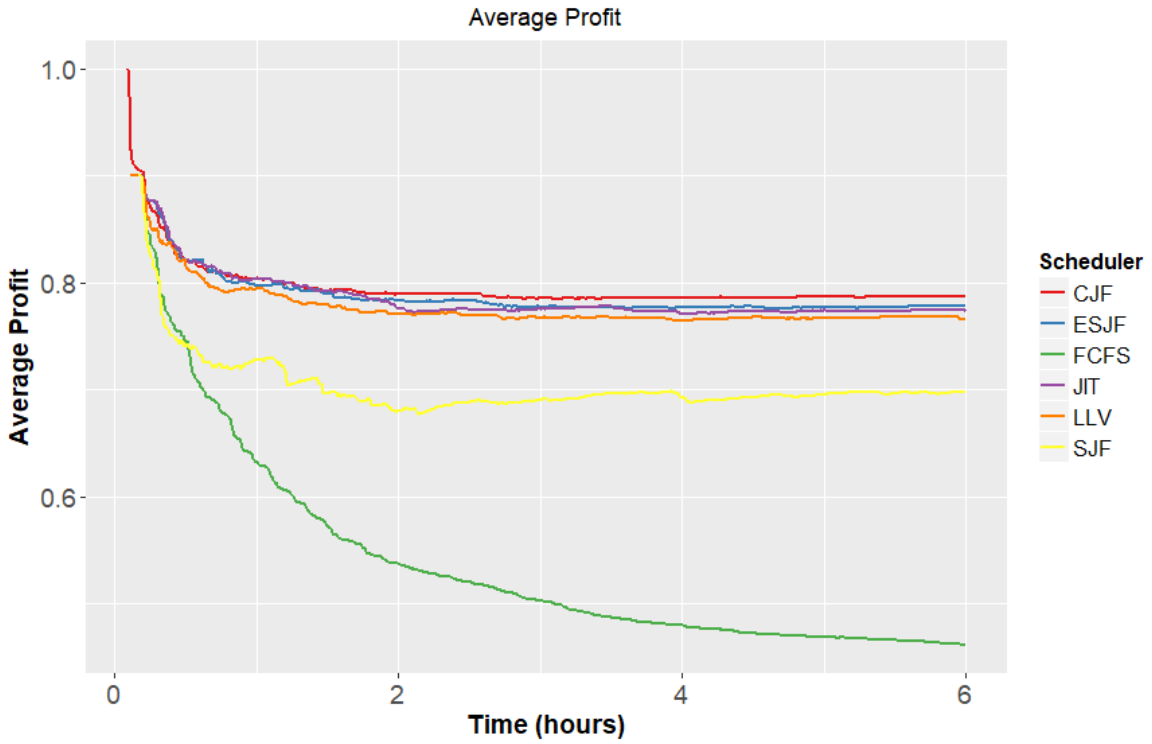

Figure 8.5: Average Profit under moderate load for Just Eat medium scenario

Heavy Load

Under heavy load, JIT and LLV maintain the highest average profit, figure 8.6. Interestingly, JIT has achieved the highest total profit (Appendix figure 11.14,)despite completing significantly fewer orders than ESJF, 609 versus 667 respectively. Additionally, LLV achieves a marginally higher profit than ESJF, whilst also completing fewer orders (644). However, throughout this scenario the order queues were at full capacity, Appendix figure 11.15.

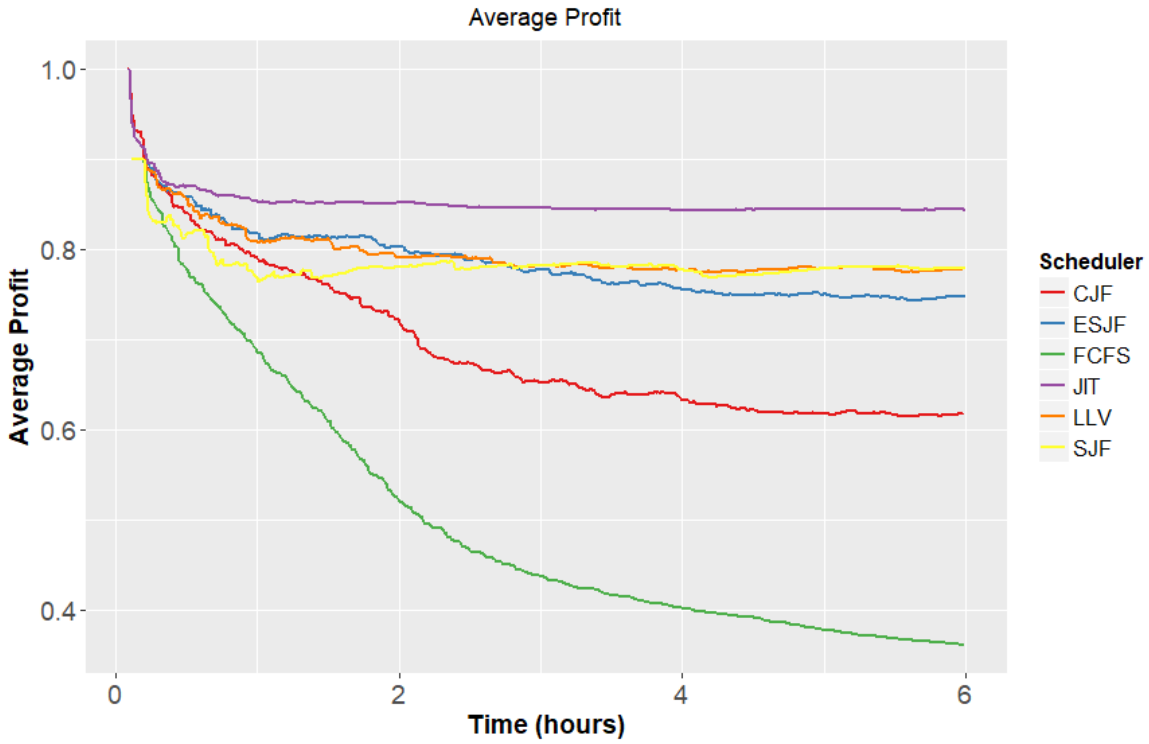

Figure 8.6: Average Profit under heavy load for Just Eat medium scenario

8.3.3 Conclusions

Our profit driven schedulers have been able to consistently outperform the other schedulers when the demand is high. CJF performs poorly a heavy load scenario due to its underlying reliance on FCFS ordering as a secondary priority. Hence when the queues are full the average service time decreases in line with FCFS. However, in the moderate load scenario the addition of restaurants has lowered the drone travel time for CJF, this allows it to outperform the other schedulers in this circumstance.

8.4 Large Scale Simulation

Our small and medium Just Eat scenarios allowed us to test how our algorithms performed when we considered a set of 6 and 25 restaurants. Moreover, it was interesting to compare how the change of assumptions effects the algorithms between the Domino's Pizza and Just Eat Scenarios. However, our medium size scenario limits orders to within our 5km world. We decided to test how our schedulers performed on a larger world.

8.4.1 Assumptions

For this larger scenario we have chosen a 10kmx10km area of London. We have randomly dispersed 50 restaurants throughout this area. Each restaurant has a service radius of 5km. In our previous scenario, each restaurant could almost cover the entire world. This scenario allows us to consider a larger world where each restaurant only serves a subsection. We have increased the number of drones to 40 and have set moderate and heavy loads of orders arriving every 15 and 10 seconds respectively. The larger world allowed us to test how increased delivery times and increased drone dispersion affected the schedulers.

Increasing the size of the world resulted in us having to add additional NO Fly Zones. We added No Fly Zones over the public parks, as well as: Kensington Palace, St Jame's Palace, Buckingham Palace, the Houses of Parliament and Westminster Abbey. The visualisation of our larger Just Eat snapshot can be seen in figure 8.7

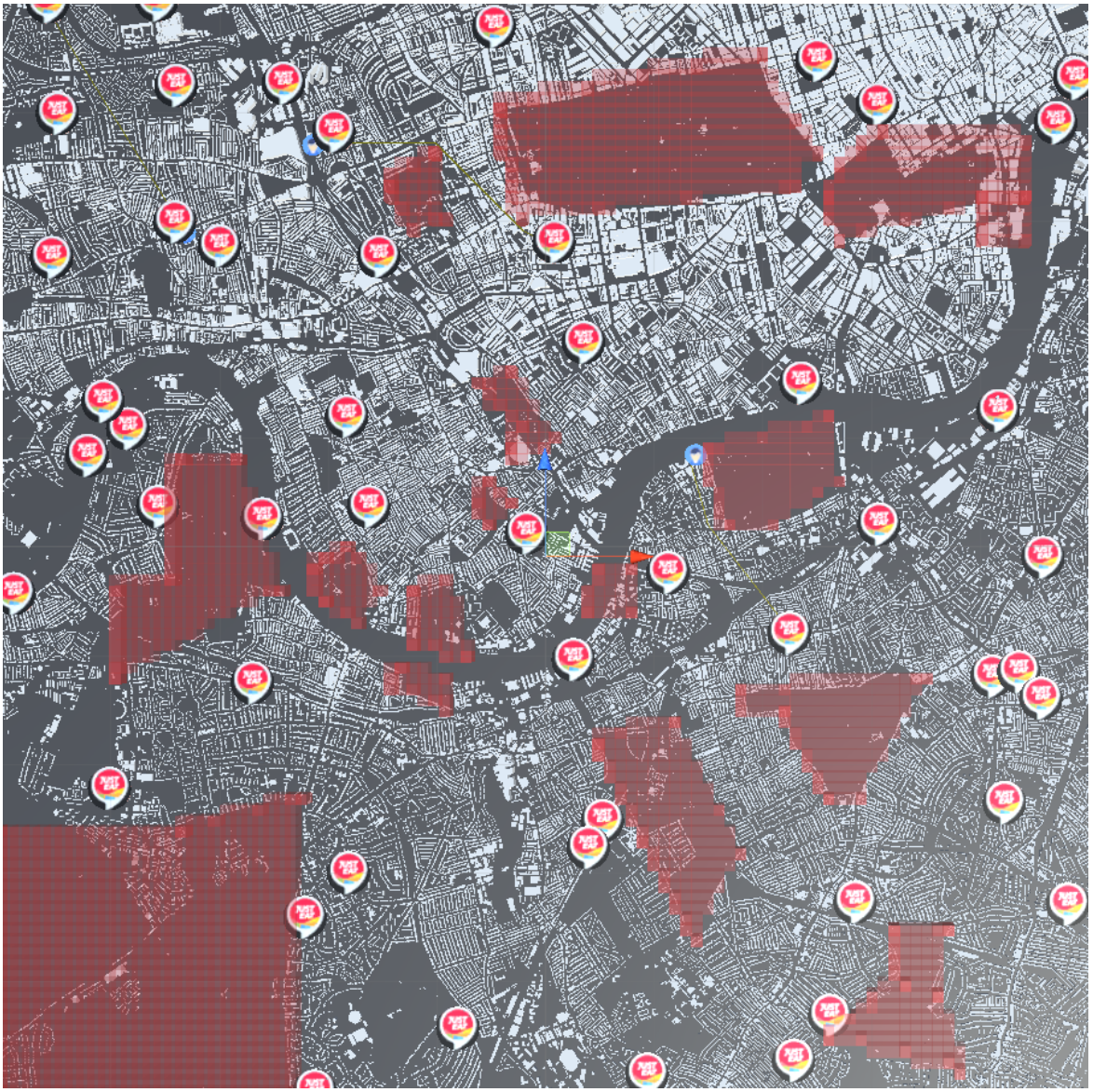

Figure 8.7: Large Just Eat simulation snapshot

8.4.2 Results

Moderate Load

In this larger scenario, the schedulers which place emphasis on minimising travel time, ESJF and CJF complete the most orders. LLV, only just falls short with 997 completed orders compared to ESJF's 1004 and CJF's 1007. JIT performs noticeably fewer orders, completing only 950, whilst FCFS and SJF are on par in this scenario, completing 812 and 822 respectively. These results can be found in appendix figure 11.16.

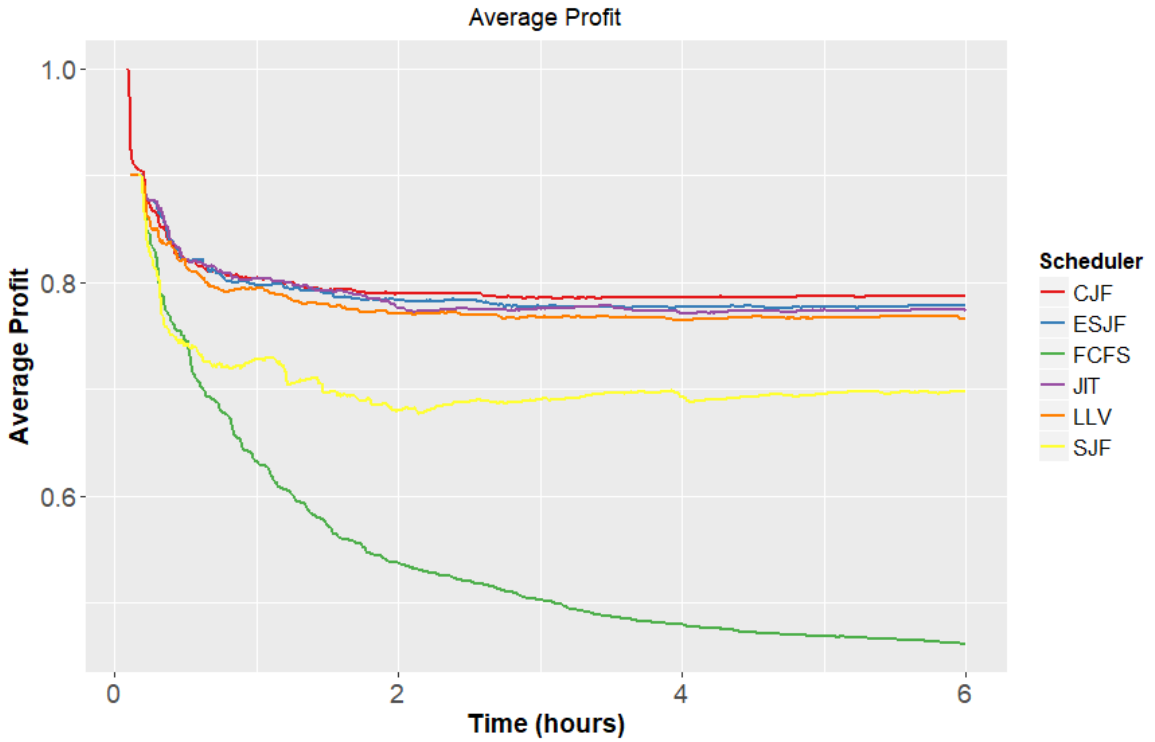

Figure 8.8: Average Profit under moderate load for Just Eat large scenario

The results show that JIT no longer achieves the highest average profit, figure 8.8. CJF consistently achieves the highest profit, followed by ESJF. The average delivery time of each scheduler is approximately the same (Appendix figure 11.18), meaning CJF has the lowest average wait time (Appendix figure 11.19. This is understandable since CJF picks the closest pick up location, and if there are multiple orders for that location, chooses the one which was placed first. In this larger scenario there are 50 restaurants distributed throughout the world. A higher concentration of restaurants results in shorter travel times between restaurants. Moreover, each restaurant is unlikely to have a large queue of orders, meaning CJF's FCFC bias does not have as significant effect as when one restaurant is under heavy load. Therefore, CJF minimises the drone travel time, and hence the order wait time, which allows it to complete orders faster than the other schedulers. The secondary reliance on

FCFS ordering also means longer jobs are not continuously postponed, as long as one restaurant is not heavily loaded. The total profit hierarchies equate to the number of orders completed in this scenario as a result of the average profit ordering. (Appendix figure 11.17)

To understand why JIT performed poorly in this scenario, we can take a look at its completion time distribution in figure 8.9. The graph shows a spike just after 18 minutes, as well as a spike after 12 minutes. We would expect these spikes to be just before each 6 minute period as JIT aims to complete an order just before it loses value. We believe the shift in completion times is a result of how JIT calculates its estimated completion time. It does not carry out a full path search for each calculation. This would require n path searches where n is the current length of the queue every time a drone requests a new order. To avoid this computation, the drone travel time is calculated based upon the Euclidean distance with a buffer time added, to account for this uncertainty. In the smaller scenarios it appears this buffer time, which was set to 60 seconds, was sufficient for JIT to complete its orders on time. However, in this larger scenario there are more No Fly Zones resulting in longer paths to avoid them. Therefore, JIT is failing to complete most of its orders when it expected to. LLV suffers from this same calculation issue.

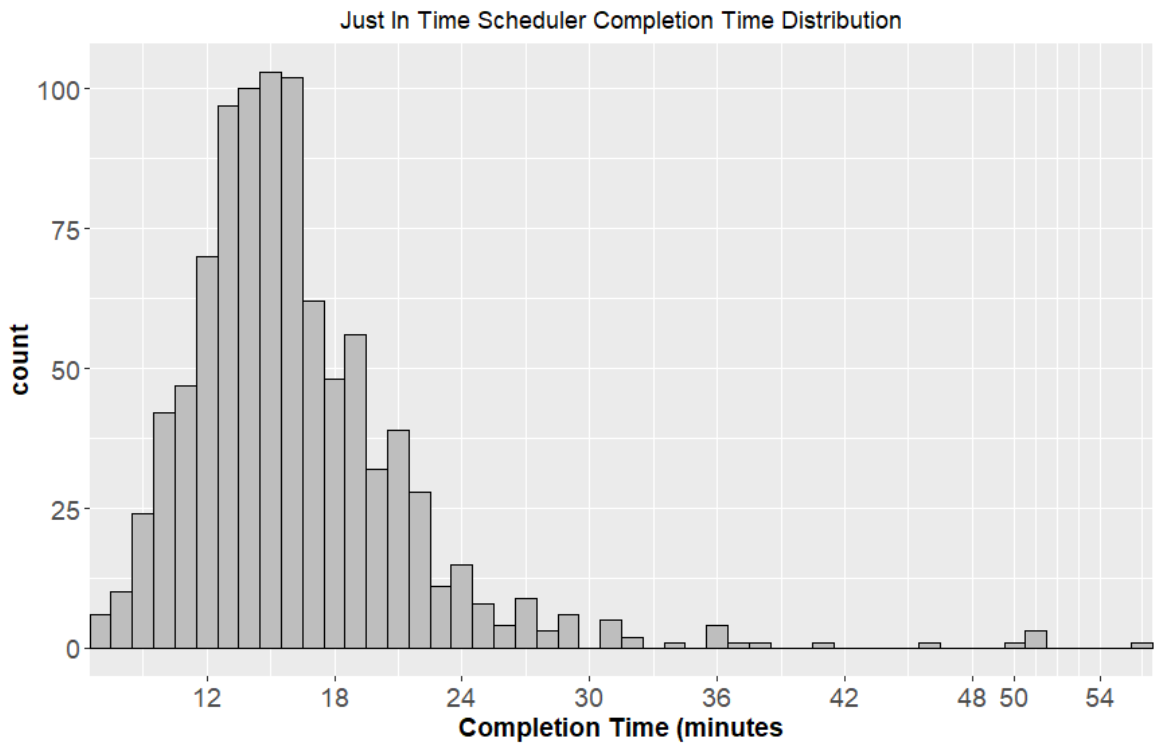

Figure 8.9: JIT Completion Time Distribution for Just Eat large scenario

Heavy Load

Interestingly, under heavy load this problem does not persist. JIT and LLV achieve the highest average profit per order, as seen in figure 8.10. We believe this is due to the queue for each scheduler being almost constantly full in this scenario. The graph of queue lengths can be found in Appendix figure 11.20 JIT will constantly be choosing the order with the highest potential that falls closest to the boundary time. When the queue is full or near capacity, any order which is added will start with 0 wait time. Assuming the order is close enough to fall into the top profit bracket, JIT will always choose this order. Hence it cherry picks the newest orders since it achieve a higher profit.

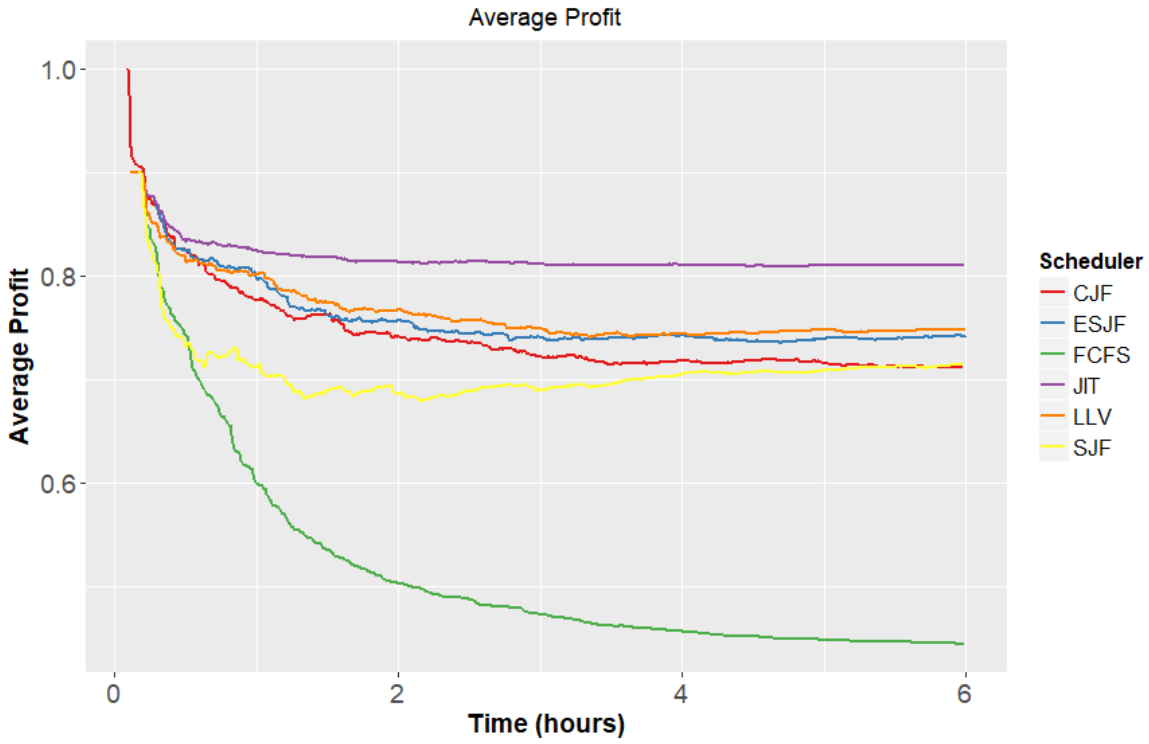

Figure 8.10: Average Profit under heavy load for Just Eat large scenario

8.4.3 Conclusions

This larger scenario has demonstrated that under moderate load CJF performs extremely well as it minimises the drone travel time. Due to the greater density of restaurants in this scenario, there is almost always a restaurant nearby which CJF chooses, provided it has an order waiting. CJF's secondary reliance on FCFS means that as long as the order queue remains steady, it can maintain a low average completion time.

The approximations in JIT and LLV have caused them to suffer under moderate load

scenarios. The extended travel time adds uncertainty to their estimates, resulting in often missed deadlines.

8.5 Evaluation Summary

These three simulations show that our schedulers perform differently in the alternative scenarios. Under heavy load, JIT and LLV are able to extract the greatest average profit from the orders they complete. This is achieved by either optimising the drones time in the case of JIT, where orders are prolonged until they are about to lose value, or by calculating which order is going to lose the most value if postponed, in the case of LLV.

However, under moderate load conditions, and in our medium and large scenarios where we have substantially more restaurants, CJF achieves the highest average profit. This comes down to two factors. Firstly, it minimises the drone travel time by only ever travelling to its closest restaurant when completing an order. Secondly, in the medium and large scenarios, there are more restaurants which results in the queue for each restaurant being shorter. When the queue for a restaurant is long, CJF suffers as it maintains an underlying FCFS ordering. This is seen in our small Just Eat scenario where CJF performs poorly. ESJF has acted as our baseline to test against, since on a global scale SJF does not take into account enough information, and FCFS suffers from impractical drone movement to adhere to its ordering.

One downside to all of these schedulers, with the exception of FCFS, is that the pay no attention to orders which are continuously postponed. In a meal delivery context, it would be unreasonable to continuously post pone a customer‘s order due to an alternative being more practical to complete. However, under a purely profit orientated approach, higher income can be achieved by postponing inconvenient orders.

Chapter 9

Project Evaluation

Whilst we were successfully in constructing a simulation for a drone delivery network, our project does have its limitations. Firstly, since we had no industry partner for this project, we had no initial data from which to base our simulations. The study of GrubHub's Meal Delivery Routing Problem(42), was our best source for realistic assumptions. Consequently, our simulation is somewhat simplified as a result of these assumptions. One improvement would be to break apart these assumptions and add a level of randomness to the simulation, in order to better reflect the real world uncertainty. However, without help from an industry partner, it is hard to decide how one would adjust these assumptions.

Creating an accurate visualisation for our simulation was by far the most time consuming part of this project. Significant time was spent ironing out bugs to create an accurate and reliable visual. Unfortunately, this meant less time was spent on simulating alternative scenarios. We were successful in assessing our schedulers on both our Domino's Pizza and Just Eat models, which was one of our main aims. However, further analysis into the Just Eat model with varying numbers of drones and restaurants may have provided more insight. Despite this, we have shown that under different order loads, and simulation sizes, there are distinct differences between scheduler performance.

Chapter 10

Conclusions

We set out to construct a simulation of a drone delivery network for multiple source nodes and assess which schedulers would be most profitable. We have successfully implemented a simulation and tested it on both a localised delivery model, our Domino's Pizza scenario, and a distributed delivery model, our Just Eat Scenario.

10.1 Global Schedulers

We have presented multiple multi-source-node oriented global schedulers, which attempt to solve the unique problems faced in meal delivery. Accounting for customer wait time as well as drone travel time, these schedulers have proven to perform better than FCFS and SJF. We have also extended LLV to a multi-source-node network, and presented JIT as a scheduler which exploits the applied time value function to maximise profit.

10.2 Global Scheduling vs Local Scheduling

Our localised delivery model covered in chapter 7 demonstrated that there is little gain in scheduling at a global level over a local level. This comes down to the nature of orders being assigned to their closest restaurant, resulting in each restaurant having a very small service area. By scheduling drones locally in a warehouse style model, where drones are tied to a location, no time is wasted moving drones between restaurants. Therefore, the potential gain from dynamically moving drones between restaurants, is outweighed by the time wasted in transit between restaurants. The only exception to this is if the load on restaurants is highly dynamic and shifts. In this scenario, local scheduling is unable to cope with the sudden load, whilst global scheduling can reassign drones as necessary.

10.3 Just Eat Model

The Just Eat model employed in this project demonstrates that under a moderate load, drone transit time between restaurants is key to decreasing lead time. This is demonstrated by our CJF model, which seeks to minimise the amount of time travelling between source nodes, as this drone time could otherwise be spent completing deliveries. However, under heavy load, or in cases where there are few restaurants and the restaurant queues are large, CJF succumbs to its underlying FCFS scheduling. In these heavy load scenarios, our JIT scheduler is able to extract the most profit from each order by prolonging orders to the point at which they are about to lose value.

Chapter 11

Future Work

Whilst we were successful in simulating a multiple source node drone delivery network, there were a few avenues we were not able to explore within the confines of this project. Here we present some of the ways our simulation and research could be extended.

11.1 Pool Size Analysis

From our research, we discovered that Just Eat uses pools of drivers to cover specific areas (32). We have been able to model one of these pools covering a fixed area of London, but it would be interesting to assess how this driver pool approach works with a drone network on a larger scale. An analysis of optimal drone pool size, or indeed fixed drone pool sizes over dynamic sizes could lead to interesting results. Additionally, a test of pooling versus global scheduling on a large scale could reveal further insight. This would reveal whether a complex global scheduler with total freedom to redistribute drones across the service area outperforms the simpler pool sub-problem approach.

11.2 Rolling Horizon Scheduling

In the study undertaken by Goergia Institue of Technology and GrubHub(42), they describe how they implemented a Rolling Horizon algorithm which would periodically recalculate and assign orders. Whilst their implementation does not directly follow a time value function like our simulation, they were still attempting to minimise delivery time whilst optimising the use of their couriers. It would be interesting to apply a similar style algorithm which determines the best order for a drone based upon its estimated completion time.

11.3 Increasing Simulation Realism

In our simulation we had to make a number of assumptions. For example, we assumed our drones would travel at constant speed. This is unlikely to be true in the real world. Therefore, in order to better model the real world, it would be beneficial to add a layer of randomness. For example, you could dynamically alter a drone‘s speed, mimicking the effects of wind in the real world, which adds a level of uncertainty and realism to the simulation. Moreover, we also assumed constant worst case handling times in line with our research. This is another area for improvement, though without sufficient data one would struggle to model it accurately.

11.4 Alternative Order Generators

We modelled our order arrival rate as a Poisson variable. This assumes that each order is independent. On the surface this may be a reasonable assumption, but this may break down at peak meal times. For example, there is likely to be a peak in orders when the work day finishes as people arrive home. Moreover, our simulation was run for 6 hours to model this evening shift. It would be interesting to apply a variable, or cyclical Order Generator which reflects the surge of orders at specific times, e.g. at lunch and dinner.

Bibliography

[1] Media Information. Technical report. URL: https://www.ocado.com/content/miscellany/pdfs/media_pack.pdf. pages 8

[2] No Fly Drones. URL: http://www.noflydrones.co.uk/. pages 15

[3] SpatialOS - Improbable. URL: https://improbable.io/spatialOS. pages 22

[4] The Air Navigation Order 2016. URL: http://www.legislation.gov.uk/uksi/2016/765/contents/made. pages 16

[5] Mike Arnot. Up in the Air: Who Determines a Plane's Altitude? URL: https://thepointsguy.com/guides/up-in-the-air-who-determines-a-planes-altitude/. pages 24

[6] Paul Balaji. Simulating Drone Delivery Networks using SpatialOS. Technical report, Imperial College London, 2018. pages 6, 19, 20, 21, 31, 33, 36, 39

[7] Paul Balaji, David Cattle, Andrea Janoscikova, Galina Peycheva, Jan Matas, and Samuel Wood. Autonomous Air Traffic Control. Technical report, Imperial College London, 2019. pages ix, 15, 17, 20, 34

[8] Lana Bandoim. Uber Plans To Launch Food-Delivery Drones, 2018. URL: https://www.forbes.com/sites/lanabandoim/2018/10/23/uber-plans-to-launch-food-delivery-drones/#6fccc614e147. pages 14

[9] David Cassel. Surprise! 7-Eleven Kicks off the Era of Commercial Drone Deliveries - The New Stack, 2016. URL: https://thenewstack.io/seven-eleven-kicks-off-era-commercial-drone-deliveries/. pages ix, 14, 16

[10] CBSNews. Amazon Drones: Amazon Unveils Futuristic Delivery Plan - CBS News - CBS News, 2013. URL: https://www.cbsnews.com/news/amazon-unveils-futuristic-plan-delivery-by-drone/. pages 4

[11] Jamie Condliffe. Delivery Drones Are Using Vans as Roving Parcel Hubs in Switzerland - MIT Technology Review, 2017. URL: https://www.technologyreview.com/the-download/608978/delivery-drones-are-using-vans-as-roving-parcel-hubs-in-switzerland/. pages 13

[12] Fintan Corrigan. 12 Top Collision Avoidance Drones And Obstacle Detection Explained — DroneZon. URL: https://www.dronezon.com/learn-about-drones-quadcopters/top-drones-with-obstacle-detection-collision-avoidance-sensors-explained/. pages 15

[13] Ben Coxworth. DHL Parcelcopter takes to Tanzanian skies, 2018. URL: https://newatlas.com/dhl-parcelcopter-africa/56663/. pages 11

[14] Daimler. Vans & Drones: Help from above, 2017. URL: https://www.daimler.com/innovation/specials/future-transportation-vans/vans-drones-2.html. pages ix, 13, 16, 25

[15] Kenny Daniel, Alex Nash, Sven Koenig, and Ariel Felner. Theta*: Any-Angle Path Planning on Grids. Technical report, 2010. URL: https://arxiv.org/ftp/arxiv/papers/1401/1401.3843.pdf. pages 17

[16] Rob Davies. Uber's self-driving car unit valued at $7.3bn as it gears up for IPO — Technology — The Guardian. URL: https://www.theguardian.com/technology/2019/apr/19/ubers-self-driving-car-unit-valued-at-73bn-as-it-gears-up-for-ipo. pages 14

[17] DHL. Drone delivery and parcelcopter technology, 2018. URL: https://discover.dhl.com/business/business-ethics/parcelcopter-drone-technology. pages ix, 10, 11, 12

[18] DHL. Rapid response from the air: medicines successfully delivered using a parcel drone in East Africa, 2018. URL: https://www.dpdhl.com/en/media-relations/press-releases/2018/rapid-response-from-the-air-medicines-successfully-delivered-using-a-parcel-dr
html. pages 11

[19] DHL Press Release. Successful Trial Integration of DHL Parcelcopter into Logistics Chain, 2016. URL: http://www.dhl.com/en/press/releases/releases_2016/all/parcel_ecommerce/successful_trial_integration_dhl_parcelcopter_logistics_chain.html. pages 11

[20] DJI. DJI Phantom 4 Specs. URL: https://www.dji.com/uk/phantom-4/info. pages 23

[21] Kevin Dorling, Jordan Heinrichs, Geoffrey G. Messier, and Sebastian Magierowski. Vehicle Routing Problems for Drone Delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017. doi:10.1109/TSMC.2016.2582745. pages ix, 20, 21

[22] EMU. Emu Analytics - Datapacks. URL: http://www.emu-analytics.com/products/datapacks. pages 46

[23] Jon Excell. Ocado Engineering reshapes retail with robotics and automation The Engineer, 2017. URL: https://www.theengineer.co.uk/ocado-engineering/. pages ix, 8

[24] Ali Fareeha. A decade in review: Ecommerce sales vs. retail sales for 2007-2017, 2018. URL: https://www.digitalcommerce360.com/article/e-commerce-sales-retail-sales-ten-year-review/. pages 3

[25] Oliver Flueler. Swiss Post is planning to use drones for the Ticino hospital group in Lugano - Swiss Post, 2017. URL: https://www.post.ch/en/about-us/company/media/press-releases/2017/swiss-post-is-planning-to-use-drones. pages 13

[26] Connie Guglielmo. Turns Out Amazon, Touting Drone Delivery, Does Sell Lots of Products That Weigh Less Than 5 Pounds, 2013. URL: https://www.forbes.com/sites/connieguglielmo/2013/12/02/turns-out-amazon-touting-drone-delivery-does-sell-lots-of-products-that-weigh-#7e6d86a5455e. pages 9, 23

[27] Alex Hern. Ocado's self-drive vehicle makes deliveries in first UK trials — Business — The Guardian, 2017. URL: https://www.theguardian.com/business/2017/jun/27/ocados-self-drive-vehicle-makes-deliveries-in-first-uk-trials. pages ix, 4, 7, 8

[28] Betsy Isaacson. Matternet Founder Paola Santana Wants To Replace The Postal System With Drones — HuffPost UK, 2013. URL: https://www.huffingtonpost.co.uk/entry/matternet-paola-santana-drones_n_2763088?ec_carp=5775853286127132729. pages 13

[29] Martin Joerss, Jrgen Schröder, Florian Neuhaus, Christoph Klink, and Florian Mann. Parcel delivery: The future of last mile, 2016. URL: https://www.mckinsey.com/~/media/mckinsey/industries/traveltransportandlogistics/ourinsights/howcustomerdemandsarereshapinglastmiledelivery/parcel_delivery_the_future_of_last_mile.ashx. pages ix, 3, 4, 5

[30] Luke Johnson. 9 things you need to know about the Amazon Prime Air drone delivery service. *Digital Spy*, 7 2017. URL: https://www.digitalspy.com/tech/a820748/amazon-prime-air-drone-delivery-service/. pages 9, 16

[31] Sunghun Jung and Kim Hyunsu. Analysis of Amazon Prime Air UAV Delivery Service, 2017. URL: https://www.researchgate.net/publication/317389269_Analysis_of_Amazon_Prime_Air_UAV_Delivery_Service. pages 9, 16, 23

[32] Just Eat. How the Integration Works. URL: https://developers.just-eat.com/docs/how-the-integration-works. pages 21, 78

[33] Allen Kim. Uber may soon deliver Big Macs to you by drone - CNN. URL: https://edition.cnn.com/2019/06/12/tech/uber-eats-elevate-food-drone-delivery-trnd/index.html. pages 14

[34] Nick Lavars. Airobotics battery-swapping platform keeps drones flying around the clock. URL: https://newatlas.com/airobotics-system-drones/43985/. pages 20, 24

[35] Mapbox. Create a map in Unity — Help — Mapbox. URL: https://docs.mapbox.com/help/tutorials/create-a-map-in-unity/. pages 46

[36] Chase C. Murray and Chu Amanda G. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research*, 2015. URL: https://ac.els-cdn.com/S0968090X15000844/1-s2.0-S0968090X15000844-main.pdf?_tid=d6eb0152-f2d0-479d-b684-08d6e638913e&acdnat=1548267461_ad6a33174d4332ce10a7b5739eef277c. pages 19

[37] NATS National Air Traffic Services. Introduction to Airspace - NATS. URL: https://www.nats.aero/ae-home/introduction-to-airspace/. pages 15

[38] NATS. Aeronautical Information Service - NATS. URL: https://www.nats.aero/do-it-online/ais/. pages 15

[39] Nathan Michael Paczan and Daniel Buchmueller. Commercial and General Aircraft Avoidance using Multi-spectral wave detection, 12 2014. URL: https://patents.google.com/patent/US20160247407. pages 15

[40] Polygon-Design. DHL Parcelcopter SkyPort a vision for improved logistics - Polygon. URL: https://polygon-design.com/en/references/projects/_projects/dhlparcelcopter-skyport.html. pages 11

[41] Reuteurs. Domino's planning drone pizza delivery service in New Zealand — Business — The Guardian, 2016. URL: https://www.theguardian.com/business/2016/aug/25/dominos-planning-drone-pizza-delivery-service-new-zealand-auckland-trial. pages 14, 50

[42] Damian Reyes, Alan L Erera, Martin W P Savelsbergh, Sagar Sahasrabudhe, and Ryan J O 'Neil. The Meal Delivery Routing Problem. *Optimization Online*, 2018. URL: http://www.optimization-online.org/DB_FILE/2018/04/6571.pdf. pages 21, 24, 75, 78

[43] Adam Satariano and Marie Mawad. Amazons Delivery Drone Project Focuses on Birds (and How to Avoid Them), 2017. URL: https://www.insurancejournal.com/news/international/2017/05/22/451737.htm. pages 9

[44] Shireen Seakhoa-King, Paul Balaji, Nicolas Trama Alvarez, and William J Knottenbelt. Revenue-Driven Scheduling in Drone Delivery Networks with Time-sensitive Service Level Agreements. Technical report, 2019. URL: https://doi.org/10.475/123_4. pages 18, 19

[45] Spencer Soper. The Little Drone Startup Beating Amazon and Google in the Home Delivery Race - Bloomberg, 2916. URL: https://www.bloomberg.com/news/articles/2016-10-05/the-little-drone-startup-beating-amazon-and-google-in-the-home-delivery-race. pages 14

[46] David Thorpe. Long-term growth in the courier industry, 2016. URL: https://www.whatinvestment.co.uk/longterm-growth-in-the-courier-industry-2536661/. pages 3

[47] Dan Wang. The Economics of Drone Delivery, 2015. URL: https://www.flexport.com/blog/drone-delivery-economics. pages 9, 13

Appendix

11.5 Domino's Pizza Scenario Graphs

11.5.1 Moderate Load

Figure 11.1: Number of Orders completed under moderate load for Domino's Pizza scenario

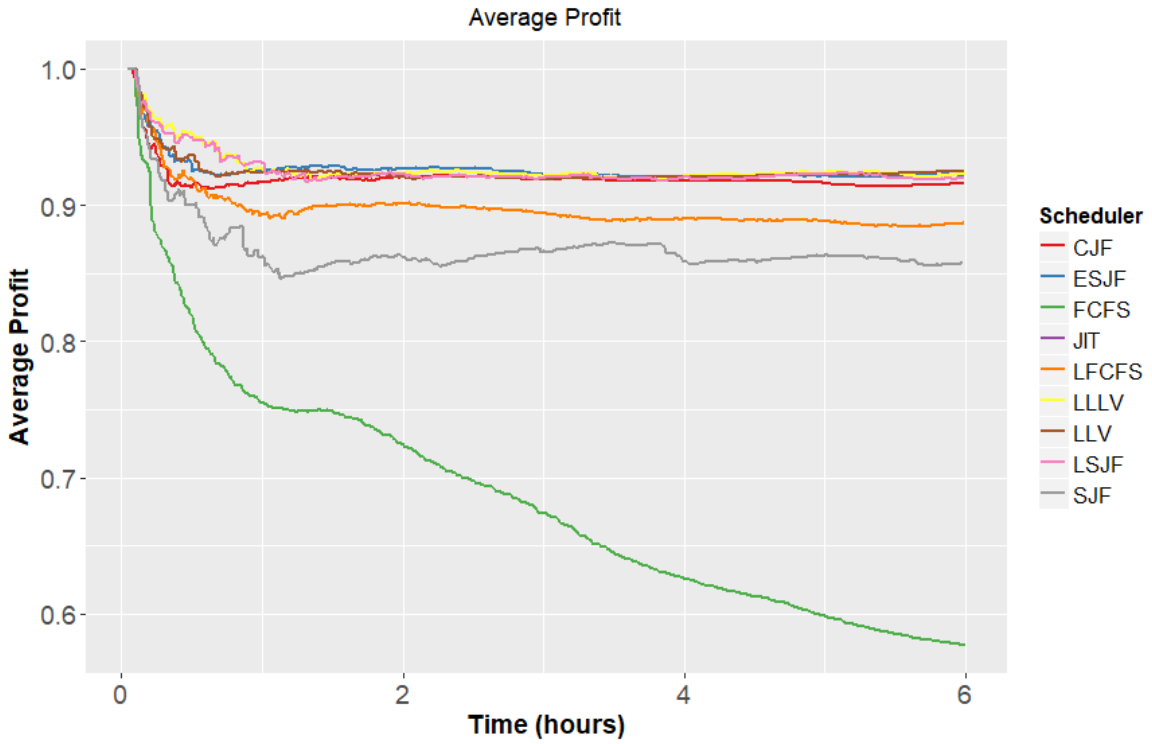

Figure 11.2: Average Profit under moderate load for Domino's Pizza scenario

11.5.2 Heavy Load

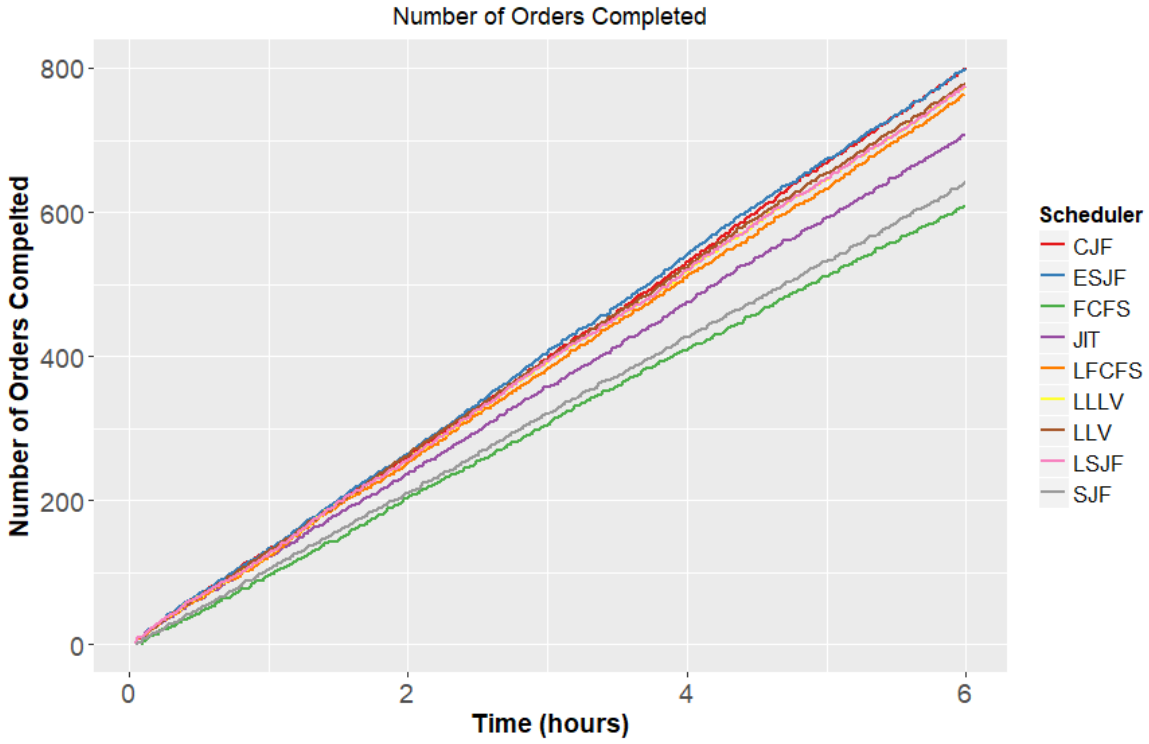

Figure 11.3: Number of Orders completed under heavy load for Domino's Pizza scenario

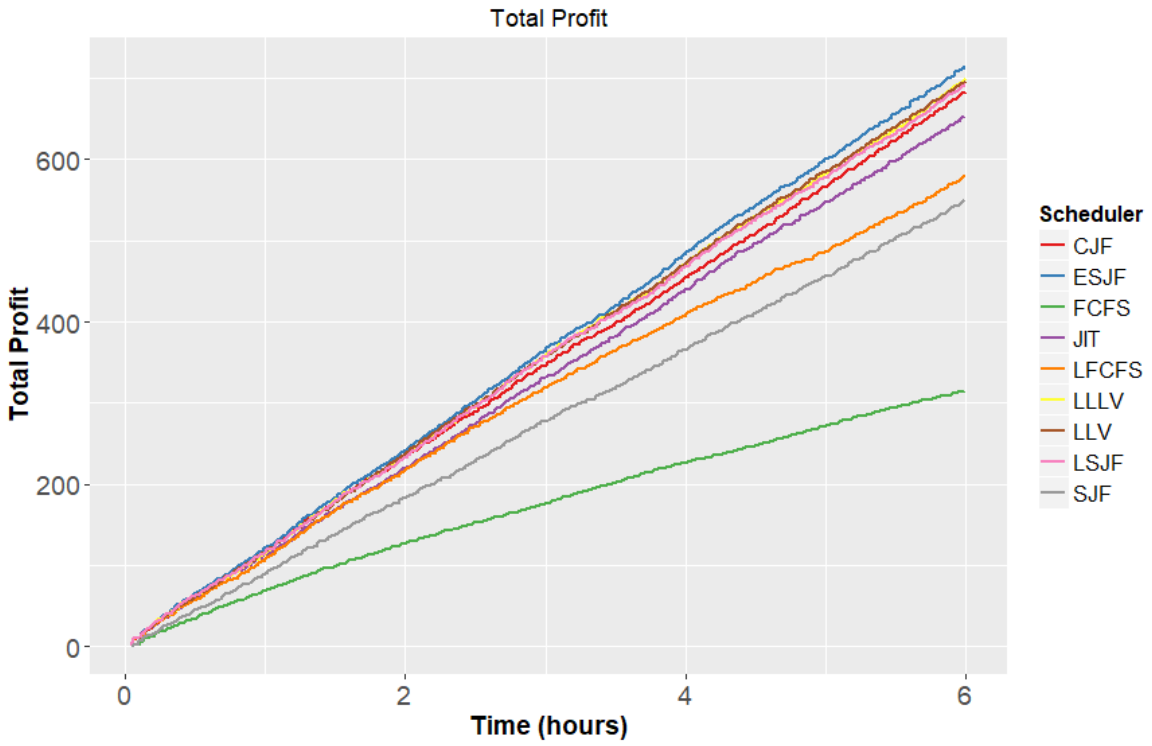

Figure 11.4: Total Profit under heavy load for Domino's Pizza scenario

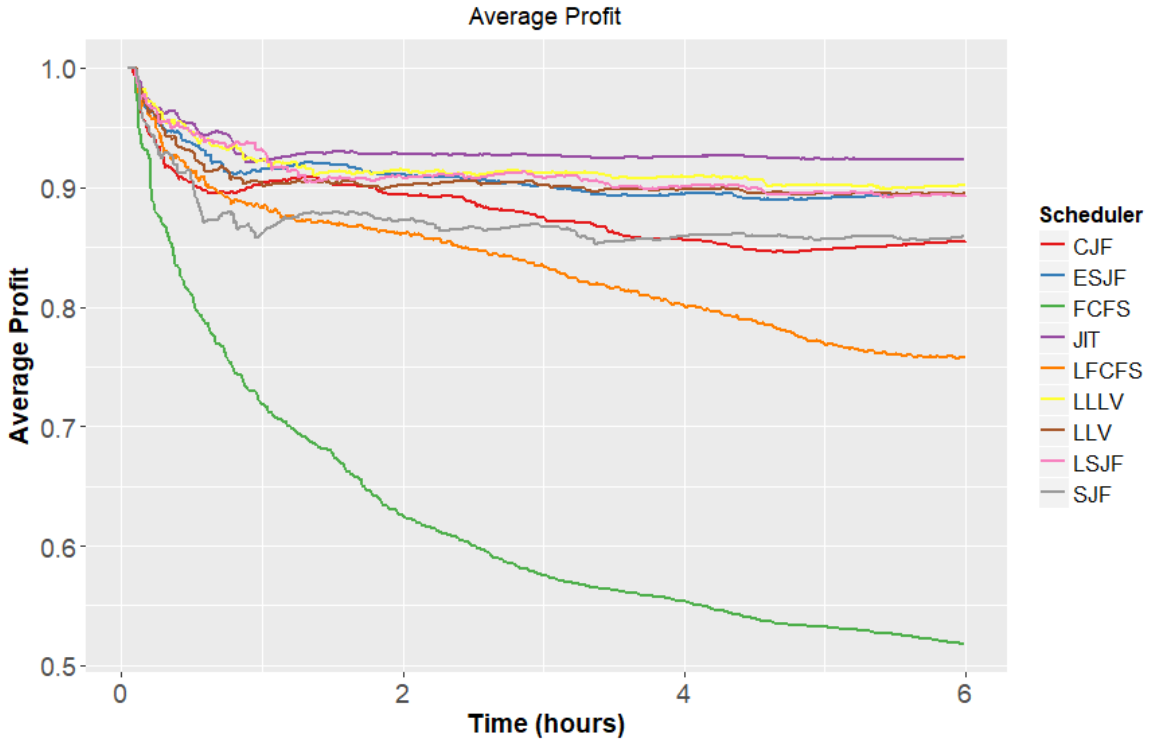

Figure 11.5: Average Profit under heavy load for Domino's Pizza scenario

11.5.3 Dynamic Load

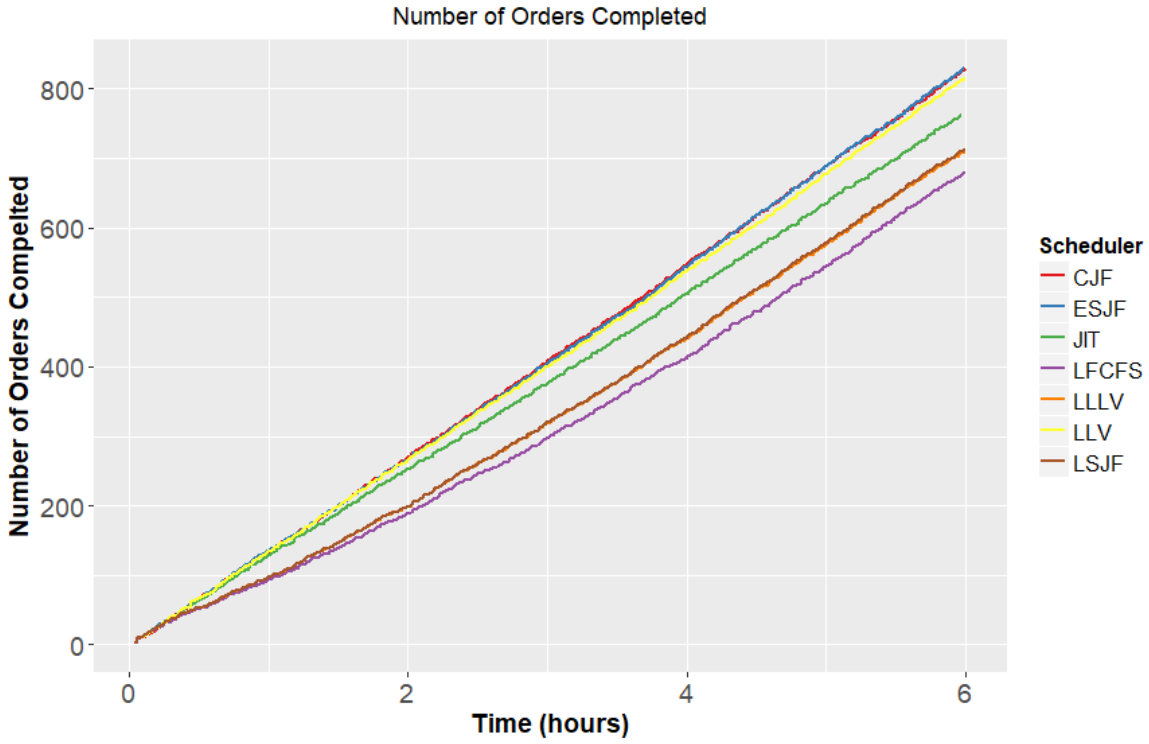

Figure 11.6: Number of Orders completed under dynamic load for Domino's Pizza scenario

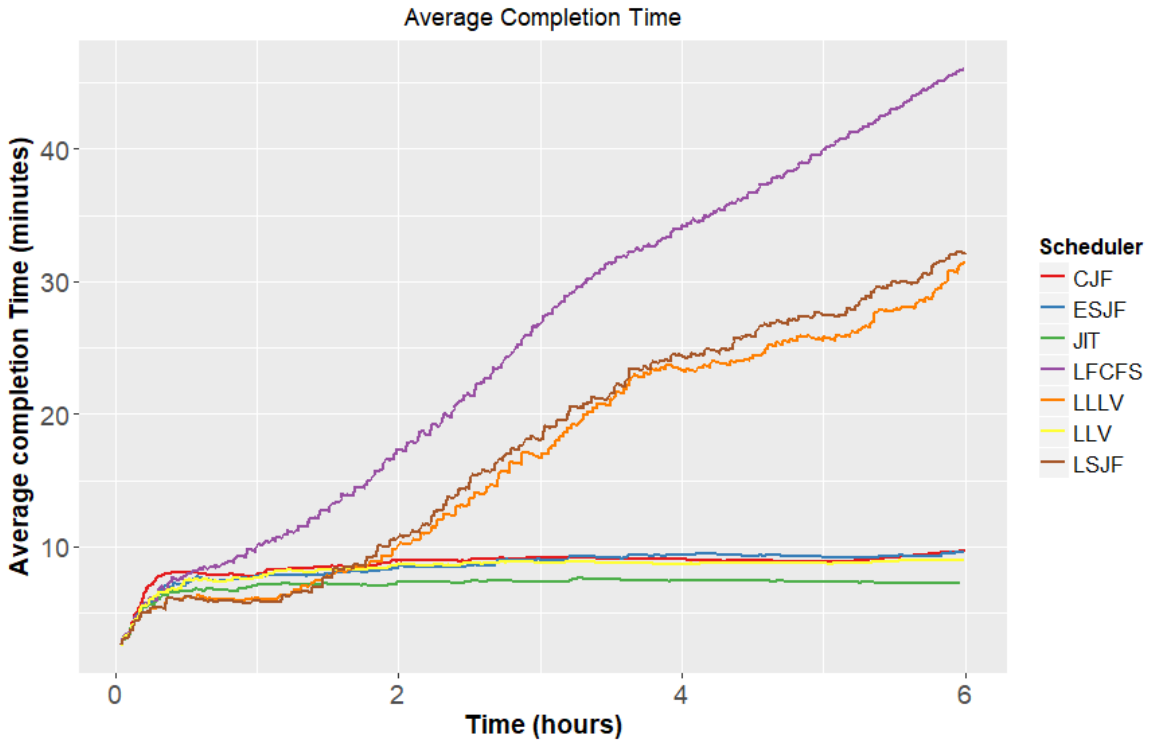

Figure 11.7: Average Order Completion Time under dynamic load for Domino's Pizza scenario

11.6 Just Eat Small Scenario Graphs

11.6.1 Moderate Load

Figure 11.8: Number of Orders completed under moderate load for Just Eat small scenario

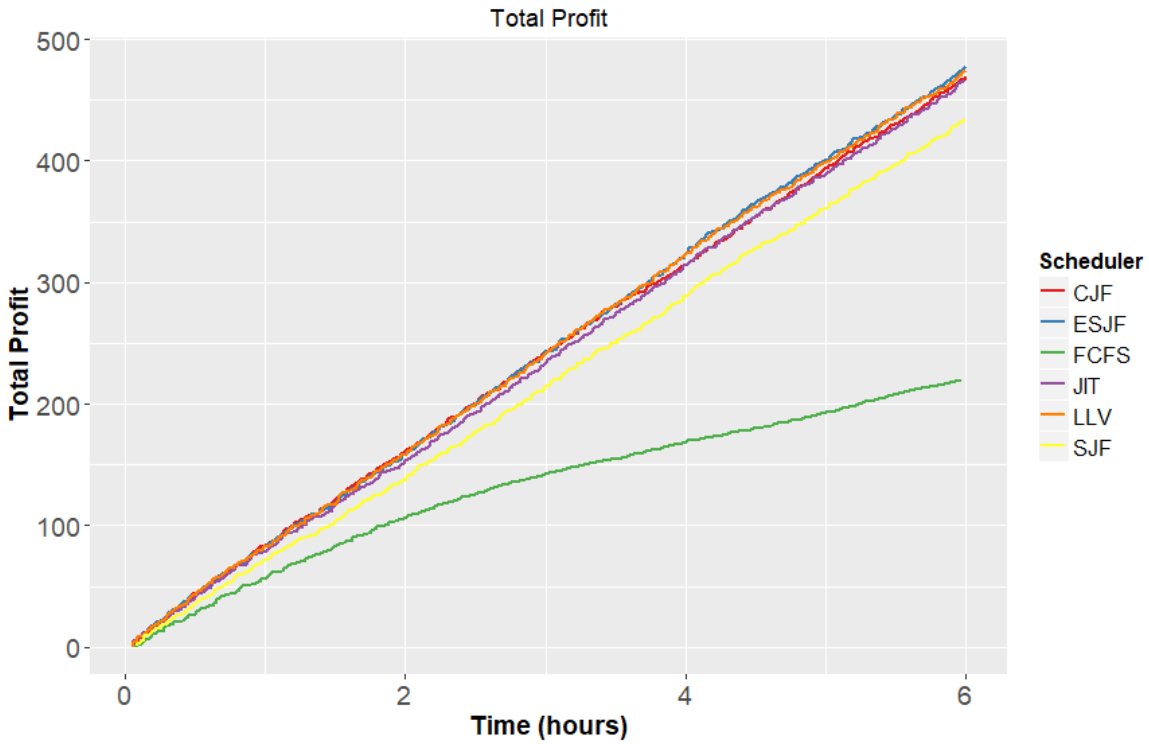

Figure 11.9: Total Profit under moderate load for Just Eat small scenario

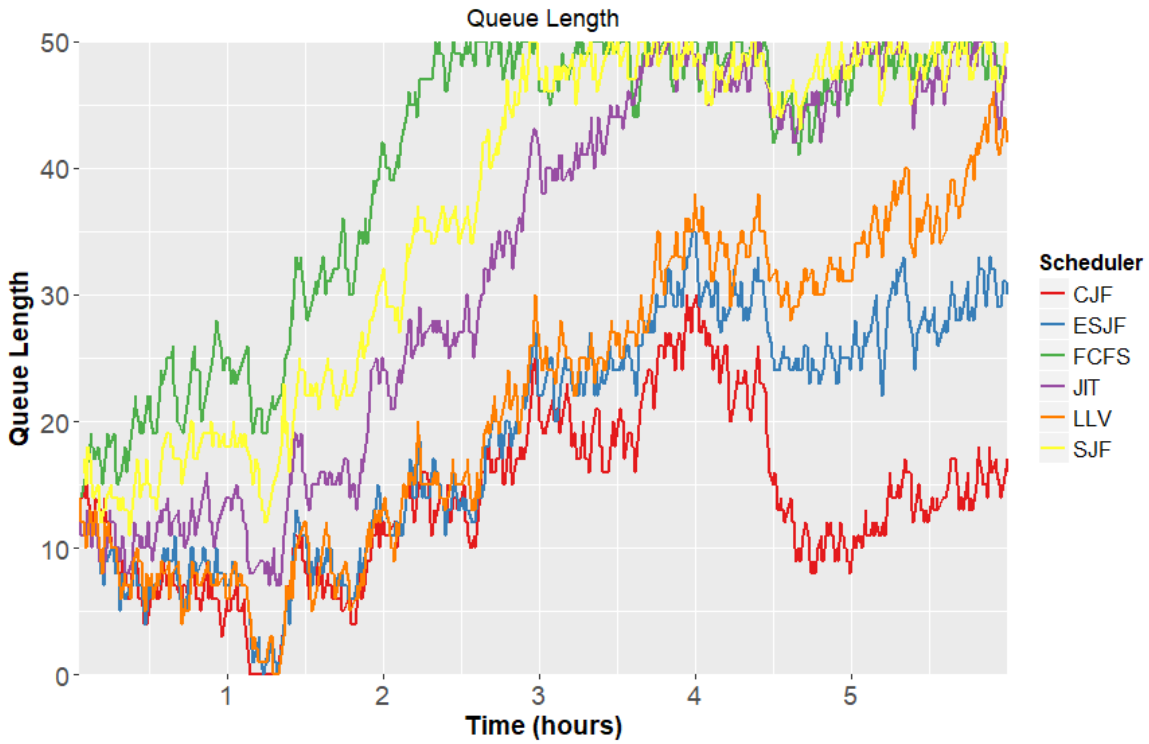

Figure 11.10: Queue Length over time under moderate load for Just Eat small scenario

11.6.2 Heavy Load

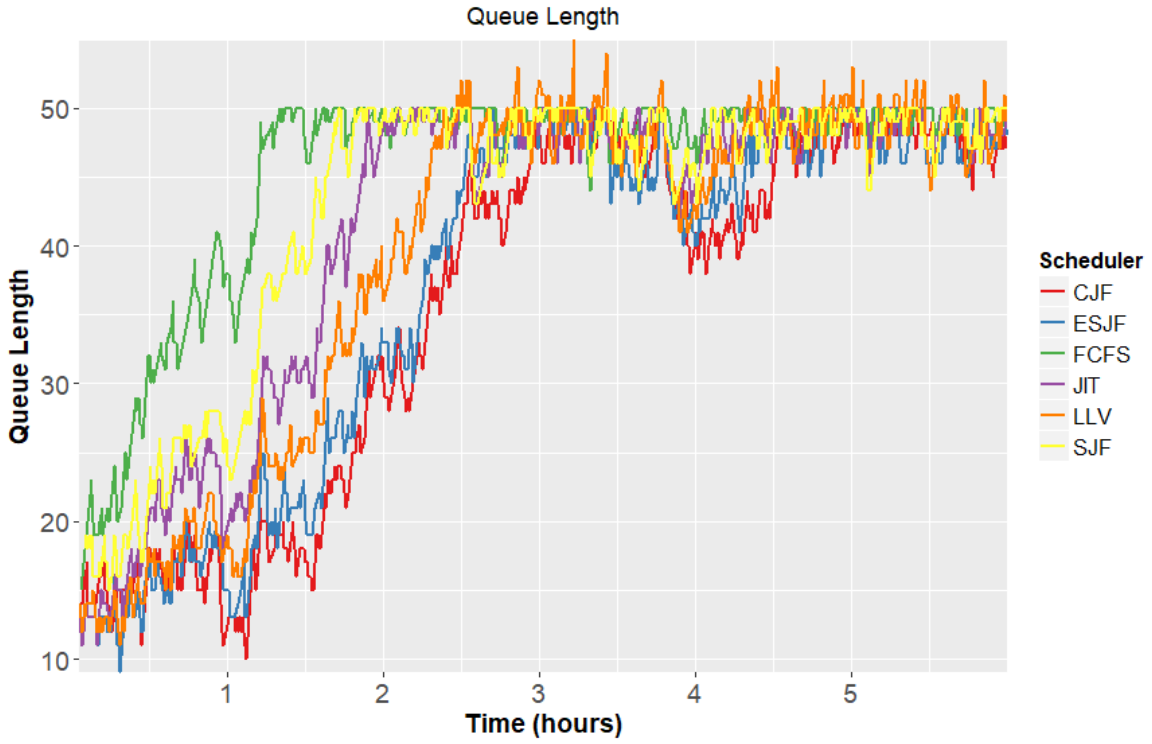

Figure 11.11: Queue Length over time under heavy load for Just Eat small scenario

11.7 Just Eat Medium Scenario Graphs

11.7.1 Moderate Load

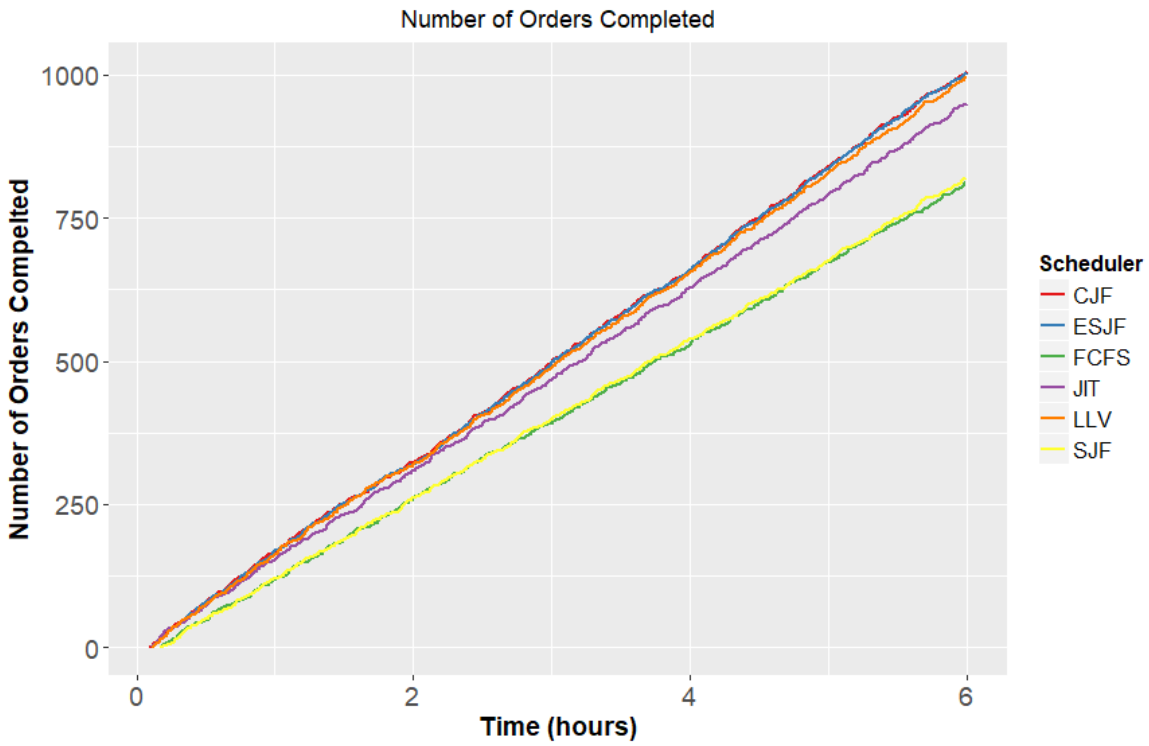

Figure 11.12: Number of Orders completed under moderate load for Just Eat medium scenario

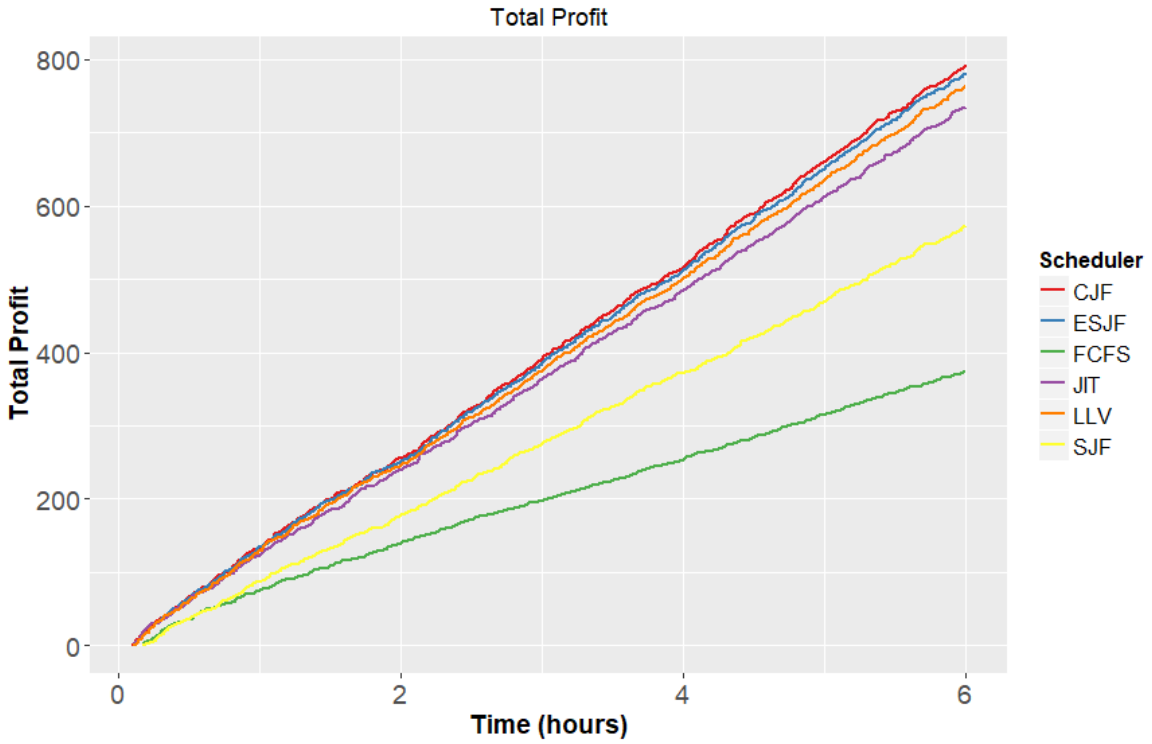

Figure 11.13: Total Profit under moderate load for Just Eat medium scenario

11.7.2 Heavy Load

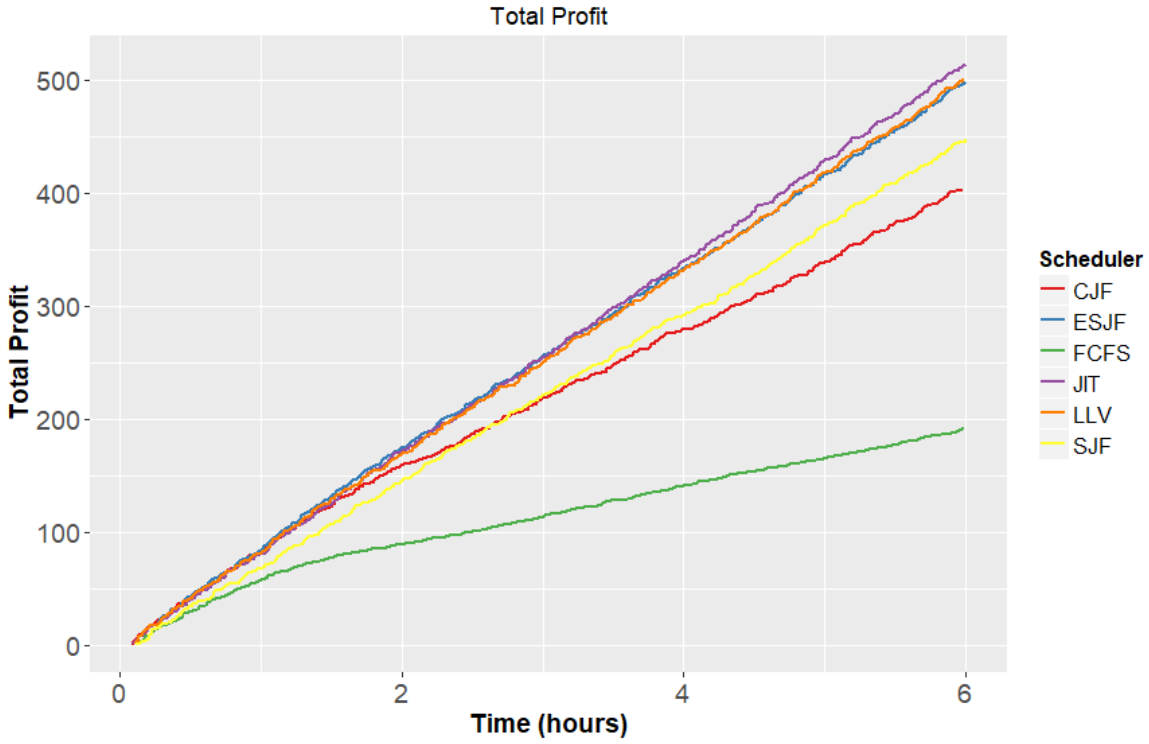

Figure 11.14: Total Profit under heavy load for Just Eat medium scenario

Figure 11.15: Queue Length under heavy load for Just Eat medium scenario

11.8 Just Eat Large Scenario Graphs

11.8.1 Moderate Load

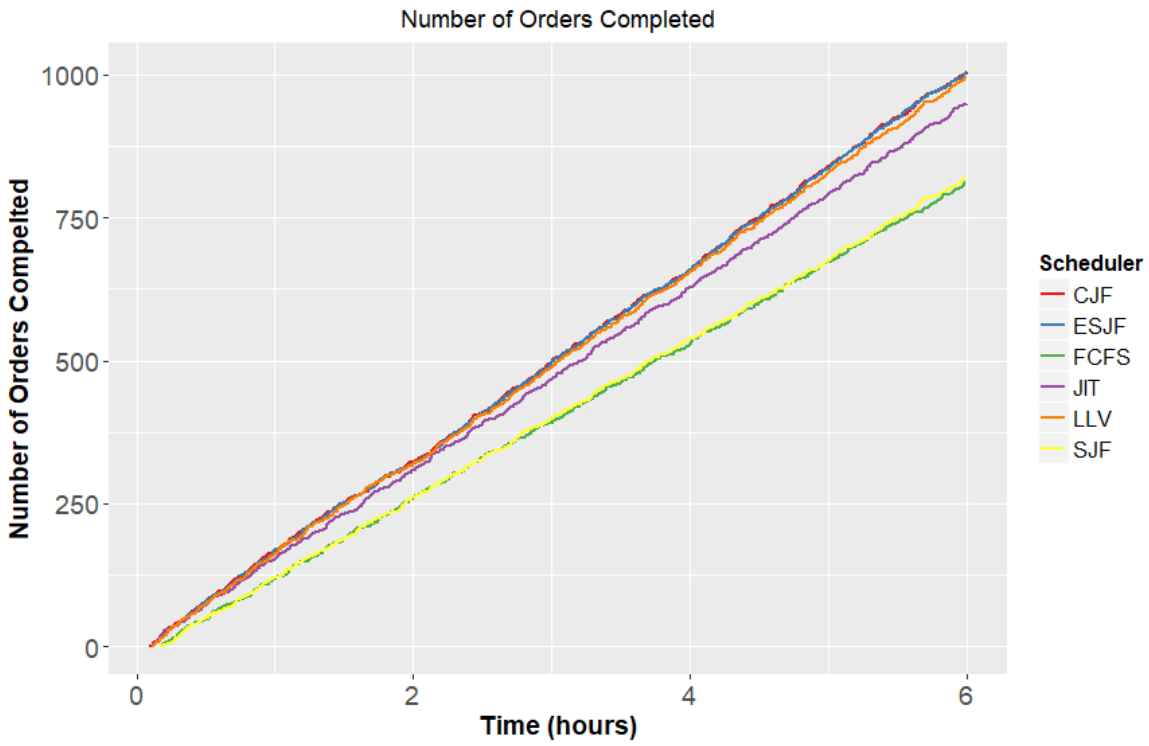

Figure 11.16: Number of Orders Completed under moderate load for Just Eat large scenario

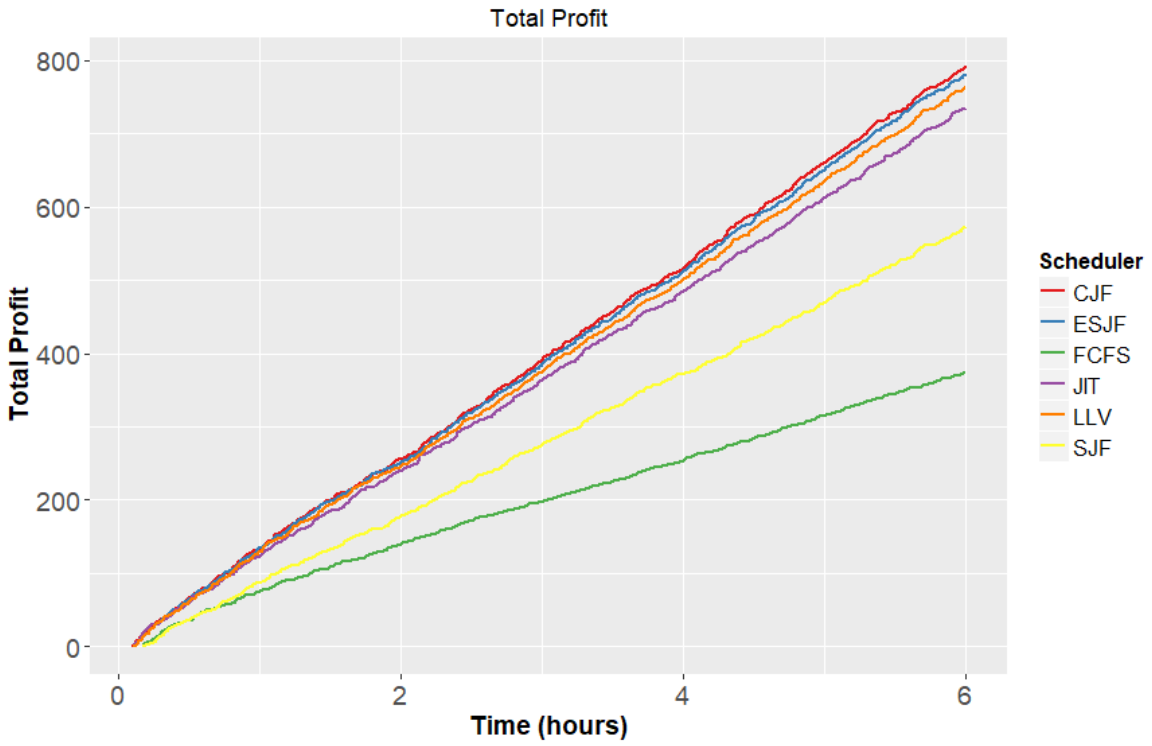

Figure 11.17: Total Profit under moderate load for Just Eat large scenario

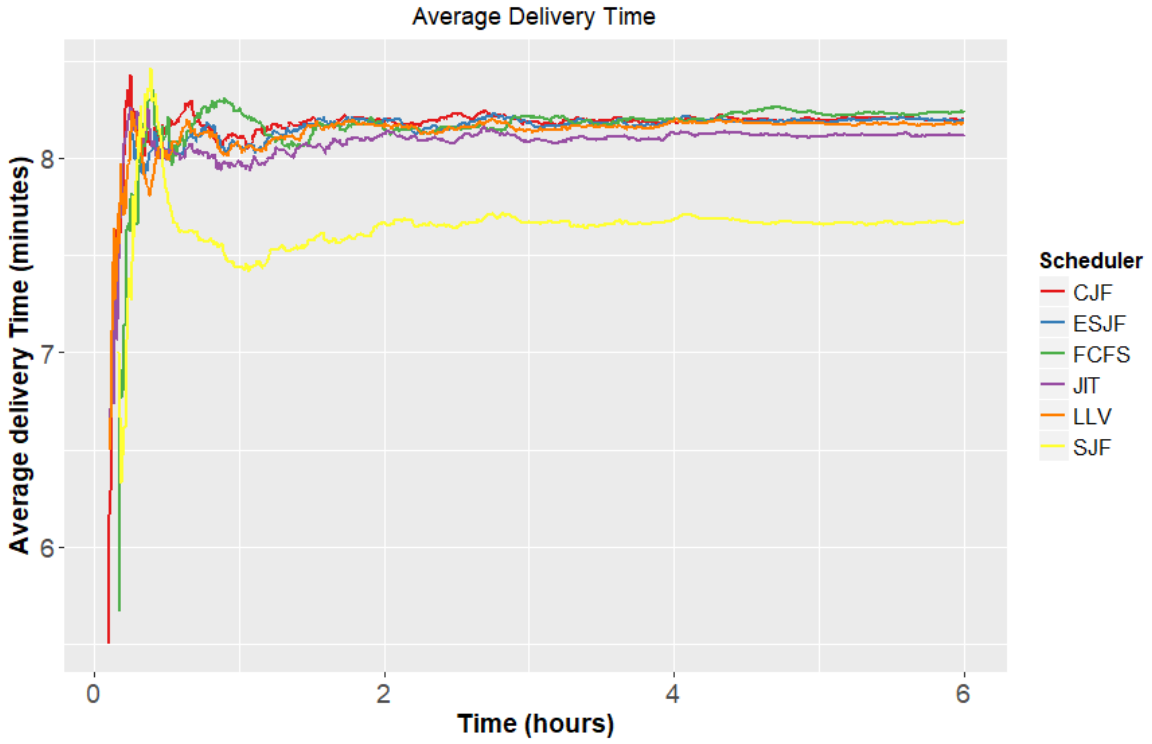

Figure 11.18: Average Delivery Time under moderate load for Just Eat large scenario

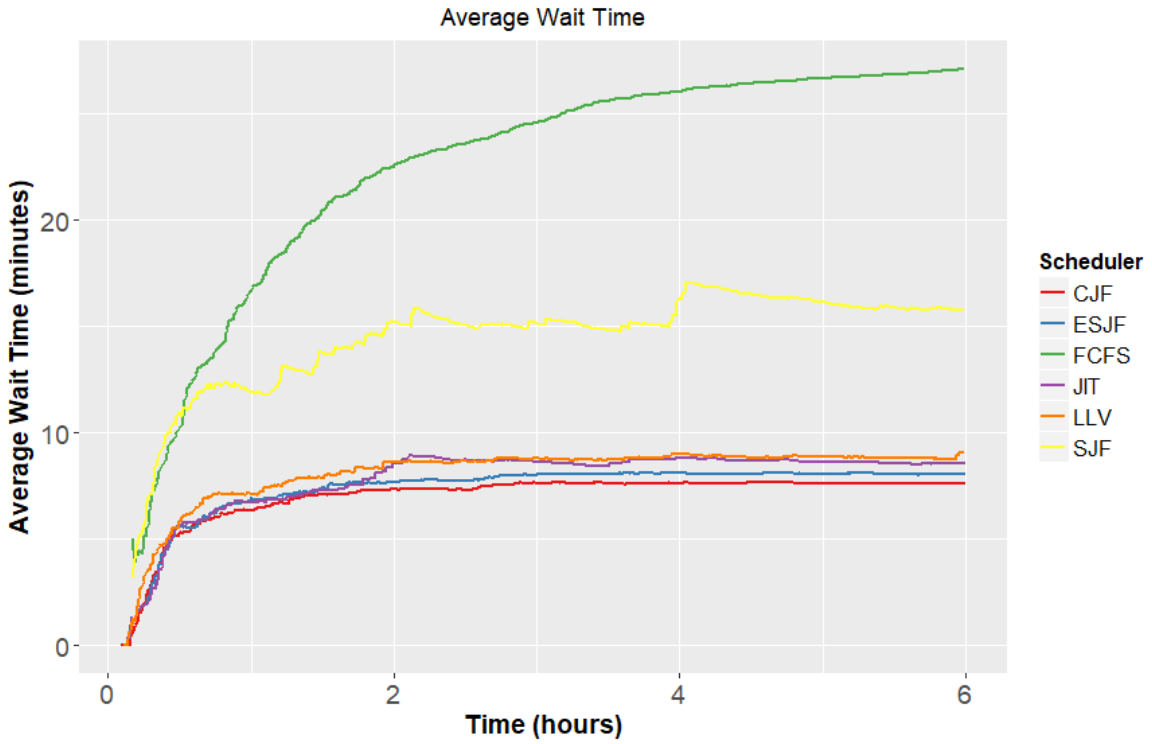

Figure 11.19: Average Wait time under moderate load for Just Eat large scenario

11.8.2 Heavy Load

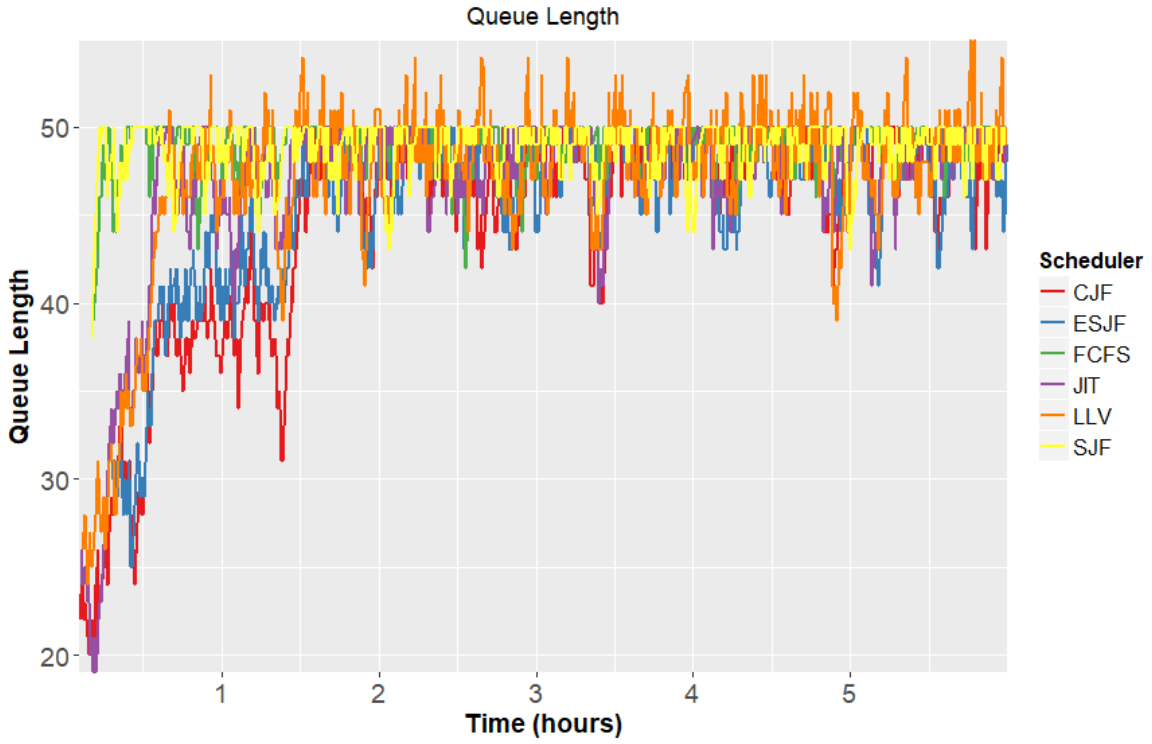

Figure 11.20: Queue Length under heavy load for Just Eat large scenario