Imperial College London

BEng Individual Project

Imperial College London

Department of Computing

Revenue-driven Scheduling in a Simulated Drone Delivery Network

Author:
Stuart McClune

Supervisor:
Prof. William Knottenbelt

Second Marker:
Prof. Kin Leung

June 17, 2019

Abstract

The use of drones in the commercial sector, specifically for last-mile deliveries, is expected to experience dramatic growth over the coming years, with successful courier and tech companies actively investing and researching in this area. The adoption of this novel technology, especially given the current scale and continued growth of the e-commerce sector, presents many challenges.

This project aims to tackle the issue of scheduling delivery jobs in a resource-limited, time-sensitive environment in order to maximise the profits of a drone courier service provider. It is pinned on the idea that the service level agreement between the customer and courier service provider would define the value of the delivery as dependent on the time taken for the package to arrive, driving the need for prompt deliveries.

In order to do this, firstly, a simulation of a drone courier network in an urban environment is built, with central controllers assigning drones to generated jobs. Then, existing scheduling algorithms are implemented to provide a benchmark for scheduler performance. Finally, thirteen novel drone delivery scheduling algorithms are developed and their effectiveness is evaluated against current solutions.

Five of the algorithms are found to be more profitable than all of the existing solutions, with a comparable or even improved quality of service when compared to the best existing scheduler. Therefore, there is no notable reduction in expected customer satisfaction if any of these scheduling policies are used, with most likely resulting in an increase.

Acknowledgements

Many people have directly or indirectly helped throughout this project and deserve acknowledgement. Therefore, I would like to thank a few key people for their support:

- My supervisor, Prof. William Knottenbelt, for proposing such an exciting idea and inspiring the direction of this work.
- My personal tutor, Anandha Gopalan, for friendly chats throughout my three years at Imperial and help when needed.
- Jack, for managing his time similarly to me, assuring me it would all work out (or we'd both fail).
- Richard and Gemma, for avoiding near death two weeks before my dissertation deadline.
- Mountain rescue, and Ben, for helping in the aforementioned avoidance of death and giving me a lift in their helicopter. True heroes.
- Martha, for being the best.
- The boys, for good laughs and excessive barbecues.
- Finally, my family, for their constant love and support, and encouraging me to always do my best.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the e-commerce industry continues to grow in scale, with global sales projected to reach 4.5 trillion USD in 2021 [2], courier service providers are looking to adopt new technologies in order to continue to provide timely deliveries. The use of drones for last-mile delivery is one option that is seeing significant interest due to their ability to move faster and more efficiently than conventional delivery vehicles. This is especially true in areas with challenging terrain or highly congested road networks, and consequently companies such as Amazon [3], DHL [4], and Wing [5] (a subsidiary company of Alphabet) are all developing and trialling their own versions.

Assuming the courier service providers are operating in an environment of limited resources, the problem of scheduling jobs in order to allocate drones to more optimally extract profit arises. In order to drive the scheduling algorithm to prioritise prompt deliveries, ensuring the customer benefits from a high quality of service, previous researchers have proposed the notion of Time Value of Service Delivery. This defines the value of a delivery as a monotonically decreasing function of time - meaning customers pay less, or even receive compensation, the longer deliveries take [6]. On the flip side, courier companies will earn more money from their customers if they keep delivery times as low as possible.

Alongside this notion, they proposed a new scheduling algorithm, Least Lost Value (LLV), which uses the Time Value of Service Delivery function to try to maximise profits for the courier company [6]. However, this is still a fresh area of research, and superior algorithms may exist which better optimise benefit to both the service provider and customer.

Testing different schedulers for a drone-based delivery network in the real world would require huge amounts of infrastructure to be set up, with cutting-edge IT systems, many starting bases, and numerous staff [7]. Even if current regulation readily allowed for real world tests, there would therefore be very large costs involved. Furthermore, in the future case of a successfully operating drone courier company, experimenting with changes of scheduling algorithm could result in a large loss of revenue if these changes turned out to be worse than their current system. This provides motivation to build a simulation of a drone courier network that can be used to benchmark and evaluate potentially better scheduling policies at minimal cost.

Therefore, the objective of this project is to develop or find advanced scheduling policies that optimise the profit earned by a drone delivery service provider, where the value gained by completing a delivery is subject to the Time Value of Service Delivery agreement between the customer and company. Since previous work has been done in this regard, where LLV was proposed as the leading algorithm [6], the aim is to devise a scheduling strategy that outperforms LLV in terms of maximising the total revenue earned by fulfilling the delivery of customers orders. Whilst the monotonically decreasing Time Value of Service Delivery should also help drive profitable schedulers to provide a high quality of service, care has to be taken to ensure this is the case, as a drone delivery company will only benefit from increased revenue if their customers are satisfied and don't turn to the services of competitors. Since real world tests will not be possible, a simulation that is sufficiently true to life must first be built before new schedulers can be implemented and tested.

1.1 Contributions

- A flexible simulation of an inner-city drone courier network, implemented using Unity, designed to easily allow different delivery schedulers to be trialled in customizable scenarios.
- Implementations of sixteen different scheduling algorithms, including some that are completely novel and some adaptations to LLV - the scheduler previously seen to be best at optimising profit in a drone delivery network [6]. Most of these algorithms are intended to maximise revenue gain, but an emphasis is also placed on customer satisfaction.
- Thorough evaluation of each scheduling policy considering many metrics relating both to profitability and service quality, with a table summarising the results to guide the selection of a scheduler based on the needs and preferences of any drone delivery service provider.

Chapter 2

Background

2.1 Delivery Network Innovations

2.1.1 Autonomous Vehicles

It is likely that automated guided vehicles (AGVs) with parcel lockers will replace current conventional forms of parcel delivery for regular, and same-day, deliveries in urban areas. They demonstrate 40 percent cost savings over current last-mile solutions. This saving is especially apparent in rural areas, but is still considerable in urban environments. This could result in a 15 to 20 percent cut in prices of deliveries to attract customers to AGV courier companies. As wages continue to rise in the future, these automated systems will create even larger savings over employee-driven vehicles [7].

In order for courier service providers to create an AGV network, they will need substantial initial investment. The costs involved are those of building an IT infrastructure capable of handling thousands of AGVs and routing them through towns and cities, acquiring the actual AGVs themselves, and paying the employees required to supervise the fleet of vehicles [7].

2.1.2 Droids

Some start-ups, such as Starship [8], are developing the use of self-driving droids in the instant delivery sector. These droids travel at pedestrian speed - around 4 miles per hour [9] - on the pavement and can serve deliveries in a 4 mile radius.

Starship was founded in 2014 and has since announced $17.2m funding in 2017, followed by $25m in 2018, clearly showing promise for the technology. Currently, over 30,000 deliveries have been completed with tests carried out in excess of 100 cities, and full service rolled out in Milton Keynes as well as the George Mason University campus [8].

To ensure safety of parcels, the droids' cargo bays are locked during transit and can only be unlocked by the intended recipients using the mobile app. The location of the orders can also be tracked for extra peace of mind.

Domino's has also shown interest in the use of droids to deliver hot pizzas to customers with the development of DOM [10]. DOM follows a similar design to Starship's robots, being a small self-driving unit with sensors to avoid obstacles on the footpath.

Even Amazon is developing this technology with their Scout robots - small six wheeled rovers that operate on sidewalks having been trained to navigate and avoid obstacles in a simulated version of the streets of Snohomish County, Washington. At current, at least six of these Scout robots are actively making deliveries in this suburban environment with Amazon employees, or "ambassadors", walking alongside to ensure they are functioning correctly and to help with public reaction to, and adoption of, this method of delivery. Eventually, the plan is to pair these droids up with Amazon delivery drivers to increase employees' efficiency at making deliveries. At the moment, Scouts are being built to operate in a suburban setting, suggesting they aren't intended for instant delivery, but rather same day delivery, since fulfilment centres are not necessarily as close to customers as in a city. There is, however, a possibility that these robots will eventually work in both city and even rural environments [11].

However, despite this interest from big businesses, it is predicted that bikes will be dominant in the field of instant deliveries in the future unless unmanned droids become much faster - achieving speeds of up to 30 kilometres per hour - or the costs of operating a droid courier service can be reduced to two to three times cheaper than what they were in 2016 [7]. The option of travelling at greater speeds would require the droids to drive on roads, since it would no longer be safe to use the pavement, and there does not seem to be any research into this area.

Therefore, it is not too likely that droids will be the instant delivery technology of choice in the near future, but they may still have their place in more niche markets, such as Aethon's TUG[12] moving materials and clinical supplies around hospitals to help ease up nurses' schedules. Only time will tell as to whether droids are adopted for suburban same day delivery, as is the current plan for Amazon's Scout rovers.

2.1.3 Cyclists

The costs involved in providing an instant delivery service are relatively high, and customers are not generally willing to pay large sums of money for better service or increasingly fast delivery times in this field [7]. Consequently, it is important to keep costs as low as possible whilst still managing to transport items quickly enough to the recipient of last-mile deliveries.

As a result of this, bicycles have emerged as the vehicle of choice for food delivery start-ups, such as Deliveroo [13]. This trend is expected to continue to grow and dominate the entire urban instant delivery industry, since bikes can travel just as fast over short distances in many cities as cars, but do not incur nearly as many costs [7] such as fuel, maintenance, and emission fees. Additionally, very little training is needed for bike couriers, making this method of delivery even more attractive to service providers.

2.1.4 Drones

Same day delivery in rural areas is currently very expensive due to the large fuel costs. The use of drones is much more efficient, and also much better for the environment. The ability to power drones on clean electricity has suggested carbon footprints could be as much as 22 times lower than traditional delivery [14]. Drones are also not limited by road networks, and could therefore provide faster delivery.

As a result of these benefits, it is likely that drones will be used for same-day delivery in rural areas [7]. Unfortunately, due to the size of drones and the difficulty of finding appropriate landing areas in urban zones, it is possible that they will not be as widely utilised in cities [7]. However, companies such as DHL have expressed that their ultimate ambition with drone technology is to use it in urban environments for the rapid delivery of small items [4]. This is likely because delivery by drone can exceed the speed of AGVs, so they could charge a greater premium for the service. One potential solution to the landing problem is the mechanism of lowering packages to the ground without ever landing, as is exhibited in the drone models made by Wing [5].

2.2 Drone Delivery Schemes

Many large tech and courier companies are researching and developing their own drone delivery services. Notably, Amazon [3], Wing [5] - a subsidiary of Alphabet, and DHL [4], have all devised numerous prototypes for how the drones could look and what role they would play in their existing, or new, courier networks.

2.2.1 Amazon Prime Air

In late 2013, Amazon announced their research and development into the field of drone-based deliveries with the concept of Amazon Prime Air [15]. They claim that using autonomous drones as delivery vehicles will allow for delivery within half an hour of purchase for packages up to a maximum weight of 5 pounds (roughly 2.27kg) [3].

On the 7th December, 2016, the first fully autonomous Prime Air delivery was carried out as part of a private trial in Cambridgeshire, England. A drone was dispatched from a special Prime Air fulfilment centre close to a customer's home and successfully completed a full delivery of a package, arriving only 13 minutes after the order was placed. The trial started with only 2 customers but Amazon planned to expand to dozens, and eventually hundreds, of customers served by their UK facility [3].

At Amazon's first re:MARS conference in Las Vegas which took place early June 2019, it was announced that Prime Air drones are expected to be making deliveries to US customers within months. However, Amazon has not been granted full authorisation by the US Federal Aviation Administration to launch an aerial delivery service. Instead, it has been granted a special airworthiness certificate, valid for one year, that allows the company to run trials for research and development in specific flight areas [16].

At this conference, the latest design of the Prime Air drones was unveiled. Resembling a hexagon, the UAVs can fly both vertically as a helicopter - during takeoff and landing, and horizontally as a plane - for fast and efficient long distance travel - using the propeller guards as wings. The range of this latest iteration is around 15 miles, and the capacity is still at the initially stated 5 pounds, aiming for delivery times under half an hour [17].

2.2.2 Wing

In 2012, the Wing team from X Development LCC., a research subsidiary of Alphabet Inc., started to explore the use of drones for the safe and efficient delivery of medicine and food. Originally, they focused on the problem of more quickly providing defibrillators to those experiencing a heart attack, hoping that the decreased delay on these devices arriving would save lives. However, they quickly learnt that integrating into emergency medical services is a huge task, so instead turned their focus to developing safe, reliable, and fast drones for the delivery of small packages [18].

Over the next few years, Wing iterated quickly on different drone prototypes in order to learn as much as possible about how to make safe, precise, unmanned aerial vehicles, with the aim to conduct real-world delivery trials [18]. Their first deliveries were made in rural Queensland, Australia in 2014, providing farmers with food, radios, batteries, and bottled water. This was followed in September 2016 with a trial at Virginia Tech delivering burritos to students in the United States. Having now graduated from the X moonshot factory to become its own Alphabet company, Wing has conducted tests in Bonython with approval from CASA, and plans were put in place to set up an ongoing delivery facility in Mitchell, Gungahlin, to make drone-based delivery in Australia a reality [19][18]. Recently, in April 2019, Australia's Civil Aviation Safety Authority granted Wing approval to launch a commercial home delivery service, with restrictions to the times of flights, where the drones can fly, and the model of drone used in the commercial service [20]. These limitations are both to reduce the problem of noise, and also to reduce any risk to the public or other aircraft. Now, Wing's drones are carrying out aerial deliveries for at least eight businesses (mostly serving food) around Canberra [21], and Wing are looking to partner with more business owners.

Wing has also been approved for deliveries in Finland, operating in parts of Helsinki. Two different featured businesses are listed on the website [22] at present: a food market, and a cafe.

Groundbreakingly, Wing was approved as an airline by the US Federal Aviation Authority - making them the first drone courier company to ever achieve this status. Airline status allows the drones to fly for longer distances and to deliver cargo, subject to the same regulations as chartered flights [23]. To date, Wing has not yet launched its commercial delivery service in Virginia, but it is due to do so later this year by commencing a delivery trial in Blacksburg and Christiansburg, partnering with local businesses [24]. The FAA granting this approval to Wing is seen as a big step forward in the field of drone delivery, paving the way for Wing's competitors to also come to an agreement with the government authority. It is expected that Wing's success will make it much easier for others to be granted the same airline status.

2.2.3 DHL Parcelcopter

DHL - the global leader in the logistics industry - is fairly critical of the hype that surrounds drone technology. However, they believe that drone delivery services in the logistics industry is a genuine business case where the adoption of drones is beneficial, claiming "the flying postman is here" [4].

The first generation of the DHL Parcelcopter [4] was created in 2013, with manual control, and was only used on a 1km river crossing flight. Since then it has been overhauled, with the third generation, which resembles an autonomous tilt-wing aircraft rather than a quadrocopter, released in 2016. It boasts the ability to carry payload up to 2 kilograms at roughly a 70km/h airspeed, and is designed to fly in the challenging conditions of hard-to-reach mountainous regions.

During a three-month trial, the Parcelcopter 3.0 was successfully integrated into DHL's existing parcel delivery network, flying parcels between the Bavarian ski resort of Reit im Winkl and the Winklmoosalm mountain plateau. In order to make this possible, DHL also developed a fully automated package loading and unloading system based on their existing Packstation technology, dubbed the 'Parcelcopter Skyport'. During the trial, private customers would load their shipments into the Skyport, which would trigger automatic delivery by Parcelcopter to Winklmoosalm. The tests were considered a success, with 130 trips flown of around 8km each, at altitudes reaching 1200m above sea level. Each delivery took approximately 8 minutes from the point of take-off, replacing a half hour trip by car in winter [4].

Despite only conducting trials in remote areas as of present, DHL's ultimate aim is to provide same-day drone delivery in urban areas. They believe the Parcelcopter 3.0 has overcome all the technical challenges necessary to carry out tests in an urban environment in the future.

2.3 Air Traffic

With regulation being a majorly restrictive factor for drone-based delivery, tech companies are keen to work with regulators and industry to find solutions to keep drone use safe and secure [3]. This has led to the idea of an air traffic management system that increases visibility of where drones are flying, and who is flying them.

In the interest of encouraging adoption of drone delivery technology, Amazon has proposed some ideas on how airspace can best by designed to allow for sustainable drone usage [25][26]. Wing are also working on their UTM platform - a system to manage complex flight paths of large numbers of drones - to allow drone operators to share the sky and still carry out different types of operations simultaneously. Their drone delivery trials in Australia rely on the UTM platform to provide routes that avoid buildings, terrain, obstacles, restricted areas, and other aircraft [27].

2.4 Time Value

The concept of time-value is the idea that the value of a resource varies with time. For example, the time-value of money states that an investor should attribute greater value to receiving an amount of money as soon as possible, rather than the same amount later, because they can then invest this money and use it to make earnings [28].

The same concept can apply to data; processing up-to-date data can yield better value for a business than processing similar, but older, data. An illustration of this is an e-commerce company interested in what customers are buying. It is more useful for them to understand the current market than the market a couple months ago, because it allows them to target their sales and advertisements towards the current more popular items, or items that are similar to these trends.

It has also been proposed that time-value applies to deliveries, with researchers putting forward the notion of Time Value of Service Delivery [6]. This defines the value of a delivery as a monotonically decreasing function of time capturing the service level agreement between the customer and the courier service provider. In other words, the longer a delivery takes, the less a customer is willing to pay.

This idea of time sensitive value has inspired research into scheduling algorithms in the field of big data that prioritise calculations and operations that result in a more maximal value for a client over those that result in less value overall.

2.5 Scheduling

Scheduling algorithms are generally associated with the topic of ordering the execution of tasks on a CPU. Understanding some common CPU scheduling policies may help with scheduling drone deliveries to maximise profit, though there are a couple key differences. Namely, a drone network will have many more drones than a CPU has cores, so revenue-based drone courier network scheduling has a higher degree of parallelism than typical CPU task scheduling. Additionally, CPUs are capable of partially completing a task and coming back to it after processing a new one, whereas, due to low maximum payloads, drones have to complete whole tasks before starting on a new one.

Regardless, it is a good idea to consider existing algorithms, especially since some of these have now seen use in a similar simulated drone environment [6].

2.5.1 First Come First Served

This very popular algorithm is possibly the most simple scheduling algorithm conceivable. It implements a First In First Out (FIFO) queue that schedules tasks such that the first to arrive is the first to be served. For this reason, it is also knows as FIFO scheduling. It does not consider any attribute of the tasks other than the time they were submitted [29].

2.5.2 Shortest Job First

Shortest Job First (SJF) [29] scheduling orders tasks in a way that means the next computed task is always the one that will take least time to complete out of the pool of available tasks. This form of scheduling ensures small tasks get completed quickly, and there is a large throughput of total tasks processed, but it can starve longer tasks of CPU time if there are always shorter jobs available.

In the context of a drone courier network, this could place an unfair bias on those who live closer to the drone bases, since deliveries to these customers will take less time. This could, however, be a good strategy for increasing profit and, with enough drone bases, it may be possible to avoid long wait times for those who live further away.

2.5.3 Priority Scheduling

A priority scheduler[29] is one that assigns a priority value to each task (or this may be defined by whatever submitted the task) and orders the pool of available jobs according to their priorities. Therefore, the task that has the highest priority will be the next to get CPU time. SJF is an example of this where the priority is inversely proportional to the time the task will take to execute.

Again, if there is a constant stream of high priority tasks, those with a lower priority may never receive any CPU time at all.

2.5.4 Multi-level Feedback Queue

Multi-level Feedback Queue (MLFQ) [29] scheduling involves multiple queues of tasks which need to be completed. Each queue corresponds to a different priority. An advantage of having multiple queues is that they can all run different scheduling algorithms internally to better match their own requirements. Tasks from the highest priority non-empty queue are scheduled before those in lower priority queues. Clearly, this can lead to starvation the same way priority scheduling can.

To combat this potential for low priority tasks to never be scheduled, MLFQ implementations use a feedback system which allows tasks to move between queues. An example of a way to manage this is to make all tasks slowly move up in priority as they wait for CPU time, and to take priority from those that have recently been given CPU time.

2.5.5 Least Lost Value

Least Lost Value (LLV) [6] is a scheduling algorithm that has been adapted for use in the context of drone deliveries, taking inspiration from the field of big data. It takes as input the Time Value of Service Delivery function for each job, and the anticipated drone flight-time distribution to serve each job. Using these, it calculates the Net Lost Value (NLV) for each job by subtracting the Potential Gained Value (PGV) from the Potential Lost Value (PLV).

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

PGV is defined as the difference between the expected value of doing the job now, and the expected value of doing the job after doing another job first.

$$PGV(job_i) = EV(job_i, time_{cur}) - EV(job_i, time_{cur} + delay_i)$$

Here, the expected value (EV) for job_i for cost function V_i , current time $time_{cur}$, and a probability density function $f_i(t)$ of the job's duration distribution is defined:

$$EV(job_i, time_{cur}) = \int_{time_{cur}}^{\infty} V_i(t).f_i(t - time_{cur})dt$$

Also, $delay_i$, the expected delay caused by choosing another job first, is defined as the mean of the expected duration of all other jobs:

$$delay_i = (\sum_{k \neq i, k=1}^{n} dur_k)/(n-1)$$

where n is the number of jobs in the job set J, and the duration dur_k of a job is:

$$dur_k = \int_{time_{cur}}^{\infty} (t - time_{cur}).f_k(t - time_{cur})dt$$

PLV is defined as the sum of the value lost from all of the other jobs that are not selected. For each of the other jobs, this value lost is the difference between the expected value of doing it now, and the expected value of doing it after the amount of time that the selected job is expected to take. For job_i:

$$PLV(job_i) = \sum_{job_k \in J, k \neq i, k=1}^{n} (EV(job_k, time_{cur}) - EV(job_k, time_{cur} + dur_i))$$

where dur_i is defined similarly to dur_k.

Deliveries with the least NLV are scheduled with a higher priority to those with a greater NLV, aiming to reduce overall net loss of value to maximise revenue for the courier service provider.

Using this scheduling algorithm was shown to extract more profit for the drone delivery company than FCFS or SJF scheduling in times of high load where the number of deliveries is great and therefore many scheduling decisions are made [6].

2.6 Unity

Unity [30] is a real-time engine that can be used for free by anyone whose revenue or funding is below \$100,000. It is exceedingly popular in the world of game development, and has been used to create half of the world's games [31]. As a result of this popularity, it is well documented and there are countless community resources and tutorials, making it a relatively easy tool to pick up and use. The engine provides support for both C# and Javascript, with the option to use both interchangeably within one project if desired.

The Unity engine allows for the creation of both 2D and 3D physics based objects that can be programmed to move around the virtual world and interact with each other, and features a powerful editor that automatically shows a graphical representation of this world and has many convenient options for configuring the behaviour of the scripts. This means it is not only useful for creating games, but also for running simulations where consistent behaviour of many moving bodies is desirable and it is valuable to be able to observe what is going on in real-time. The ability to access and modify public variables within scripts from the Unity editor GUI is very useful for changing the settings of a simulation to explore how the outcome responds to certain changes.

Chapter 3

Simulation

The first step in this project was to build a simulation of an inner-city drone delivery network, to later be used to test the behaviour of various delivery scheduling algorithms where the profit earned by fulfilling the delivery of packages is subject to a Time Value of Service Delivery function. This was created using Unity and C#.

Below, the key components of the simulation implementation are explained in turn.

3.1 Controller

The job of the controller is to handle the core logic of the simulation. This involves initialisation of the city and drones, as well as coordinating the scheduling of delivery jobs and providing drones with their appropriate waypoints for each of their flight paths.

The controller is also considered to be the physical building from which the drones originate, and must return to between deliveries. In this respect, it can be thought of as the fulfilment centre of the delivery company, and it is placed in the geographical centre of the simulation space.

3.1.1 City Generation

At the beginning of the simulation, the controller procedurally generates a grid style city environment by spawning building and road objects. The user can specify the size of the city by choosing the number of "cells", and also how many buildings are in each of these cells. A random number generator is used to determine the height of each building, with a range of possible values defined by the user.

3.1.2 No Fly Zones

No fly zones (NFZs) are parts of airspace where drones are not allowed to fly. For example, drones may not be allowed to fly in the vicinity of an airport or prisons. To simulate this, a number of rectangular areas are generated which flight paths cannot cross. This is achieved by picking random bounded dimensions, and then picking a random location from within the city which will result in the entire NFZ being within the bounds of the simulation space. If the resulting area intersects with the controller building no drones would be able to fly, so it is simply discarded and we try again if this is the case. Figure 3.1 shows a top-down view of a generated city, with the NFZs represented as translucent red cuboids.

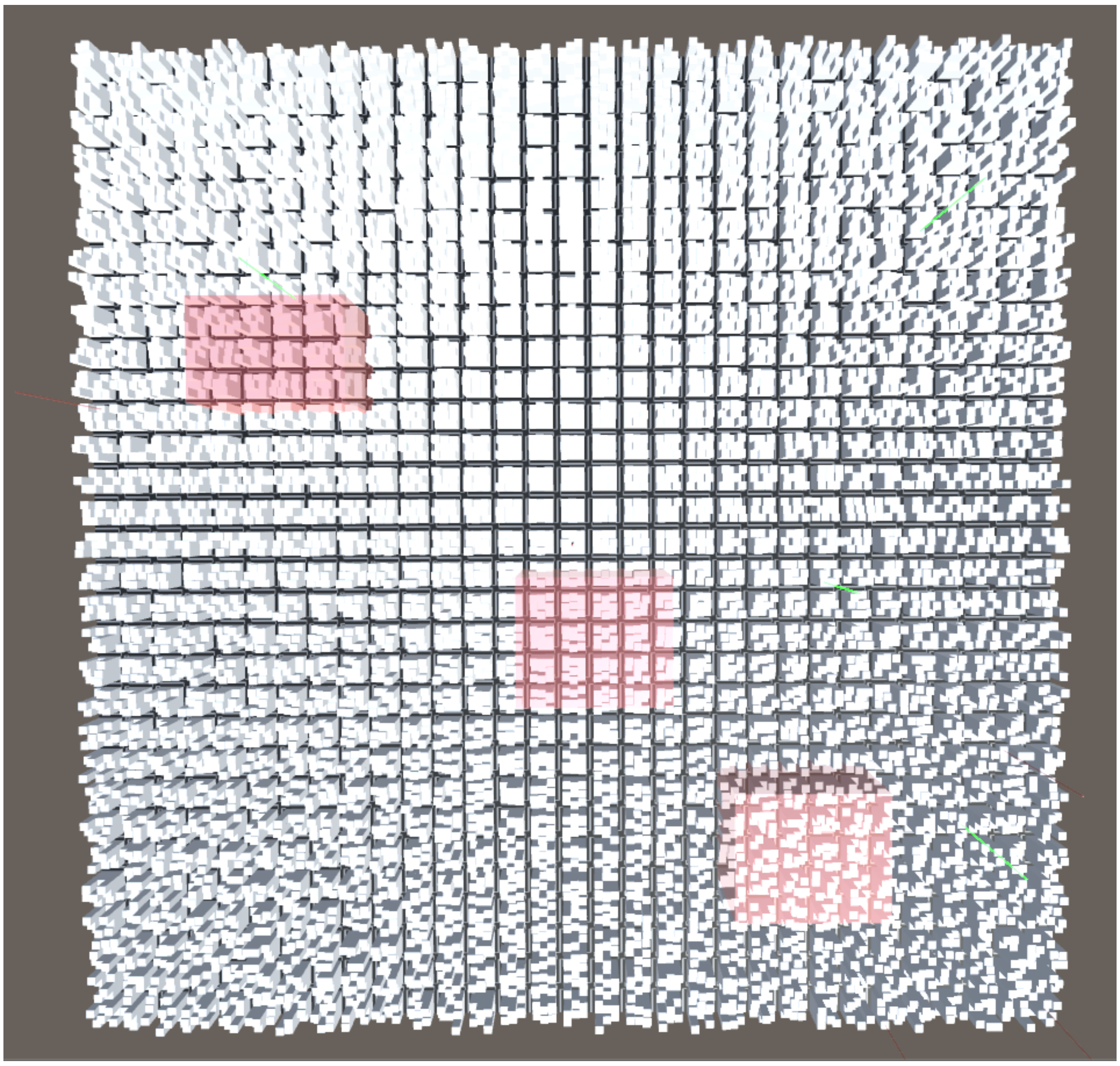

Figure 3.1: Bird's eye view of a generated city, demonstrating the grid layout and no fly zones shown in red

3.1.3 Path Planning

Every update, the controller checks if there are any drones currently waiting at the fulfilment centre, and whether there are any requested deliveries ready to be scheduled. If so, the controller asks the scheduler for the next job to be performed, and assigns it to an available drone.

A dictionary mapping each drone to an instance of the DroneInfo class is used to track the state of each drone the controller is responsible for. This class contains a fight path represented by a list of waypoints, an index tracking how many waypoints the drone has successfully navigated to so far, as well as a method to return the next waypoint in the flight path. When a drone returns back to the controller building, it is added to a list of waiting drones and its entry in the drone info dictionary is cleared. A new entry with a fresh list of waypoints is inserted whenever a drone is assigned a new delivery, but to do this the controller has to first calculate an appropriate route in the sky.

Every flight path starts with a waypoint telling the drone to take off and ascend vertically to a specified flight altitude. This avoids collision with other buildings, and prevents drones from flying low above other areas of the city, which could lead to unwanted levels of noise in the real world. Then, an efficient route to the destination of the delivery that avoids NFZs is computed using the Lazy Theta* algorithm [32]. This is an adaptation of the A* algorithm [33] to allow for any angle path planning, whilst triggering less line of sight checks than the normal Theta* algorithm [34]. These line of sight checks are implemented using raycasts that check for collision with any NFZ. The Lazy Theta* algorithm was chosen over A* because it is shown to create shorter paths than A* with post-smoothing, and the paths appear natural. Waypoints are also added to tell the drone to land, deliver, take off again, reverse the route, and land back at the fulfilment centre, ready for another delivery job.

If needed, the length of the produced flight plans can be calculated by traversing the returned list of waypoints and summing the straight line distance between each. The gain in height from the fulfilment centre to flying altitude and the loss in height back down to the destination building have to be added on. Since drones fly at a constant speed, this distance can be used to accurately figure out how long any given delivery will take once accepted.

3.1.4 World Representation

Three different two-dimensional arrays are used to store a representation of the simulated world. Firstly, in order to calculate the length of flight paths, the change in altitude needs to be known for each possible destination. Therefore, the height of each building is stored in a square array of floats where each element naturally corresponds to a building (or road) in the city grid.

The positions of roads also need to be known so, later, the delivery generator will not create new delivery requests where there is not a building. This is an array of booleans of the same shape as the the array of building heights, called the isRoad array.

Lastly, the positions of NFZs need to be known since it doesn't make sense to generate delivery requests from within these zones. One option would be to store the corners of each NFZ and later test whether buildings are located inside the bounds. However, this containment test could be fairly complicated and requires extra computation. Instead, by restricting NFZs to either fully filling each subsection of the city grid or not filling them at all, the NFZ locations can be stored similarly to the positions of roads, and checking if a building is in a NFZ can be achieved by simply indexing the isNFZ array. Furthermore, this simplifies the idea of visible neighbours in the Lazy Theta* implementation. Of course, this somewhat limits what shape the NFZs can be, since they have to be aligned with the grid and their size must be a multiple of the grid unit size, but this is still sufficient for the purposes of the simulation.

3.2 Drone

3.2.1 Flight

The only real task drones have to perform is physically flying between the controller building and their assigned delivery destination, following their planned path along the way. In order to give the controller authority over the exact path each drone takes - effectively acting as an automatic air traffic controller - drones do not store their full flight path. Instead, on arrival at each waypoint, they query the controller for their next destination.

Since drones also have to complete deliveries and inform the controller when they have returned back to the fulfilment centre, different types of waypoints are used for takeoff, landing, flying to a location, delivering, and to indicate the end of the flight path.

Initially, the controller sets the first waypoint (a takeoff instruction) and tells the drone to perform the task. After performing each type of task, with the exception of the END waypoint type, the drone will request a new waypoint from the controller and start the new task.

Performing the task for each waypoint type involves different steps. END waypoints simply involve informing the controller that the drone is waiting for a new delivery. DELIVER waypoints are similar - the drone informs its assigned delivery that it is complete.

TAKEOFF (flying up until flight altitude reached), LAND, and COORD (flying to a specified position) waypoints are the only instructions that cause the drone to actually move. In the case of a LAND waypoint, the drone will set its velocity to vertically down. Later, Unity will detect the drone colliding with a building, which effectively signals that the drone has arrived at its destination. On receipt of this message, the drone stops and requests a new waypoint. TAKEOFF and COORD, however, specify a point in the air at which to fly to, not the roof of a building. Since waiting for a collision with a trigger is such an effective way of telling a drone that it has completed its current flight segment, drones spawn physical waypoint entities at the target location and set their velocity towards them, waiting to be told they have collided with the triggers (Figure 3.2). On this collision event, the drone has to simply check that the waypoint belongs to itself, not another drone, to know it has completed the task of its current waypoint instruction.

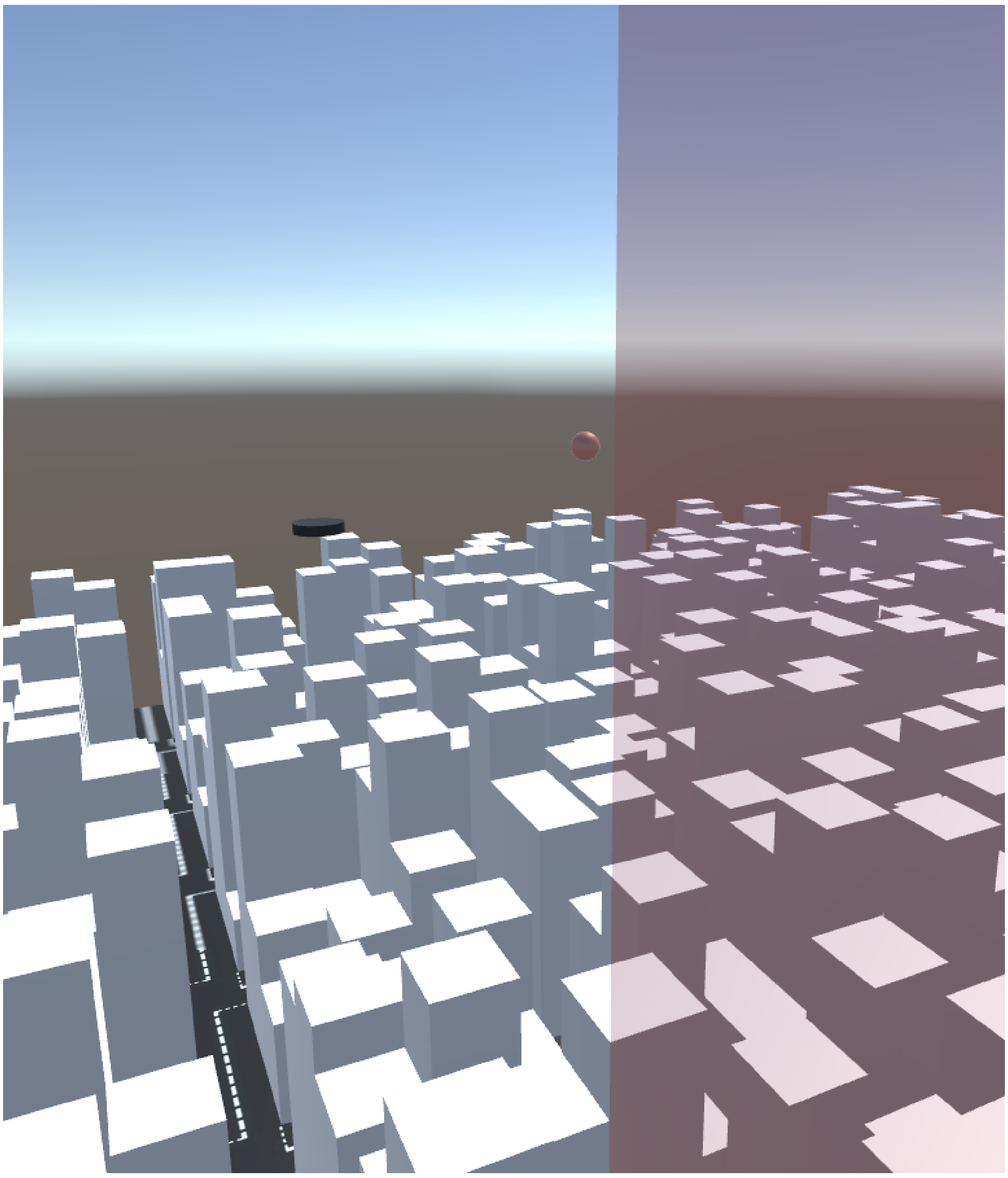
Figure 3.2: A drone (black disc) flying towards its next waypoint (red sphere) near a no fly zone

3.3 Delivery Generator

The delivery generator is required to provide a stream of delivery requests throughout the city, simulating real world customers placing orders in an online shop. For simplicity, new deliveries are created at a fixed, user specified, frequency. Each delivery request is given a unique integer ID. The location is decided by randomly choosing a position within the city grid. The chosen coordinates are then looked up in the isRoad and isNFZ arrays and if either returns true, a new random location is picked.

3.3.1 Package Type

In order to somewhat root the profit earned from each delivery in reality, the delivery generator selects a package type randomly, and also a weight bracket if applicable. The pool of types and weights to choose from is restricted to reflect the sort of parcels a drone could realistically deliver since they have both a limited size and weight capacity.

The package type, alongside its weight, decides the base value of each delivery - the maximum profit that can be earned by carrying out the job. This is determined by looking up the current fulfilment fee that Amazon Prime charges to deliver such a parcel [1], as shown in Table 3.1.

Package Type	Weight /g	Fee /£
Small Letter	≤ 100	0.60
Large Letter	≤ 250	0.80
Small Envelope	≤ 100	1.27
Standard Envelope	≤ 100	1.40
	≤ 250	1.54
	≤ 500	1.63
Large Envelope	≤ 1000	2.05
Standard Parcel	≤ 250	2.02
	≤ 500	2.14
	≤ 1000	2.30
	≤ 1500	2.45
	≤ 2000	2.68
	≤ 3000	3.83

Table 3.1: Amazon Prime Fulfilment Fees for Smaller Packages [1]

3.3.2 Time-value Functions

Time value functions establish a relationship between how long a delivery takes to be carried out and the proportion of the base value that is actually paid to the courier. Choosing monotonically decreasing functions captures the idea that the customer will pay less the longer it takes an item to reach them after ordering. This can later be used to drive a scheduler to encourage prompt deliveries, resulting in a satisfied customer.

The exact shape of these monotonically decreasing functions can differ. For example, a stepwise decrease at regular time steps could be used to represent a linear loss of value with time which may suit a customer who just wants delivery to be as fast as possible. Alternatively, the value halving at some point in time after the order was placed, and then eventually dropping to zero, might better suit somebody who desires delivery before a certain soft deadline.

Of course, many different functions can be imagined but the delivery generator picks from only one of these two. It may be better in the future to add more options, but just a linear stepwise decrease and a halving pattern should be sufficient for benchmarking different schedulers.

To represent these time value functions, an array of booleans is used. Each successive element in the array dictates whether or not the value decreases in a specific time step, with all reductions being of an equal amount and the final value always being zero. This implementation finds a nice trade-off between simplicity and flexibility - allowing many unique functions to be expressed. Finally, the delivery generator randomly assigns a time step to each delivery request which determines how much time each item in the array represents.

Figure 3.3 plots the proportion of the full base value received from a delivery job against the index of the underlying array of booleans. To convert this index to the time since the creation of a delivery request, one must add one to it before multiplying it by the time step. The value is 100% of the base before the first of these times. The array representation of both functions are also provided:

```
stepwiseDecrease =
    [true, true, true, true, true, true, true, true, true, true]

halvingDecrease =
    [false, false, false, false, true, false, false, false, false, true]
```

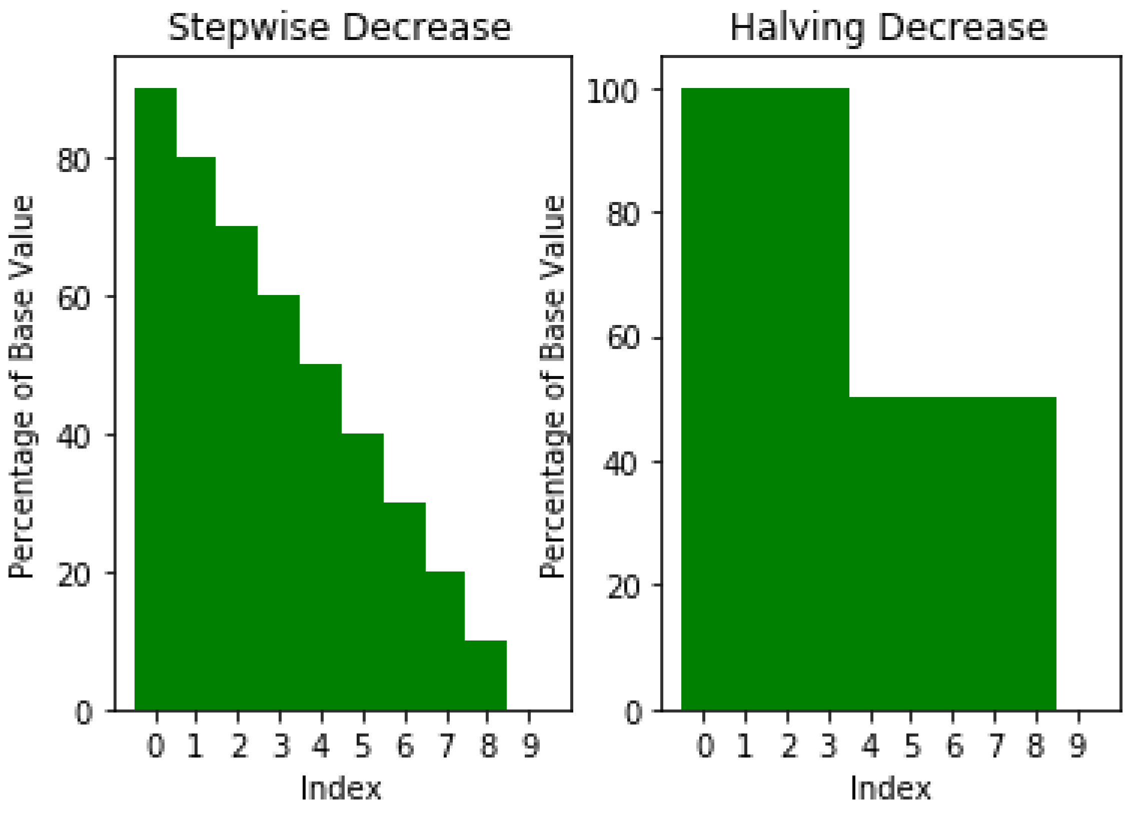

Figure 3.3: Stepwise Decrease and Halving Decrease Time Value functions

3.4 Delivery

A delivery stores information about itself which can later be looked up by the delivery scheduler or the controller. This includes its ID, base value, time value function and corresponding time step, and flight path. The flight path is actually a specific object that allows lazy evaluation of the waypoints and length - only executing the path planning algorithm when first needed and subsequently returning the stored result. A timeout is also included in each delivery such that once it has elapsed, the delivery deletes itself if not already accepted. This could be interpreted as the time at which a delivery should be carried out by conventional means, rather than a drone.

Additionally, deliveries implement a method to enumerate their value after a certain delay. To achieve this, the total number of reductions in the time value function are counted up and the base value is divided by this to determine how much value is lost per reduction.

$$decreaseAmount = \frac{baseValue}{totalDecreases}$$

Then, the index corresponding to the time after the delay is calculated.

$$index = \left\lfloor \frac{timeSinceCreation}{timeStep} \right\rfloor$$

Traversing the array until this index and again counting the number of reductions can be used to calculate the final value at the given time.

$$value = baseValue - (decreaseCount \times decreaseAmount)$$

3.5 Delivery Scheduler

To choose what order the delivery requests are executed in, a delivery scheduler is required. An abstract class is utilised to specify the required functionality of all schedulers, making it easy to implement and test different scheduling policies by extending this class and implementing its methods. At their core, schedulers have to be able to queue new delivery requests, and provide the next delivery to be carried out according to some criteria. It is also useful to be able to remove delivery requests to support timeouts, and to check that the scheduler is currently able to return the next delivery.

At this stage, only a simple First In First Out (FIFO) scheduler is implemented - choosing the oldest delivery request as the one to be executed next.

3.6 Logger

Simulating a drone courier network would be useless without the ability to extract metrics about the performance and profitability of the delivery scheduler. Therefore, a logger runs during the entire duration of the simulation, calculating the total profit, number of complete and accepted deliveries, total number of deliveries requested, various timings, and the drone utilisation, at a fixed frequency. The results are saved to the Unity logs and also shown on screen.

3.7 Seeding

Throughout early tests it was observed that due to the random nature of the generation of the city and NFZs, and delivery values and locations, the performance of each scheduler would vary greatly every time the simulation was run, especially on shorter executions. This variance could often exceed the effect from changing the delivery scheduling algorithm, making it very difficult to compare their effectiveness.

To solve this, the user can specify a seed for the random number generator before running the simulation such that choosing the same seed multiple times will result in the exact same scenario being generated. This allows fair comparison between schedulers but it should be noted that generating the same sequence of delivery requests each time reduces the generality of the test.

On short tests this may result in scheduler A outperforming scheduler B using a given seed, but scheduler B outperforming scheduler A when another seed is chosen. Therefore, it is advisable to run longer simulations, or, if time does not allow this, to repeat shorter simulations using different seeds.

Chapter 4

Scheduler Implementations

Now that a simulation framework is in place to facilitate the evaluation of scheduling algorithms' behaviours, the focus of the project shifts to devising and implementing numerous different policies in an effort to generate more revenue than Least Lost Value.

Since most of the schedulers to be tested are some form of priority scheduler, it makes sense to add a priority field to the delivery class such that the scheduler implementation can set this and use it to order the list of all delivery requests. This also requires the delivery class to adhere to C Sharp's IComparable interface and implement the CompareTo method such that a delivery with a higher priority is considered to come after one with a lower priority.

4.1 First In First Out

The first and most basic scheduling algorithm implemented is the popular First In First Out (FIFO) scheduler. This maintains a list of all delivery requests and new requests are added to the end of the list. By scheduling deliveries at the front of the list first, the oldest request is always the next to be served.

4.2 Shortest Job First

Another very common scheduling technique is to prioritise the shortest job. This should allow the maximum number of deliveries to be completed which is good for customer satisfaction and may lead to a large revenue since it creates the most opportunities to extract profit. However, due to ignoring the actual value gained from each delivery job, an SJF scheduler may lead to large numbers of deliveries that are not particularly, or at all, profitable.

When a new delivery request is added to the list of all delivery requests, the scheduler reads its flight path length from the delivery's flight path object and sets this as the priority. The actual time a drone spends out for delivery need not be calculated, since it is directly proportional to the flight path distance so would have no effect on the ordering. This approach means that to choose the shortest job, the delivery with the lowest priority should be scheduled first. To avoid this slightly confusing situation, the reciprocal function (or negation) of the flight path distance could be used, but this requires slightly more computation and would be more likely to cause issues with the precision of floats. If the maximum queue length is exceeded by the addition of the new delivery, its priority is compared to the maximum element in the set of deliveries, and, if it is lower, it evicts this element.

When the controller asks for the next scheduled delivery, the delivery in the set with the minimum priority is removed and returned.

Due to the priority of each delivery being the flight path distance, which never changes, the SJF delivery scheduler can take advantage of C Sharp's SortedSet collection to store the queue of deliveries. This collection allows easy and efficient access to both the minimum and maximum element, which are the only elements the SJF implementation ever needs to access, and maintains these automatically as new deliveries are added and removed.

4.3 Least Lost Value

The Least Lost Value (LLV) scheduling strategy was developed using the Constrained Scheduling Problem method and aims to use the cruciality of the problem in order to find the least-impact policy [6]. To attempt to minimise the negative impact, both the Potential Gain Value (PGV) - the value gained by doing a job now instead of later, and Potential Lost Value (PLV) - the sum of the value lost from all other jobs by doing them later, rather than now, are considered. By subtracting the PGV from the PLV, the Net Lost Value (NLV) is found, and jobs are scheduled by minimising this value.

In the context of drones carrying out deliveries in a simulated city, the problem is simplified by removing any stochastic nature from the time taken to carry out a delivery. In a real life scenario, there may be some variance in how long it takes a drone to fly a given distance with a package, in which case the expected value of the task duration distribution should be used.

The implementation of this scheduler cannot utilise the SortedSet collection as SJF does because the addition and removal of delivery requests changes the NLV of every other delivery in the queue. Therefore, a list of deliveries is used to store the delivery requests and this is sorted when needed. When a new delivery request is queued, it is simply added to this list.

To provide the next delivery to be executed, the LLV scheduler calculates the NLV of each delivery request in the list, sets the result as the priority, and then sorts the list. Similar to the SJF scheduler, the item with the least priority is actually the one that is carried out next. If, after sorting, the length of the list is greater than the maximum queue length, delivery requests with the highest NLVs are evicted until there are few enough elements in the list. This means that the list of all delivery requests is often longer than the maximum size, but it will only exceed the limit by the number of new deliveries requested between each scheduling decision.

Calculating the NLV for a given job first requires the PLV to be enumerated. To find the sum of the values lost from all other deliveries by first doing the given delivery, the length of time it takes a drone to fly from the fulfilment centre to the order destination and back is calculated. Then, for all other deliveries in the queue, the time taken to fly one way is calculated and the value earned by doing the other delivery after doing the given delivery is subtracted from the value earned by doing the other delivery now. The PLV is the sum of all these differences.

```
int PLV(Delivery delivery, float currentTime)
{
  int lostValue = 0;
  float twoWayJobTime = jobTime(delivery) * 2;
  foreach (Delivery otherDelivery in deliveries)
  {
    if (otherDelivery != delivery)
    {
      float otherJobTime = jobTime(otherDelivery);
      lostValue += (otherDelivery.GetValueAfterTime(otherJobTime)
        - otherDelivery.GetValueAfterTime(otherJobTime + twoWayJobTime));
    }
  }
  return lostValue;
}
```

The PGV also has to be calculated. To do this, the mean of the time taken to fly both ways for every other delivery is found as the estimated delay caused by scheduling another job first. The time taken to fly the parcel to the recipient of the chosen delivery is also found in order to evaluate the value gained by doing this delivery at the current time. The value gained by carrying out this delivery after the estimated delay is calculated and subtracted from the current time value to find the PGV.

```
int PGV(Delivery delivery, float currentTime)
{
  float jobTime = jobTime(delivery);
  float totalTimeOfOthers = 0;
  foreach (Delivery otherDelivery in deliveries)
  {
    if (otherDelivery != delivery)
    {
      totalTimeOfOthers += jobTime(otherDelivery) * 2;
    }
  }
  float avgTimeOfOthers = totalTimeOfOthers / (deliveries.Count - 1);
  return (delivery.GetValueAfterTime(jobTime) -
    delivery.GetValueAfterTime(jobTime + avgTimeOfOthers));
}
```

The list of deliveries is sorted using the NLV: the difference between the PLV and the PGV - maximising the "won" value, whilst minimising the "lost" value.

It is believed that these three schedulers - FIFO, SJF, and LLV - are the only algorithms that have previously been tested in a similar simulated environment [6] with drones launched from static fulfilment centres to carry out deliveries.

4.4 Least Lost Value Variants

4.4.1 Approach

Whilst the LLV scheduling algorithm seems to consider all attributes that may affect overall profitability, there are a few potential areas for improvement that are worth experimenting with. These are generally rooted in the time durations that the algorithm uses both in its calculation of Potential Lost Value and Potential Gain Value.

4.4.2 Previous Delivery Duration

When evaluating the PGV of a job, LLV estimates the delay caused by executing another delivery first as the mean of the total durations of each other job in the queue. However, it is likely that to extract optimal profit, the duration of accepted delivery requests will be skewed towards those closer to the fulfilment centre, and therefore the mean will overestimate the delay. One possible improvement is to maintain an average of the duration of jobs previously accepted as a prediction of future accepted delivery durations, and to use this value instead in the PGV calculation.

4.4.3 Next Drone Dispatch Time

Another thing to note about the LLV policy is that it is naive to there being multiple drones carrying out deliveries. When considering deliveries being performed at a later time, the delay is always calculated as if a single drone has to perform another delivery and return back to the controller building before any more scheduling occurs. This is not the case - a second drone may return back ready for dispatch before the first completes its flight. It may be worth tracking the expected return time of each drone to find an accurate delay in both the PLV and PGV calculation.

4.4.4 Least Lost Value Density

It can be shown that the schedule in which, at all times, the job with the highest value density is executed first will result in a total value as high as, or even higher than, any other schedule [35]. This relies on the value gained by completed each job being known, and the scheduler being able to "pause" any given task and return to it later, after processing other tasks for any amount of time. Inspired by this, despite the inability to pause a delivery, a variant of LLV is devised that aims to minimise the net loss of value density - the value gained divided by the time invested to do so - to hopefully drive the scheduler towards a schedule where the sum of the value density of every delivery carried out is maximal.

To do this, the concepts of Potential Gain Value and Potential Lost Value are tweaked to be Potential Gain Value Density (PGVD) and Potential Lost Value Density (PLVD). The calculations of these scores are similar to previous, except the PGV is divided by the round trip time (RTT) of a drone performing the given delivery to find the PGVD, and each value in the sum forming the PLV is divided by the round trip time for the respective other delivery, resulting in the PLVD.

$$PGVD(job_i) = \frac{PGV(job_i)}{RTT(job_i)}$$

$$PLVD(job_i) = \sum_{job_k \in J, k \neq i, k=1}^{n} \frac{value(job_k, time_{cur}) - value(job_k, time_{cur} + RTT(job_i))}{RTT(job_k)}$$

4.5 Highest Value First

4.5.1 Highest Initial Value

The Highest Initial Value scheduling strategy sets the priority of each delivery as its base value - the profit that would be extracted if the parcel was delivered instantaneously. Since this value never changes, the SortedSet collection can be utilised and the maximum element can efficiently be found to be scheduled. The minimum element of the delivery set is evicted if a new delivery request of a higher initial value is added.

This method is greedy and also has no concept of time, meaning it may suffer due to not considering the later impact of choosing a certain delivery, and also very old deliveries may be chosen which are no longer profitable.

4.5.2 Highest Current Value

This policy is similar to the Highest Initial Value scheduler, but by prioritising the delivery that yields the most value to the courier at the earliest time a drone can arrive at the destination, instead of the delivery that would have yielded the most value if carried out as soon as requested, it attempts to address the potential downfalls of the previous approach.

Unfortunately, a SortedSet can no longer be used since the current attainable value of each delivery will vary as time passes, so has to be recalculated every time a new delivery is scheduled. Consequently, a list of deliveries is used which is sorted during scheduling decisions and excess elements are also pruned at this time.

This method is still greedy, but will no longer schedule old deliveries that are not profitable anymore.

4.5.3 Highest Value Density

Another similar approach is to consider the value density of each delivery as its priority and attempt to maximise this. The value density is defined as the value gained from a job divided by the length of time it takes to carry out this job, the round trip time (RTT).

$$valueDensity(job_i) = \frac{value(job_i, time_{cur})}{RTT(job_i)}$$

However, since drones fly at a fixed speed, we can remove this constant factor and reduce the amount of calculation required. Additionally, using the one-way distance of the delivery means it can be directly looked up from the flight path object and the calculation can be further simplified. This results in the priority being equal to the value gained divided by the one-way distance flown to deliver the package.

$$valueDensity(job_i) \propto \frac{value(job_i, time_{cur})}{oneWayFlightDist(job_i)}$$

As mentioned earlier, the schedule with the highest overall value density will be the one that extracts maximum profit, making this a promising metric, but it is a greedy algorithm naive to the impact of carrying out a delivery on the value density that can later be achieved.

Two versions of this scheduler are implemented - one considering the initial value density, and the other considering the current value density. It is likely the current value density version will be superior, but both are included for completeness.

4.5.4 Two Step Value

In an attempt to consider the effect of performing a given delivery on the value of later deliveries, the concept of Two Step Value is proposed as a novel metric to be maximised. This is defined as the sum of the profit earned by fulfilling a delivery, and the maximum profit that can then potentially be gained from one of the remaining options after the drone has returned.

```
foreach (Delivery delivery in deliveries)
{
  float jobTime = jobTime(delivery);
  int thisValue = delivery.GetValueAfterTime(jobTime);
  int nextValue = 0;
  foreach(Delivery nextDelivery in deliveries)
  {
    if (nextDelivery != delivery)
    {
      float nextJobTime = jobTime(nextDelivery);
      nextValue = Mathf.Max(nextValue,
                 nextDelivery.GetValueAfterTime(jobTime * 2
                                  + nextJobTime));
    }
  }
  delivery.Priority = thisValue + nextValue;
}
```

This Two Step Value should encourage deliveries that have less of a detrimental impact on later performance, but it is not perfect since it is not aware of other drones also performing deliveries, and it cannot predict what other delivery requests will be made whilst the drone is out for delivery for the first job.

The Two Step Value principal could be extended to any N-Step Value up to the maximum queue size. However, this would vastly increase the size of the solution space to be searched, and therefore the complexity of computing the priority. As larger N are considered, the algorithm looks ahead to a time further in the future when $N-1$ deliveries have been completed. Not only does this mean the current list of delivery requests will become increasingly stale due to their value decreasing over time, but more new delivery requests will also have been added to the pool of options, meaning there is more uncertainty. Therefore, it is argued that any $N > 2$ results in a priority that is no more useful than the Two Step Value.

A similar metric - Two Step Value Density - can also be derived, aiming to maximise the value density over two scheduling decisions. It is expected that this will lead to shorter jobs being accepted, and it is hoped that the resulting schedule will have a greater average value density than if the Highest Value Density policy were used.

4.6 Last In First Out

A LIFO scheduler is the opposite of the FIFO scheduler. The most recent delivery request will always be the next to be executed. This may be somewhat advantageous in terms of profit since it inadvertently prioritises jobs that will still have value close to their base (maximum) value. Unlike density based approaches and SJF, LIFO is geographically fair - placing no emphasis on carrying out deliveries with shorter flight times. It's also fair towards customers requesting less expensive deliveries, though this is clearly disadvantageous for the courier company's profit.

4.7 Splitwise Scheduling

Each of the scheduling policies implemented above will exhibit their own behaviours. For example, some, such as LLV and Highest Current Value, are designed purely around extracting as much profit as possible over time. Others, such as SJF and value density based schedulers, are likely to cause a higher total volume of delivery requests fulfilled. Meanwhile, FIFO is focused on being fair - effectively not allowing anyone to queue jump.

Many of these characteristics are desirable in the context of a drone courier company. Maximising profit is only maximising the short-term value to the service provider, and doesn't directly benefit the customer (though the monotonically decreasing nature of the Time Value of Service Delivery functions creates a correlation with short delivery times). From the perspective of the courier company, it is also important to reach as many customers as possible - both in terms of delivering many packages, and covering a wide geographical area - and to keep the average wait time low. This will result in a greater customer satisfaction, helping the company to grow its user base and make more money in the long term.

Therefore, new splitwise schedulers are implemented, using other scheduling policies as building blocks to create a more rounded behaviour. The general approach involves taking a number of schedulers' priorities and attaching a weight to each, creating a blend of them all and allowing the overall performance to be tuned by adjusting the weights. Unfortunately, this weighted sum is not ideal since the priority from each scheduler is not normalised, or even necessarily linear, so the effect of adjusting the weights may not be intuitive. For example, setting the weights as equal does not imply that the final behaviour will be an even split of the characteristics of each involved scheduler's policy.

4.7.1 LLV-SJF

The LLV-SJF splitwise scheduler takes a list of delivery requests and calculates the net lost value for each, as in the LLV policy, as well as the round trip time, similar to the SJF policy. The values of both scores are found and multiplied by their corresponding weight before being added together and the result multiplied by negative one to produce a priority to be maximised. The round trip time is used instead of the one-way path length in an attempt to make the weights more intuitive.

The aim of this join is to combine the profitability of LLV with the delivery volume of SJF in a way that allows the courier company to adjust which one they place most emphasis on. It is expected that buildings further from the fulfilment centre will not have many of their delivery requests accepted no matter how the weights are set since a greater distance is detrimental to both profitability and the number of deliveries performed.

4.7.2 HVD-SJF

This scheduler is very similar to the LLV-SJF combination, but uses the Highest Value Density scheduler as the component that maximises profit in place of LLV. The only difference in implementation is that the score from the HVD policy will not be inverted since it is already designed to be maximised.

4.7.3 SJF-LIFO

By combining the SJF scheduler with the LIFO scheduler, a new slitwise scheduler is created that prioritises short deliveries that were recently requested. This may be good for profit due to the value not having a chance to decrease from its base, and also more deliveries being performed to allow more customers to be charged. Interestingly, an SJF-LIFO split would achieve this without unfairly preferring customers who have payed more. The geographical fairness can be tuned using the weights - the more SJF is weighted, the less fair in this regard.

Since LIFO isn't naturally a priority scheduler, a priority that exhibits the same behaviour has to be considered. Minimising the time since the request was made should achieve this, so the negative weighted sum of this and the one-way flight distance is maximised to achieve the overall intended behaviour.

Chapter 5

Evaluation

In order to evaluate the performance of each individual scheduler and compare them to one another, the Unity simulation saves numerous key metrics to a log file. A python notebook tool is then used to extract these statistics and plot graphs to provide a visual insight into each policy's behaviour.

The simulation was configured with 10 drones and a city grid with 31 cells, each consisting of a 3 by 3 square of buildings with roads separating each cell. The buildings were between 5 and 25 units tall and the drone flight altitude was set to be 50 units above street level, where 5 units is the side length of a building or road segment. In this city, 3 no fly zones were randomly generated, with each side being between 10 and 20 times the length of a building. This setup is the same as in Figure 3.1. The drone speed was set to be 10 units per second. The delivery generator was set up to spawn a new delivery every 4 seconds, and each delivery's time value function would have a time step between 5 and 8 seconds long, meaning no profit can be made from a delivery 50 to 80 seconds after its creation. The deliveries were set to time out after 5 minutes to avoid too many entities slowing things down.

Each of the implemented schedulers were run for 6 hours (with the exception of FIFO which was stopped after approximately 1 hour), with the logger printing out metrics every 5 seconds, resulting in 4,320 data points per policy. Every test was performed with the same seed, since it was found that otherwise the variance induced by the randomness of the simulation was too great to fairly compare the results between simulations.

5.1 Profit

Perhaps the most important factor, especially in the short term, for a drone courier company choosing a scheduling algorithm, is the amount of money made for the service provider.

5.1.1 Total Profit

The total profit is simply defined as the sum of the value extracted from each completed delivery.

By taking a look at Figure 5.1, it is clear that the FIFO scheduler implemented is extremely inefficient at making money for the service provider. A small amount of money is made and then no further profit is accrued until the simulation was stopped at around the 1 hour mark. The performance of the other scheduling policies is a bit unclear due to the tight grouping, so Figure 5.2 shows a zoomed in view by plotting only the last few time points. Here, it can be seen that LLVDispatch, LIFO, SJF, Highest Initial Value First, Highest Initial Value Density, and the SJF-LIFO splitwise scheduler are not too effective in terms of total profit earned. Interestingly though, SJF-LIFO performed the best out of these options despite not directly prioritising the

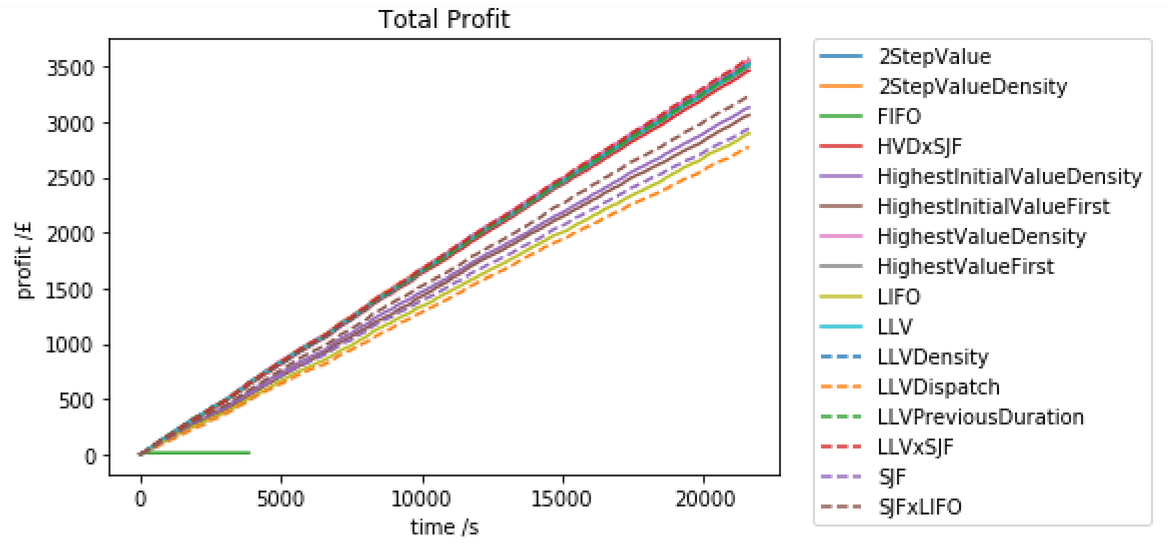

Figure 5.1: Total profit of all schedulers

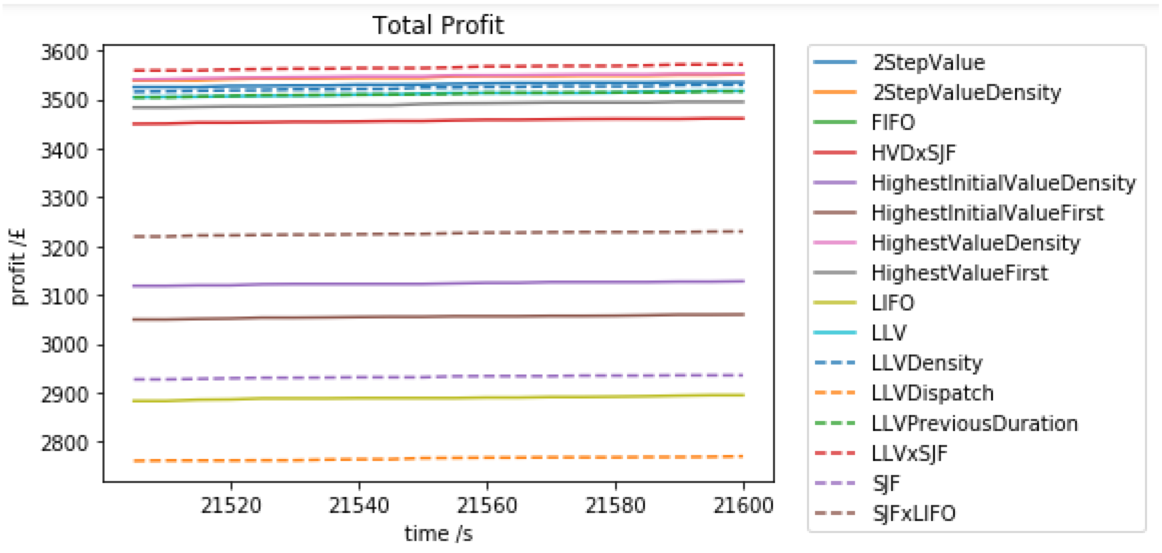

Figure 5.2: Final time slice of total profit of all schedulers

value of deliveries when making scheduling decisions, earning approximately 90% of LLV-SJF - the best scheduler in this regard. The worst of these is LLVDispatch which makes roughly 23% less revenue than LLV-SJF. Because it is still difficult to see the exact ranking amongst the top group of scheduling policies in this graph, Figure 5.3 is provided which only plots the values for the most profitable scheduling algorithms, also in the zoomed in view. Here, it is apparent that the splitwise LLV-SJF scheduler was able to extract the most profit over a 6 hour simulation. It is worth noting that this scheduler was built from LLV for its profitability, and SJF for its volume of deliveries served, in the hopes of creating a more rounded scheduler that exhibits both of these properties, but to less of an extent. Surprisingly, this combination did not result in any reduction in total profit as compared to LLV, but rather an increase, implying LLV does not put enough priority on reducing the time it takes to carry out each delivery when optimising purely for profit. This is further evidenced by the LLVDensity scheduling policy outperforming LLV, when value density only differs from pure value by punishing longer flight times more. Finally, it is mentioned that, on the whole, density based algorithms seem to have performed best and the simplest - Highest Value Density - even ranked second whereas LLV, the policy previously thought to be the best in terms of profit, ranks sixth.

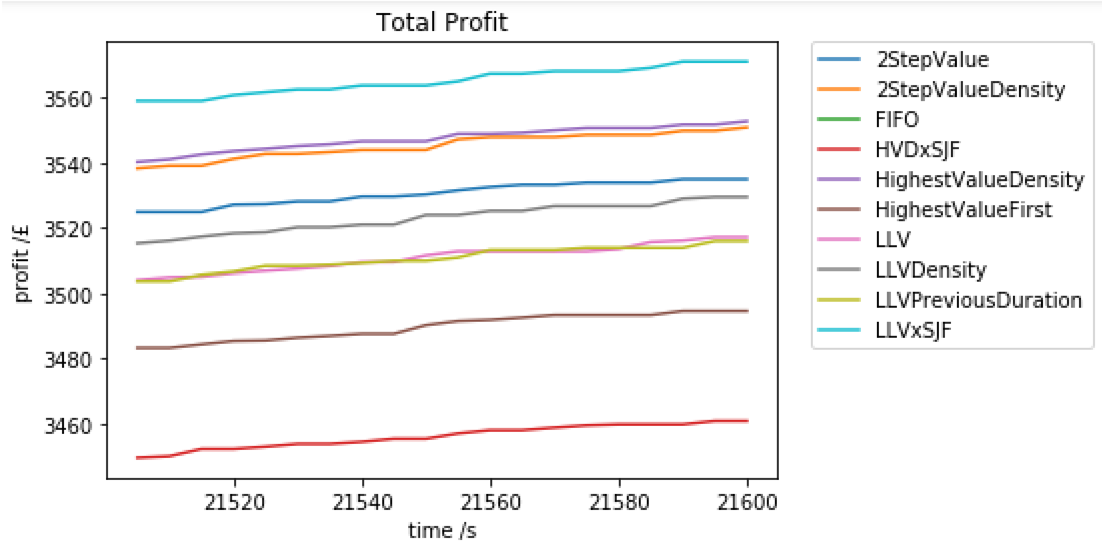

Figure 5.3: Final time slice of total profit of best schedulers

5.1.2 Profit per Delivery

The average profit per delivery gives an insight into how the scheduler makes its money, and how much it utilises the available profit of each delivery. It is defined as:

$$averageProfitPerDelivery = \frac{totalProfit}{numberOfCompleteDeliveries}$$

Three similar graphs have been plotted showing the results of the average profit made per delivery. Figure 5.4 displays all the data collected across every simulation. FIFO is seen to initially make profit on the first few deliveries but the average quickly reduces, tending towards zero, likely due to it serving old delivery requests that are no longer money making opportunities. The spike seen in most schedulers' values at the start can be explained by an early delivery request being very valuable, having a great affect on the mean since there are very few, if any, less profitable deliveries that have been carried out at this stage. Figure 5.5 is zoomed in on the last portion of the previous graph, allowing the order of most of the schedulers to be seen, whilst Figure 5.6 only includes the results of the 8 highest ranking schedulers which all display similar values for the average profit per delivery, ranging from just over 97 pennies per delivery, to just over 99 pennies. This top tier is full of the same policies as the best group for total profit with the exception of HVD-SJF, which earns about 92 pennies per complete delivery, a whole 5 pennies outside the best performers. Inside the group of top ranking schedulers, the density based schedulers proved most effective at earning the most total profit, but the opposite is seen in terms of profit per delivery. Here, the 3 value density based algorithms are the 3 worst ranking, with a clear separation between these and the value based schedulers. The LLV-SJF splitwise scheduler forms a medium between the two approaches, which makes sense given that it is formed by a policy that aims to maximise pure value, and a policy that minimises the time of jobs, which results in a behaviour that is similar to maximising value density. The fact that this scheduler is not amongst the value density schedulers may evidence that the weights used to combine the priorities of each of its components favour the score from LLV - placing a higher precedence on optimising for pure value on each scheduling decision.

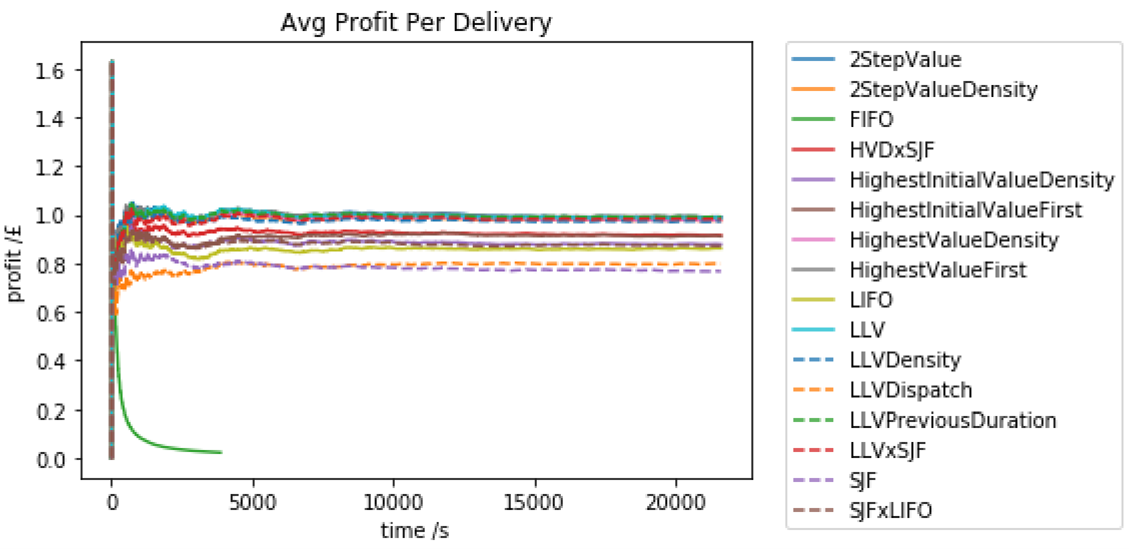

Figure 5.4: Average profit per delivery of all schedulers

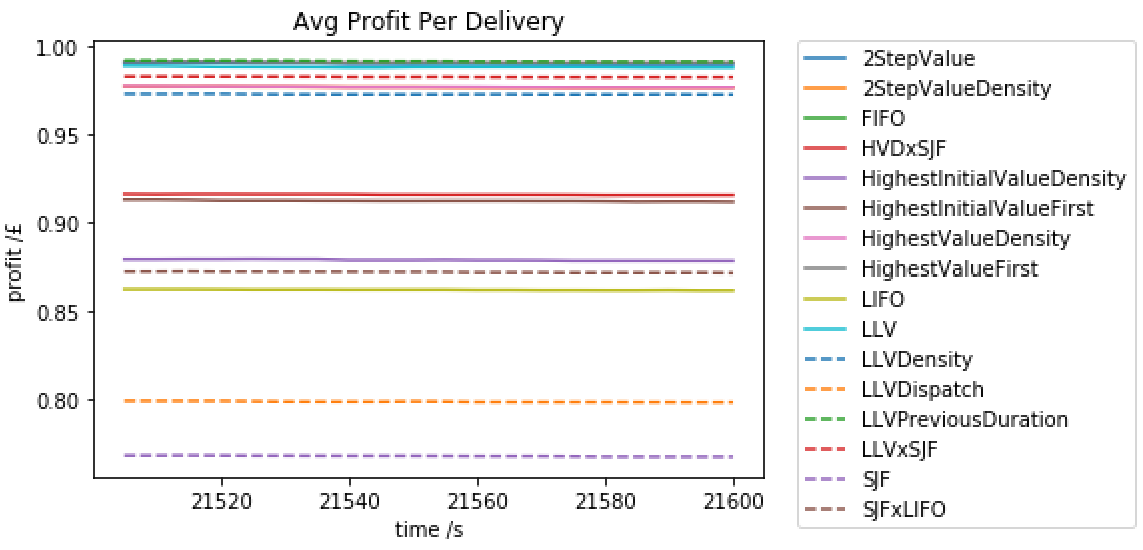

Figure 5.5: Final time slice of total profit per delivery of all schedulers

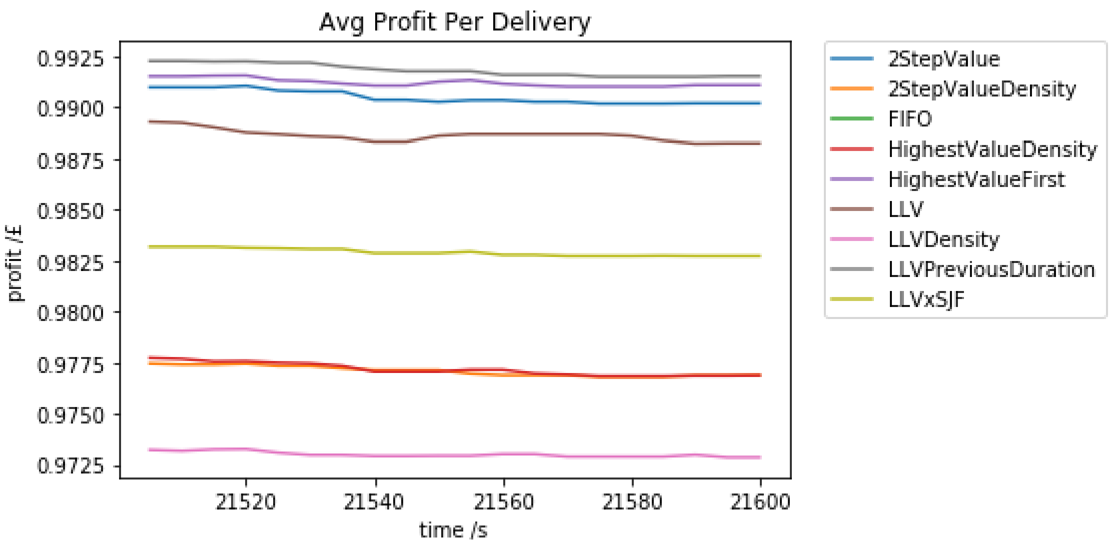

Figure 5.6: Final time slice of total profit per delivery of best schedulers

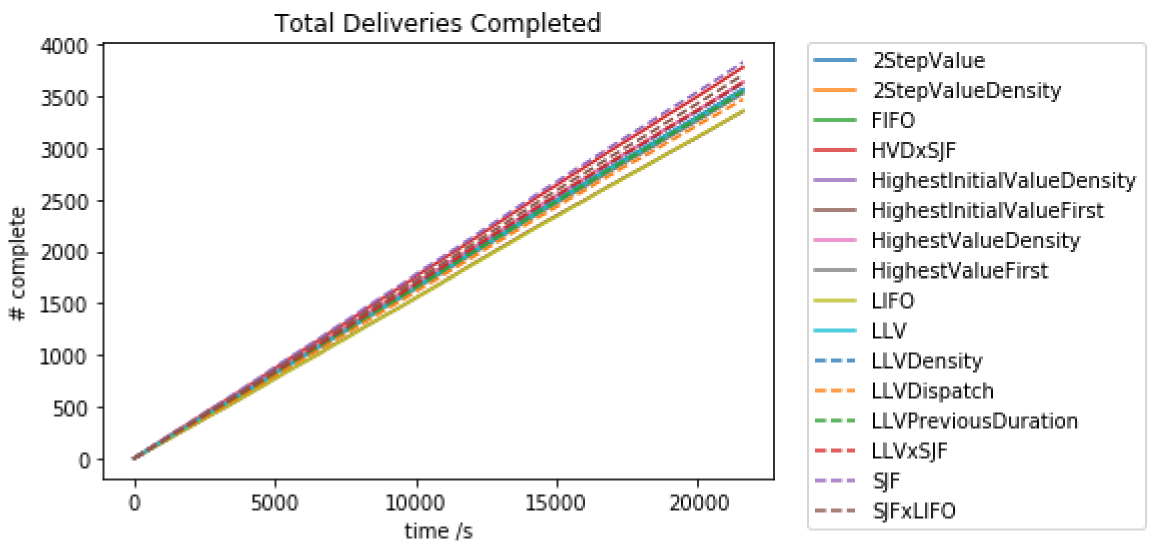

Figure 5.7: Number of deliveries completed for all schedulers

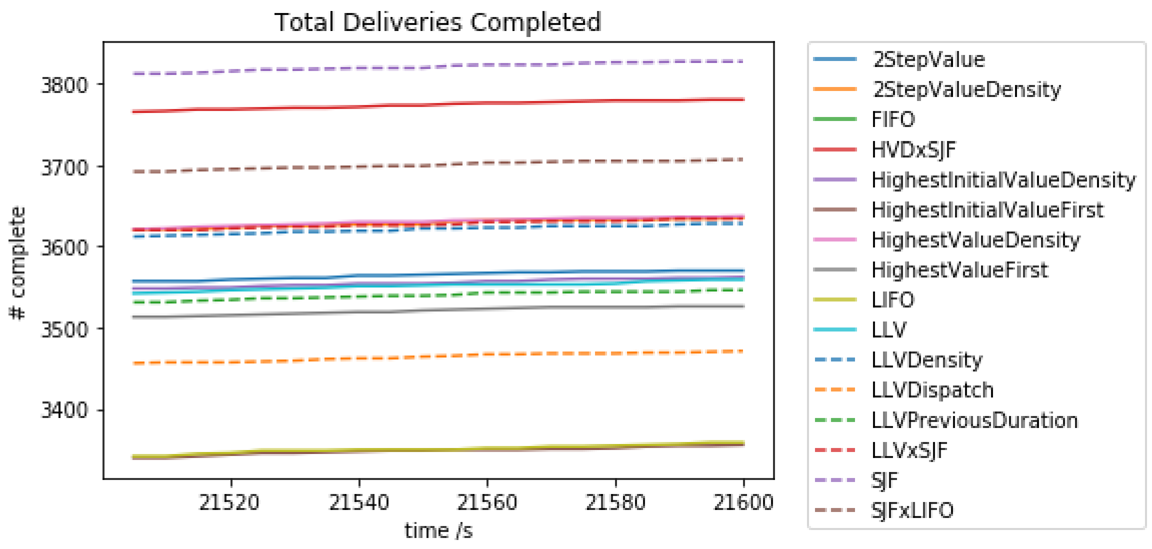

Figure 5.8: Final time slice of number of deliveries completed for all schedulers

5.2 Deliveries Completed

The quantity of deliveries completed by the drones is an important metric for understanding how large of a customer base is being reached. By performing more deliveries, more customers are served, which is advantageous for the growth of the company. Furthermore, it implies that each customer will have more of their orders delivered quickly by drone, so they will be happier with the service provided by the courier.

Figure 5.7 is a plot of how the number of deliveries completed by each scheduler varies with times, and Figure 5.8 shows only the final time slice of this for clarity. Unsurprisingly, the Shortest Job First scheme serves the highest volume of delivery requests due to its aim simply being to carry out as many jobs as possible. In this sense, this would be a very strong choice for a drone courier service provider to use in order to improve customer satisfaction and grow the customer base. The three splitwise schedulers also perform well with respect to fulfilling as many orders as possible since they all combine SJF with another policy, with all three outperforming their second component when considered on its own.

5.3 Timing

To better understand how the algorithms are working, three key timings are considered. The Time to Accept is how long it takes for a delivery request to be accepted (the time at which a drone is dispatched) after the request is made. A low average Time to Accept indicates that customers are not waiting a long time for their orders to be dispatched. It is good to note that this time is independent of the distance between the customer and the fulfilment centre, meaning it cannot be artificially reduced by generally serving only customers who are nearby, perhaps making it the most important timing metric to minimise to provide a greater level of service to more customers.

$$timeToAccept = acceptedTime - creationTime$$

The Time to Complete is the amount of time that passes between a customer placing an order and the package arriving at their house. When considering an individual case, this is the time that a customer will care most about reducing, since they want their parcel to arrive as soon as possible. However, the average Time to Complete cannot be interpreted quite so simply. This is because the flight time has an effect on the value, so a scheduler that prioritises recipients who live close to the controller building will naturally produce a smaller average than a controller that takes just as long to dispatch drones but serves all customers fairly based on distance. In other words, this metric indicates how long deliveries take, but only for those customers who are actually provided a delivery service.

$$timeToComplete = completedTime - creationTime$$

Finally, the (one way) Flight Duration is considered: the amount of time elapsed between a drone being dispatched, and arriving at the delivery destination. Reducing this time has no direct benefit to the customer, as it is effectively a measure of the radius of the delivery service provided by a fulfilment centre. In fact, if two schedulers manage to generate the same profit, the one with the higher average Flight Duration would likely be preferable, since it serves customers in a wider geographical area. This metric could be used by a courier company to decide how far apart their controller buildings should be in order to provide a similar level of service for everyone in a city.

$$flightDuration = completedTime - acceptedTime$$

5.3.1 Time to Accept

Every scheduling algorithm exhibits a similar trend in the average time to accept delivery requests throughout the simulations, growing to a given value and settling fairly quickly, as shown in Figure 5.9. The magnitude of the values that are settled at are generally fairly similar with the exception of that of FIFO, which grows to a level a whole order of magnitude greater than the others. This is expected since FIFO chooses the oldest pending delivery request to be executed and cannot keep up with the volume of orders made. Thus, increasingly old deliveries are carried out by the drones until every delivery is accepted just before it times out after five minutes; hence the growth rate of the average accept time decreases, tending towards this 300 second mark.

Figure 5.10 better shows the results for all other schedulers. LIFO is the best at reducing the average accept time - again, this is unsurprising since it schedules the newest delivery request - and SJF-LIFO places second due to partly exhibiting the behaviour of LIFO. This suggests that a fast service is provided to all customers served a large proportion of the time, regardless of their geographical location. Most of the remaining scheduling algorithms perform very similarly to one another, producing values in the 7.5 second to 10 second range. Highest Initial Value Density, LLVDispatch, and SJF perform disappointingly, with between 21 and 22.5 seconds average accept time. This behaviour of SJF has increased the accept time in each of the splitwise schedulers, though the effect is minimal in the LLV-SJF combination.

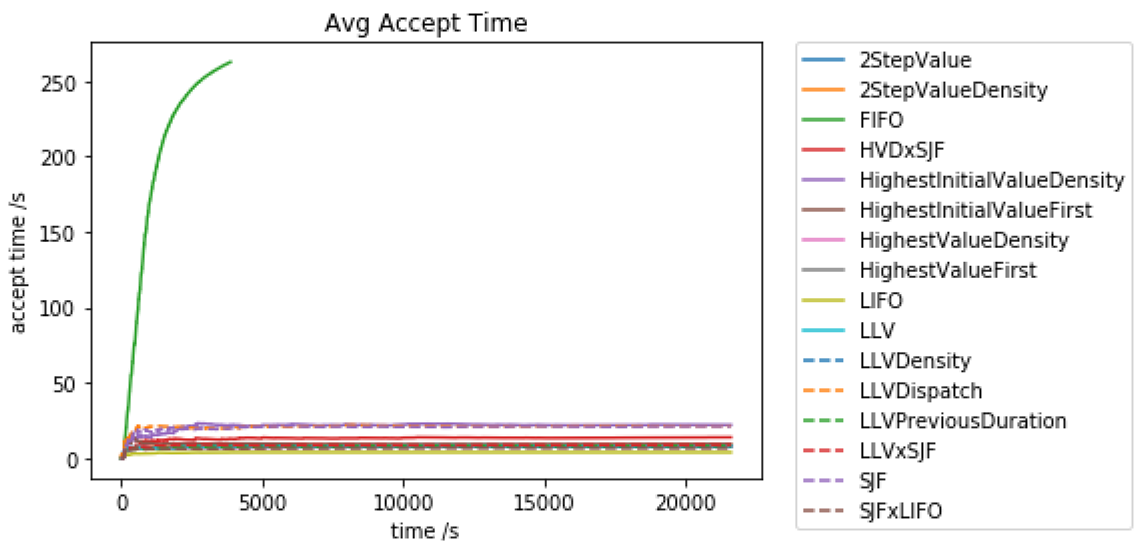

Figure 5.9: Average time to accept deliveries for all schedulers

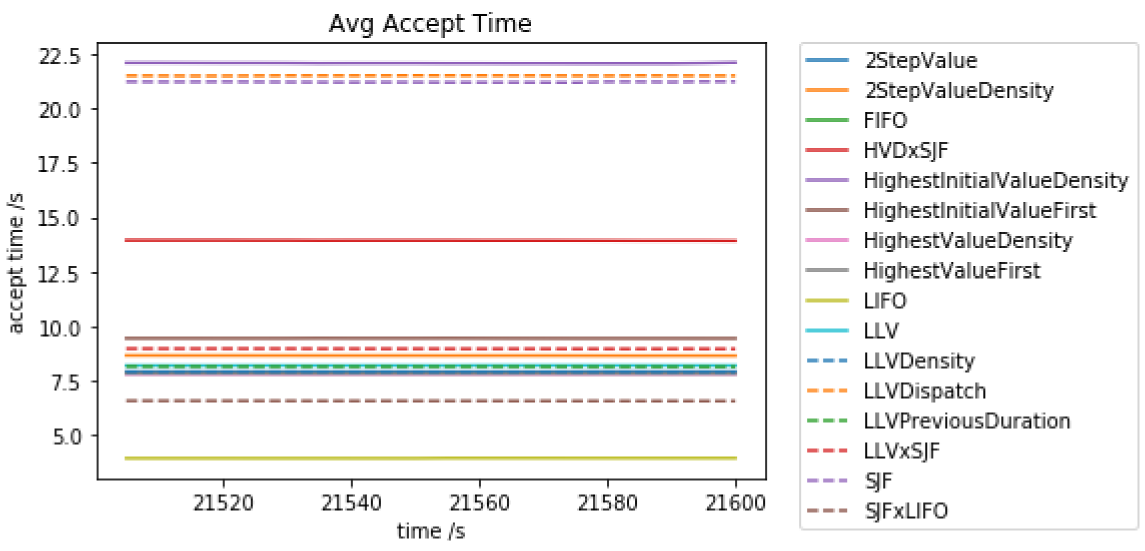

Figure 5.10: Final time slice of average time to accept deliveries for all schedulers

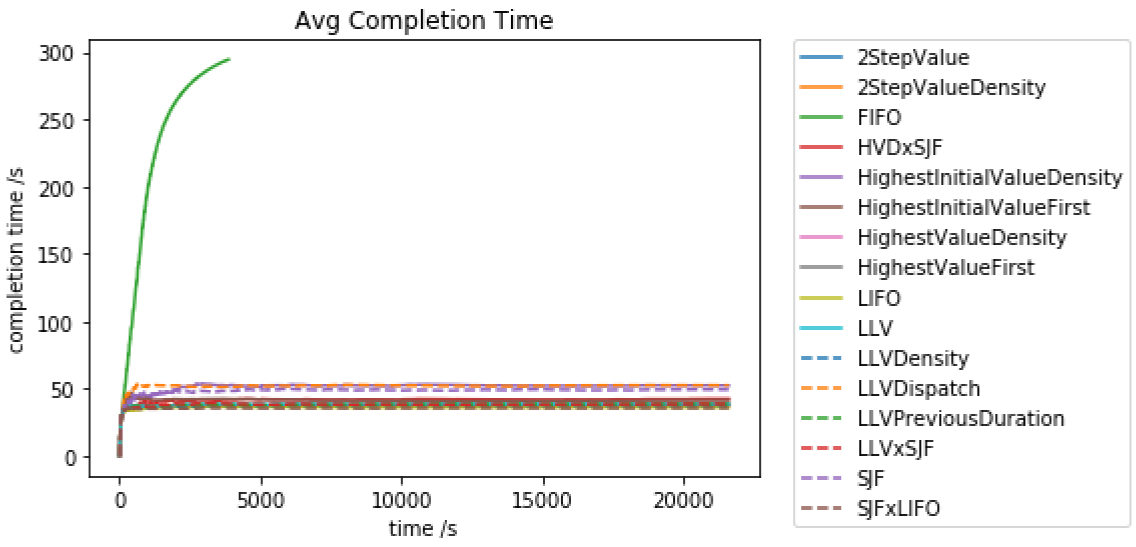

Figure 5.11: Average time to complete deliveries for all schedulers

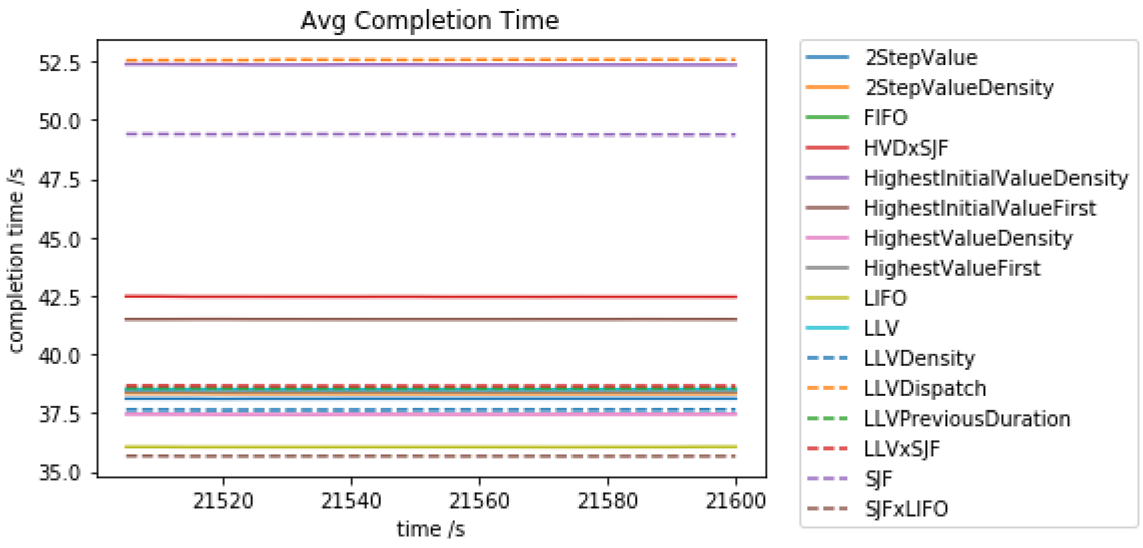

Figure 5.12: Final time slice of average time to complete deliveries for all schedulers

5.3.2 Time to Complete

The overall trends in completion time appear to be very similar to the trends in the average time to accept, just with values about 30 seconds larger, as shown in Figure 5.11. When zoomed in, as in Figure 5.12, it can be seen that the distribution of different completion times between schedulers is generally skewed towards lower times, compared to the previous metric. The best policy for minimising the average completion time is the SJF-LIFO splitwise scheduler, meaning this is the best algorithm for delivering packages as fast as possible to customers. However, the rank improvement from the average time to accept to the average time to complete suggests a smaller physical radius of deliveries are completed. The size of this radius will be explored more using the following metrics. The LLVDispatch, Highest Initial Value Density, and SJF are again the worst for carrying out fast deliveries, with HVD-SJF also displaying less than ideal results as before. Highest Initial Value First performs worse in terms of completion time than accept time, moving away from the group of similar results, likely because it makes no attempt to select shorter deliveries.

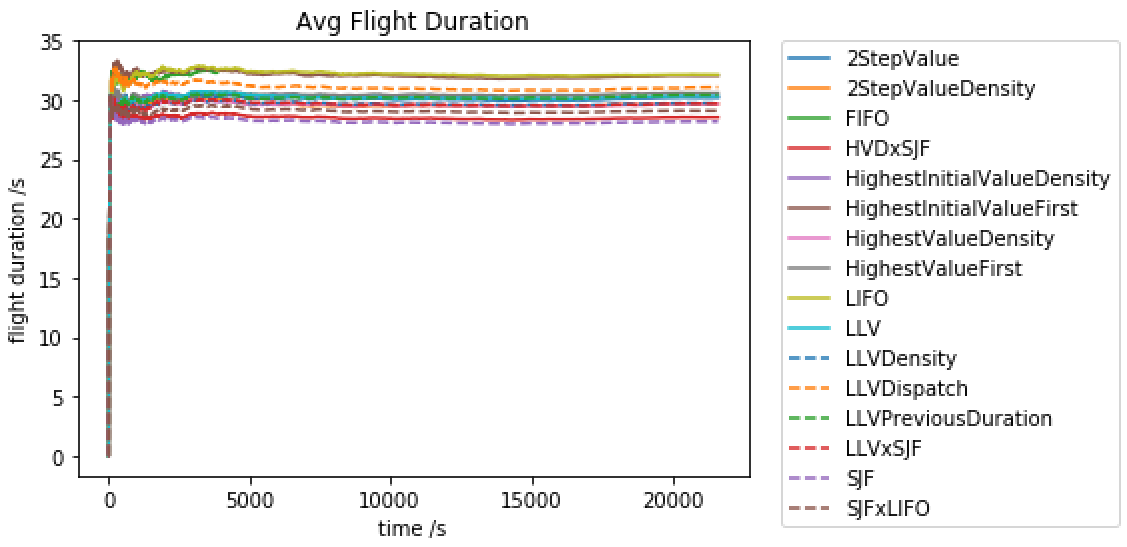

Figure 5.13: Average one way flight duration for all schedulers

5.3.3 Flight Duration

Figure 5.13 contains information about the average one way flight duration of each scheduling behaviour. It is observed that FIFO shows early signs of exhibiting one of the higher flight durations before its simulation was stopped, an indication that it allows further away customers to receive drone deliveries. Figure 5.14 displays the final time slice from Figure 5.13, so does not contain data for FIFO, but the performance of all other schedulers can be seen with more ease. Scheduling with SJF leads to the shortest flight times, approximately 28.2 seconds, and the SJF splitwise schedulers are also amongst the lowest values. Value density based policies record the next lowest values overall, with pure value schedulers displaying times roughly 0.5 to 0.75 seconds greater than these. LLVDispatch further results in longer flight durations, likely due to its relaxed method of calculating the negative impact of fulfilling one delivery on the value of the others in the queue, allowing for longer deliveries to be executed where another drone return will return back before their completion. Finally, the longest flight times are found when simulating the use of scheduling algorithms that are non-discriminate to distance or flight time: LIFO, Highest Initial Value First, and presumably FIFO. This is because no implemented schedulers view longer deliveries as a positive attribute, so those that involve the distance or flight duration in their decision making will inevitably reduce it in some way. This is a drawback in terms of reaching a larger area of customers per fulfilment centre but, since none of the flight duration neutral schedulers are particularly effective at generating revenue, appears to be essential in order to increase the profitability of the drone courier network.

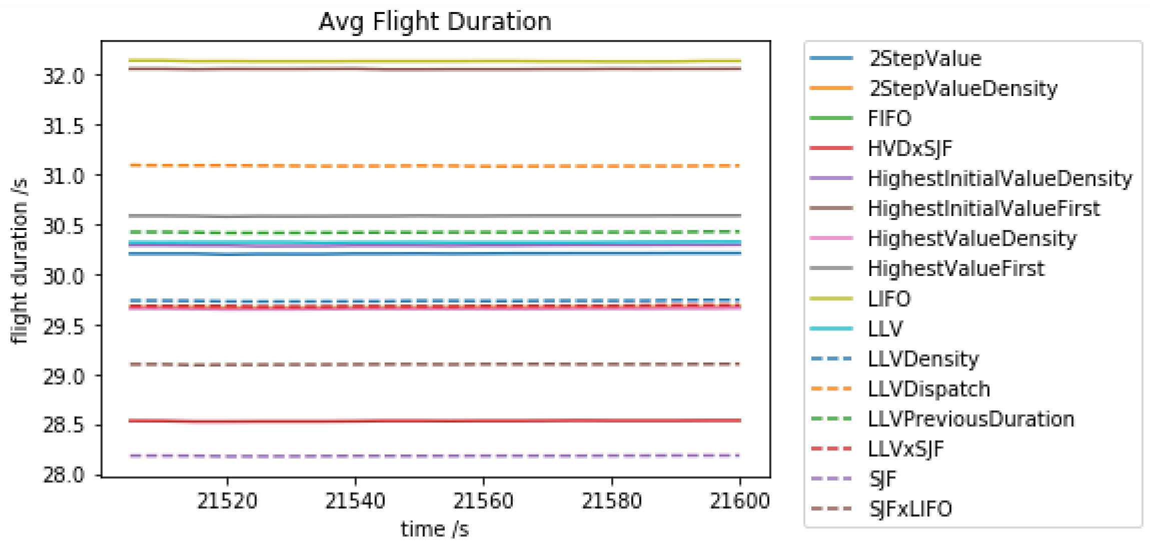

Figure 5.14: Final time slice of average one way flight duration for all schedulers

5.4 Spatial Distribution of Accepted Deliveries

By logging the location of every accepted delivery, a heatmap can be created to provide a visual representation of the spatial distribution of the delivery requests that are served by the drone couriers. The value to the delivery service provider from this is similar to that of the Flight Duration metric - information is provided about the geographical fairness of the scheduling algorithms, and this can also be used to inform the company as to where they should place their fulfilment centres throughout a city.

The advantage of the spatial distribution heatmap over just the Flight Duration metric is the level of detail and amount of information available. Whilst the average Flight Duration gives an indication of the spread of accepted deliveries, assumptions about the shape of the distribution have to be made for the value to be meaningful, alongside knowledge of the average building height and flight altitude. However, a heatmap displays exact information about the location of every accepted delivery, showing precisely which customers are served and how often. Smoothing the results by convolving the raw heatmap image with a Gaussian filter (a sigma of 5 is chosen here) allows the courier company to estimate the likelihood of a customer's delivery being fulfilled, given their position in the city. This could then be used to find a controller building layout that results in even coverage of drone deliveries across a city, even accounting for the positions of no fly zones. Since the raw heatmaps are still valuable due to containing complete information, they can be found in Appendix A.1.

From the results of the metrics previously discussed, certain algorithms are considered unlikely to see use in a drone courier network. The heatmaps of the accepted delivery locations for these scheduling policies are omitted. They are, however, included in Appendix A.2.

5.4.1 Two Step Value Density

Figure 5.15 shows that the Two Step Value Density scheduler is not geographically fair to the customers of the drone courier company. The generally darker centre (other than the white patches marking no fly zones) indicates a preference towards customers who live close to the controller building. This is expected from a density based scheduler since a lower total time invested to carry out a delivery results in an increase in value density, assuming the value earned from the delivery is the same.

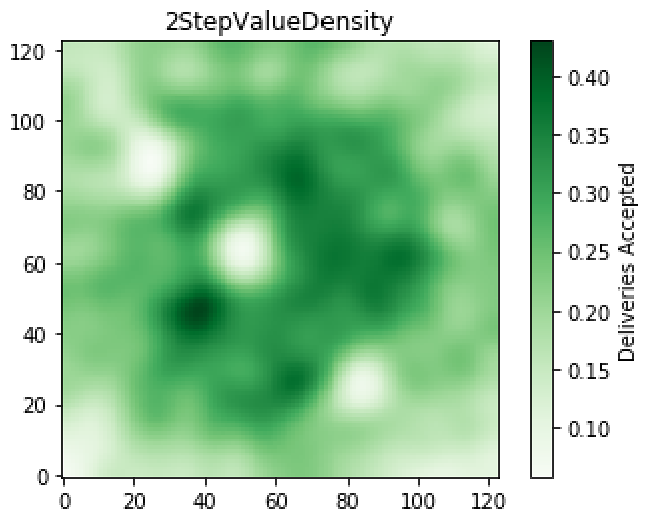

Figure 5.15: Smoothed spatial distribution of accepted deliveries when using the Two Step Value Density scheduler

5.4.2 Highest Value Density

The heatmap of accepted delivery request locations for the Highest Value Density policy, Figure 5.16, is very similar to that of the Two Step Value Density algorithm, showcasing a moderate level of unfairness. It is observed that, at least over the course of a six hour simulation, Highest Value Density is very slightly less fair.

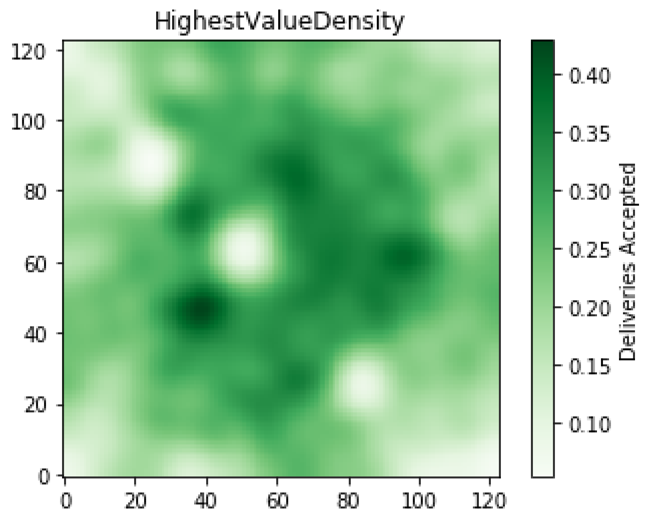

Figure 5.16: Smoothed spatial distribution of accepted deliveries when using the Highest Value Density scheduler

5.4.3 Highest Value First

In contrast, Figure 5.17 shows a nearly uniform spatial distribution, suggesting Highest Value First is close to offering an equal frequency of service to all customers within the city. It seems to show a slight bias towards drones delivering packages close to the fulfilment centre, which is explained by the fact that the monotonically decreasing time value functions cause the profit earned from a delivery to decrease even after a drone is dispatched. As a consequence, given the choice between a number of delivery requests that are identical other than flight duration, Highest Value First would choose the delivery that can be completed in the shortest time since this is the one with the highest attainable value.

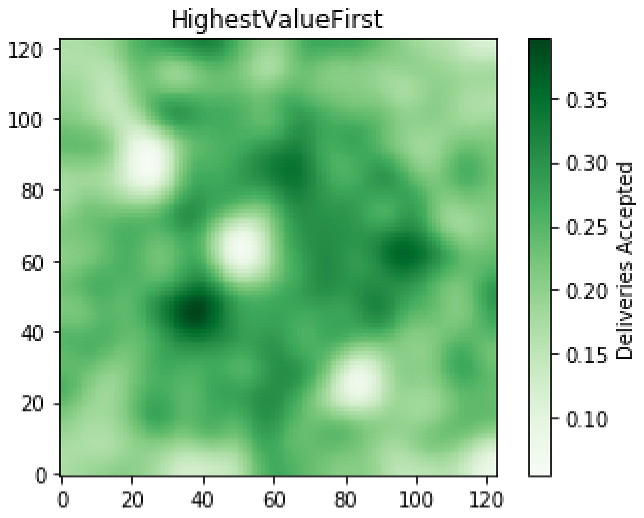

Figure 5.17: Smoothed spatial distribution of accepted deliveries when using the Highest Value First scheduler

5.4.4 HVD-SJF

By combining the Highest Value Density scheduler, which has been shown to be geographically unfair, with the behaviour of Shortest Job First, which is the most unfair due to picking the closest jobs, HVD-SJF can only exhibit unfair behaviour in this regard. Figure 5.18 shows a clear example of a scheduler that has an extreme preference for fulfilling orders close to the controller building. Customers on the extremities of the city received little to no service from the drone couriers, and would have to wait for delivery by a conventional method.

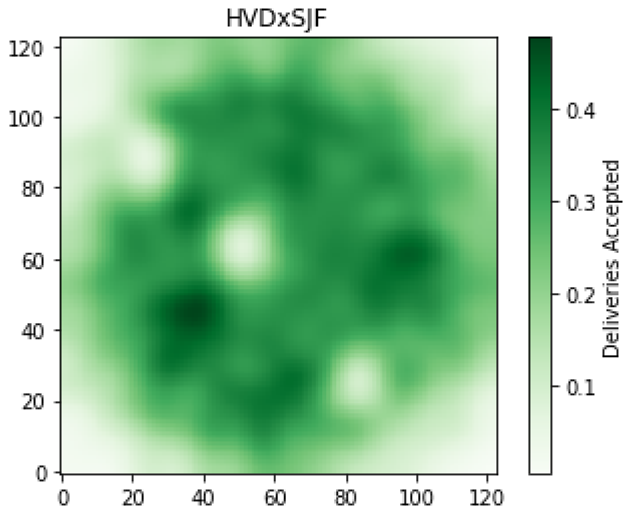

Figure 5.18: Smoothed spatial distribution of accepted deliveries when using the HVD-SJF splitwise scheduler

5.4.5 LLV

Surprisingly, the LLV policy is almost as fair as Highest Value First. This is somewhat expected due to not being a density based, or splitwise, scheduler. Therefore, not a lot of emphasis is placed on minimising the flight duration of the deliveries carried out. However, LLV considers the length of time a drone is out for delivery when computing the negative impact on the attainable value of the other delivery requests, but this is not majorly reflected in the spatial distribution of accepted deliveries (Figure 5.19).

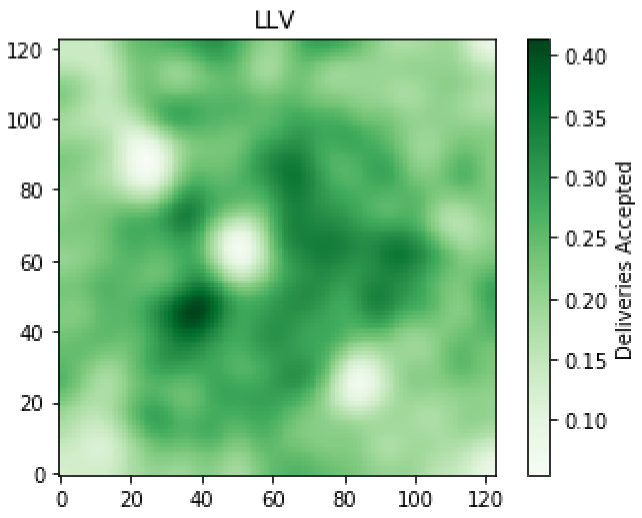

Figure 5.19: Smoothed spatial distribution of accepted deliveries when using the LLV scheduler

5.4.6 LLVDensity

Least Lost Value Density's heatmap, Figure 5.20, portrays the expected results. Using the LLV-Density scheduler results in a drone courier network that is less fair than if LLV were used, but prioritises short deliveries to a similar degree as other value density based policies.

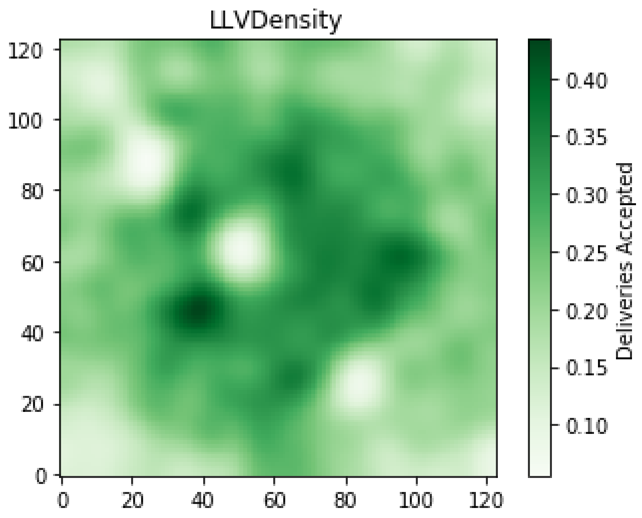

Figure 5.20: Smoothed spatial distribution of accepted deliveries when using the Least Lost Value Density scheduler

5.4.7 LLV-SJF

Amazingly, the most profitable of the implemented schedulers, a splitwise combination of LLV and SJF, manages to achieve this level of total profit without biasing customers in close proximity to the fulfilment centre to a massive extent - only slightly more than the LLVDensity algorithm. This does admittedly make it the least geographically fair of the schedulers discussed so far in Section 5.4 aside from HVD-SJF, but customers near the corners of the city may still be chosen for drone delivery at times, as seen in Figure 5.21.

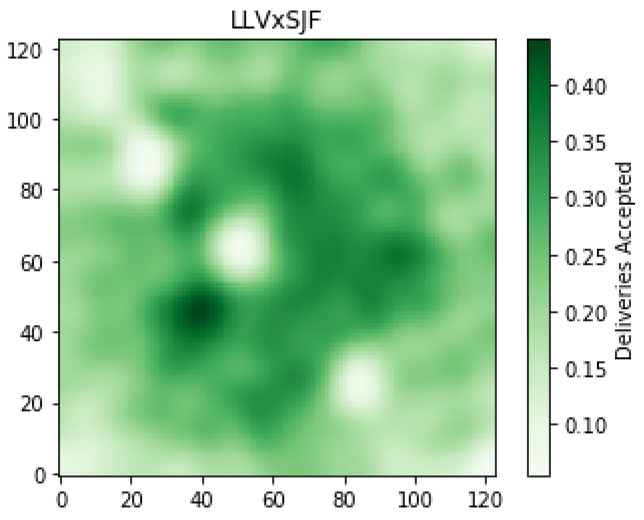

Figure 5.21: Smoothed spatial distribution of accepted deliveries when using the LLV-SJF splitwise scheduler

5.4.8 SJF-LIFO

The final scheduling policy whose accepted delivery spatial distribution will be discussed is the SJF-LIFO splitwise scheduler. Whilst not being the most successful in terms of extracting revenue from deliveries, the behaviour of this scheduler resulted in the shortest average completion time of customers' delivery requests. Figure 5.14 makes it apparent that a substantial part of the reason for this is that the shortest deliveries are heavily favoured, meaning further away potential customers are not provided with a consistent service. Only HVD-SJF is more unfair in this sense from Section 5.4, moving LLV-SJF up into sixth out of the eight highlighted algorithms.

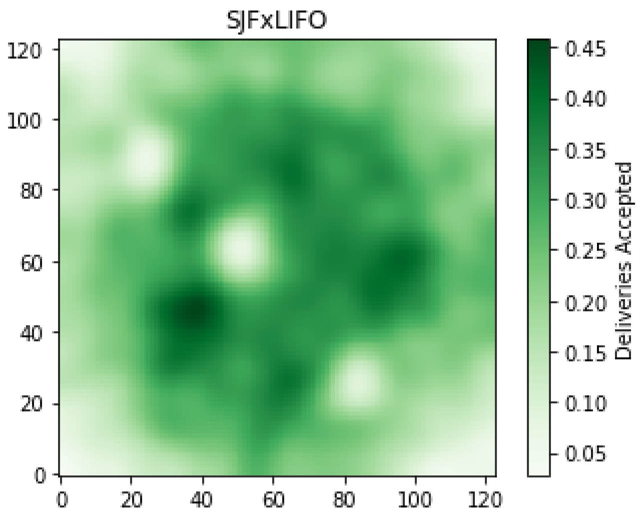

Figure 5.22: Smoothed spatial distribution of accepted deliveries when using the SJF-LIFO splitwise scheduler

5.5 Summary of Schedulers

Using all of the above metrics, a table can be produced that highlights the strengths and weaknesses of each scheduling policy benchmarked in this research (Table 5.1). Often, the schedulers that extract the highest profits are the least fair to customers, which may harm customer satisfaction. Drone delivery courier companies can use this resource to select a scheduler that works for their specific needs and priorities, since there is no perfect solution.

Numerical entries in the table represent the percentage of the highest ranking scheduler's score that was achieved after the 6 hour simulation, so a high score is preferable when considering profit and the number of deliveries complete, whereas a low score is best for the two timing columns. Spatial fairness is a qualitative description based on observation of the heatmaps representing the spatial distribution of accepted deliveries for each scheduling policy. Since FIFO was only run for one hour, the number of deliveries completed cannot be compared in this case.

Scheduler	Profit	#Complete	Accept Time	Complete Time	Spatial Fairness
2StepValue	99.0	93.3	201.5	106.9	Neutral
2StepValueDensity	99.4	95.0	220.0	107.5	Poor
FIFO	0.4	N/A	6685.3	826.1	Best
HVD-SJF	96.9	98.8	354.0	119.1	Terrible
HVDinitial	87.6	93.0	562.9	146.8	Neutral
HVFinitial	85.7	87.7	240.3	116.4	Best
HVD	99.5	95.0	198.6	105.0	Poor
HVF	97.9	92.1	199.0	107.7	Good
LIFO	81.0	87.8	100	101.1	Best
LLV	98.5	93.0	208.5	108.0	Neutral
LLVDensity	98.8	94.8	200.9	105.5	Poor
LLVDispatch	77.6	90.7	547.4	147.5	Good
LLVPreviousDuration	98.5	92.7	207.0	108.1	Neutral
LLV-SJF	100	95.0	228.0	108.4	Poor
SJF	82.2	100	540.4	138.5	Worst
SJF-LIFO	90.4	96.9	166.9	100	Bad

Table 5.1: Summary of performance of each scheduling algorithm

Chapter 6

Conclusion

6.1 Summary

The goal of this project was to devise different scheduling algorithms for use by a drone delivery company in order to optimise revenue, whilst remaining cautious of the potential negative impact this might have on customer satisfaction. It was apparent that this would require a simulation of a drone courier network to be developed so key metrics alluding to the behaviour of each scheduling policy could be recorded and used to evaluate their performance. Ideally, a new scheduler would be created that beats the existing LLV scheduler in terms of profitability, without excessively sacrificing the satisfaction of customers.

These goals have been met, with a simulation built using the Unity engine that enables drone courier service companies to test schedulers without investing the large quantities of money and resources required to carry out real world trials. The Unity editor's interface also makes it simple to change the settings of the simulated environment so extreme or predicted future scenarios can be played out, keeping drone delivery companies prepared for what might happen.

Additionally, sixteen schedulers were implemented, including three that were tested in previous work as benchmarks (LLV, SJF, and FIFO). Of these, some were found to not be particularly effective, but many proved strong at least in some regard. From the evaluation, it was concluded that value density based schedulers appear to be the most profitable in a drone delivery network in a general sense, and not one, but five, scheduling policies were devised that outperform LLV in terms of generating a large total profit. All of these also carried out a higher volume of deliveries than LLV, a bonus in terms of the service quality. Clearly, this surpasses the initial aim of the project. In order to not neglect the quality of the service provided, and thus the customer satisfaction, statistics were collated about the fairness and delivery-speed of every scheduler. Only one of the five algorithms that were more profitable than LLV caused customers to wait longer, on average, to receive their parcels than LLV does. Unfortunately, this was LLV-SJF, the most profitable of all, so there was no one clear best delivery scheduler. Therefore, a table was produced to assist companies with their choice of algorithm.

6.2 Opportunities for Future Research

Despite meeting the goals for this project, there is still room for improvement to the simulation and further research into drone delivery scheduling algorithms.

6.2.1 Drone Battery Level

In reality, drones have a limited battery life before they have to be recharged. This is omitted by the simulation implemented in this research. The naive way of coping with a limited battery life is to allow a drone's battery level to continuously reduce until it does not have the range to carry out a delivery assigned by the controller, at which point the battery is charged until full. However, considering the battery life and recharge time in the scheduling behaviour could result in improvements to the service quality provided by the courier company, and the revenue generated. It may be found that there are cases where a usually sub-optimal delivery request should be executed by a drone that is low on battery if this delivery request has a sufficiently short flight duration such that the drone can carry it out before recharging. Also, it is possible that there are times it would be best for a drone to wait and charge even when not running low on battery in order to perform more deliveries back to back at a later time without having to recharge then.

6.2.2 Stochastic Flight Time

Most of the schedulers trialled in the context of drone delivery rely on accurately predicting the time taken to deliver a package to any given location in a city. Due to being developed initially as a big data tool, LLV is an exception to this, having been designed to take a duration distribution as part of its input. However, this simulation does not introduce any variation to flight times, so the implementation instead uses the exact value. When tested in a previous study [6] there was a small amount of randomness in the flight durations as a result of the drones adapting their paths to avoid one another, but the LLV implementation still did not consider a duration distribution, instead using the straight line distance from the controller to the destination as an estimate of the flight path distance with a constant drone speed.

There are many factors that mean the time a drone is out on a delivery job is actually highly stochastic in the real world, including weather conditions, aerial obstacles, and, depending on the method used to allow customers to collect their parcels, how long it takes for the package to be retrieved from the drone when at the destination. Including this in a simulation could be a good way of understanding the effect this would have on the current schedulers, and to develop new schedulers that are more robust to this variation if need be.

6.2.3 Testing Under Differing Load

The configuration used for the simulations run resulted in 5400 delivery requests being made over the 6 hour period. SJF, the algorithm that proved best at maximising the volume of deliveries served, managed to fulfil 3827 of these, around 70%. This can be used as a measure of workload. It is likely that the relative performance of the different schedulers would vary with the amount of load placed on the drone courier network, so it makes sense to test under different conditions. Even if a company knows what frequency of delivery requests to expect, it will vary throughout the day, so a different scheduling algorithm may be best at night than at peak hours. Based on intuition, greedy schedulers should exhibit a better relative performance when under high load, whereas schedulers that attempt to minimise the negative impact of their decisions, such as LLV and its variants, would be superior when the frequency of orders is lower. However, this is only speculative without evidence and the significance is unknown without further research.

6.2.4 Experimentation with Splitwise Schedulers

The concept of a splitwise scheduler has been shown to be effective in this study, with LLV-SJF extracting the most profit overall. The version of this splitwise scheduling algorithm that was benchmarked utilised equal weights to combine its component's priorities, but there is nothing to suggest this is the best balance. Thus, it would be prudent to further investigate the effect of modifying these weights in hope of creating an even more profitable policy.

Furthermore, the weighted sum of priorities is not the only conceivable way to combine two scheduling algorithms. Other approaches, such as running N schedulers concurrently on the same controller and partitioning the drones between them, or using one policy to shortlist a number of potential delivery requests and using another to pick from them, may prove to be superior.

Finally, there are many more possible combinations of splitwise schedulers that can be made which have not been tried out. This is especially true when more than 2 algorithms are mixed.

6.2.5 Multiple Controller Buildings

Another interesting avenue of research is to simulate a scenario where there is more than one controller building. This opens up the opportunity for drones to be shared between fulfilment centres to compensate for imbalances in workload across a city which could lead to some exciting and complex new schedulers. Potentially, the individual stock level of each fulfilment centre could also be modelled, creating situations where a delivery has to be carried out from a further away controller building if the closest is out of stock in a certain item, which results in even more interesting interactions between controllers.

Bibliography

[1] Amazon.com Inc., "Fulfilment by amazon fees." https://m.media-amazon.com/images/G/02/FBA_Files/2019/190529-FBA-Rate-Card-UK.pdf?ld=NSGoogle_null, June 2019. Accessed: 16/06/19.

[2] Wonsang Yoo, Eun Yu, Jaemin Jung, "Drone delivery: Factors affecting the public's attitude and intention to adopt." https://www.sciencedirect.com/science/article/pii/S0736585318300388#b0250, 2018. Accessed: 16/06/19.

[3] Amazon.com Inc., "Amazon prime air." https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011, 2019. Accessed: 16/06/19.

[4] DHL, "Dhl parcelopter." https://discover.dhl.com/business/business-ethics/parcelcopter-drone-technology, June 2018. Accessed: 16/06/19.

[5] Wing, "Wing." https://wing.com/, 2019. Accessed: 16/06/19.

[6] S. Seakhoa-King, P. Balaji, N. T. Alvarez, and W. J. Knottenbelt, "Revenue-driven scheduling in drone delivery networks with time-sensitive service level agreements," in *Proceedings of the 12th EAI International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS 2019, pp. 183–186, ACM, 2019. http://doi.acm.org/10.1145/3306309.3306339. Accessed: 16/06/19.

[7] Martin Joerss, Florian Neuhaus, Christoph Klink, Florian Mann, "Parcel delivery the future of last mile." https://www.mckinsey.com/~/media/mckinsey/industries/travel%20transport%20and%20logistics/our%20insights/how%20customer%20demands%20are%20reshaping%20last%20mile%20delivery/parcel_delivery_the_future_of_last_mile.ashx, 2016. Accessed: 16/06/19.

[8] Starship Technologies, "Starship company." https://www.starship.xyz/company/, 2019. Accessed: 16/06/19.

[9] Jade Perry, JWT Intelligence, "Droid delivery." https://www.jwtintelligence.com/2016/07/droid-delivery/, July 2016. Accessed: 16/06/19.

[10] Domino's Pizza Group, "Dom autonomous delivery vehicle." https://www.dominos.com.au/inside-dominos/technology/dru, 2019. Accessed: 16/06/19.

[11] Ben Fox Rubin, CNET, "Amazon's scout robots: That's no cooler, that's your prime delivery." https://www.cnet.com/news/amazons-scout-robots-thats-no-cooler-thats-your-prime-delivery/, June 2019. Accessed: 16/06/19.

[12] Aethon, "Mobile robots for healthcare." https://aethon.com/mobile-robots-for-healthcare/, 2019. Accessed: 16/06/19.

[13] Deliveroo, "Deliveroo: Takeaways delivered from restaurants near you." https://deliveroo.co.uk/, 2019. Accessed: 16/06/19.

[14] Wing, "Wing finland." https://wing.com/finland/, 2019. Accessed: 08/02/19.

[15] Amazon.com Inc., "Amazon prime air." https://youtu.be/98BIu9dpwHU, 2013. Accessed: 16/06/19.

[16] Lisa Lacy, ADWEEK, “Amazon still needs regulatory approval for drone deliveries.” https://www.adweek.com/digital/amazon-still-needs-regulatory-approval-for-drone-deliveries/, June 2019. Accessed: 16/06/19.

[17] Natashah Hitti, dezeen, “Amazon prime air drone to deliver purchases by drone "within months".” https://www.dezeen.com/2019/06/06/amazon-prime-air-drone-news/, June 2019. Accessed: 16/06/19.

[18] Wing, “Wing x.” https://x.company/projects/wing/, 2019. Accessed: 16/06/19.

[19] Wing, “Wing australia.” https://wing.com/australia/, 2019. Accessed: 08/02/19.

[20] BBC News, “Google wing launches first home delivery drone service.” https://www.bbc.co.uk/news/technology-47880288?CMP=share_btn_me, April 2019. Accessed: 16/06/19.

[21] Wing, “Canberra - australia - wing.” https://wing.com/australia/canberra/, 2019. Accessed: 16/06/19.

[22] Wing, “Helsinki - australia - wing.” https://wing.com/finland/helsinki/, 2019. Accessed: 16/06/19.

[23] BBC News, “Google wing drones approved for us home deliveries.” https://www.bbc.co.uk/news/technology-48029396?CMP=share_btn_me, April 2019. Accessed: 16/06/19.

[24] Wing Medium, “Wing becomes first certified air carrier for drones in the us.” https://medium.com/wing-aviation/wing-becomes-first-certified-air-carrier-for-drones-in-the-us-43401883f20b, April 2019. Accessed: 16/06/19.

[25] Amazon.com Inc., “Determining safe access with a bestequipped, best-served model for small unmanned aircraft systems.” https://www.documentcloud.org/documents/2182312-amazon-determining-safe-access-with-a-best.html, 2015. Accessed: 16/06/19.

[26] Amazon.com Inc., “Revising the airspace model for the safe integration of small unmanned aircraft systems.” https://utm.arc.nasa.gov/docs/Amazon_Revising%20the%20Airspace%20Model%20for%20the%20Safe%20Integration%20of%20sUAS[6].pdf, 2015. Accessed: 16/06/19.

[27] Wing, “Wing utm platform.” https://wing.com/utm-platform/, 2019. Accessed: 08/02/19.

[28] Investopedia, “Time value of money - tvm.” https://www.investopedia.com/terms/t/timevalueofmoney.asp. Accessed: 16/06/19.

[29] John Bell, “Cpu scheduling.” https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/6_CPU_Scheduling.html, 2018. Accessed: 16/06/19.

[30] Unity, “Unity real-time development platform.” https://unity.com/, 2019. Accessed: 16/06/19.

[31] Unity, “Products - unity.” https://unity3d.com/unity, 2019. Accessed: 16/06/19.

[32] A. Nash, S. Koenig, and C. Tovey, “Lazy theta*: Any-angle path planning and path length analysis in 3d.,” in *Proceedings of the Third Annual Symposium on Combinatorial Search*, vol. 1, January 2010. https://www.researchgate.net/publication/220743619_Lazy_Theta_Any-Angle_Path_Planning_and_Path_Length_Analysis_in_3D. Accessed: 16/06/19.

[33] P. Hart, N. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968. http://dx.doi.org/10.1109/TSSC.1968.300136. Accessed: 16/06/19.

[34] A. Nash, K. Daniel, S. Koenig, and A. Feiner, "Theta*: Any-angle path planning on grids," in *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pp. 1177–1183, AAAI Press, 2007. http://dl.acm.org/citation.cfm?id=1619797.1619835. Accessed: 16/06/19.

[35] E. Jensen, D. Locke, and H. Tokuda, "A time-driven scheduling model for real-time operating systems," *RTSS*, vol. 85, pp. 112–122, February 2003. https://www.researchgate.net/publication/2570854_A_Time-Driven_Scheduling_Model_for_Real-Time_Operating_Systems. Accessed: 16/06/19.

Appendix A

Appendix

A.1 Raw Spatial Distribution Heatmaps of Accepted Deliveries

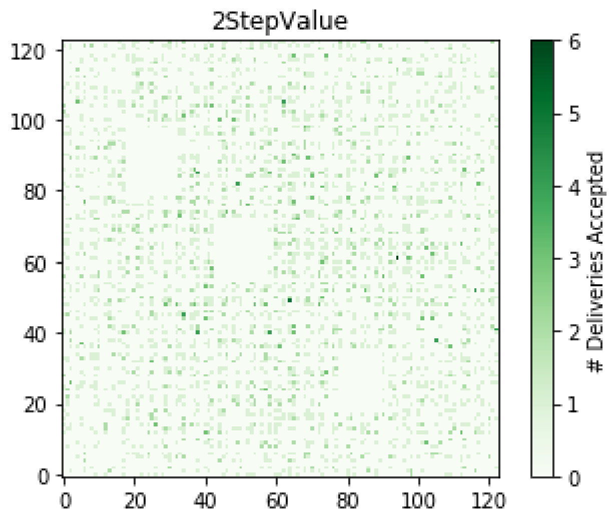

Figure A.1: Spatial distribution of accepted deliveries when using the Two Step Value scheduler

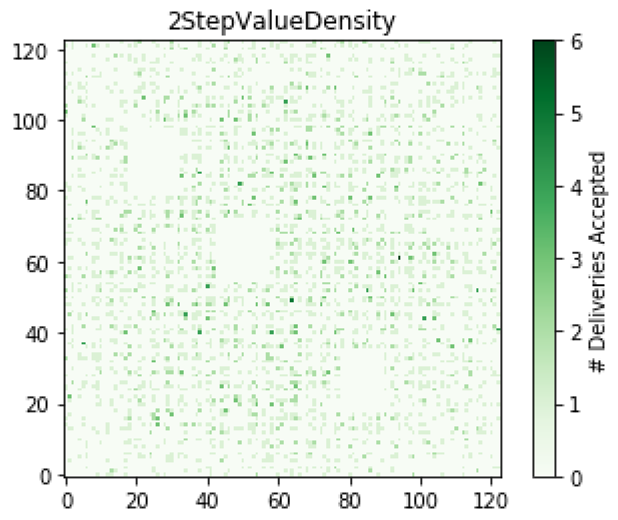

Figure A.2: Spatial distribution of accepted deliveries when using the Two Step Value Density scheduler

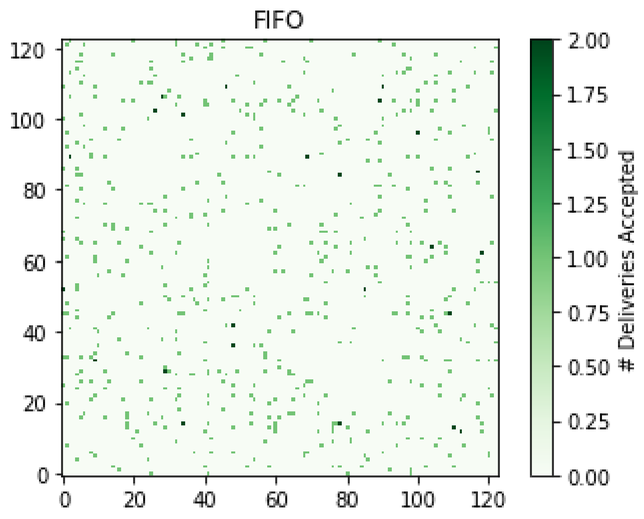

Figure A.3: Spatial distribution of accepted deliveries when using the FIFO scheduler

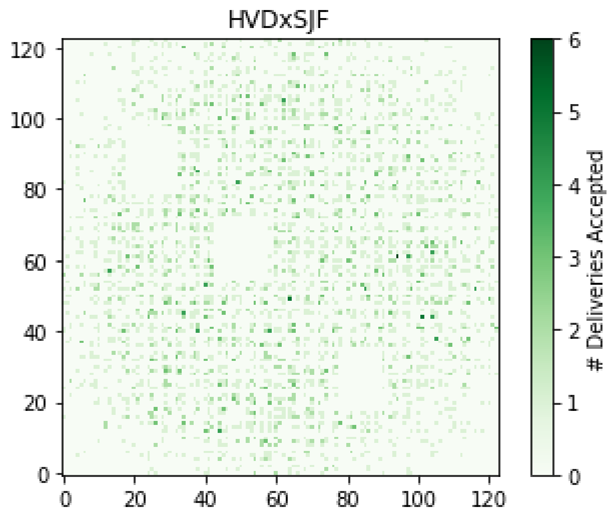

Figure A.4: Spatial distribution of accepted deliveries when using the HVD-SJF splitwise scheduler

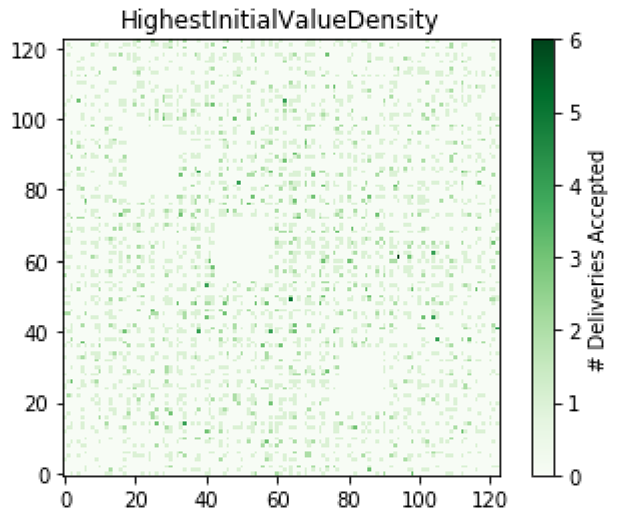

Figure A.5: Spatial distribution of accepted deliveries when using the Highest Initial Value Density scheduler

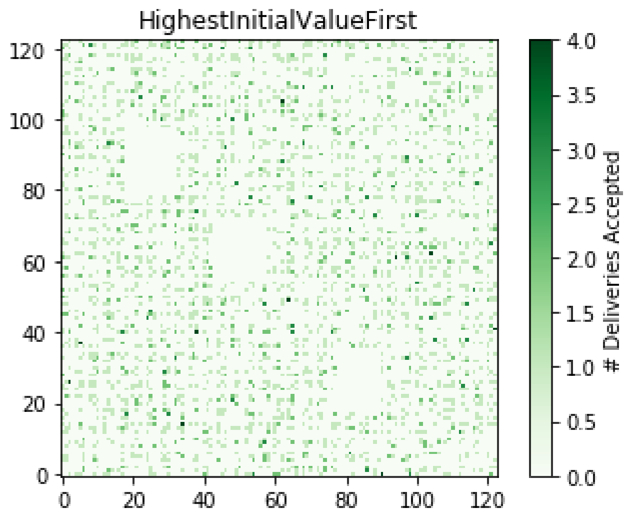

Figure A.6: Spatial distribution of accepted deliveries when using the Highest Initial Value First scheduler

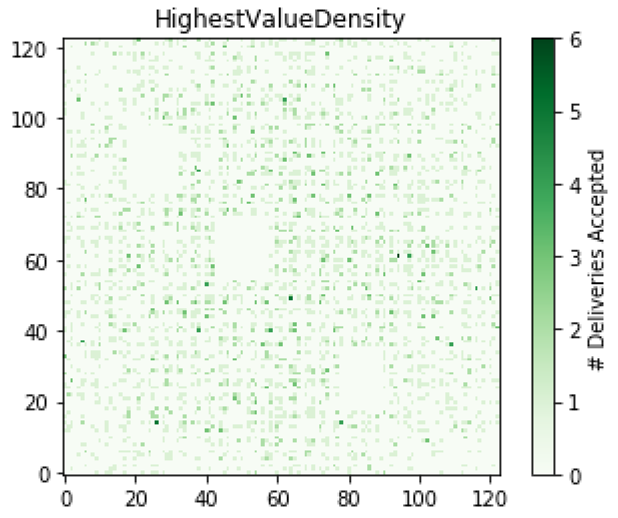

Figure A.7: Spatial distribution of accepted deliveries when using the Highest Value Density scheduler

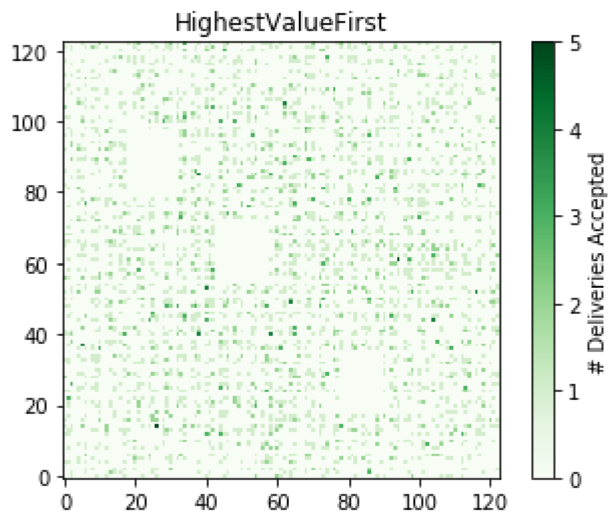

Figure A.8: Spatial distribution of accepted deliveries when using the Highest Value First scheduler

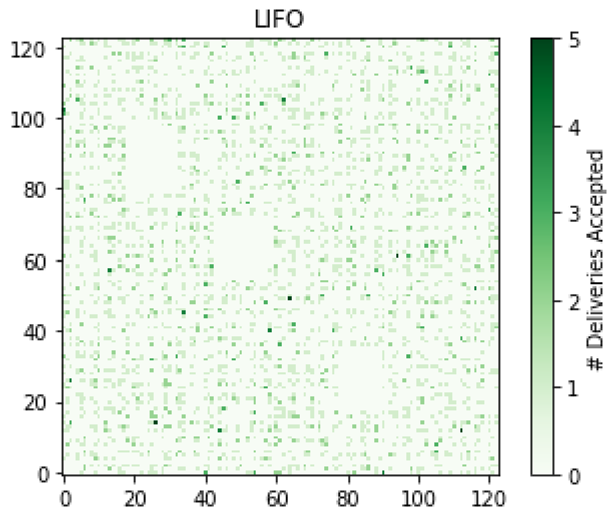

Figure A.9: Spatial distribution of accepted deliveries when using the LIFO scheduler

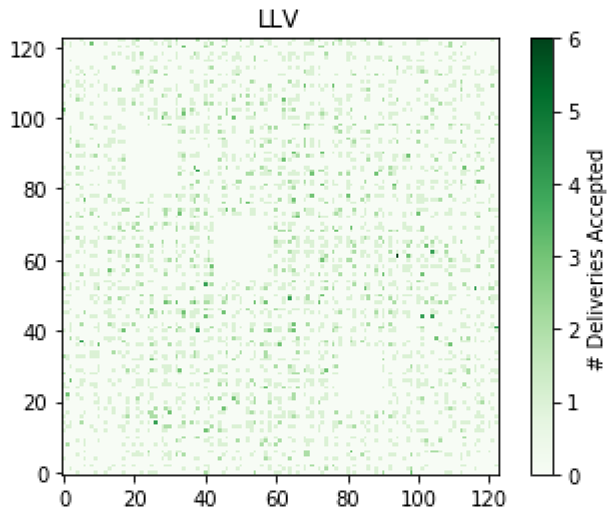

Figure A.10: Spatial distribution of accepted deliveries when using the LLV scheduler

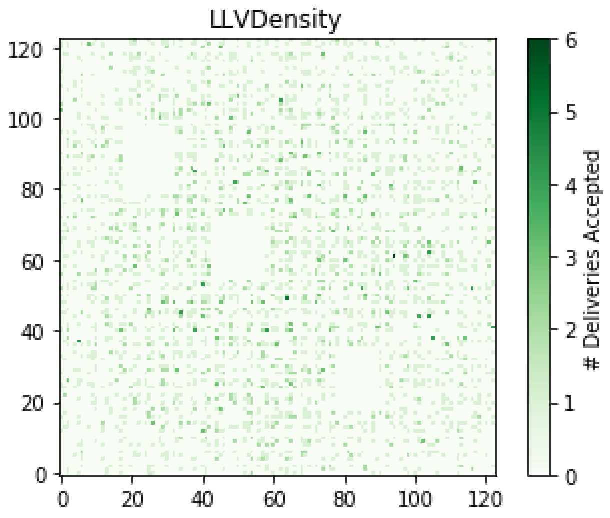

Figure A.11: Spatial distribution of accepted deliveries when using the Least Lost Value Density scheduler

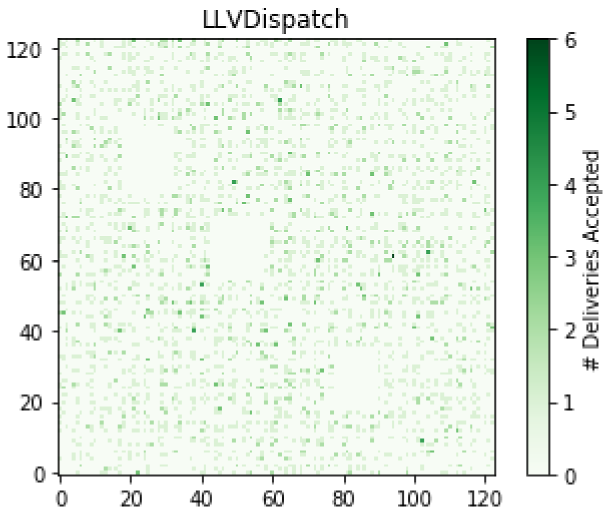

Figure A.12: Spatial distribution of accepted deliveries when using the LLV Dispatch Time scheduler

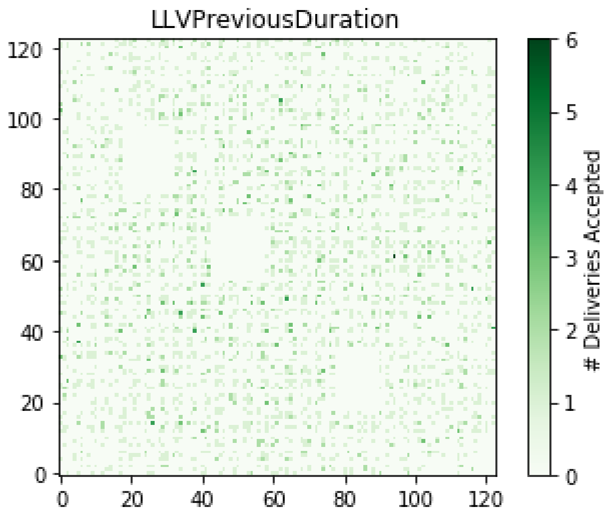

Figure A.13: Spatial distribution of accepted deliveries when using the LLV Previous Duration scheduler

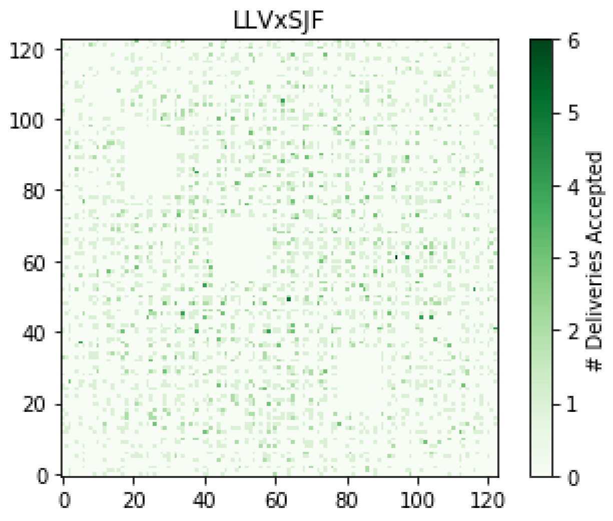

Figure A.14: Spatial distribution of accepted deliveries when using the LLV-SJF splitwise scheduler

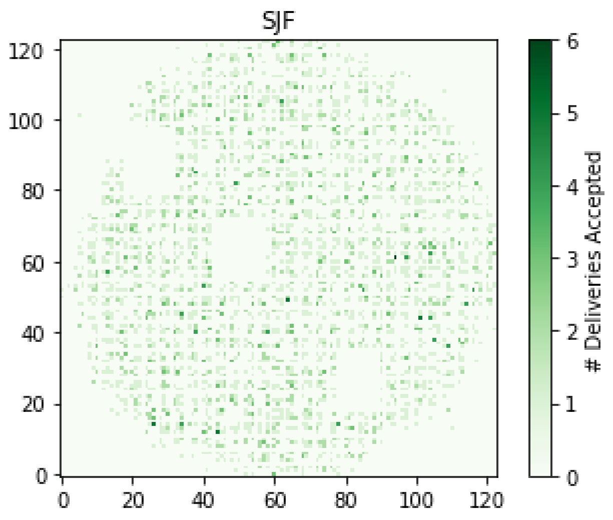

Figure A.15: Spatial distribution of accepted deliveries when using the SJF scheduler

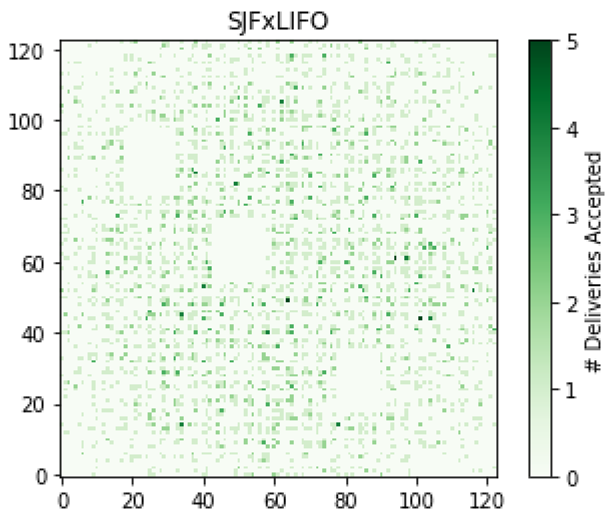

Figure A.16: Spatial distribution of accepted deliveries when using the SJF-LIFO splitwise scheduler

A.2 Smoothed Spatial Distribution Heatmaps of Accepted Deliveries

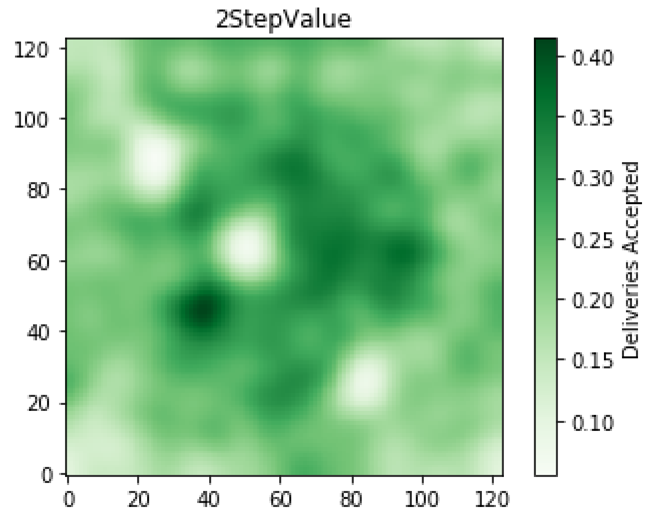

Figure A.17: Smoothed spatial distribution of accepted deliveries when using the Two Step Value scheduler

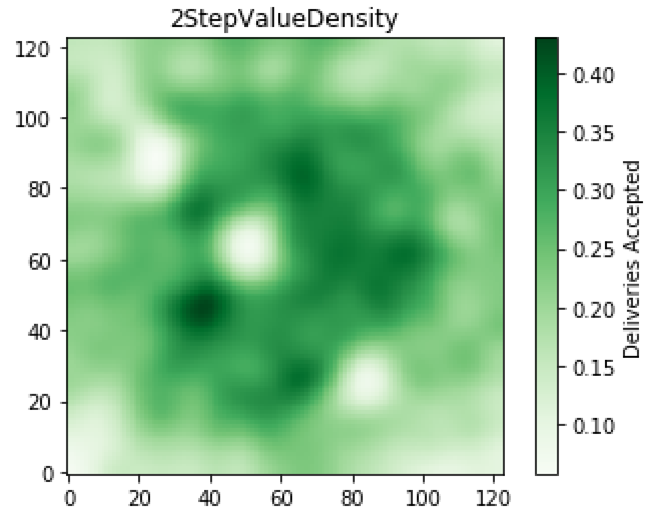

Figure A.18: Smoothed spatial distribution of accepted deliveries when using the Two Step Value Density scheduler

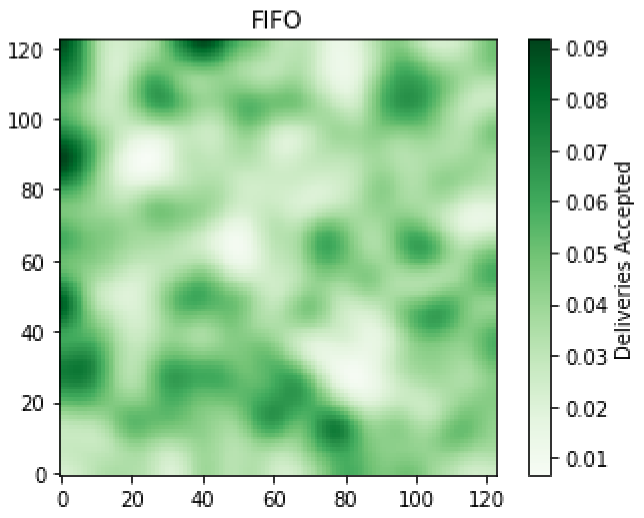

Figure A.19: Smoothed spatial distribution of accepted deliveries when using the FIFO scheduler

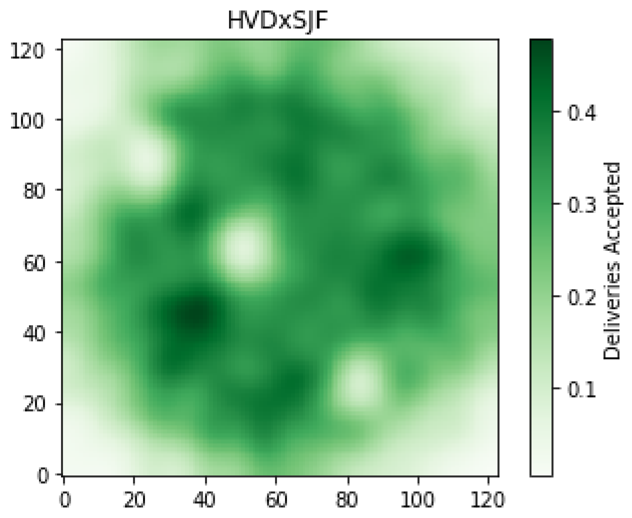

Figure A.20: Smoothed spatial distribution of accepted deliveries when using the HVD-SJF split-wise scheduler

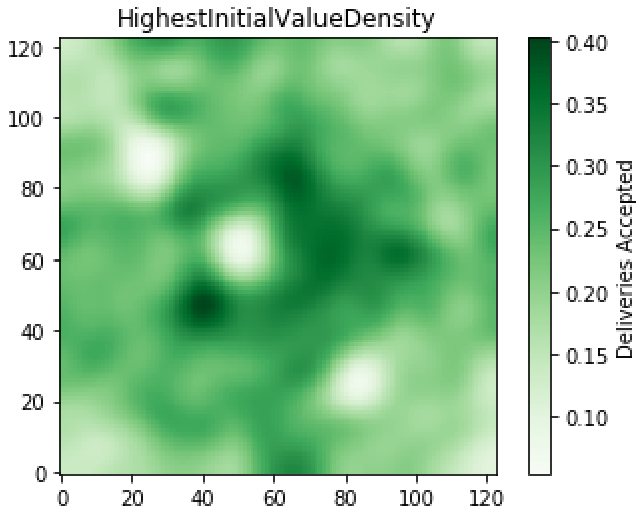

Figure A.21: Smoothed spatial distribution of accepted deliveries when using the Highest Initial Value Density scheduler

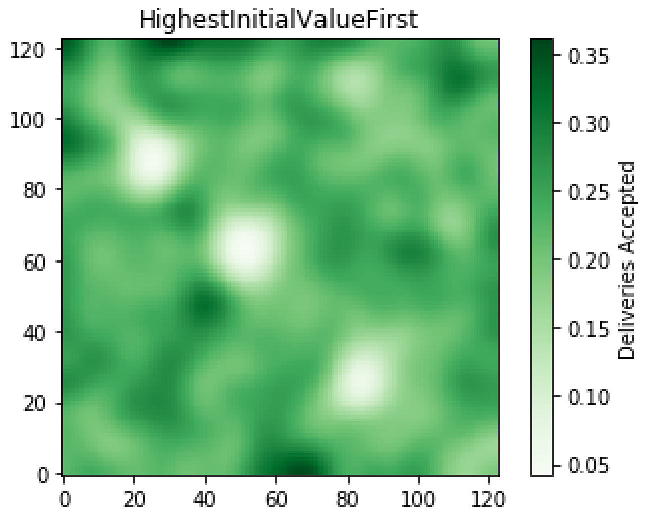

Figure A.22: Smoothed spatial distribution of accepted deliveries when using the Highest Initial Value First scheduler

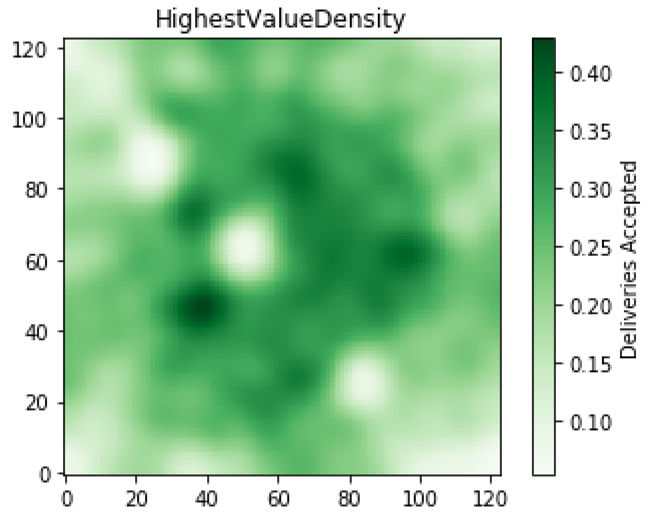

Figure A.23: Smoothed spatial distribution of accepted deliveries when using the Highest Value Density scheduler

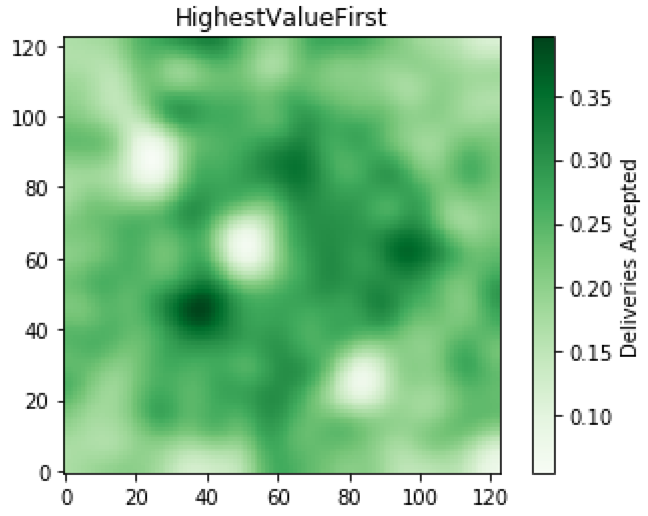

Figure A.24: Smoothed spatial distribution of accepted deliveries when using the Highest Value First scheduler

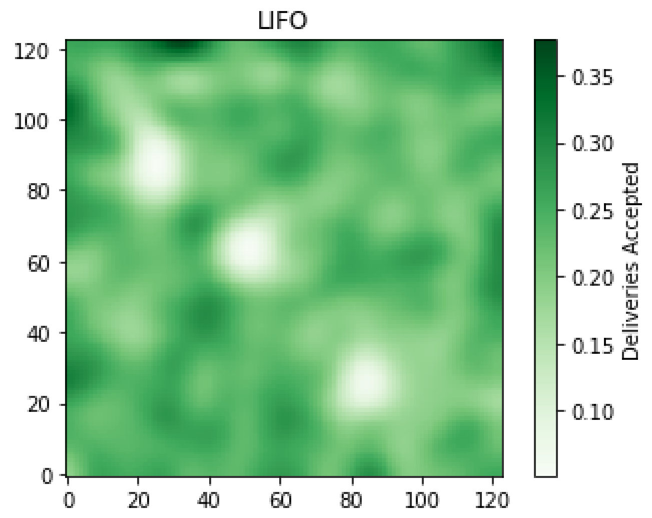

Figure A.25: Smoothed spatial distribution of accepted deliveries when using the LIFO scheduler

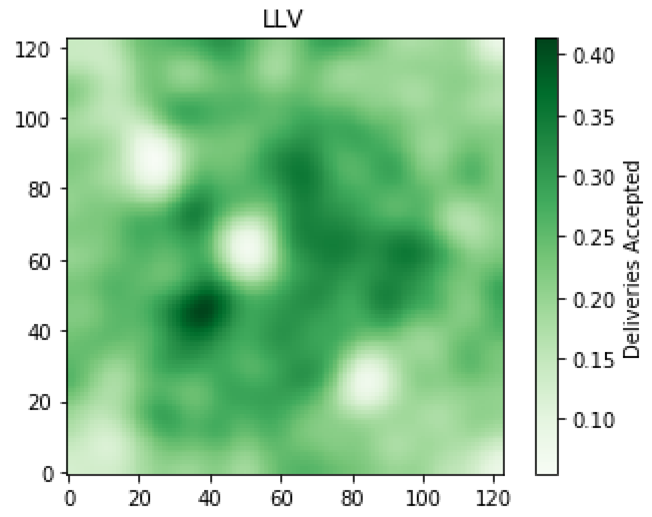

Figure A.26: Smoothed spatial distribution of accepted deliveries when using the LLV scheduler

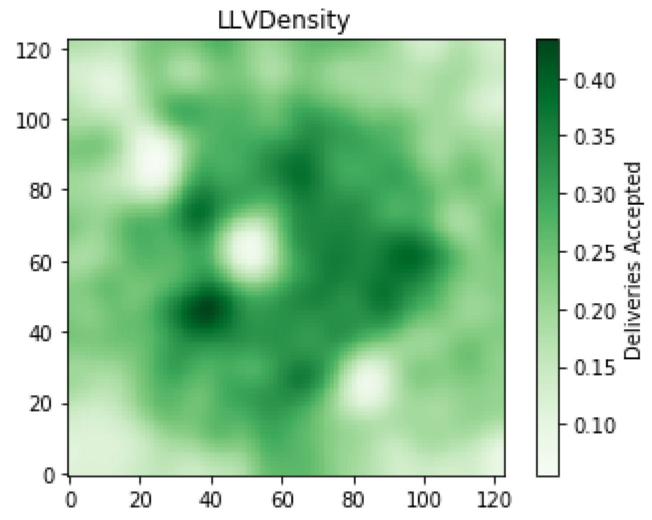

Figure A.27: Smoothed spatial distribution of accepted deliveries when using the Least Lost Value Density scheduler

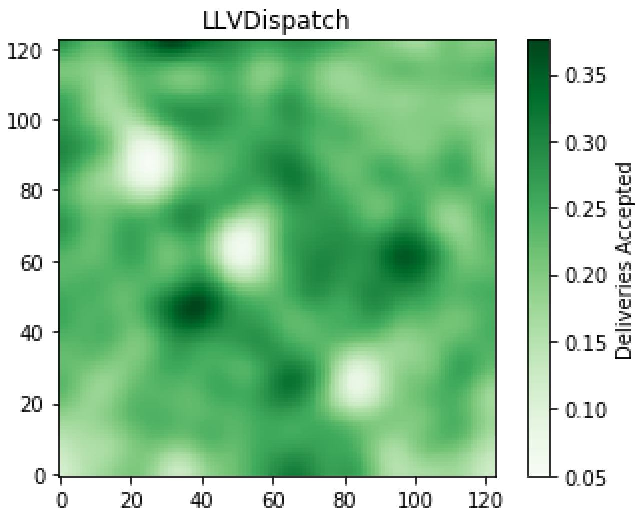

Figure A.28: Smoothed spatial distribution of accepted deliveries when using the LLV Dispatch Time scheduler

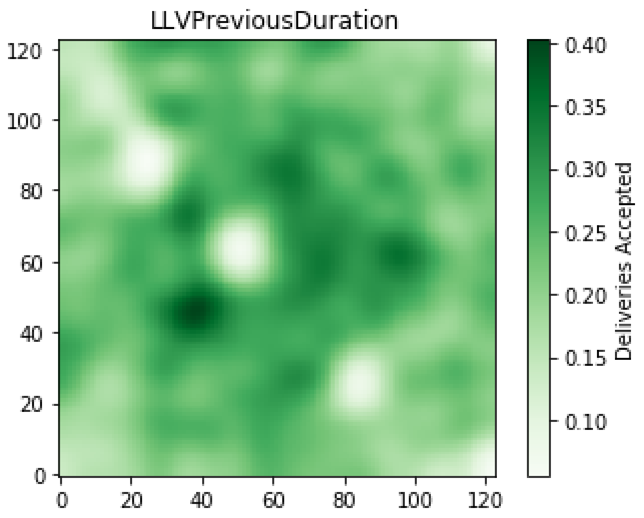

Figure A.29: Smoothed spatial distribution of accepted deliveries when using the LLV Previous Duration scheduler

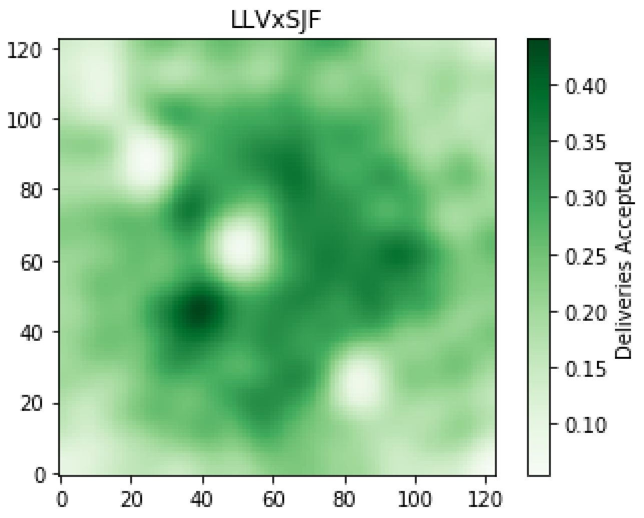

Figure A.30: Smoothed spatial distribution of accepted deliveries when using the LLV-SJF splitwise scheduler

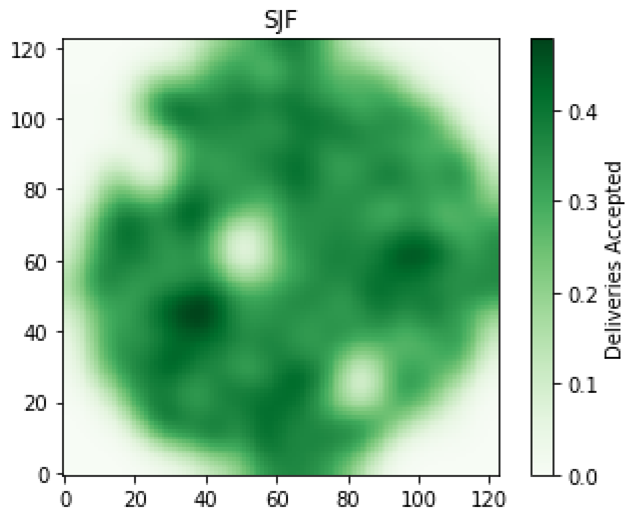

Figure A.31: Smoothed spatial distribution of accepted deliveries when using the SJF scheduler

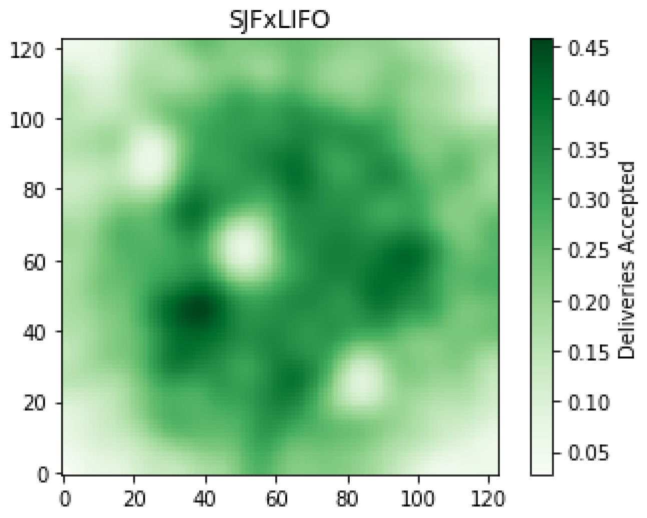

Figure A.32: Smoothed spatial distribution of accepted deliveries when using the SJF-LIFO split-wise scheduler