MSc Project Report

The London Bicycle Sharing System: Efficiency and Balance in a Commuter-Dominated BSS

Joshua Hill
MSc Data Science

Department of Computer Science and Information Systems
Birkbeck College, University of London

2020

1 Abstract

Bicycle Sharing Systems (BSS) are popular utilities in many cities worldwide, and allow users to easily rent bicycles and make journeys between docking stations in the system. Operators need to actively 'rebalance' systems: relocating bikes from stations that are too full to stations that are too empty. This is usually performed by a fleet of motorized vehicles and can detract from the environmental sustainability, as well as the profitability, of a BSS. On weekdays the London BSS requires a large amount of mid-day rebalancing due to concentrated, polarized demand from commuters travelling to and from work.

This project investigated whether it is possible to improve the weekday balance of the London BSS in the absence of mid-day rebalancing. Demand profiles, derived from five years of journey data, were used to estimate the ideal capacities and morning bicycle allocations for stations in the system. A simulator was built to model typical user demand, and various modifications to the system were simulated to evaluate performance.

It was concluded that many central stations have insufficient capacity to accommodate commuter demand, and that increasing the capacity of certain stations would theoretically reduce the need to rely on mid-day rebalancing.

Supervisor: Alessandro Provetti

2 Table of Contents

3 Acknowledgments

I would like to thank:

Alessandro Provetti and Andrea Ballatore for their feedback and suggestions.

Enrique Jiménez Meroño and Francesc Soriguera, for their thoughts and explanations.

My employer for their support and positive attitude towards self-development.

Rachael for supporting and motivating me, and for sharing the sacrifices that came with two years part-time study.

4 Introduction

Bicycle Sharing Systems (BSS) allow users to quickly and conveniently rent bikes, typically for the duration of individual journeys. Third Generation BSS, such as the London 'Santander Cycles' BSS, are characterised by fixed docking stations located throughout the extent of system (DeMaio, 2009). Users can use membership or bank cards to undock a bike from a docking station, undertake a journey, and end their rental by docking the bike at another station in the system. The stations communicate with a central system which recognises when the rented bike has been returned, and the bike is immediately available for other users. The London BSS is owned by Transport for London (TfL).

BSS offer users a cycling option merged with some of the advantages of public transport. People who own private bikes may still choose to take advantage of BSS. Users can flexibly use the system as part of a multi-modal journey without having to carry a bike with them throughout. Users do not need to maintain bikes or protect them against theft, and retain the option of making a one-way journey without the concern of leaving a private bike at their destination.

BSS must be 'rebalanced' by their operators. Users rely on being able to find an available bike at the start of their journey and an available dock at the end of their journey. Completely empty or completely full docking stations may be undesirable depending on current user demand, and the system operator may need to intervene to prevent or resolve these states. This is typically achieved using a fleet of motorised trucks to relocate bikes, so the extent of rebalancing may impact the environmental sustainability of a BSS.

The need for rebalancing may vary depending on the size, topology, and usage of a system. Imbalance may arise gradually from usage which is inherently polarized, for example: a tendency for bikes to migrate downhill as users prefer not to make uphill journeys. Imbalance may also arise because of concentrated one-way demand within a given time period. For example, commuters may overwhelmingly take bikes from a docking station near a transport hub and travel towards employment centres in the morning. Even if the same commuters return in the evening, the acute imbalance during the morning may overwhelm the supply of bikes at the docking station in question. This necessitates urgent mid-day rebalancing to keep the docking station supplied with bikes, which is a preventable and inefficient form of rebalancing (Médard de Chardon, Caruso and Thomas, 2016).

The objective of this project was to investigate the ability of the London BSS to accommodate user demand over the course of a typical weekday, without relying on mid-day rebalancing. Reducing the need for preventable mid-day rebalancing could help to reduce the BSS's contribution to congestion and air pollution in the city, and help make BSS journeys more environmentally sustainable. A number of potential interventions were investigated and evaluated using simulations of the London BSS, and it is hoped that the conclusions of this project could contribute to a discussion on how to improve the sustainability of the system.

5 Background and Literature

On weekdays the London BSS is known to require a particularly large amount of concentrated mid-day rebalancing to the extent that it is likely to generate more vehicle miles than it offsets (Fishman, Washington and Haworth, 2014). The high demand for mid-day rebalancing is largely due to the fact that the system's weekday usage is dominated by commuters. Commuter demand is concentrated and polarised: in the morning there is a high net-demand to travel from the periphery of the system, particularly from transport hubs, towards the dense employment centres. In the evening the demand reverses (Médard de Chardon, Caruso and Thomas, 2016; O'Brien, Cheshire and Batty, 2014). Polarized demand is problematic because it makes the system less self-balancing: a station may have a net tendency to lose or gain bikes at a given time of day.

Even if a station is theoretically balanced over the course of a whole day, commuter demand can overwhelm the finite capacity of the station: filling docks or depleting bikes prematurely. This may force the operator to perform what Médard de Chardon *et al.* term 'Station Extension Rebalancing' (SER). With SER, the operator will compensate for insufficient station capacity by ferrying bikes from near-full stations to near-empty stations during the rush-hour, despite the fact that those stations are theoretically self-balancing over the course of a day. This is particularly inefficient because the SER will generally have to be performed twice: in the morning and then again, in reverse, during the evening.

The topic of rebalancing in a BSS is generally divided into two problems in the literature: the *static* rebalancing problem and the *dynamic* (or 'reactive') rebalancing problem. Both problems set the objective of minimising the number of journeys which are hindered due to stations being full ('failed ends') or empty ('failed starts'), or alternatively will seek to minimise the time that a station spends full or empty. The static rebalancing problem looks at how to arrange the system at a point in time when its state is considered to be constant (e.g. deciding how bikes should be allocated overnight ahead of the morning rush, or what the optimal bike-to-dock ratio system-wide is). The dynamic rebalancing problem looks at *reactively* rebalancing the system whilst it is in use, i.e. how the rebalancing fleet should respond to changing system states during the day to prevent disrupted journeys. The dynamic rebalancing problem is more complex and is addressed by fewer papers in the literature.

For the static rebalancing problem of how bikes should be allocated at the start of the day, (Schuijbroek, Hampshire and van Hoeve, 2013) used a Markov chain-based approach to derive a 'service level' for each station. That is: an acceptable range of bikes that the station should start the day with to minimise the time spent full or empty. They extend the problem to address how vehicles should be routed to efficiently achieve these service levels overnight. (O'Mahony, 2015) addressed both the static problem (overnight rebalancing) and dynamic problem (mid-day rebalancing), whilst working closely with the operators of the New York City BSS. (Jian *et al.*, 2016) built upon O'Mahony's work and used a simulation-optimisation approach to incrementally improve the overnight allocation of bikes by running simulations and re-allocating bikes between stations where the largest number of simulated faults occurred.

In addition to Markov-chain based allocations, (O'Mahony, 2015; Jian *et al.*, 2016) both trialled bike and dock allocations based on the typical demand profile of a station: the typical net of arrivals and departures over time. This project uses a similar demand profile-based approach to allocate bikes.

Authors will commonly validate their proposed rebalancing approach by simulating the BSS, applying their rebalancing strategy, and measuring a reduction in the number of 'failed starts' and 'failed ends'. (Soriguera, Casado and Jiménez, 2018) built a particularly detailed simulation framework using the MatLab programming language, including the implementation of 'truck agents' which reactively rebalance the simulated system mid-rush.

This project focuses on the static rebalancing problem, applied to the London BSS, and investigates: how bikes should be allocated at the start of a day, and which stations require additional capacity. Past literature has investigated the temporal profile of user demand of stations in the London BSS (Médard de Chardon, Caruso and Thomas, 2016). This project produced similar demand profiles and, building on this, used them to make recommendations on ideal station capacity and bike allocations using techniques inspired by those trialled on the New York City BSS (Jian *et al.*, 2016; O'Mahony, 2015). The project focuses on weekday usage, given the particular strain that commuters place on the system.

6 Project Synopsis

The project made use of two key datasets: ‘journey data’ relating to individual rentals that occurred in the system, and ‘station data’ relating to the dock capacity and bicycle fill-level of stations over time. For the journey dataset, eight years’ worth of open datasets were downloaded, cleaned and aggregated. The station dataset was collected by scheduling calls to the TfL API (Application Programming Interface) and recording real-time station information for a number of months. Both datasets were loaded into a SQLite database and were indexed to optimise queries.

The current BSS and its usage were analysed and the concentrated commuter demand could be seen clearly through geospatial visualisations. By merging journey data and station data, it was possible to obtain an accurate count of total bikes circulating in the system over time, which gave insight on the existing strategy of releasing bikes into circulation during weekday mornings, then removing them into depots again in the evenings. To my knowledge, this total bike-count through time has not been previously derived in the literature. The use of depots explains how TfL can accommodate huge demand for bikes at Waterloo and King’s Cross Stations.

Weekday demand profiles, showing the net tendency for bikes to leave or arrive at a station over time, were produced using five years of journey data. These profiles were used to estimate ideal station capacities, and ideal morning allocations of bikes, for all 791 stations in the system. A key conclusion was that considerably more capacity is required at certain stations in employment centres such as City of London and Westminster.

A BSS simulator was developed using object-oriented programming, making it possible to stochastically simulate typical user demand in the London BSS. The simulator uses station data to model the setup of the true system, and simulates user demand / journeys using parametric models based on five years of journey data.

Using the simulator, it was possible to model various scenarios / interventions and evaluate whether they improved the ability of the system to accommodate user demand by reducing failed starts and failed ends. A baseline scenario, which aimed to model the true system but without mid-day rebalancing, was evaluated. Other scenarios included: the existing system but with adjusted morning bike allocations, a system which increased capacity at certain stations, and a system with increased capacity at some stations as well as a reduced bike fleet.

The greatest improvement in performance was seen in the scenario where certain central stations had their capacity increased, but where the existing bike fleet was maintained (but reallocated). The results suggest that it is theoretically possible to reduce the need for mid-day rebalancing by improving the capacity of popular central stations so that they can accommodate commuter demand passively. None of the scenarios were faultless and it appears that there is a legitimate need for *some* mid-day rebalancing.

In the report, section 8 describes how the datasets were gathered and processed. Section 9 features exploratory analysis of the system and explains how demand profiles were created to estimate ideal station capacities and bike allocations. Section 10 describes how the simulator was developed and validated, and how user demand was modelled. Section 11 explains how various ‘scenarios’ based on the system analysis were evaluated using the simulator. The results are given and discussed in section 12, and section 13 gives an overall conclusion and critical evaluation.

7 Project Approach

Project tracking used aspects of the ‘Agile’ framework, whereby broader tasks are defined as ‘epics’ and are sub-divided into ‘user stories’. The main aim of this approach was to divide the work into manageable tasks and focus on near-term development, whilst keeping sight of broader objectives. The Jetbrains YouTrack software was used for tracking, and a board was set up, with Github integration, to track the progress of user stories (see Figure 1).

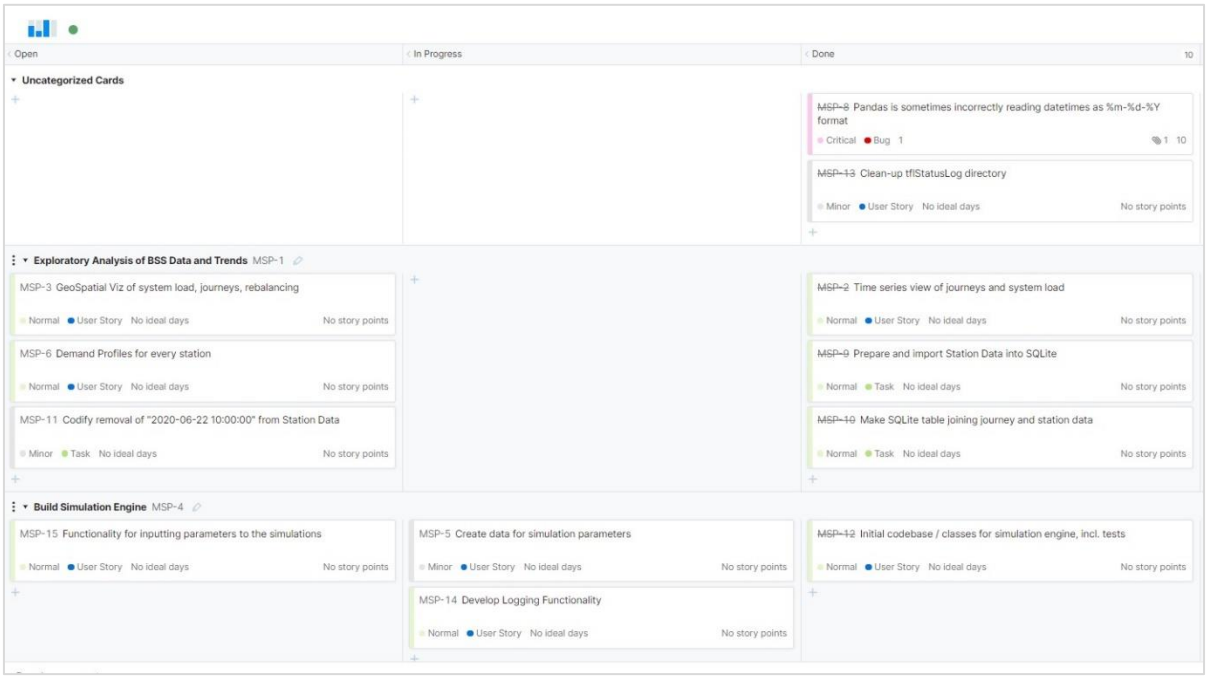

Figure 1- Screenshot of the YouTrack Agile board used for project tracking, as of August.

Git and a remote GitHub repository were used for version-control; software development took place in the Jetbrains Pycharm IDE; and exploration of an interactive or visual nature was generally conducted in Jupyter Notebooks. Automatic testing was used, particularly during the development of the BSS simulations (see section 10.1).

A design principle was to make the project repeatable and robust by scripting as much as possible, and minimising the number of ‘manual’ actions taken. This approach could generate more work in the short term but ultimately helped make the project more robust, particularly when data preparation steps had to be repeated due to bugfixes etc.

8 Data Gathering and Processing

The project featured two key datasets:

- *Journey data* –user rentals: where did they rent a bike, for how long, where did they dock it, and when? The dataset is a comprehensive list of rentals.
- *Station data* – at a point in time, how many bikes were docked at a given station and how many docks were available? The dataset has discrete 'snapshots' of the system at regular points in time.

	Row-count	Date Range
Journey Data	81,432,771	04/01/2012 - 26/05/2020
Station Data	5,463,667	26/01/2020 - 22/06/2020

The gathering and processing of both datasets will be described in more detail in sections 8.1 to 8.4, but is summarised schematically in Figure 2. Once the two datasets were processed as CSVs, they were converted to tables in a SQLite database for the analysis and simulation phase.

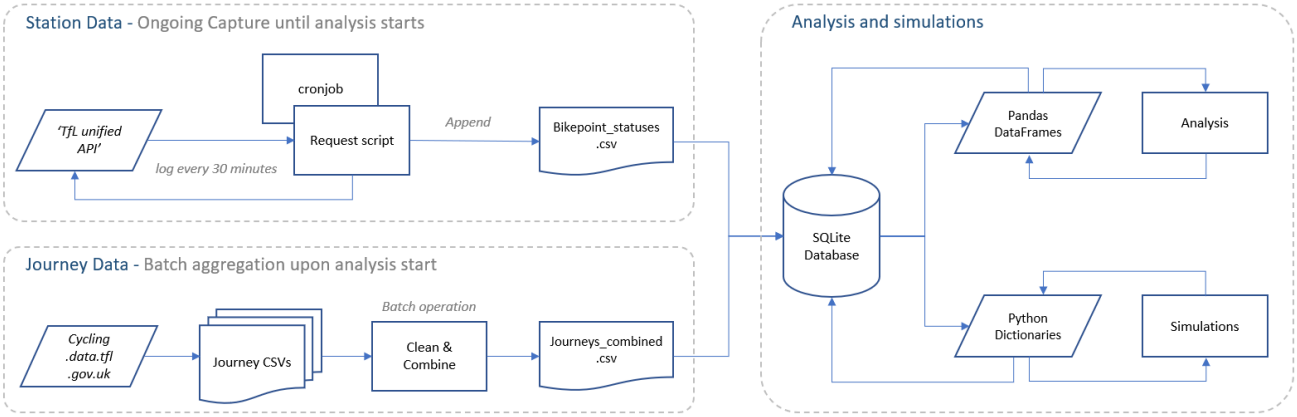

Figure 2 - Schematic summary of data gathering and processing: figure re-used from the project proposal.

8.1 Collecting Journey Data

TfL make their journey data openly available at: https://cycling.data.tfl.gov.uk/

However, the data is stored as hundreds of individual CSVs, each covering a set timespan. Additionally, the webpage lists many datasets in addition to the journey data that we are specifically interested in. It was therefore necessary to automate the process of identifying, downloading and aggregating these datasets. Source code is located in *tfl_project/cycle_journey_prep* in the project repository.

8.1.1 Downloading and Combining the Journey CSVs

The *combineCycleData.py* script was created to recognise the addresses of journey data within the web-page's HTML using a regular expression. After extracting all applicable dataset addresses, the script requested the datasets directly from those web addresses and saved them to disk individually.

Next, the script was used to aggregate the CSVs into one single CSV. This task was complicated by the fact that the format of the CSVs was inconsistent, including:

- Additional unexpected columns, sometimes due to the addition of blank columns
- Renamed columns (e.g. the column usually called 'Duration' sometimes being called 'Duration_Seconds')

The script was iteratively modified to handle these cases. Columns were renamed according to some known substitutions if necessary (e.g. 'Duration_Seconds' to 'Duration'). The header length was then checked against the expected length. If additional columns were detected, it was firstly checked to see if they were all blank (as was often the case) and could simply be omitted. If the header was still too long, the user was prompted by the script to manually input the name of the columns that should be omitted.

It should be noted that although this allowed the CSVs to be successfully coerced into a single file, there was still a need for further data cleansing, as will be described in section 8.3.

8.2 Collecting Station Data through API calls

A historical log of station data is not openly available from TfL. However, it is possible to request real-time station information from the TfL API: https://api.tfl.gov.uk/bikepoint, which is how travel apps such as Google Maps and Citymapper obtain real-time information from TfL. In order to collect a historic station dataset, it was necessary to request and save data periodically from the API.

A custom module, *logging_functions.py* and a script: *bikeStationStatus.py* were written to request and save the output of API calls. The script submits a request to the API, receiving a JSON (JavaScript Object Notation) string in response. It then extracts the desired information from this, adds a timestamp, and appends the resulting data to a CSV in tabular format.

To log data continuously over many months a low-power 'Raspberry Pi' computer was set up with a mounted hard drive disk (Figure 3). A 'cron' (scheduler) job was set up to run the *bikeStationStatus* script every 30th minute. This logging commenced from February 2020 until the analysis phase of the project began in June 2020.

Figure 3 – Photo of the Raspberry Pi computer with mounted hard-drive which was set up to log data from the TfL API every 30th minute. The computer and the solid-state hard drive have very low power consumption.

8.3 Data Cleansing

The script *clean_combined_cycle_data.py* was written to iterate through the combined journey data in batches, apply various cleansing functions, and append the output to a second 'clean' CSV of the same dimensions. Data cleanliness issues encountered included:

- Some EndDates set to 01/01/1970 or zero instead of being null, where a journey was not completed
- Impossible start dates, such as 1900s.
- Text strings instead of integers.
- Cases where different station identifier keys were used instead of the typical bikepoint ID.

The issue with station identifiers was particularly problematic and affected weeks of data at a time. The typical station identifiers were in the 1-800 range, but some batches gave longer identifiers such as: 300077. It was eventually determined that this second key was the 'TerminalName' attribute which was also present in the data returned by the TfL API. Since the API gives both bikepoint IDs and TerminalNames, it was possible to produce an inverse TerminalName-to-bikepoint ID mapping to resolve the issue.

To aid with cleansing, and to ensure a reliable foreign-key relationship, an 'authority' list of active stations was created from the station data. -1 was used as a pseudo key for any *non-null* station ID in the journey data that was not recognised as a contemporary station (i.e. not present in the station data at any point) to ensure a clean and predictable foreign-key relationship between the station data and the journey data.

8.4 SQLite Database and Schema

Given the size of the datasets, it was infeasible to operate on them as CSVs. CSVs have the disadvantage that they must be read sequentially and lack indexing. Therefore, the datasets were uploaded as tables to a SQLite database. SQLite offers the functionality of a relational database but without the need for a remote server. The major strengths of this format were:

- The ability to index data for efficient querying and so that subsets of records could be loaded from disk as needed (e.g. only loading weekday journeys, or journeys starting at a specific station).
- The ability to summarise/aggregate many rows of data using a B-tree, without holding all the component rows in memory simultaneously.
- Optimised join operations.
- Data typing and the ability to impose checks and foreign key relationships if needed

The three main tables produced were: the journeys data, the stations data, and a *station_metadata* table, using *bikepoint_id* as a primary key, which lists station attributes such as: name, co-ordinates and capacity. The ER Diagram in Figure 4 summarises the schema.

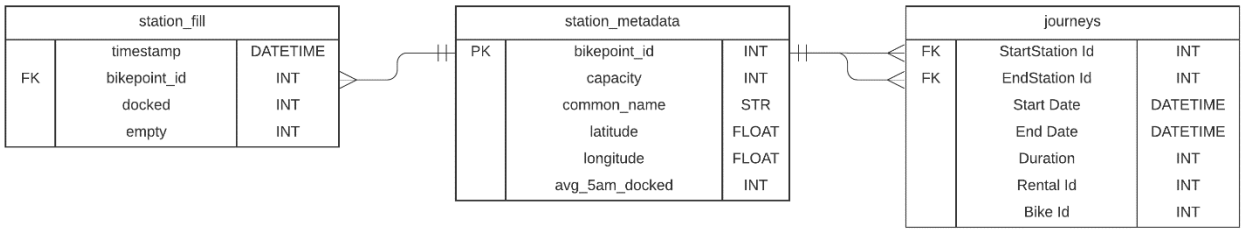

Figure 4 - ER Diagram for the core datasets as implemented in SQLite. The diagram is slightly simplified and omits various attributes used for indexing. For example: journeys table has 'year', 'weekday_ind' and 'hour' attributes which are used for indexing.

8.4.1 Indexing

Various table indices were built to optimise queries. Certain columns such as: *year, hour* and *weekday_ind,* were created which were informationally-redundant (given that there was already a datetime attribute), but which could be indexed for faster retrieval. Inspecting the query plans of slow queries occasionally revealed the

need for a new covering index which could substantially improve performance. For example, when fitting Gumbel distributions to journey durations, hundreds of individual queries were executed to fetch the weekday journeys for a given station. Creating a composite index on (*StartStation Id, weekday_ind*) increased the speed of these queries several fold.

9 Analysis

Analysis began with exploration to profile general patterns of system usage, as well as identify a suitable ‘cut-off date’ where the usage of the system began to shift due to the COVID-19 pandemic (see section 15.1 in appendix).

9.1 Typical System Usage

The BSS sees seasonal usage with monthly volumes peaking in summer and troughing in winter, as can be seen in Figure 5.

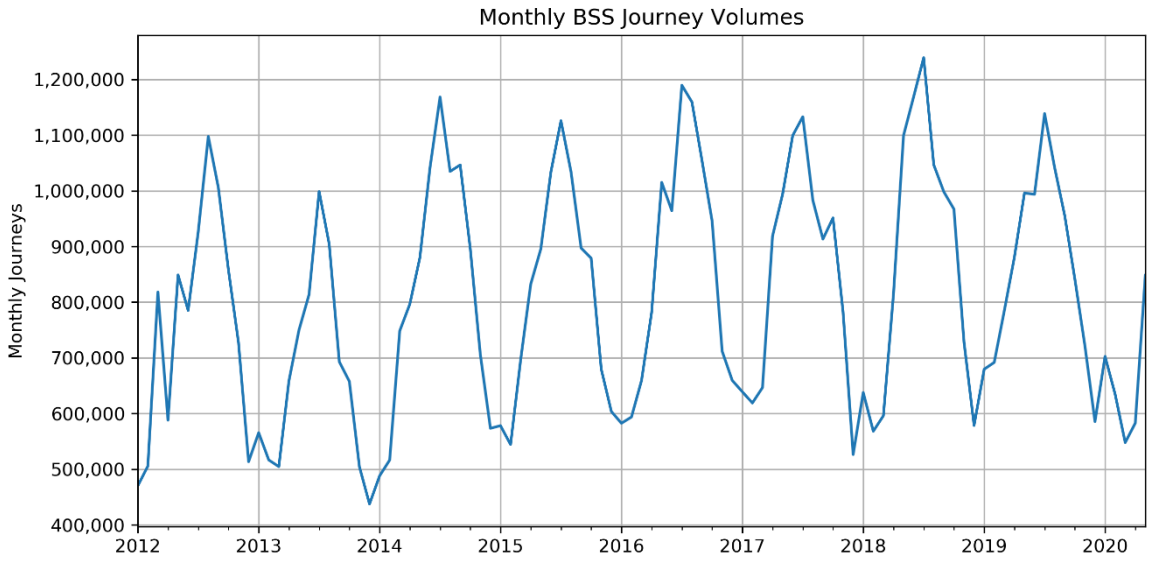

Figure 5- Monthly BSS journeys. Note: the y-axis scale does not originate at zero.

Typical weekdays feature prominent rush-hour peaks in the mornings and evenings, as can be seen in Figure 6.

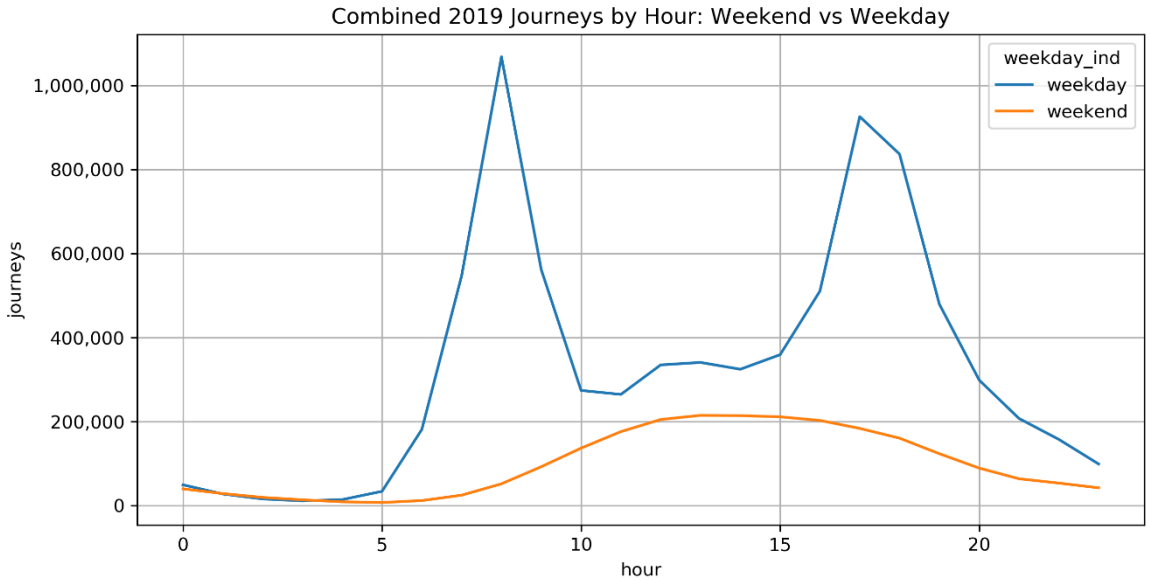

Figure 6 - Combined 2019 journeys by hour, weekdays vs weekends

As has already been noted in previous literature, e.g. (Médard de Chardon, Caruso and Thomas, 2016; O'Brien, Cheshire and Batty, 2014) there is a weekday 'pulse' where users migrate bikes from the periphery of the system to the commercial centres such as the City of London and Holborn. Figure 7 shows two frames from a 24-frame animation: midnight versus midday for the average weekday. During the weekday morning there is a migration of bikes to commercial centres as commuters ride them to work. Then in the evening rush, the bikes migrate back out to the periphery of the system and to transport hubs, leaving fewer bikes and consequently more available docks in the centre. This is not undesirable, and in fact given the polarized demand of the system is efficient in principle.

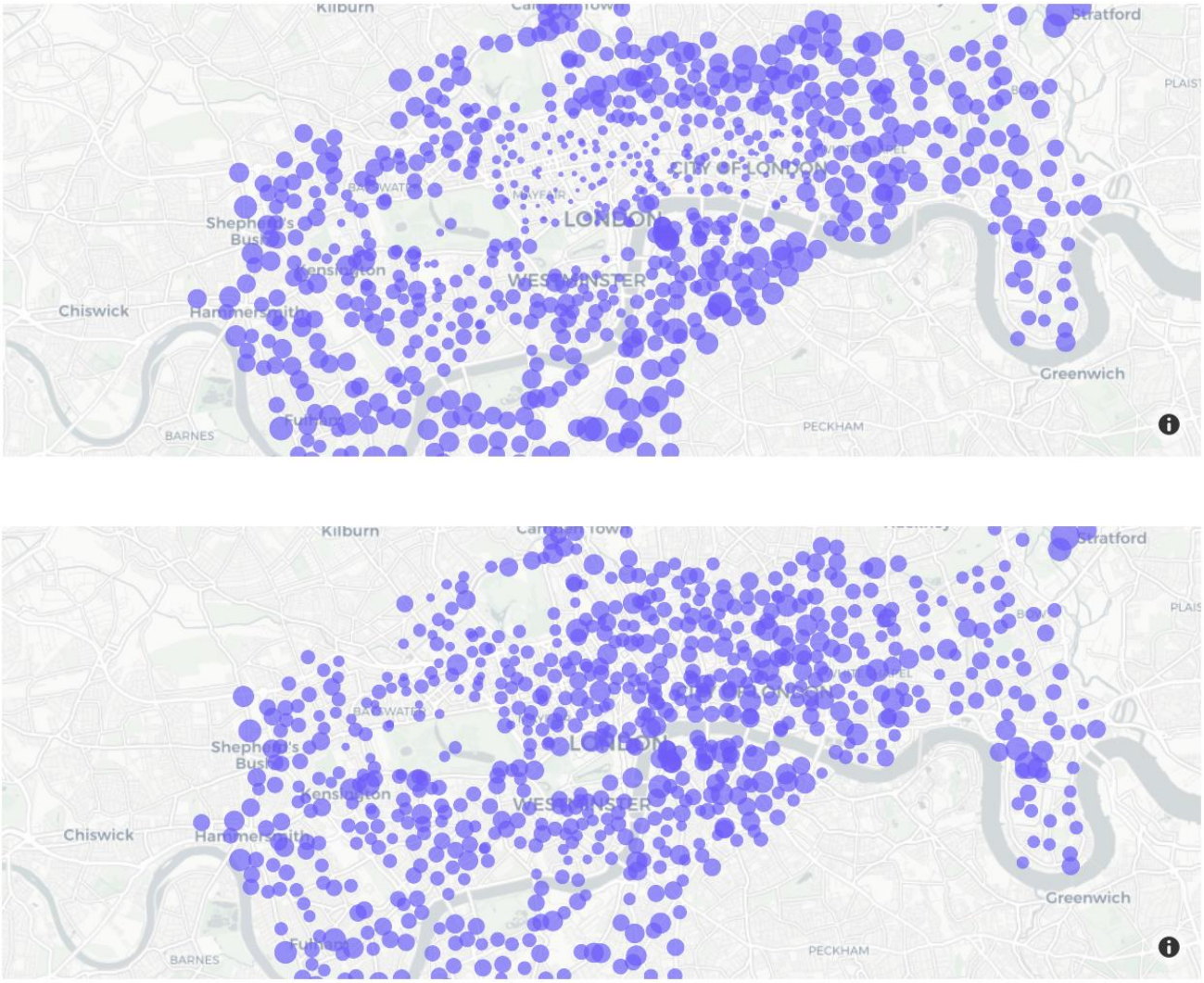

Figure 7- Changes in the distribution of bikes. Mean docked bikes by station at midnight (above) and midday (below), based on pre-COVID weekday observations in the station data. The size of the bubbles are proportional to the absolute number of bikes docked at the station on average.
Central stations are permitted to fill-out during the day and empty during the evening.
The background tile is property of CARTO.

The importance of transport hubs in typical weekday usage becomes very apparent when the most frequent (non-circular) journeys are plotted geospatially. In Figure 8 it can be seen that on weekday mornings, journeys overwhelmingly originate from bike stations around King's Cross and Waterloo train stations and travel into the central business districts of London.

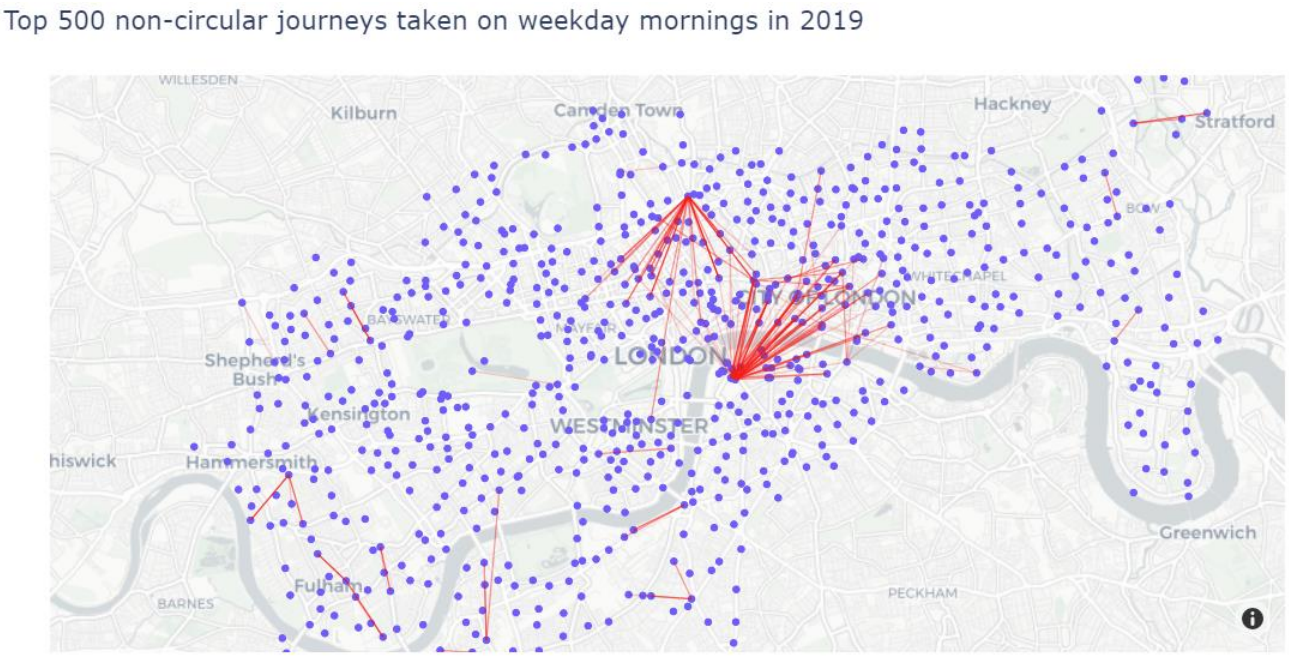

*Figure 8 - The top 500 non-circular journeys, as measured by volume of trips, on weekday **mornings** during 2019. Opacity of an invidiual line is proportional to the number of times that journey was observed.*
The background tile is property of CARTO.

Figure 9 shows the same plot for weekday *afternoons*. The same transport-hub emphasis can be seen (i.e. a reversal of the morning pattern), but there is also more activity on the rest of the network. Some 'weekend-style' use, potentially from tourists, can be seen in the afternoons: including heavy use in Hyde Park and the Queen Elizabeth Olympic Park near Stratford: both of which have opportunities for scenic, car-free cycling.

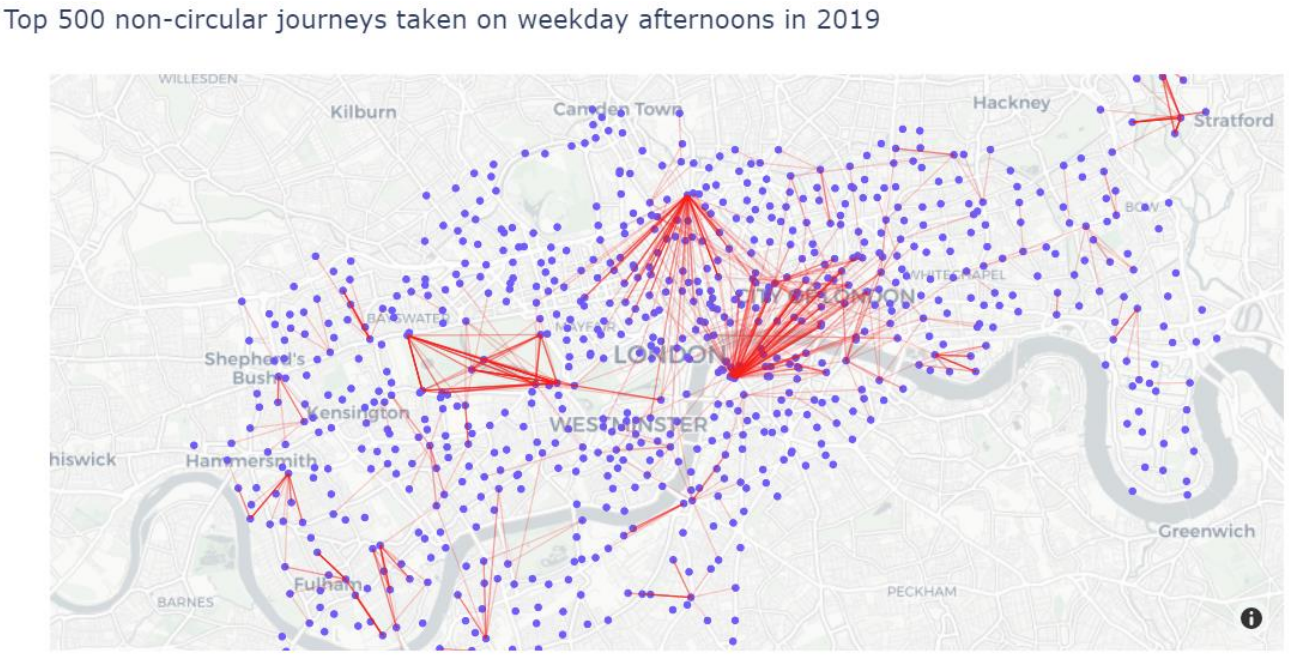

*Figure 9- The top 500 non-circular journeys, as measured by volume of trips, on weekday **afternoons** during 2019. Opacity of an individual line is proportional to the number of times that journey was observed.*
The map tile used is property of CARTO.

9.2 Tackling Station Extension Rebalancing with Depots

9.2.1 The Problem with Polarized Use and Station Extension Rebalancing

If dock and bike availability are unconstrained, then commuter demand is not necessarily an issue in terms of keeping the system balanced. To illustrate with a realistic example, if the average commuter rides a bike from Waterloo to Holborn in the morning, they will likely ride a bike in the reverse direction in the afternoon. Thus, Waterloo has a bike and Holborn has a free dock ready for the next day's commute.

Whilst commuter demand is balanced over a day, it is highly polarized within the mornings and afternoons. So, the problem with commuter-dominated use is often not one of balance, but one of finite capacity. Continuing the illustration: bikes at Waterloo are exhausted and docks at Holborn are saturated during the morning rush. In this scenario, the operator is forced to perform what has been referred to as 'Station Extension Rebalancing', SER, (Médard de Chardon, Caruso and Thomas, 2016). The operator has to actively redistribute bikes between Waterloo and Holborn, during rush-hour, to avoid empty / full docks. A further compounding factor is that, given the commuter demand is balanced over the day, this SER activity must be performed *twice*: in opposite directions during both rush-hours.

Therefore, the fact that commuter demand exceeds station capacity could introduce a significant efficiency penalty: the demand between Waterloo and Holborn, which should be self-balancing in theory, must instead be facilitated with two rounds of frantic rush-hour rebalancing every weekday. So, reducing the need for SER could notably improve the efficiency of the system.

9.2.2 TfL's Buffering of the System using Depots

There is more demand for bikes (docks) at Waterloo and King's Cross on weekday mornings (afternoons) than there is capacity in the docking stations[1]. TfL appears to ease this issue with the use of extra bike storage at: King's Cross Station, Waterloo Station and Holborn. These depots effectively provide the system with 'buffer storage' to aid with commuter demand. It is described in a blog post (IanVisists, 2014) that TfL 'releases' bikes from King's Cross and Waterloo in the mornings, whilst to some lesser extent 'absorbing' bikes into a central depot at Holborn during the day, before reversing the action in the evenings. This effectively provides the extra capacity needed to 'passively' accommodate the commuter demand and reduce the need for SER. A similar strategy is used at transport hubs in New York City, where bikes are given to, or taken from, commuters and corralled by valets (Médard de Chardon, Caruso and Thomas, 2016).

To observe this in action, both journey and station data were used to count the *total* number of bikes circulating in the system at a given point in time: the figure would be expected to vary as TfL release bikes into the system from depots. It was necessary to merge the two datasets: station data can tell us how many bikes were docked in the system at a point in time, but it cannot tell us how many were mid-journey. By performing a carefully-optimised join operation between the station data and the journeys data, it was possible to obtain a count of *total* bikes circulating in the system during snapshots in time, including bikes that were mid-journey.

It can be seen from the resulting data, illustrated in Figure 10, that the number of bikes circulating in the system does indeed vary in a cyclic pattern on weekdays.

[1] see online addendum to: (Médard de Chardon, Caruso and Thomas, 2016), or see appendix section 15.2 in this project.

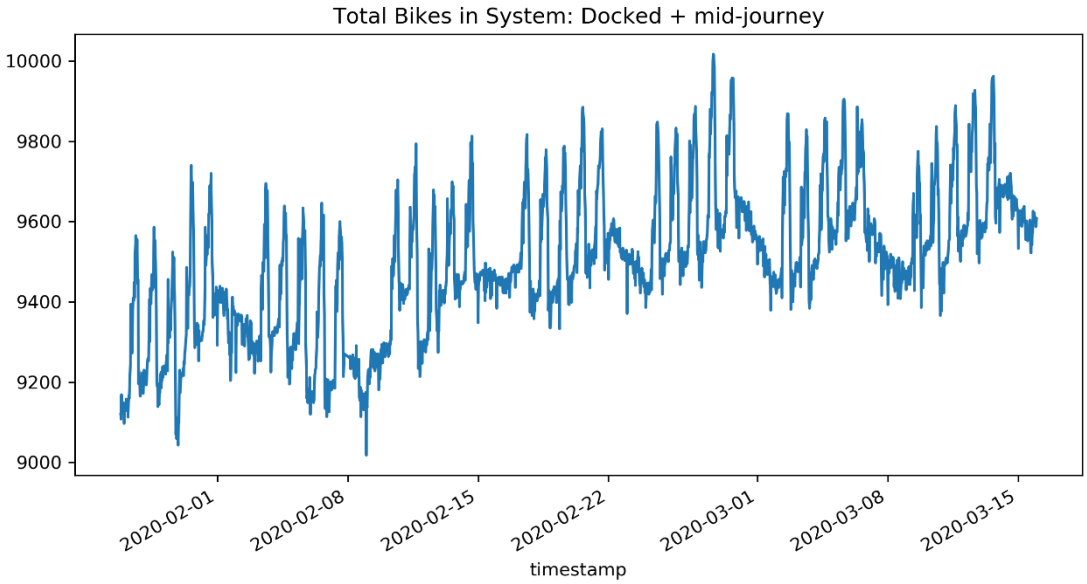

Figure 10- Time series of the total bikes in the system for seven (pre-COVID) weeks in 2020.

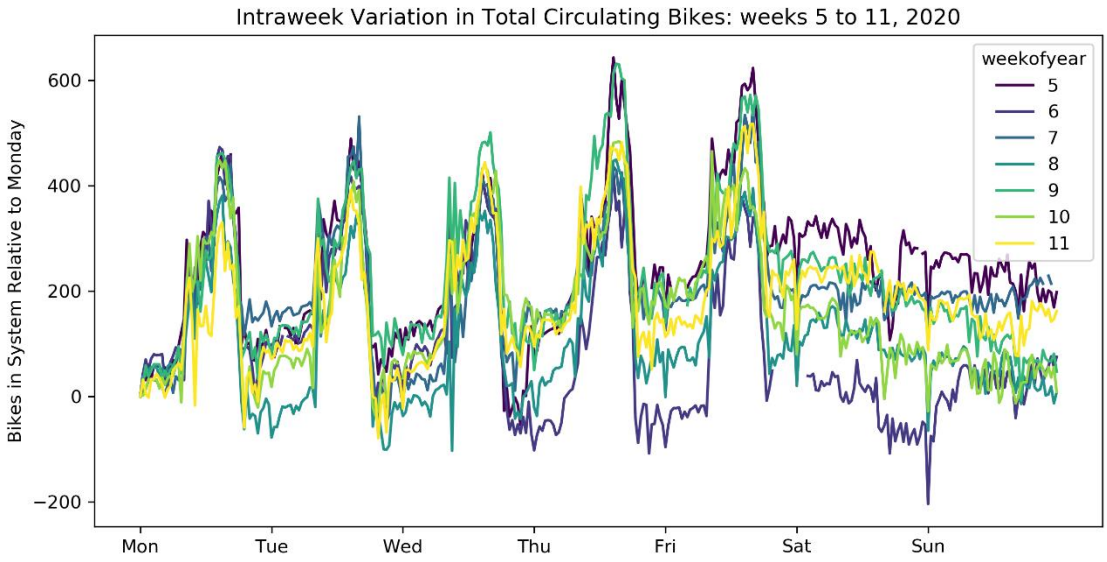

Figure 11 – Variation in the count of total bikes circulating in the system for seven (pre-COVID) weeks in 2020. The count of bikes is stated relative to midnight on Monday of that week. The apparent noise is likely to be due to imprecision around the start and end times of journeys, and/or possibly due to rebalancing actions.

Figure 11 shows clearly that TfL release at least 500-600 bikes into circulation on weekday mornings and retrieve a similar volume in the evening. Over weekends the volume of bikes is relatively stable. Over the observed time period there was an upwards trend, although it appears that most weekends would see a slight reduction in the number of bikes in the system. To my knowledge, this count has not been observed previously in the literature.

This enhances our understanding of TfL's strategy, showing clearly that a buffer of *at least* 600 bikes is used to support the system on weekdays. Despite this, the system still requires a lot of mid-day rebalancing during peak hours (Médard de Chardon, Caruso and Thomas, 2016; Fishman, Washington and Haworth, 2014; O'Brien, Cheshire and Batty, 2014).

9.3 Using Demand Profiles to Determine Idealized Capacities and Allocations

To address the static rebalancing problem, it is necessary to estimate for each station:

- What capacity that station should have in terms of docks.
- How many bikes should ideally be docked at the station before the morning rush.

These parameters should be set to best accommodate the typical usage of that station over the course of a day. If well-optimised, a station should ideally make it through a day without needing any mid-day rebalancing by the operator. Setting an ideal morning-allocation of bikes does imply that overnight rebalancing may need to take place, but it can be argued that overnight rebalancing is preferable to reactive mid-day rebalancing since: roads are less congested, there is ample time for the task to be undertaken by a reduced truck fleet, and the inactivity of the BSS means that the fleet does not have to react to a changing system state.

9.3.1 Rebalanced Effective Usage / Demand Profiles

To determine idealised capacities and allocations, an approach based on demand profiles, as described in (Médard de Chardon, Caruso and Thomas, 2016; Jian *et al.*, 2016; O'Mahony, 2015), was used. The first step was to derive what Médard de Chardon *et al.* refer to as 'Rebalanced Effective Usage' (REU) profiles for all stations based on their weekday journeys in 2019. For clearer interpretation the report will often refer to REU profiles as 'demand profiles', and the two terms are used interchangeably. Some papers such as (O'Mahony, 2015) describe a very similar profile which they refer to as 'demand curves': the interpretation is the same.

Producing this demand profile involves taking the difference between cumulative arrivals and cumulative departures from a station over a day. The net of these cumulative sums gives the demand profile of the station on a given day. This section will briefly explain the derivation of these profiles, before the next section elaborates on their interpretation and use.

To illustrate, Figure 12 shows the daily demand profiles for the station at Broadcasting House in 2019. Each line is the cumulative sum of arrivals, minus the cumulative sum of departures up the given minute of that day.

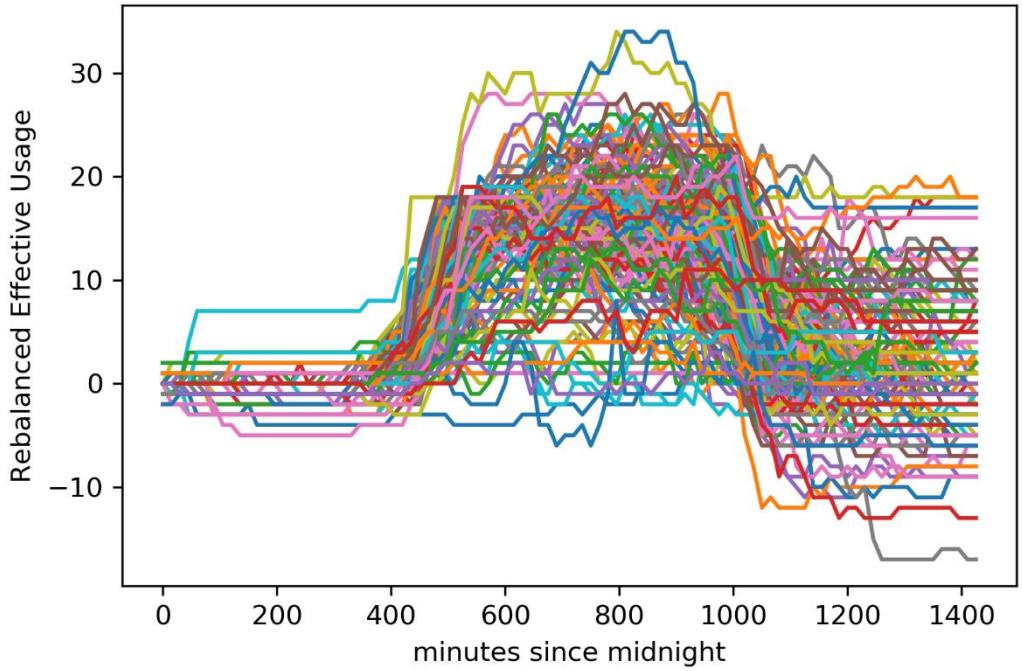

Figure 12 - Daily REU profiles (cumulative arrivals minus cumulative departures) for Broadcasting House station in 2019.

These daily profiles can be aggregated to give a profile of typical usage. Taking the mean gives the profile that would be expected on a 'typical' day (left subplot of Figure 13). Alternatively, quantiles can be used to capture a sense of the variation in daily usage. The right subplot of Figure 13 gives the 25th, 50th and 75th percentile (as ranked per each time interval) for Broadcasting House in 2019.

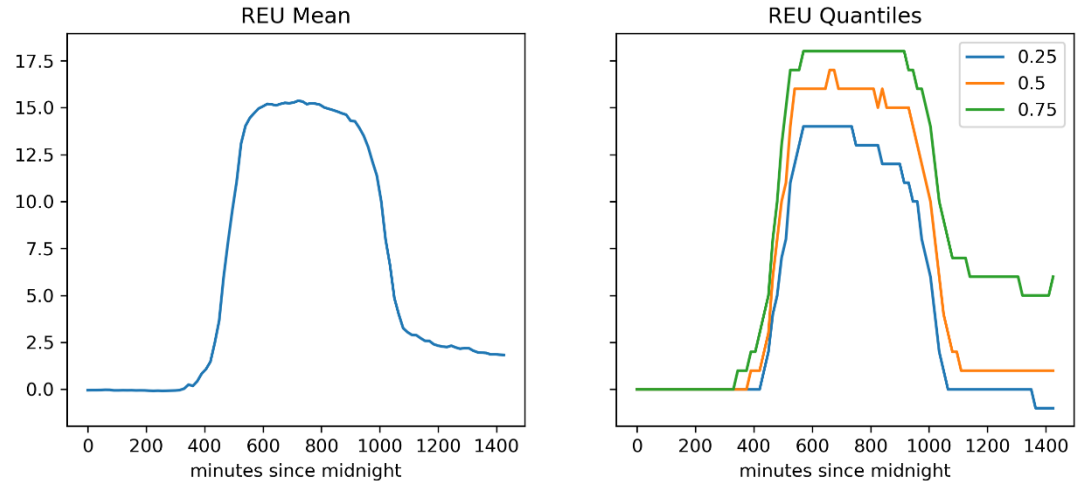

Figure 13 - Summarising the REU Profile of Broadcasting House using mean average (left) and quartiles (right).

9.3.2 Deriving Idealised Capacities and Allocations from Profiles

If a station has perfectly-balanced departures and arrivals over the entire day it will have a flat demand profile. In all other cases, the amplitude of the profile tells us the polarity of its demand over time. A positive amplitude indicates that the station has been a net-recipient of bikes up to that point, whilst a negative amplitude indicates that there have been net departures. To elaborate using three examples:

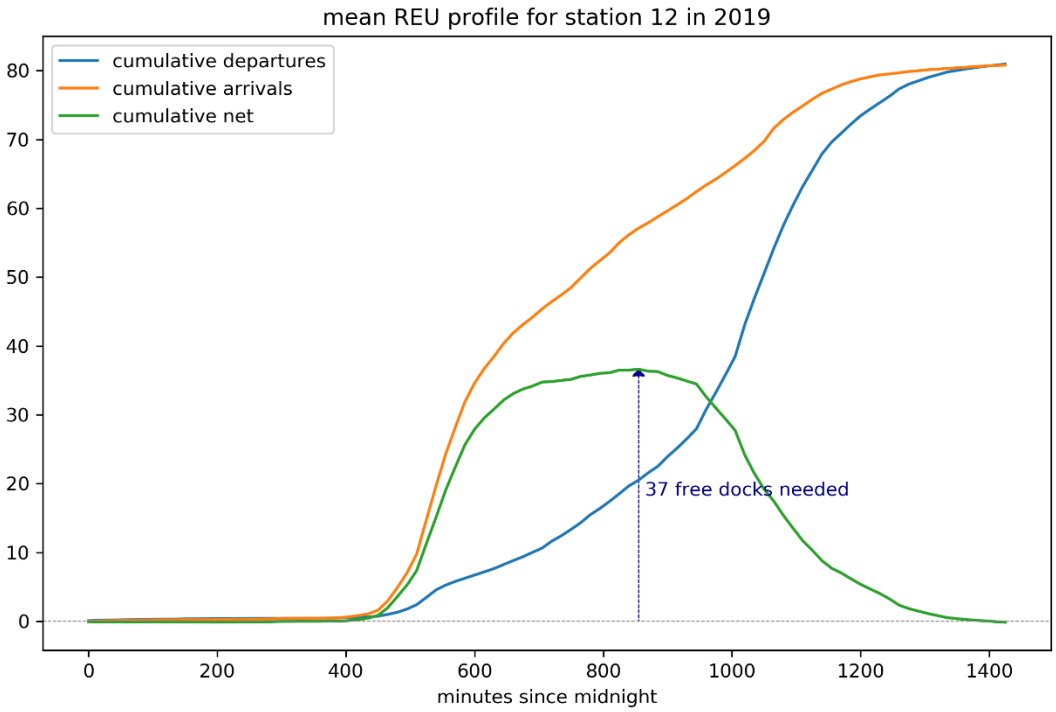

Figure 14 – Mean weekday REU profile for Malet Street.

Figure 14 shows the mean demand profile for the bike station at Malet Street. This station typically has perfectly balanced departures and arrivals over 24 hours, but the maximum amplitude of the demand profile is +37. The station experiences a net-influx of 37 bikes during the day and therefore it needs at least 37 free docks at the start of the day to accommodate this without mid-day rebalancing (before allowing any buffer for inter-day variation in usage).

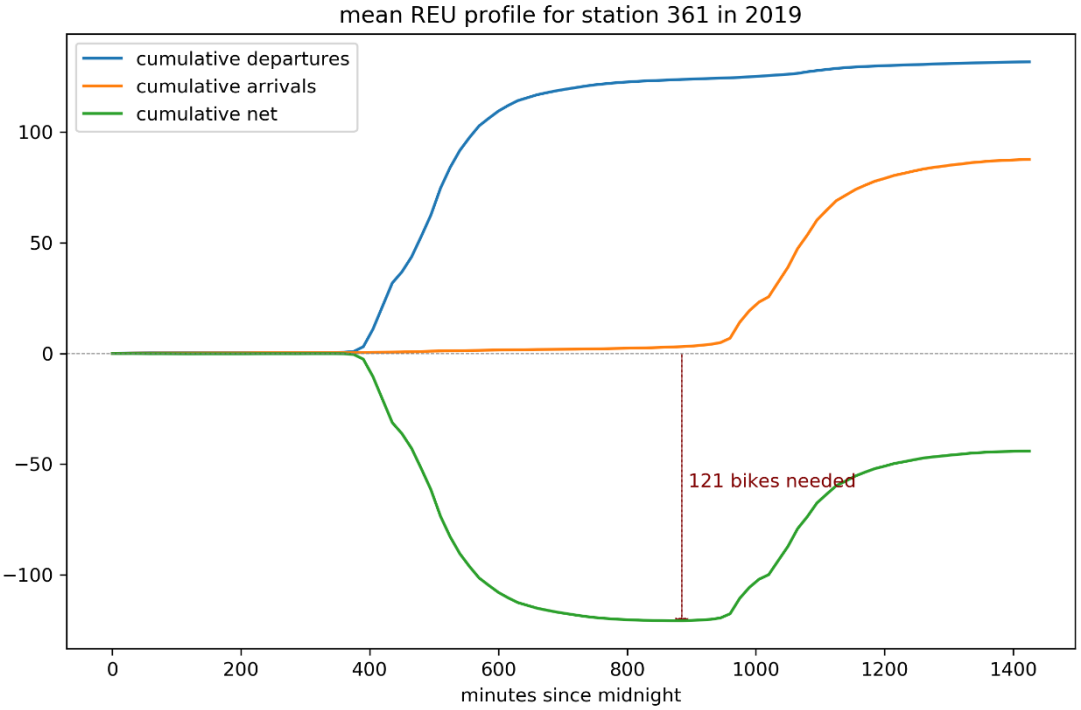

Figure 15 - Mean weekday REU profile for Waterloo Station 3

Figure 15 shows the equivalent profile for Waterloo: bike station 3. This station has a net outflux of bikes during the day. The minimum amplitude is -121 indicating that that the station could start its typical weekday with 121 bikes and zero free docks. As has already been explored in section 9.2, this is facilitated in reality using depots which are used to keep bikes topped-up. To use the terminology of Chardon *et al.*, this station is 'transaction negative' meaning that it typically loses more bikes than it receives over a day. This means that the station either needs:

- Additional buffers of bikes to meet the demands of the next day, or:
- Overnight rebalancing to replenish its stock of bikes (from stations which were transaction positive)

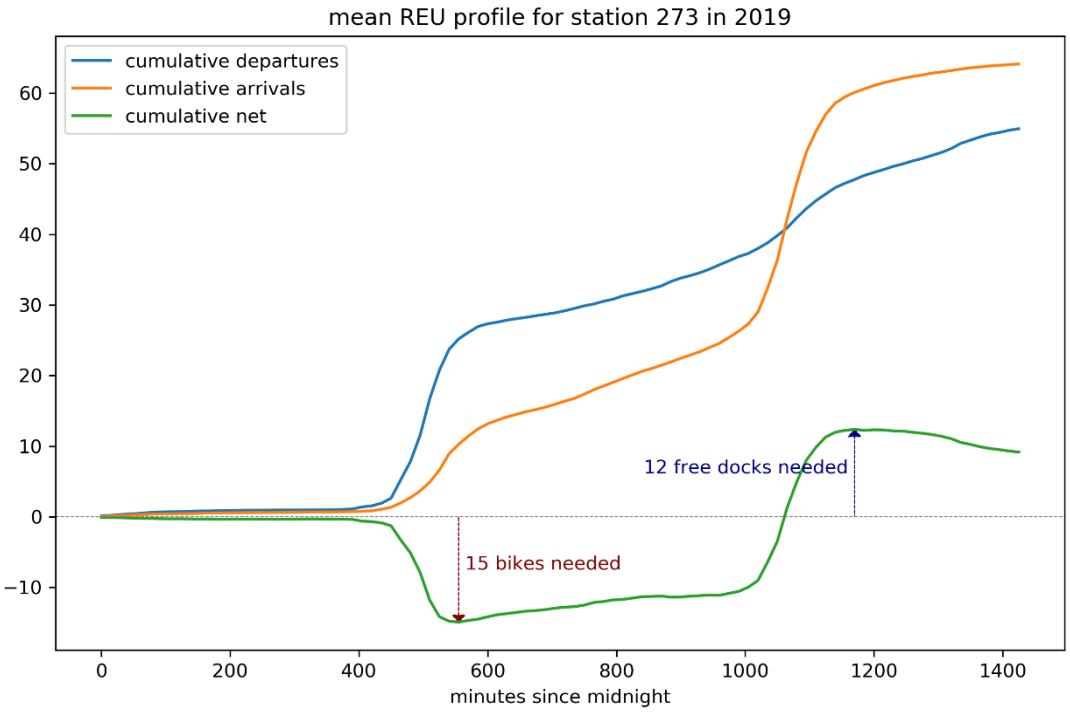

Figure 16- Mean weekday REU profile for Belvedere Road, Southbank

Finally, Figure 16 shows the mean profile for Belvedere Road, Southbank. This station initially sees a net outflux of bikes in the morning, potentially due to its proximity to Waterloo Station. However, in the evening it sees an even greater influx of bikes, potentially due to its proximity to attractions at the Southbank. The demand profile implies the station should (typically) start its day with at least 15 bikes and 12 free docks, totalling a capacity of 27 (because the minimum amplitude is -15 and the maximum amplitude is +12).

To summarise, the ideal morning bike allocation is the absolute value of the minimum negative amplitude, the ideal dock allocation is the maximum positive amplitude, and naturally the ideal capacity is the sum of the two. However, basing these values on the *mean* demand profile fails to account for the variance in usage that occurs in reality.

Rather than relying on mean profiles, more conservative estimates were made using the following approach:

1. The bike allocation is based on the minimum negative amplitude of the 25th percentile-based demand profile (see Figure 13 for example).
2. The dock allocation is based on the maximum positive amplitude of the 75th percentile-based demand profile (see Figure 13 for example).

By being more conservative in both directions, we can accommodate more of the variance in daily station demand.

9.3.3 Resulting Recommendations

Figure 17 plots the idealised station capacities and morning bike allocations that were estimated from station demand profiles.

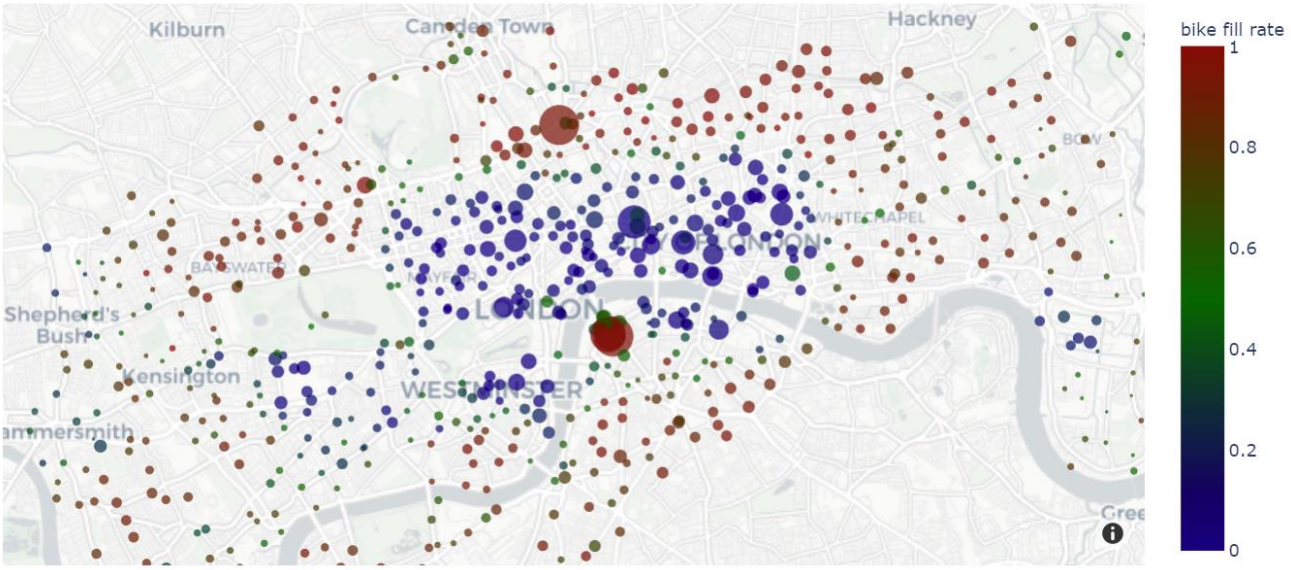

Figure 17 – Ideal station capacity and starting bike allocations according to the conservative estimates. Size of bubbles are proportional to the required capacity (largest value is Waterloo Station 3 at 276 bikes). Colour indicates how much of the total capacity should be filled by bikes at the start of the day. Map tile is property of CARTO.

The ideal weekday allocation begins with most bikes being located at the periphery of the system and around transport hubs, and the employment centres being largely devoid of bikes. This arrangement is the one that accommodates the commuter demand. The transport hubs need the most capacity (which is already catered-for with depots in reality), but it can also be seen that stations in employment centres tend to require more capacity than those on the periphery of the system. It can also be observed that there is a perimeter of stations, surrounding the employment centres, which should have a roughly equal bike-to-dock allocation (appearing as green on the plot).

Comparing real-life capacities with idealised capacities, there are 78 stations whose estimated ideal capacity is at least 10% greater than their true capacity.

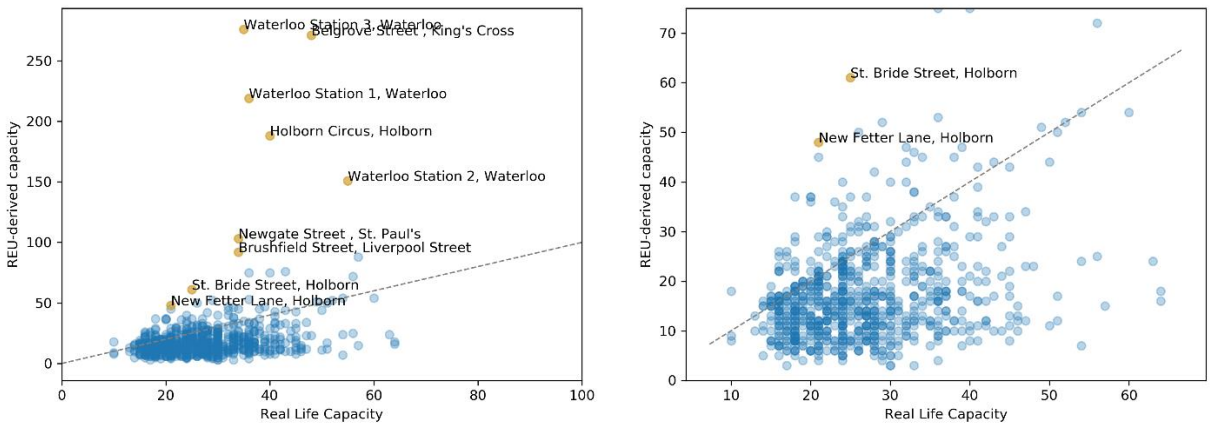

Figure 18- Comparison of real-life capacities versus the conservative capacities derived from demand profiles. The plots are identical but with rescaled axes. The dashed lines show equality.
Stations are highlighted and annotated where the ideal is 2.2 times greater than reality (this threshold was chosen to keep labels readable).

Figure 18 gives a comparison of conservative REU-derived capacities versus real life capacities. It should be noted that the most extreme discrepancies are in fact for stations which are known to have depots in operation, namely: King's Cross, Waterloo and Holborn. The real-life capacities stated do not account for depots, so the discrepancy is effectively overstated for these stations. The top 20 stations for estimated shortfall in capacity are listed in the appendix: section 15.2.

Many stations already have sufficient capacity, and may simply require a modified bike allocation. Plotting the *extra* capacity required geospatially results in the plot in Figure 19.

*Figure 19- Stations requiring **extra** capacity according to the conservative estimates. Existing depots / storage are not accounted for.*
Size of bubbles are proportional to the required extra capacity (largest value is Waterloo Station 3 at 241 bikes).
Colour indicates how much of the total capacity should ideally be filled by bikes at the start of the day.
Map tile is property of CARTO.

As with Figure 18, the increase is overstated for stations with depots. Aside from these stations, there is also an apparent need for more capacity around the city of London (e.g. 69 extra docks at St Paul's), Liverpool Street Station (58 extra docks), as well as several apparent clusters of stations around Westminster, St. James's Square, Soho and Knightsbridge. If these stations cannot accommodate extra capacity, then the implication is that they would benefit from a local depot to act as a buffer.

Having analysed the system and potential interventions, it was necessary to develop a simulator to estimate the impact of any hypothetical changes to the system.

10 Simulation Implementation

The source code for simulations can be seen in the project repository under *tfl_project/simulation.* This section will initially describe the program, including classes and their behaviours. In section 10.4 the modelling of user demand is discussed.

10.1 Development Approach

The simulator was implemented using object-oriented programming in the Python language. The simulator was developed using a bottom-up approach in test-driven cycles. Automated tests using the pytest framework were used to ensure that behaviour was as expected (e.g. that if a User attempts to dock a bike at a full Station, an exception is raised), and as classes and functionality were added, so were tests. The following sections give an overview of the classes that were developed for the simulator, including some of the key attributes and behaviours of those classes.

10.2 Classes and Behaviours

The classes developed for the simulation can be divided into two rough groups:

1. ‘core’ classes concerned with enacting the simulation:
    a. **Agent**, **User and City** in the **city.py** module
    b. **Store**, **Station** and **WarehousedStation** in the **station.py** module
2. ‘management’ classes used to set-up and run simulations:
    - **LondonCreator** and **SimulationManager** in the **sim_managment.py** module

10.2.1 Core Classes

A UML (Unified Modelling Language) class diagram for the core classes is given in Figure 20

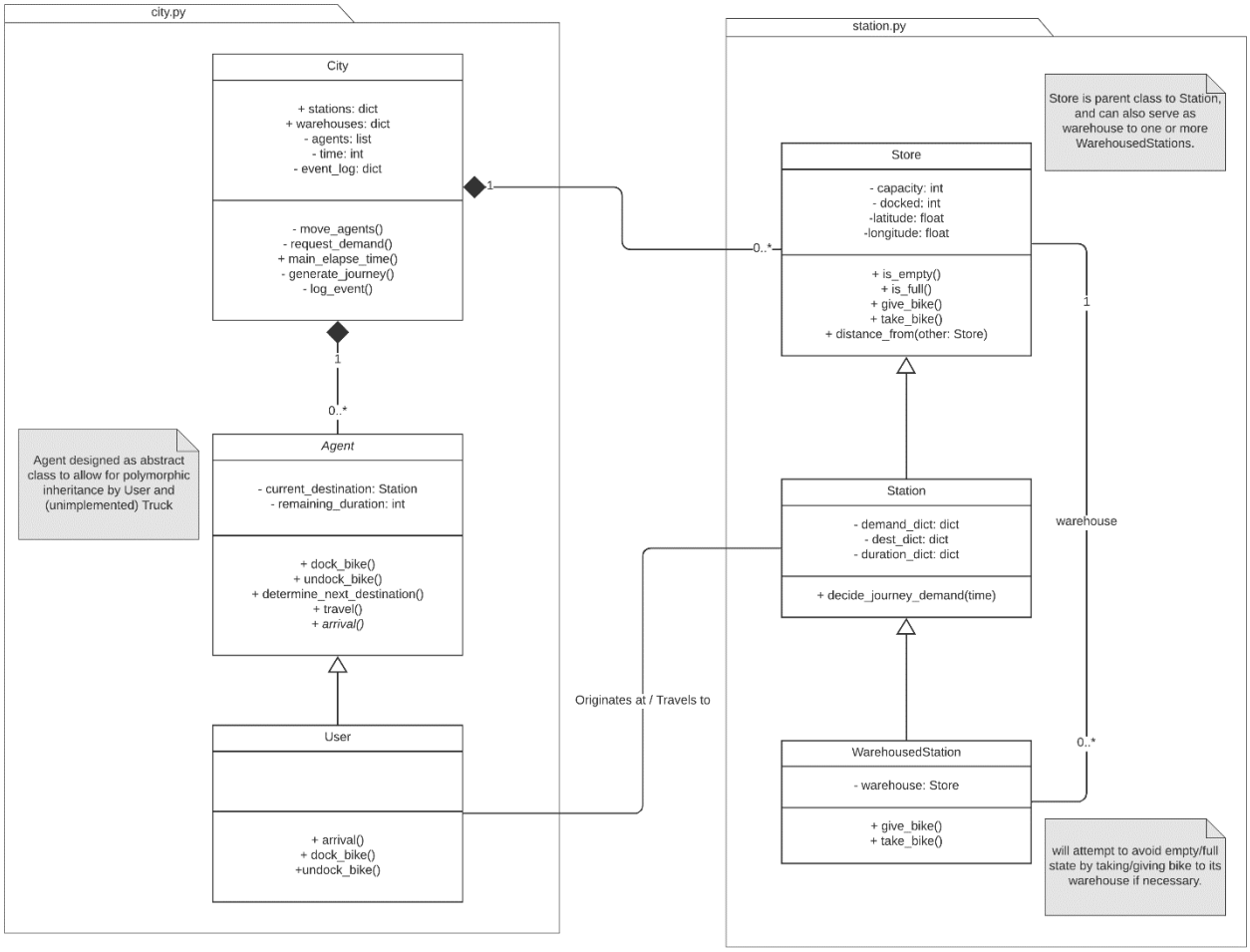

Figure 20 - UML Class diagram for the 'core' simulation classes

10.2.1.1 Store / Station Classes

The **Store** class implements the functionality around taking bikes, storing them up to a finite capacity, and giving bikes out from the store. Bikes are tracked using counters. When directly instantiated, the Store class can function as a depot or 'warehouse' in the simulated BSS.

The **Station** class inherits from Store, and extends the class with attributes and methods relating to user demand generation. A station instance has dictionaries used for deciding which journeys, if any, will originate from it at this time. The modelling behind journey generation will be discussed in more detail in section 10.4, but the key information given by these dictionaries is:

- *demand_dict*: for a given time period, what is the average number of journeys that originate at this station per minute.
- *destination_dict*: for a given time period, what is the multinomial distribution of destinations that users will travel to from this station.
- *duration_dict*: given a destination, what are the parameters for the probability distribution of journey durations from here to there.

A Station does not directly create any journeys or Users, but it decides what journeys should be created and passes their details to its City for generation.

The **WarehousedStation** class models stations which are directly served by a nearby depot, like King's Cross. The class inherits from Station and adds a warehouse attribute: which points to an instance of the Store class. The WarehousedStation behaves slightly differently when giving or taking a bike. If giving or taking a bike will leave the WarehousedStation empty or full, it will attempt to exchange a bike with its warehouse, subject to availability/capacity, to avoid being empty/full. Many instances of WarehousedStation can share a single Store as a warehouse (as is the case with the three bike stations at Waterloo in reality).

10.2.1.2 City and User Classes

The **City** class 'aggregates' Stations and Users into a cohesive context. City also tracks time, stores logs of events, and has the main logic for advancing time in the simulation.

The *main_elapse_time* method increments the current time counter and calls various other methods which advance the state of the simulation: the *move_agents* method calls Users to perform their movement actions, *request_demand* calls stations to supply the parameters for any new journeys. Note that the procedures performed by the Users and Stations in response to these calls are 'encapsulated' within those classes themselves.

The City class is able to generate new Users who are beginning a journey (or logs a failed start if their originating station is empty). Although the demand is supplied by Stations, the actual creation of the journey is performed by City in order to maintain encapsulation and prevent Stations from directly accessing and modifying the list of agents.

The **Agent** class serves as an 'abstract interface' between the City class and any agents (i.e. Users). This means that the Agent class is not intended to be directly instantiated, but has various methods which any child-class should implement or inherit. This approach was chosen so that it would be possible to benefit from the concept of 'polymorphism' if rebalancing trucks were implemented in the simulation as well as users. The City would be able to interface with Users and Trucks by calling abstract Agent methods on them such as *determine_next_destination*, whilst the two child classes could 'override' the method and perform different actions as applicable. Ultimately only the User class was implemented, but the abstract Agent class still means that Trucks could be implemented with minimal difficulty in any future work.

The **User** class inherits from the Agent interface. A User is instantiated by City with a predefined destination and a duration in which to travel there. When their duration has elapsed (meaning they have arrived), they will attempt to dock their bike, log a successful journey, and flag themselves for removal from the City. Otherwise, they will log a failed end and choose a new destination (and duration) based on whichever non-empty station is nearest to them.

10.2.2 Management Classes

The management classes in the **sim_managment.py** module, described by the UML diagram in Figure 21, are used to set-up and oversee simulations.

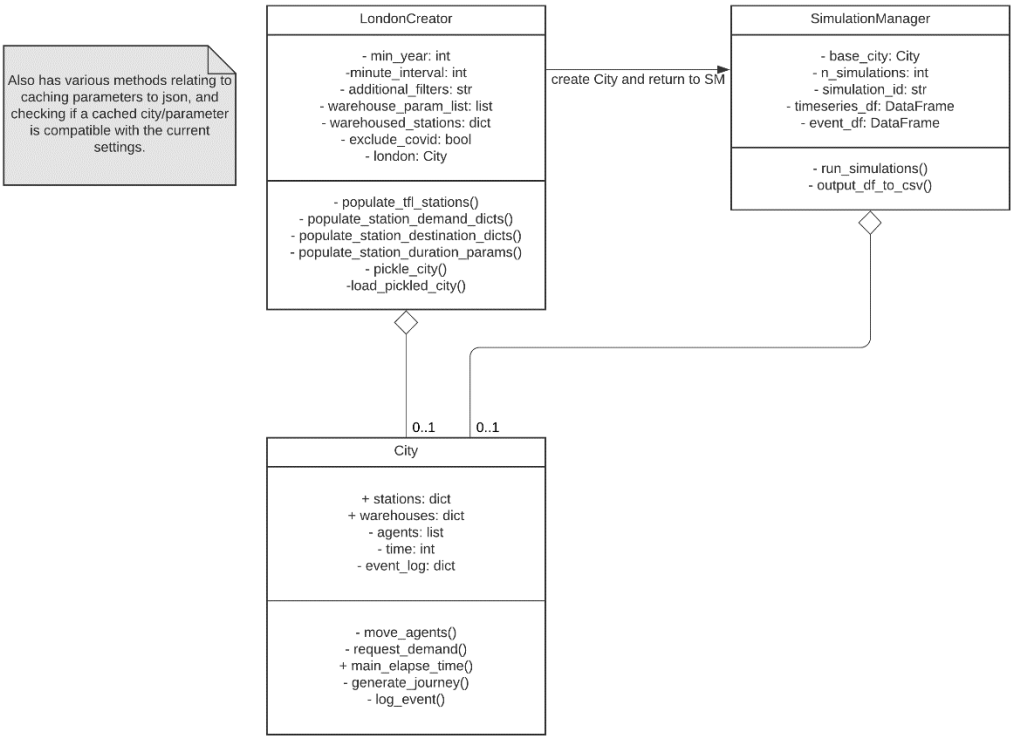

Figure 21 - UML Class Diagram for Managment Classes

LondonCreator is designed to instantiate a City with attributes modelling the real-life London BSS based on the historic journey and station data, keeping the specific logic separate from the City class itself. When creating a City from scratch, the LondonCreator class will execute SQL (Structured Query Language) queries against the SQLite database to fetch the required data, process it, and pass the resulting parameters to the City.

The LondonCreator class also has the ability to 'save' an instantiated City to a 'pickle' file on disk. Another instance of LondonCreator can then load the City from the file without pulling the data from scratch. A system of saving cities along with a json describing their parameters was used to protect against accidentally loading a city with different parameters (e.g. a minimum year of 2015) to those specified in the LondonCreator instance (e.g. minimum year of 2014).

The **SimulationManager** class is passed a City instance when it is instantiated. It runs *n* independent '24 hour' simulations starting with the base City each time. The SimulationManager aggregates the log DataFrames from each of these simulations and can output the combined logs to CSV.

10.3 Running Simulations

To run simulations the user has to prepare a City instance, typically using a LondonCreator, then pass it to a SimulationManager. The following code is from the *sim1_warehouses_bigger_5am.py* script, which is used in the project as a baseline model of the true BSS.

```
from tfl_project.simulation.sim_managment import LondonCreator, SimulationManager
from tfl_project.simulation.scenario_scripts.describe_city import describe_city


def main():
    """
    Using conservative warehouse capacities as suggested by the REU profiles.
    See 'total system use' data for validation around whether this is realistic.
    :return:
    """
    warehouse_params = [
        {'capacity': 520, 'docked_init': 520, 'st_id': 'WATERLOO'}
        , {'capacity': 220, 'docked_init': 220, 'st_id': 'KINGSX'}
        , {'capacity': 150, 'docked_init': 0, 'st_id': 'HOLBORN'}
    ]

    warehoused_stations = {
        # Waterloo Stations 1, 2, 3
        374: 'WATERLOO',
        361: 'WATERLOO',
        154: 'WATERLOO',
        # Belgrove Street, King's Cross
        14: 'KINGSX',
        # Holborn Circus, New Fetter, Stonecutter
        66: 'HOLBORN',
        546: 'HOLBORN',
        112: 'HOLBORN',
    }

    base_london = LondonCreator(
            min_year=2015
            , minute_interval=20
            , exclude_covid=True
            , warehouse_param_list=warehouse_params
            , warehoused_stations=warehoused_stations) \
        .get_or_create_london(pickle_loc='simulation/files/pickled_cities/london_big_warehouses')
    describe_city(base_london)
    sm = SimulationManager(city=base_london, n_simulations=20,
                           simulation_id='SIM1_BIG_WAREHOUSE_5AM_NO_REBAL')
    sm.run_simulations()
    sm.output_dfs_to_csv()


if __name__ == '__main__':
    main()
```

The script defines parameters describing three depots linked to seven stations. The LondonCreator *get_or_create_london* method will check the given file directory for a pre-saved *london.pickle* file. If does not find one, it will create it from scratch using fresh SQL queries, save it to disk, then return it as a function output. The SimulationManager instance then takes the city and runs 20 simulations and saves the logs to a CSV in a file directory named after the simulation_id attribute. In other scripts, functions were defined to modify attributes in *base_london* to simulate other hypothetical scenarios.

10.4 Modelling Stations, User Demand and Journeys

10.4.1 BSS Attributes

The stations present in the simulated London BSS were based on the 791 stations observed in the station data between 26/01/2020 and 22/06/2020. Default capacities were set to the maximum capacity observed for that station in the available data (some station capacities varied due to maintenance etc.).

The default bike allocation between stations was set to the mean number of bikes docked at each station at 05:00 on weekdays between 26/01/2020 and 15/03/2020 (the COVID-19 cutoff). 05:00 was selected because it is the latest hour before user activity begins to increase for the next working day, and therefore reflects the mean 'starting state' of the system. The assumption here is that if TfL perform any overnight rebalancing, it is likely to be done by 5am.

In total the default station attributes summed to 21,087 docks of station capacity and 9,458 bikes pre-allocated between stations.

10.4.2 User Demand

As with many papers in the literature (O'Mahony, 2015; Jian *et al.*, 2016; Schuijbroek, Hampshire and van Hoeve, 2017) user demand was modelled as a Poisson process. A Poisson distribution gives the probability distribution of the number of discrete events that may occur within a time interval, parameterised by the expected rate of occurrences (in this case: mean journeys-per-minute). This lends itself well to modelling the number of journeys that will begin at a station during a given minute.

The rate at which journeys begin at a station is time-dependent, so each station, *i*, must have its journey-rate, $\lambda_{i,t}$, calculated for every given time interval, *t*, within a 24-hour period. Rates were calculated over 20-minute intervals, meaning each station has 72 'journeys-per-minute' rates to reflect every interval in a 24-hour period. Note that in practice the simulation proceeds on a minute-by-minute basis, but the Poisson process by which a station generates journeys will only change every 20 iterations. Narrower intervals would have meant demand changes with more granularity, but would potentially increase the risk of overfitting by calculating rates over shorter time intervals. The mean journeys-per-station-per-interval were calculated from all journeys undertaken on weekdays from 2015 onwards.

10.4.3 Destinations

When a given station, *i*, generates a journey within time interval *t*, the probability of the destination being station *j* is estimated as:

$$P_{t,i,j} = \frac{r_{t,i,j}}{\sum_k (r_{t,i,k})}$$

Where $r_{t,i,j}$ is the number of rentals since 2015 from station *i* to station *j* which began during interval *t*, and where $\sum_k (r_{t,i,k})$ is the sum over all destinations for the same time interval. So, after a simulated journey is generated at station *i* at time *t*, its destination is sampled from a multinomial distribution of discrete probabilities: $P_{t,i,k}\ \forall\ k$

As with demand, 72 sets of parameters were stored per station: one for each 20-minute interval in a 24-hour period.

10.4.4 Journey Durations

The distribution of journey durations *from* a given station, *i*, *to* a given station, *j*, were assumed to be independent of the time of day. The distributions of durations were estimated separately in both directions, so the duration from *i* to *j* was modelled independently from the duration from *j* to *i*.

Previous papers have estimated journey durations in discrete minutes using Poisson distributions (Jian *et al.*, 2016; O'Mahony, 2015). After experimentation, it was decided that this project should use the right-skewed Gumbel distribution instead of the Poisson distribution because:

- This reflected the right-skew present in the true journey durations, unlike the Poisson distribution which is generally symmetrical.
- The Poisson distribution tended to be too wide when durations are short, and could over-estimate the probability of very short trips (see stations 30, 240 and 330 in Figure 22).
- The Poisson distribution tended to be too narrow when durations are long, and could under-estimate the probability of longer trips (see station 150 in Figure 22). This was particularly apparent if durations were measured in seconds instead of minutes.

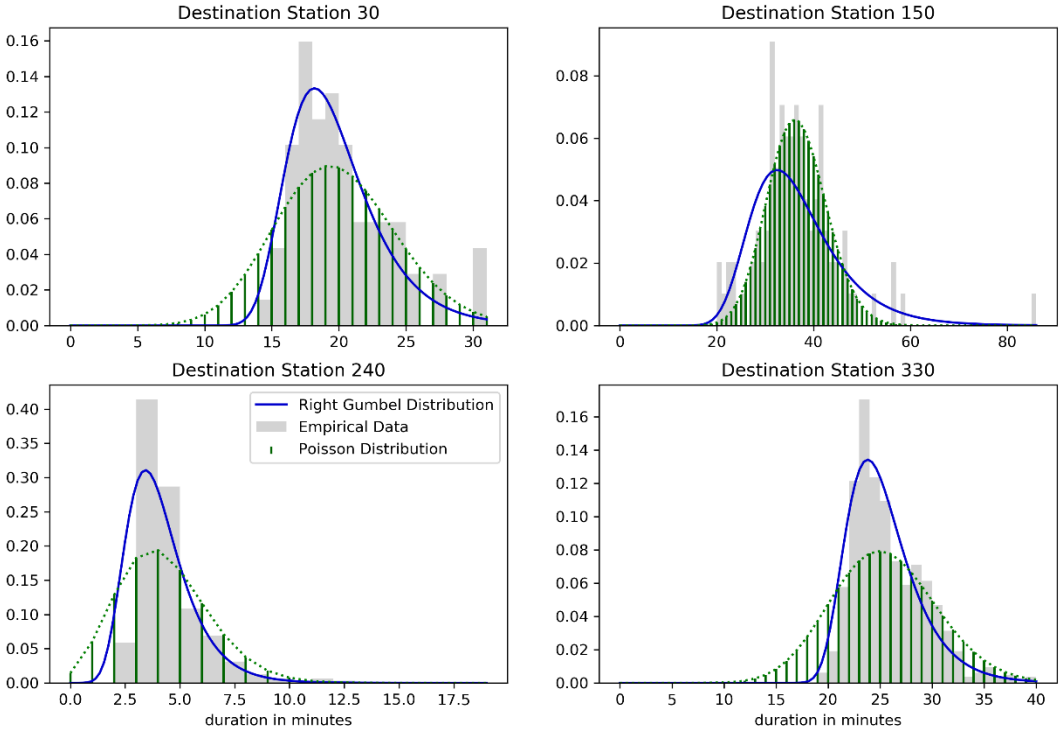

Figure 22 – Journey durations from Waterloo Station 3 to a number of destinations of varying distance. Fitted probability density functions for Gumbel distributions, and probability mass functions for Poisson distributions, are shown for comparison. The y-axis gives density / mass.

The disadvantage to using the Gumbel distribution was that it significantly increased the computation required. A Poisson distribution can be parameterised using just the mean journey duration as a rate, which is quick to compute in a SQL query. In contrast, the Gumbel distribution necessitated loading the whole array of journey durations into memory so that the distribution could be fitted to the data using the Scipy.stats package, before the best-fitting parameters could be obtained. For this reason, the durations were queried and fitted for one station at a time, and the resulting parameters were 'cached' as JSON files so that future instantiations of LondonCreator could avoid computing them from scratch.

During simulations the duration of a journey is sampled from the applicable Gumbel distribution for the given start and end stations. Since the simulation proceeds in discrete minutes, and the Gumbel distribution is a continuous probability distribution, the resulting duration is rounded to the nearest minute (and also floored at one minute as a precaution against edge-cases).

10.5 Simulation Validation

To validate that the simulator was emulating the behaviour of the BSS realistically, the mean behaviour of 20 simulations was compared with the mean daily behaviour of the true system on 2019 weekdays. The simulation used for the validation ran:

- The default 5:00am bike allocations (described in section 10.4.1)
- The default station capacities
- Three depots with generous capacity:
    - A King's Cross depot of 220 bikes, linked to the Belgrove Street bike station
    - A Waterloo depot of 520 bikes, linked to the three Waterloo bike stations
    - A Holborn depot of 150 empty spaces, linked to: Holborn Circus, New Fetter Lane and Stonecutter Street[2].

Since the true capacity of the TfL depots is not publicly available, the simulated depots were set to accommodate the demand suggested by the station demand profiles. It is reasonable to assume that even if the depots are smaller than this in reality, TfL will compensate for any shortfall in the buffer with SER rebalancing (which is absent from the simulator).

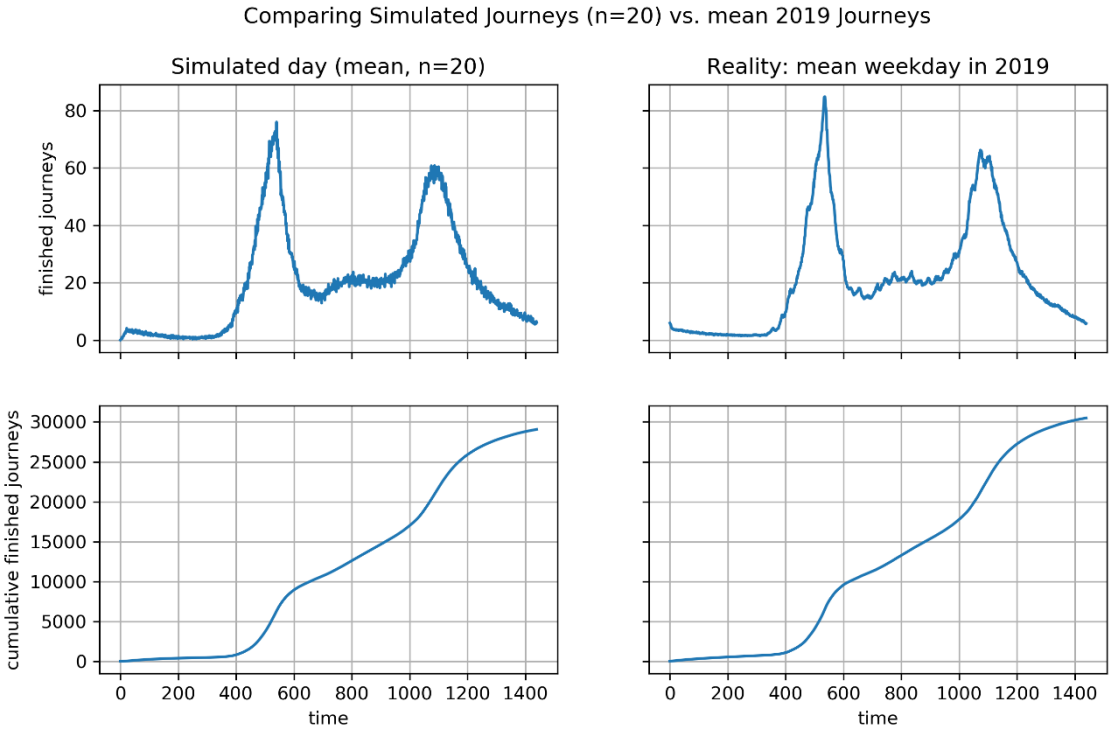

Figure 23 - Comparing the timelines of an average simulation day (n=20) with an average day in reality (2019 average). The metric is finished journeys and time is given in minutes since midnight. y-axes are shared by charts in the same row.

[2] Stonecutter Street was included as linked to the Holborn depot because of its proximity and because images of corralled bicycles can be seen there at: https://goo.gl/maps/TXXPiiAJhKcs1aQV7, as noted in the online addendum to: Médard de Chardon, C., Caruso, G. and Thomas, I. (2016) 'Bike-share rebalancing strategies, patterns, and purpose', *Journal of Transport Geography*, 55, pp. 22-39. .

As can be seen in Figure 23, the temporal profile of journeys in the simulator matches very closely with the 2019 reality. To validate the realism of station-level demand, both in terms of journey origins and destinations, scatterplots were produced to compare the simulated and empirical journeys per station.

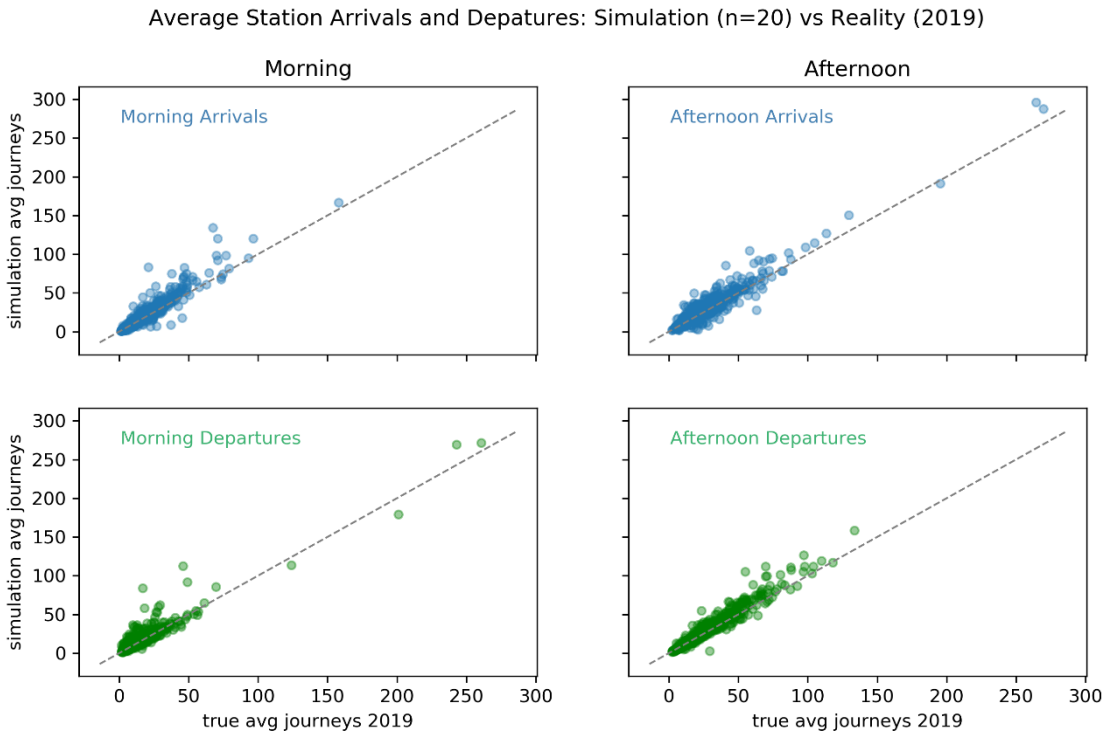

Figure 24 - Station daily journeys, split by arrivals / departures and morning / afternoon.
Each point is a station, the x-axis is the true average daily journeys in 2019 and the y-axis is the simulator average from 20 simulations.

Figure 24 shows that the station-level demand behaves realistically. The linear relationship in departures (green) shows that the simulator originates journeys realistically, and the linear relationship in arrivals (blue) shows that those journeys' destinations are chosen realistically. Finally, the fact this linear relationship is maintained in both the morning and afternoon shows that the temporal profiles of stations are realistic. Note that the large demand *from* transport hubs can be seen in the morning departures (note the points with > 100 journeys), and the reversed demand *to* those same stations can be seen in afternoon arrivals.

Some slight deviation from reality is expected, because mid-day rebalancing was not simulated, so certain stations may have filled or emptied more rapidly in the simulation than in reality.

11 Simulated BSS Scenarios

A number of scenarios were simulated to evaluate the impact of different interventions on the BSS. Each simulation will be described in some detail, and a combined summary is given in Table 1. To give a brief synopsis:

- Scenario 1 – the baseline, intended to simulate the true system.
- Scenario 2.1 – an 'idealised' system based on demand profiles, with more capacity and fewer bikes.
- Scenario 2.2 – an 'idealised' system with more capacity, but retaining the original number of bikes (allocating them *pro-rata* to spare docks).
- Scenario 3 – trialling improved allocation of bikes given the constraints of the existing system: no additional capacity and retaining the original number of bikes.

11.1 Scenario 1 – Baseline: True System, including depots

This scenario is intended to serve as a realistic baseline, including three depots, and is the same setup that was used for simulation validation in section 10.5. The depot capacities of the true BSS are not publicly available, although analysis in section 9.2.2 shows there is a buffer of *at least* 600 bikes in depots. The simulated depots were given generous capacities (see section 10.5 for details) based on the demand-profile analysis. The rationale here was, given a lack of exact information, to avoid penalising the baseline through insufficient depot capacity.

- Station capacities: true-to-life stations (as derived in section 10.4.1) and 890 docks of depot capacity
- Number of bikes: true-to-life, and 740 stored in depots.
- Initial bike allocations: true-to-life, based on 5:00am average (section 10.4.1).

11.2 Scenario 2.1 – Idealised allocations with extended capacity and *reduced* bikes

This setup used the idealised capacities and bike allocations derived from demand profiles in section 9.3.3 and extended the baseline system with an additional 916 docks (see geospatial visualisation in Figure 19). The capacity of stations was unconstrained and the largest station, excluding depots, had 103 docks (for reference the largest station in reality has 64 docks).

- Station capacities: Additional 916 docks over baseline.
- Number of bikes: A *reduction* of 2,624 bikes
- Initial bike allocations: RUE-based allocation Type A as described below.

I will refer to the bike allocation strategy as '*REU-based type A*' to distinguish it from modified versions used in scenarios 2.2 and 3. The allocation algorithm was as follows:

RUE-based allocation – Type A Algorithm

1. For each station:
    a. If the station is linked to a depot: no adjustment needed (continue to next station).
    b. Take the 'ideal capacity' c_i and 'ideal bike allocation' b_i derived in section 9.3.3 and the true station capacity c_t.
    c. If $c_i > c_t$
    then increase the station's capacity: $c_t \coloneqq c_i$
    d. Set bike allocation to: b_i

In other words, retain any existing capacity but extend stations as needed. Set the bike allocations to the exact value derived and leave any 'spare' capacity (when $c_i < c_t$) empty.

Whilst capacity was increased, the number of bikes was actually reduced by 2,624: a significant reduction which will be discussed in more detail with the results in section 12.

11.3 Scenario 2.2 – Redistributing bikes, with extended station capacity

This scenario was performed as an adjustment after observing the results of Scenario 2.1. It uses a similar allocation strategy, but modifies it to use all available bikes, by allocating 'spare bikes' to 'spare capacity' *pro rata*. I refer to this allocation algorithm as '*REU-based type B*'

RUE-based allocation – Type B Algorithm

1. Define an overall bike budget: B
2. For each station:
    a. If the station is linked to a depot: no adjustment needed (continue to next station).
    b. Take the 'ideal capacity' c_i and 'ideal bike allocation' b_i derived in section 9.3.3 and the true station capacity c_t.
    c. If $c_i > c_t$ then increase the station's capacity: $c_t \coloneqq c_i$
    d. Else if $c_i < c_t$ then note the station's spare capacity: $c_s \coloneqq c_t - c_i$
    e. Set station's bike allocation to b_i
    f. Subtract the allocated bikes from the remaining bike budget: $B \coloneqq B - b_i$
3. Sum the spare capacities across all stations to obtain total spare capacity: $S \coloneqq \sum c_s$
4. For each station with spare capacity:
    a. Increase bike allocation by: $round\left[\, B\left(\frac{c_s}{S}\right)\right]$

This scenario was intended to leave the size of the bike fleet true-to-life. Rounding the bike allocations to integers introduced rounding error which accumulated to having 141 more bikes than the baseline. This is a relative increase of 1.5% and was not thought to be significant enough to warrant complicating the algorithm further, so was left as-is.

There is a strong rationale for allocating spare bikes instead of discarding them: the demand profile of a station gives the near-minimal allocation of bikes and docks that will tightly accommodate expected net demand over the course of a day. However, as noted in previous literature (e.g. (Jian *et al.*, 2016)), it does not leave additional 'buffer' for daily stochasticity in demand. In practical terms this means that, so long as a station has the docks and bikes available to accommodate the expected net demand of the day, it is useful to 'pad' any spare capacity with bikes and docks, to buffer against stochasticity.

11.4 Scenario 3 – Redistributing existing bikes using existing capacity.

This setup, as with Scenarios 2.1 and 2.2, used bike allocations based on the demand-profile-derived recommendations, but in this case was constrained to use only the existing station/depot capacity and did not modify the size of the bike fleet. In other words, this scenario tested whether adjusting the distribution of bikes could yield any benefit, whilst controlling for other attributes in the system.

- Station capacities: true-to-life stations and 890 in depot capacity
- Number of bikes: true-to-life including 740 stored in depots.
- Initial bike allocations: REU-based type C, as described below

The bike allocation policy was similar to 'REU-based allocation Type B', but if true capacity was less than ideal then it would not be increased and instead the available capacity would be filled *pro rata*.

REU-based allocation Type C

1. Define an overall bike budget: B
2. For each station:
    a. If the station is linked to a depot: no adjustment needed (continue to next station).
    b. Take the 'ideal capacity' c_i and 'ideal bike allocation' b_i derived in section 9.3.3 and the true station capacity c_t.
    c. If $c_i > c_t$:
        i. Set station's bike allocation to: $round\left[c_t \frac{b_i}{c_i}\right]$
        ii. Subtract the allocated bikes from the remaining bike budget: $B := B - round\left[c_t \frac{b_i}{c_i}\right]$
    d. Else**:**
        i. Note the station's spare capacity: $c_s := c_t - c_i$
        ii. Set station's bike allocation to b_i
        iii. Subtract the allocated bikes from the remaining bike budget: $B := B - b_i$
3. Sum the spare capacities across all stations to obtain total spare capacity: $S := \sum c_s$
4. For each station with spare capacity:
    a. Increase bike allocation by: $round\left[B\left(\frac{c_s}{S}\right)\right]$

Table 1 - Summary of simulated BSS scenarios which were evaluated. Scenario 1 serves as the baseline. The key adjustments from baseline are highlighted in bold.

	Capacity		Bikes		Bike-to-dock ratio		Allocation Policy
	Station (initial)	Depot (intial)	Station (initial)	Depot (intial)	Excluding warehouses	Including warehouses	
Scenario 1	21,087	890	9,458	740	0.45	0.46	Empirical (5am avg)
Scenario 2.1	**22,003**	890	**6,834**	740	**0.31**	**0.33**	**REU-based (type A)**
Scenario 2.2	**22,003**	890	9,599	740	0.44	0.45	**REU-based (type B)**
Scenario 3	21,087	890	9,599	740	0.46	0.47	**REU-based (type C)**

12 Results

Each simulated scenario was run 20 times. The results are summarised in Table 2 and Table 3.

Table 2 - Simulation outcomes. Mean and confidence intervals, from 20 simulations per setup. 95% confidence intervals based on two-tailed t-distribution for 19 degrees of freedom

	Failed Starts		Failed Ends		Combined Faults		Finished Journeys	
	mean	95% CI	mean	95% CI	mean	95% CI	mean	95% CI
Scenario 1	1,507	± 38	2,981	± 110	**4,487**	± 115	29,064	± 77
Scenario 2.1	2,320	± 49	637	± 46	**2,957**	± 75	28,157	± 68
Scenario 2.2	1,243	± 31	1,017	± 57	**2,260**	± 70	29,292	± 53
Scenario 3	1,280	± 35	2,535	± 92	**3,815**	± 103	29,217	± 64

Table 3 - Simulation outcomes. Absolute and relative difference from baseline mean.

	Failed Starts		Failed Ends		Combined Faults		Finished Journeys	
	Absolute	Relative	Absolute	Relative	Absolute	Relative	Absolute	Relative
Scenario 1	-	-	-	-	-	-	-	-
Scenario 2.1	+813	+54%	-2,343	-79%	**-1,530**	**-34%**	-907	-3%
Scenario 2.2	-264	-18%	-1,964	-66%	**-2,227**	**-50%**	+228	+1%
Scenario 3	-227	-15%	-445	-15%	**-672**	**-15%**	+153	+1%

The following figures illustrate the average distribution of simulated faults within a given scenario. Within a given plot: the size of the marker is proportional to the number of combined faults, and the colour indicates whether those faults tended to be failed starts or failed ends.

Note: the size of markers should not be compared between plots as the scale is different for each plot. Please refer to the captions or Table 2 for the total number of faults.

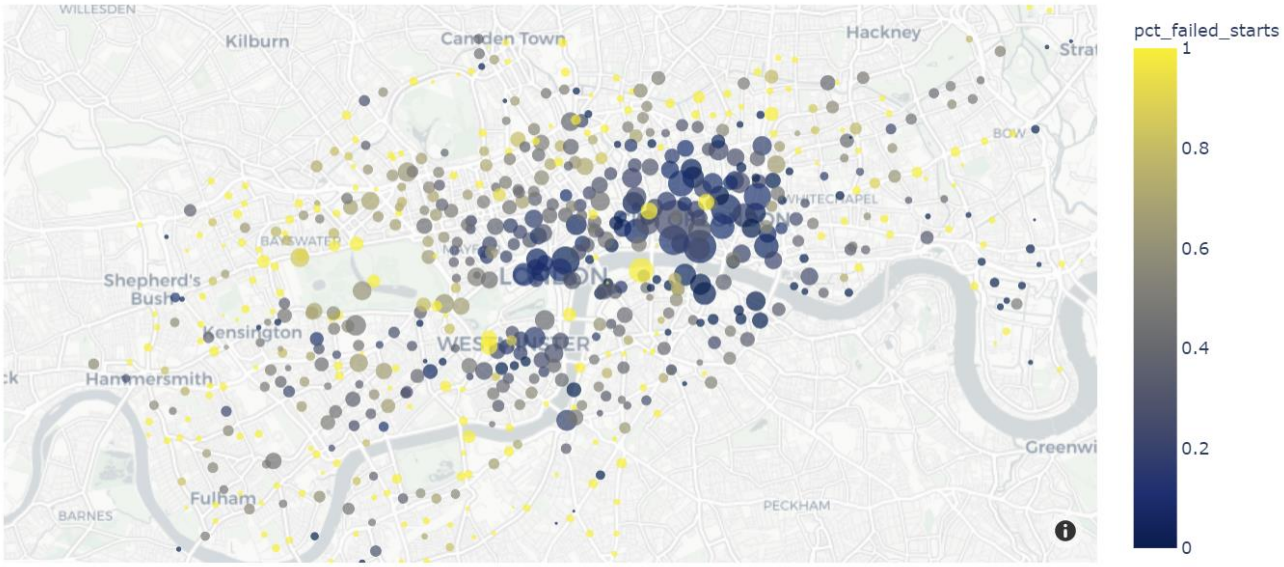

Figure 25 - Scenario 1 (baseline). Total of 4,487 faults. Map tile is property of CARTO.

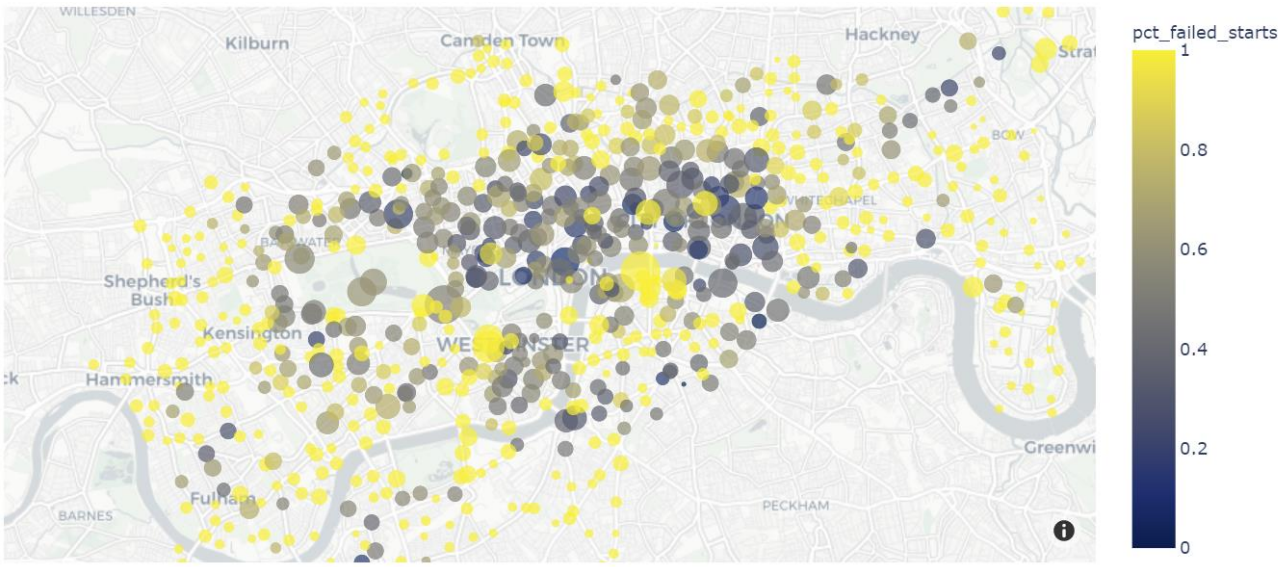

Figure 26 - Scenario 2.1. Total of 2,957 faults. Map tile is property of CARTO.

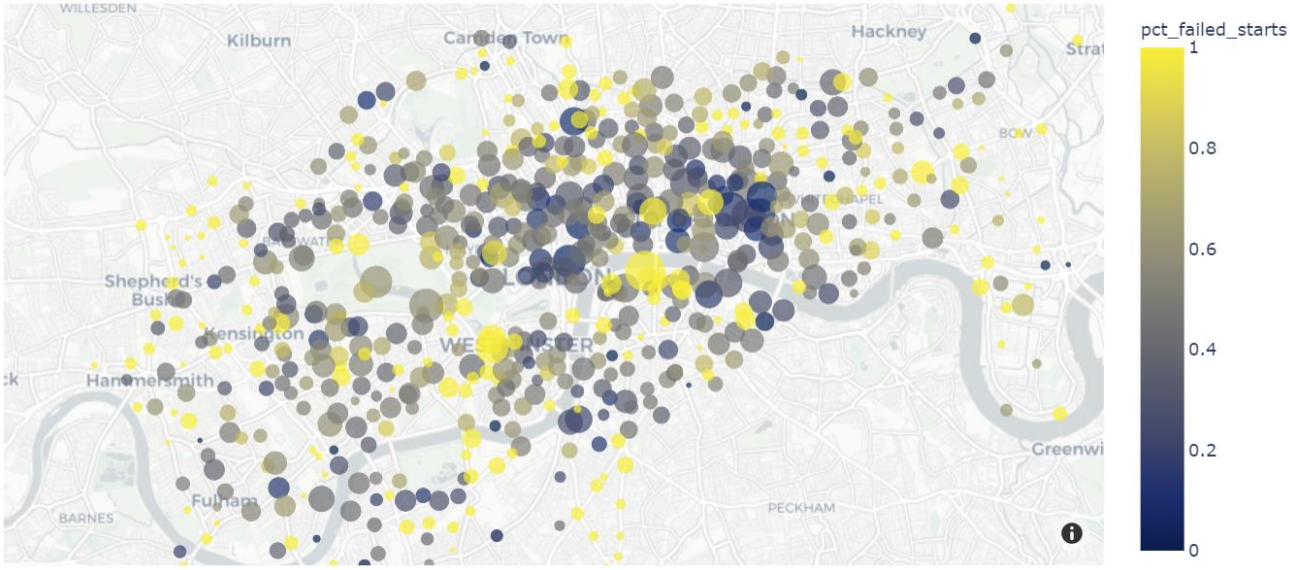

Figure 27- Scenario 2.2. Total of 2,260 faults. Map tile is property of CARTO.

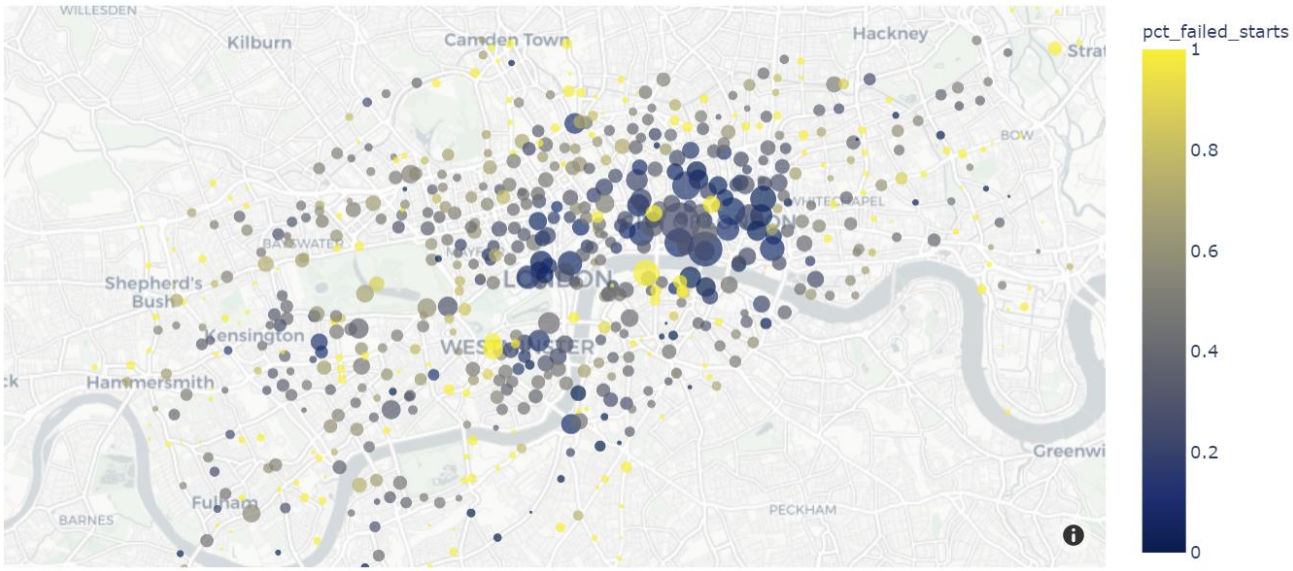

Figure 28 - Scenario 3. Total of 3,815 faults. Map tile is property of CARTO.

12.1 Discussion of Evaluated Scenarios

Logistically, Scenario 3 was the smallest intervention as it simply involved distributing bikes differently ahead of the morning rush without changing bike numbers or station capacities. Despite being constrained to the existing system, Scenario 3 saw a modest but statistically significant 15% reduction in faults. Failed starts and failed ends dropped in equal proportion. This suggests that some adjusted overnight rebalancing without increasing system capacity could slightly improve performance in the system.

Logistically, Scenario 2.1 was the most significant adjustment to the system. This scenario was the most literal interpretation of the demand profiles and involved an increase to station capacity as well as a considerable reduction to the bike fleet. This scenario reduced combined faults by 34%, and reduced failed ends by the largest amount (-79%). However, a side effect of reducing the bike fleet was that failed starts actually increased. Some of the reduction in failed ends must be seen with the context that an additional 813 journeys failed to begin at all. Operators generally consider failed ends to be more disruptive to users than failed starts (Médard de Chardon, Caruso and Thomas, 2016), so reducing the bike fleet to increase failed starts and reduce failed ends may in fact be a legitimate strategy, but without further information on the relative penalty it is better to treat both faults as equal.

Continuing the evaluation of Scenario 2.1: the geospatial distribution of the faults (Figure 26) showed that the increased capacity in employment centres helped to reduce the ‘hotspots’ for failed ends. However, stations on the periphery of the system in particular seemed to now be suffering from a shortage of bikes. Despite the attempts to make the bike allocations conservative, it appears that the demand profiles still under-estimated the number of bikes needed. Furthermore: reducing the bike fleet to this extent as a strategy to accommodate commuters is unlikely to be a viable solution, and would likely have a negative impact on BSS performance on weekends. This conclusion motivated the formulation of Scenario 2.2.

Scenario 2.2 struck a balance between Scenario 2.1 and Scenario 3. It still involved an increase in station capacity, but did not make the intervention of reducing the bike fleet. Instead, any spare bikes were allocated to stations that had spare capacity. This provided an extra buffer to many stations where the demand profiles had under-estimated the need for bikes. As a result, this scenario had the greatest reduction in combined faults (-50%). Crucially, it achieved a reduction in failed ends (-60%) whilst still reducing failed starts (-18%) which

means its performance compares favourably to Scenario 2.1. Scenario 2.2 resulted in the closest balance between failed starts and failed ends at 1,243 and 1,017 respectively, and the geospatial plot (Figure 27) showed a fairly even distribution of faults with the former ‘hotspots’ significantly less prominent.

13 Conclusions and Critical Evaluation

13.1 Limitations of Measuring Demand

It is important to view these results with the knowledge that the simulations will be an imperfect model of the counterfactual scenarios that were trialled.

The Poisson demand generation in the simulator was derived from past journeys, as were the demand profiles in the analysis. A limitation of this approach is that past journeys are themselves shaped by availability in the system, which introduces a circular element to measuring past demand.

If a station has periodically been allowed to be full or empty:

1. Journeys may have been directly prevented from beginning or ending at the station.
2. A secondary effect is that would-be-users may not form the habit of using the station, if it is regularly seen to be unusable.

(O'Mahony, 2015) addressed the first point by overlaying demand with station data and censoring records where station fill was likely to have suppressed observed demand. The station data available for this project had a short time span, so O'Mahony's approach couldn't be used without severely restricting the journey data to a handful of months. Even if censoring could have been used, it is impossible to fully control for the effect of existing service-level on demand. The results of this project, as with many results in the field, should be viewed with this limitation in mind.

Additionally, it should be noted that the simulator models station demand as independent from other stations, but in reality there is likely to be complex interactions between stations. For example: if a popular destination station, *a*, is often full or empty then some users may opt to use a nearby station, *b*, as a more reliable alternative. If *a* had its capacity increased then the demand for station *b* might fall as a consequence. The simulator developed during this project will not reflect these interactions so will not be a perfect model of such counterfactual scenarios. However, the results still demonstrate that certain changes could accommodate *existing* demand more effectively.

13.2 Implications of Results

The project sought to investigate whether the London BSS's reliance on truck-based mid-day rebalancing could be reduced by improving the ability of the system to passively accommodate the pulse of commuter demand. The results support that this is theoretically possible if capacity is increased in central stations. In simulations where there was *no* reactive rebalancing by trucks, performance could be still be considerably improved by increasing certain station capacities, particularly those in employment centres.

Increasing station capacities in real life is, of course, not always practical. There is a cost-benefit trade-off to adding docks to a station, both financially and in terms of how the space could otherwise be used. Naturally there are also physical constraints on how large a docking station can be, particularly in those dense employment centres which would benefit most from increased capacity. Still: if a station must have its capacity subsidised by a vehicle once or twice a day for most weekdays in the year, then the cost of failing to increase capacity will accumulate with time.

None of the simulated scenarios were fault-free, so it is apparent that there is a legitimate need for reactive mid-day rebalancing. Stochasticity in demand will inevitably result in some stations filling, or emptying, faster than is expected. Given the fact that it is not always practical to extend stations, mid-day rebalancing could also

be used to extend the reach of nearby depots as a compromise. Indeed, this is very likely to already be a tactic applied to the Waterloo, Holborn and King's Cross depots.

Another consideration is: even if increased capacity can offset *mid-day* rebalancing, it doesn't necessary eliminate the need for some end-of-day rebalancing. Setting a desired bike allocation for stations carries the assumption that this allocation will be restored each day, so inevitably there will still be a need to rebalance the stations that are not self-balancing (see Figure 15 for an example of a transaction negative station). This rebalancing could be performed overnight without contending with time pressure, road-congestion and a rapidly changing system state, so is likely to be more efficient than the frantic mid-day rebalancing which this project has focused on offsetting. But, in reality there will be an optimal trade-off between striving for an ideal static allocation during the night, and reacting to actual demand during the day. (Schuijbroek, Hampshire and van Hoeve, 2013) addressed both the problem of ideal morning allocations *and* how to route vehicles to achieve them overnight, so this is clearly an area that could be investigated in more detail.

Deriving suggested capacities and bike allocations using demand profiles was shown to be reasonably effective: Scenario 3 showed a modest improvement by simply reallocating bikes in the existing system. However, it was apparent in Scenario 2.1 that the derived allocations were suboptimal and underestimated the number of bikes that were needed. If time had permitted, additional approaches such as the Markov chain-based allocations used in (O'Mahony, 2015; Schuijbroek, Hampshire and van Hoeve, 2013) could have been trialled.

To conclude, the results of this project suggest that increased capacity, and/or modified overnight rebalancing could improve the ability of the London BSS to accommodate commuter demand during the day. This implies that such interventions could offset some, but likely not all, of the mid-day rebalancing which currently takes place. Further study, ideally in partnership with TfL, could shed more light on the cost-benefit trade-off between increasing station capacities and relying on station-extension-rebalancing: potentially in the form of an optimisation problem. This is a pertinent area of investigation as London continues to combat air-quality issues; and as cycling adoption in the UK surges in response to the COVID-19 pandemic.

14 References

DeMaio, P. (2009) 'Bike-sharing: History, Impacts, Models of Provision, and Future', *Journal of Public Transportation,* 12.

Fishman, E., Washington, S. and Haworth, N. (2014) 'Bike share's impact on car use: Evidence from the United States, Great Britain, and Australia', *Transportation Research Part D: Transport and Environment,* 31, pp. 13-20.

IanVisists (2014) 'A look around the Barclays Cycle Hire Repair Depot'. Available at: https://www.ianvisits.co.uk/blog/2014/01/20/a-look-around-the-barclays-cycle-hire-repair-depot/ (Accessed 03/31 2020].

Jian, N., Freund, D., Wiberg, H. M. and Henderson, S. G. 'Simulation optimization for a large-scale bike-sharing system', *2016 Winter Simulation Conference (WSC)*, Washington, DC, pp. 602-613.

Médard de Chardon, C., Caruso, G. and Thomas, I. (2016) 'Bike-share rebalancing strategies, patterns, and purpose', *Journal of Transport Geography,* 55, pp. 22-39.

O'Brien, O., Cheshire, J. and Batty, M. (2014) 'Mining bicycle sharing data for generating insights into sustainable transport systems', *Journal of Transport Geography,* 34, pp. 262-273.

O'Mahony, E. (2015) *Smarter Tools for (Citi)Bike Sharing.* Computer Science, Cornell University, Cornell University Library [Online] Available at: https://ecommons.cornell.edu/handle/1813/40922 (Accessed: 03/04/2020).

Schuijbroek, J., Hampshire, R. and van Hoeve, W.-J. (2013) 'Inventory Rebalancing and Vehicle Routing in Bike Sharing Systems', *European Journal of Operational Research*, 257.

Schuijbroek, J., Hampshire, R. C. and van Hoeve, W. J. (2017) 'Inventory rebalancing and vehicle routing in bike sharing systems', *European Journal of Operational Research,* 257(3), pp. 992-1004.

Soriguera, F., Casado, V. and Jiménez, E. 'A simulation model for public bike-sharing systems'. 139-146.

15 Appendices

15.1 Appendix 1 - Altered Usage due to COVID-19

The core aims of the project relate to a ‘typical’ weekday and it was therefore important to establish when the COVID-19 pandemic began to impact usage of the system, so that affected data could be excluded from analysis and the parameterisation of simulations when necessary. Figure 29 and Figure 30 show how usage of the system changed in late March.

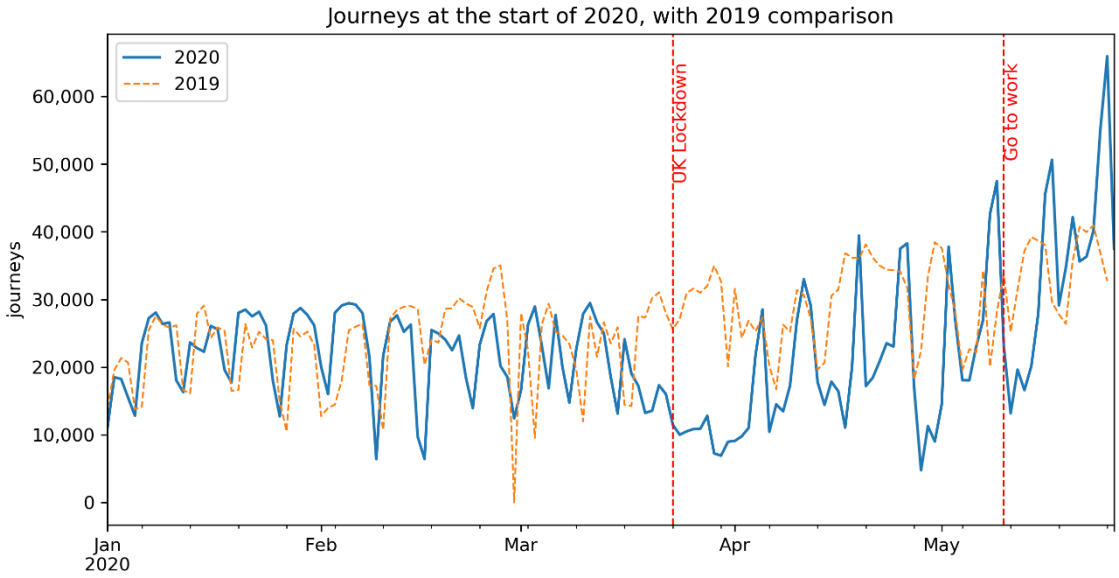

Figure 29 - Daily journey volumes in the initial months of 2020, contrasted with 2019. The volume of usage drops substantially before and during lockdown. Then volumes increase but with a noticeably altered weekly profile.

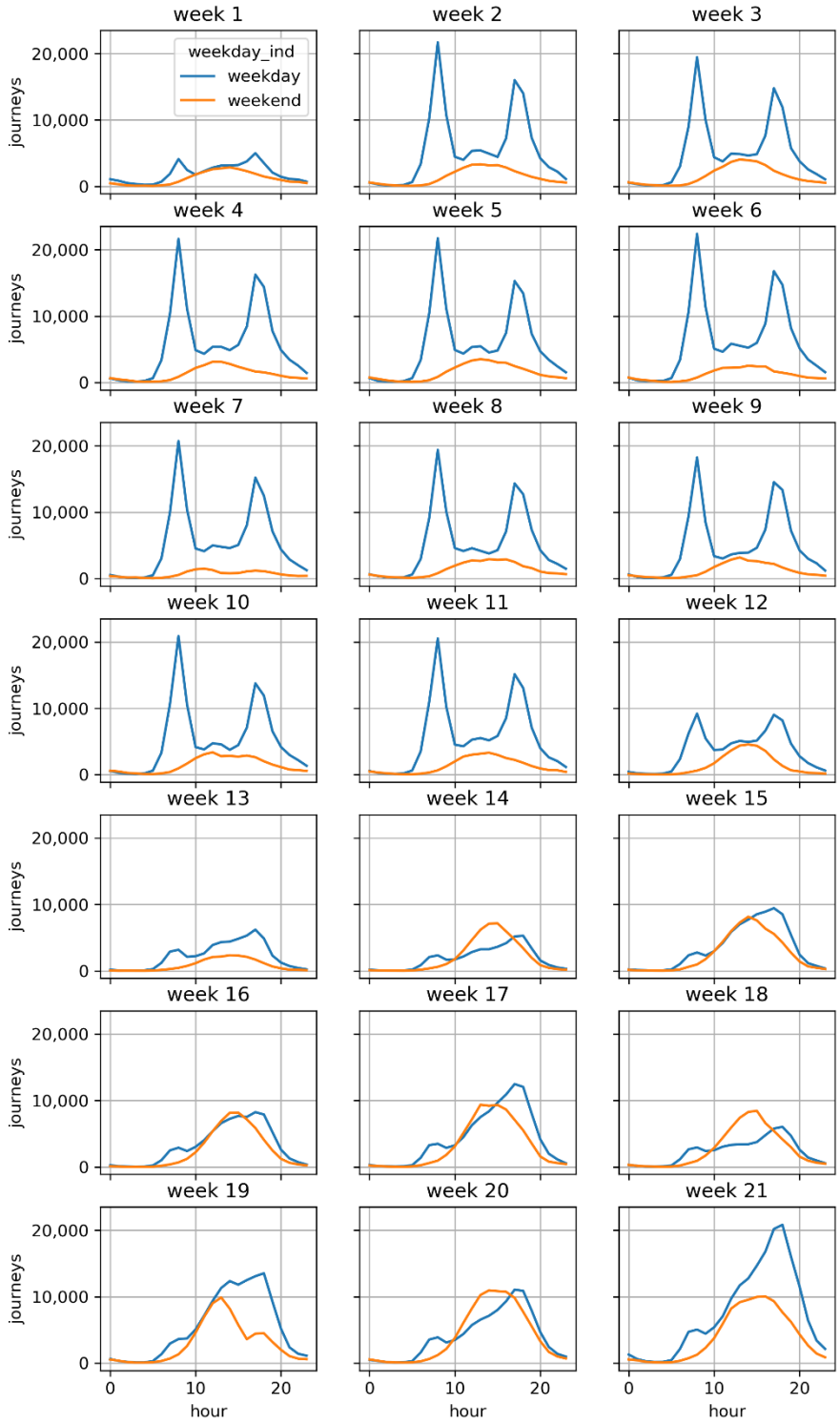

Figure 30 - Journey volumes over time of day, aggregated per week in 2020. The impact of COVID-19 can be observed around week 12 onwards.

It can be seen that weekday journey volumes dropped noticeably during the 12th week of 2020, and remained suppressed for a number of weeks. Even when journeys eventually began to pick-up, usage was significantly altered from the norm. The two rush-hour peaks on weekdays were considerably smaller and in general the weekday usage skewed towards the evenings. Meanwhile, usage on weekends actually increased: potentially due to reduced road traffic and the fact that people were permitted to leave the house for one form of exercise,

including cycling, during the lockdown. It was decided to define 15/03/2020 (Sunday of week 11) as the last day of typical usage. Subsequent analysis and the data used to parameterise simulations excluded later dates.

15.2 Appendix 2 – Top 20 Stations for Estimated Capacity Shortfall

The following table lists the top 20 stations as ranked by the estimated shortfall in capacity. The italicised stations are understood to already be served by depots: the shortfall is overstated in these cases.

bikepoint id	common name	true capacity	estimated ideal capacity	estimated ideal morning bikes
154	*Waterloo Station 3, Waterloo*	*35*	*276*	*267*
14	*Belgrove Street, King's Cross*	*48*	*271*	*257*
374	*Waterloo Station 1, Waterloo*	*36*	*219*	*211*
66	*Holborn Circus, Holborn*	*40*	*188*	*0*
361	*Waterloo Station 2, Waterloo*	*55*	*151*	*151*
71	Newgate Street, St. Paul's	34	103	0
251	Brushfield Street, Liverpool Street	34	92	1
579	Queen Street 2, Bank	36	75	0
703	St. Bride Street, Holborn	25	61	0
228	St. James's Square, St. James's	40	75	0
427	Cheapside, Bank	43	76	2
109	Soho Square, Soho	57	88	1
546	*New Fetter Lane, Holborn*	*21*	*48*	*8*
45	Boston Place, Marylebone	26	50	50
762	Storey's Gate, Westminster	21	45	0
108	Abbey Orchard Street, Westminster	29	52	0
336	Concert Hall Approach 2, South Bank	18	37	19
428	Exhibition Road, Knightsbridge	20	37	0
436	Red Lion Street, Holborn	36	53	7
583	Abingdon Green, Westminster	20	37	1

15.3 Appendix 3 – Project Repository Overview

The following gives an overview of the key packages and files in the project repository. The list is not comprehensive but is intended to give a good orientation. /-- denotes objects that are nested below the directory above. Various 'README' files are also present in the repository and can be consulted for further details.

notebooks/	Notebooks that were used for exploration and visualisation. Notebooks ending with 'MSP-n' are more structured: others are more experimental.
tfl_project/	Root for source code and data.
/-- create_sqlite_database.py	If csv data is ready, this script will prepare the SQLite database by calling various other scripts.
/-- database_creation/	Various scripts called by create_sqlite_database.py.
/-- cycle_journey_prep/	subpackage for downloading, combining and preparing cycle data as CSVs.
/--/-- combineCycleData.py	Script downloads and combines journey CSVs.
/--/-- clean_combined_cycle_data.py	Script cleans the combined journey CSV if it exists.
/-- data/	Various data inputs and outputs. Data itself is not stored in version control.
/-- simulation/	Subpackage for simulations.
/--/-- scenario_scripts/	Scripts for running the various scenarios that were trialled in the paper.
/--/-- tests/	Automated tests for the simulator.
/--/-- city.py	Module for the City and Agent classes.
/--/-- sim_managment.py	Module for the LondonCreator and SimManager classes.
/--/-- station.py	Module for the Store classes.
/-- tfl_api_logger	Subpackage for calling TfL API data.
/--/-- bikeStationStatus.py	Script for calling the TfL API and appending station data to CSV.
/--/-- logging_functions.py	Multipurpose logging functions.
environment.yml	YAML file for recreating the Conda environment.