

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

London's Bus Network Design and Frequency Setting After the Pandemic

Manuel Caetano Mariano Filgueiras

WORKING VERSION



Mestrado Integrado em Engenharia Civil

Supervisor: Sara Maria Pinho Ferreira

September 27, 2021

London's Bus Network Design and Frequency Setting After the Pandemic

Manuel Caetano Mariano Filgueiras

Mestrado Integrado em Engenharia Civil

September 27, 2021

Abstract

The impact of Covid-19 on bus transit network systems in cities worldwide is characterised by the low affluence of passengers and the ensuing disruptions of regular service. A Transit Network Design Frequency Setting model is proposed to cope with the post-pandemic crisis and the new expected demand.

The multi-objective transit network design and frequency setting problem (TNDFSP) involves finding routes and their associated frequencies to operate public transport systems in an urban area. The TNDFSP is a complex combinatorial optimisation problem with an ample search space and multiple constraints. The proposed model aims to mitigate the impacts of the crisis, where people are still reluctant to travel in public transport, mainly if in-vehicle crowding exists.

The objectives are to improve the network without incurring massive changes to its composition and minimise passenger costs, which essentially concerns the travel times: waiting time, in-vehicle travel time and transfer time. The proposed model includes a Route Generation Algorithm (RGA), a simple heuristic that generates additional lines to the existing ones to serve the most demanding trips directly, and a Passenger Assignment (PA) and Frequency Setting (FS) models, two Mixed-Integer Programmes (MIP) that distribute the demand through the updated bus network and set the optimal number of buses for each of the lines.

Computational experiments are conducted on a test network and further on the network comprising the boroughs of Kensington and Chelsea, London. The model provides good results in the test network with a list of the specified operator and Covid-19 variables that are not considered in other literature. However, due to the formulation of the PA and FS models, the model has limitations for more prominent representations in terms of memory and computational times, including Kensington and Chelsea's Network.

Keywords: Transit Network Design, Frequency Setting Problem, Covid-19 Pandemic, Bus Optimization, Route Generation algorithm, Mixed-Integer Programme

Resumo

O impacto da Covid-19 nas redes de autocarros pelas cidades de todo o mundo é caracterizado pela baixa afluência de passageiros e pelas interrupções do serviço regular que se lhe seguiram. Propõe-se um modelo para definir as linhas e determinar as frequências (TNDFSP) para uma rede de autocarros de forma a lidar com a crise pós-pandemia e acautelar os novos padrões de procura esperados.

O planeamento de rede com multi-objectivos e o problema de configuração de frequência (TNDFSP) envolve encontrar rotas e suas frequências associadas para operar num sistema de transporte público de uma área urbana. O TNDFSP é um problema complexo de otimização combinatória com amplo espaço de busca e múltiplas restrições. O modelo proposto visa mitigar os impactos da crise, enquanto as pessoas ainda mostram relutância em viajar no transporte público, principalmente se houver sobrelotação dos veículos.

Os objectivos deste trabalho são melhorar a rede sem provocar alterações maciças na sua composição e minimizar os custos do ponto de vista dos passageiros, que dizem essencialmente respeito aos tempos de deslocação: tempo de espera, tempo de viagem num veículo e tempo de transferência entre linhas. O modelo proposto contempla um Algoritmo de Geração de Rota (RGA), uma heurística simples que gera linhas adicionais às existentes para atender as viagens mais procuradas diretamente, e os modelos de Atribuição de Passageiros (PA) e Configuração de Frequência (FS) que distribuem a procura pela rede de autocarros (após esta ser atualizada) e define a frequência de autocarros para cada uma das linhas.

Foram conduzidos testes computacionais numa rede de teste e, posteriormente, na rede que compreende os bairros de Kensington e Chelsea, em Londres. Os modelo fornece bons resultados na rede de teste tendo em conta especificações provenientes do operador e novas variáveis relacionadas com o período pós Covid-19 não contempladas na literatura existente. Contudo, devido à natureza matemática dos modelos PA e FS, existem limitações para representações mais proeminentes, inclusive na rede que serve a zona de estudo.

Palavras-chave: Desenho de Rede de Autocarros, Problema de Configuração de Frequência, Pandemia Covid-19, Otimização de Redes de Autocarros, Algoritmo de Geração de Rotas, Modelo Linear Misto

Acknowledgements

Throughout the writing of this dissertation, I have received a great deal of support while learning a lot.

I would like first to thank my supervisor at the University of Twente, Professor Konstantinos Gkiotsalitis, for the opportunity to take part in such an interesting project. His expertise was invaluable in formulating the research steps and methodology, and his insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

Secondly, I would like to thank my tutors, Dr Menno Yap from Transport for London and Dr Oded Cats from the University of Delft, for their valuable guidance throughout these studies, providing me with the tools and insights needed to successfully complete my dissertation.

My supervisors at the University of Porto, Professors Sara Ferreira and António Lobo, are thanked for their availability and support throughout this work, especially during my stay in The Netherlands. Their review and guidance towards the dissertation were most helpful.

Completing this dissertation would also have not been possible without the help of my good friend Joel Coelho who taught me how to programme through many days and nights struggling with the models. Thank you for everything!

In addition, I would like to thank my parents, brother, cousin and uncle for their wise counsel and sympathetic ear. You are always there for me. Finally, I could not have completed this dissertation without the support of my friends from FEUP and from the UT/Erasmus that provided stimulating discussions and happy distractions to rest my mind outside of my research. Last, a special thanks to Filipa, the dearest person who accompanied me closely on this journey.

This work is a result of project DynamiCITY: Fostering Dynamic Adaptation of Smart Cities to Cope with Crises and Disruptions, with reference NORTE-01-0145-FEDER-000073, supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

Manuel Filgueiras

“The best way out is always through”

Robert Frost

Contents

1	Introduction	1
2	Literature Review	5
2.1	Contributions	10
3	Solution Framework	11
3.1	Line Generation Model	13
3.2	Lower-Level Model: Passenger Assignment Model	15
3.2.1	Assumptions for the Passenger Assignment Model	15
3.2.2	Objective Function for the Passenger Assignment Model	18
3.3	Upper-Level Model: Frequency Setting Model	19
3.3.1	Assumptions for the Frequency Setting Model	19
3.3.2	Objective Function for the Frequency Setting Model	22
4	Model Application	25
4.1	Case Study - Toy Network	25
4.1.1	Results Regarding the Toy Network	26
4.2	Case Study - Area of the London Network	29
4.2.1	Results Regarding the London Network	29
5	Concluding Remarks	35
A	Code Script	37
A.1	Line Generation Script	37
A.2	Passenger Assignment and Frequency Setting Script	43
	References	55

List of Figures

1.1	London’s Bus and Tube Passenger Ridership Through the Pandemic	2
3.1	Solution Framework	12
3.2	Adapted Route Generation Algorithm	14
3.3	In-Vehicle Crowding Penalty Function	20
3.4	Passengers’ Assumed Waiting Time <i>Versus</i> Vehicle Time-Headways	21
4.1	Example of a Test Network	25
4.2	Component Representation of the Test Network	26
4.3	Volume of Direct and Transfer Passengers	27
4.4	Chelsea’s Bus Lines	29

List of Tables

2.1	Summary of the TNDFSP Literature	9
4.1	Example OD Matrix	26
4.2	Parameters Considered for the Test Network	27
4.3	Output of The Bi-Level Model Using the Test Network	28
4.4	Parameters Considered for The Chelsea Network	30
4.5	Output of the Line Generation Model	30
4.6	Output of the PA and FS Models in Chelsea's Network	31
4.7	Output of the PA and FS Models in the Generated Network	32
4.8	Output of the PA and FS Models in the Combined Networks	33

Abbreviations

TfL	Transport for London
OD	Origin-Destination
TNDFSP	Transit Network Design and Frequency Setting Problem
NP	Nondeterministic Polynomial Time
TNDP	Transit Network Design Problem
TNFSP	Transit Network Frequency Setting Problem
MIP	Mixed-Integer Programme
MINLP	Mixed-Integer Nonlinear Programme
MILP	Mixed-Integer Linear Programme
LP	Linear Programme
UTDNP	Urban Transit Design Network Problem
TND	Transit Network Design
FS	Frequency Setting
TNT	Transit Network Timetabling
VSP	Vehicle Scheduling Problem
DSP	Driver Scheduling Problem
DRP	Driver Rostering Problem
ABC	Artificial Bee Colony Algorithm
GA	Genetic Algorithm
MOPSO	Multi-Objective Particle Swarm Optimisation
SA	Simulated Annealing
MA	Memetic Algorithm
TS	Tabou Search
NS	Neighbourhood Search
GS	Greedy Search
LG	Line Generation
RGA	Route Generation Algorithm
PA	Passenger Assignment
PTN	Packet Transport Network
BPR	Bureau of Public Roads
PWL	Piecewise Linear Function
RAM	Random-Access Memory

Chapter 1

Introduction

After the detection, in December 2019, of the COVID-19 virus, the risks of transmitting the airborne infectious disease have held an enormous impact on global mobility [Zhou et al., 2020]. The accelerated propagation of the virus is due to the extent of connections [Cartenì et al., 2021] and the urgency to restrict contact called for many governments to act and implement travel limitations and lockdowns [Przybyłowski et al., 2021]. Although the epidemic situation still varies from region to region, as mentioned in [Ghosh et al., 2020] and in [Cartenì et al., 2021], cities with high affluence to public transport are prone to a higher transmission rate. This sector is, in fact, one of the most disrupted, following implications like reduced patronage, operational impacts, and emerging financial issues [UITP, 2020].

A sharp decline in the service's affluence is depicted in the user's statistics of almost every city, whose pandemic situation is not under control [Cui et al., 2021]. For instance, in the early stages, cities like London saw a reduction of 85% of bus passengers and 95% of the subway [TfL, 2019]. In New York, there was a 94% drop in the tram ridership [Wilbur et al., 2008], while in Budapest and Santander, there was a 90% and 93% reduction in public transport usage, respectively [Aloi et al., 2020] [Bucsky, 2020].

Regarding operational impacts, the decrease in ridership took some operators to limit their service, close stations, and suspend routes [UITP, 2020]. In London, a reduction of the service frequency generated criticism, as some bus lines became overcrowded [UITP, 2020]. When limiting the service, bus operators should not reduce bus frequencies by a fixed percentage of the complete network frequency. This option is not wise as lines have different demands, requiring a weighted distribution of buses to avoid overcrowding when the demand exceeds the supply and empty lines when the supply exceeds demand.

Besides the lessened number of passengers, the public transport operations were subjected to new health guidelines that introduced capacity restraints to the vehicles and requested operators to increase service frequency to ensure social distancing [UITPb, 2020]. For example, London and Beijing limited the number of available seats in-vehicle by 50% [TfL, 2020], [UITP, 2020]. In Poland, the public transport capacity changed accordingly to the stage of spread, 100%, 70%, and

50%, respectively [Wielechowski et al., 2020]. The rapid adaptations in cities like London have mainly been formulated with ad-hoc knowledge [Gkiotsalitis and Cats, 2020], which leaves room for improvements and optimization in the post-Covid-19 future.

Now public transport companies want to tackle decisions that accommodate the near-future scenario in which the expected demand will grow as measures are released and as signs of progress to contain the virus are made in the medical field. However, the predicted turnout is sharpened down compared to the pre-pandemic levels before 2020 [Ratho and Lourdes John, 2020]. In Figure 1.1, one can observe that both the tube and bus service usage in London since the pandemic began until the writing of this text is still fluctuating under the pre-levels of January 2020.

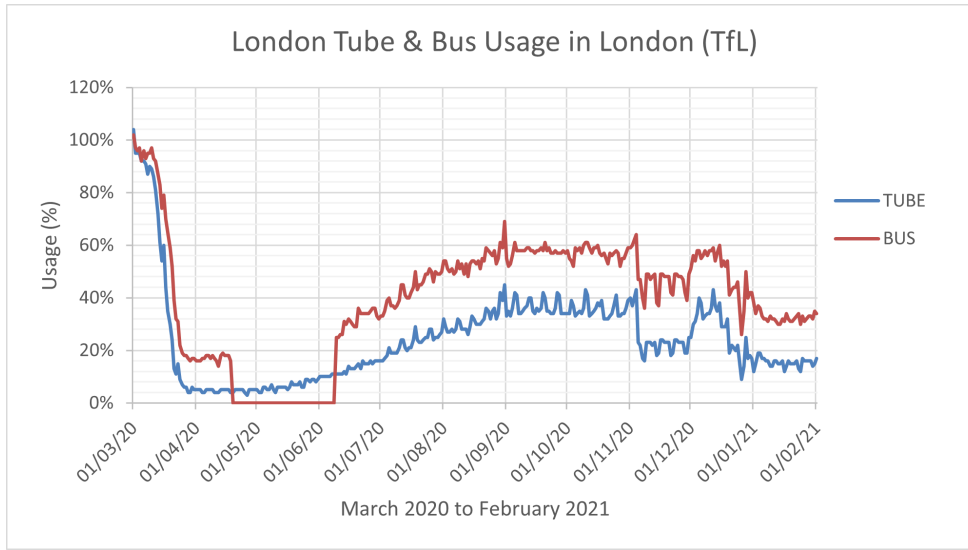


Figure 1.1: London's Bus and Tube Passenger Ridership Through the Pandemic (Data imported from UK's Government public database)

The figure is based on London's public transport operator (Transport for London) and the UK government's publicly available data.

A combined route design and frequency setting model is proposed with the aim of optimizing the bus service by reducing the perceived travel time¹ and contemplating the predicted passenger demand for the post-Covid-19 scenario. The model is based on the presumption of having an existing bus network that is to be optimized given an expected projection of the future trips that travellers are likely to take, i.e. an Origin-Destination matrix (OD matrix). The model re-designs the network on the principle of adapting and not incurring massive changes to the current composition. The latter is done by adapting the existing routes to meet the demand and creating an extra set of short lines tailored to satisfy the most demanding trips more directly. The frequency setting assigns the number of vehicles to the lines according to the distributed volume of passengers through the network.

¹[Meng et al., 2018] have defined the former as "the duration that the passenger felt that he/she was spending between the departure and arrival."

The Transit Network Design and Frequency Setting Problem (TNDFSP) is an NP-hard problem in which every sub-problem is hard to solve. The complexity of an extended network holds several operational variables and restrictions that are difficult to tackle all at once. The hierarchy of the formulation separates the Transit Network Design Problem (TNDP) from the Transit Network Frequency Setting Problem (TNFSP). The TNDP is addressed with a simple heuristic to generate extra lines based on the shortest paths of the highest demand OD pairs. The independent output of this model is used to resolve the TNFSP. The TNFSP iteratively comprises two mixed-integer programmes (MIP) that combined sort passengers through the lines and assign their optimal frequency to reduce passenger costs. Further analysis of the results can help the planner cut operational costs by discarding low-frequency lines, mainly the pre-existing little-used ones.

The model is tested on an example test toy network and then applied to an area of London comprising the Kensington and Chelsea boroughs. TfL provided the supply data of this network. In the absence of real OD demand data, an artificial OD matrix has been used in this study. The results show that the model is capable of providing reasonable solutions to the test toy network. Nevertheless, for computational feasibility, the MIP programmes make it infeasible to process the complete demand on Chelsea's network. A partial OD was used for this network.

This model is different from the existing literature combined models to attend TfL specifications and the post-Covid-19 scenario, where passengers' perception towards using public transport is changed.

The paper's remaining framework is divided into chapters—the literature review regarding different approaches of the TNDFSP is presented in Chapter 2. Chapter 3 regards the followed solution framework and the formulation of the models. Numerical experiments on different networks are conducted, and the individual results are depicted in Chapter 4. Chapter 5 concludes the dissertation with additional remarks for future research and development. Last, Appendix A provides the code script used to run the models in Python language.

Chapter 2

Literature Review

To our best knowledge, not much work has been done on the post-COVID-19 impact on the spatio-temporal passenger perception and demand. To better tackle the question in the short future, operators need to re-plan and improve the service in order to meet passengers' specifications and regain their trust in public bus transport systems.

According to [Gkiotsalitis and Cats, 2020], the planning decisions of the Urban Transit Design Network Problem (UTDNP) in the post-shutdown phase will be of three different types: strategic, operational, and tactical.

Our planning is done strategically by combining the transit route design and the frequency setting problem. According to [Guihaire and Hao, 2008], both aspects are correlated and shall be attended to improve the solution quality simultaneously. Therefore, the transit network designing and frequency assignment should be considered simultaneously to minimise passenger perceived travel time and meet all the expected demand. The output is the set of routes and frequency associated with our specific network topology, the set of desired passenger trips, and the travel time between the links connecting stops.

The vast literature on planning the bus services, considering a sequential methodology of the typical sub-problems¹, is retrieved in detail in [Ibarra-Rojas et al., 2015]. In this section, the aim is to synthesise studies that can contribute to the after-pandemic situation and our specific-case scenario, meaning the ones that tackle the TNDFSP.

Upon examining the literature on the TNDFSP, one can identify three different approaches: mathematical, heuristics and meta-heuristics formulations. Sometimes, authors even use more than one strategy to get more efficient or optimal solutions. One example of the latter is the work of [Szeto and Wu, 2011], which aims to reduce the passenger waiting, transfer and travel times in the small area of Tin Shui Wai in Hong Kong. The problem formulated as a mixed-integer nonlinear program uses a genetic algorithm hybridised with a neighbourhood search heuristic to design the network and solve the frequency setting problem. Another example is the work of [Wei Fan and Machemehl, 2004], and [Fan, 2006], who formulated multi-objective nonlinear

¹Due to the complexity, the UTDNP is usually sub-divided into six sub-problems: Transit Network Design (TND), Frequency Setting (FS), Transit Network Timetabling (TNT), Vehicle Scheduling Problem (VSP), Driver Scheduling Problem (DSP) and Driver Rostering Problem (DRP).

mixed-integer models applying a genetic algorithm to solve the combinatorial and NP-hard problem. The meta-heuristics involved prevent a non-guaranteed globally optimal solution in the context of several sources of non-linearities and nonconvexities. Besides the heuristics, they employed a neighbourhood search, the simulated annealing algorithm to minimise passenger, operator, and unmet demand costs.

Concerning mathematically formulated problems, [Hasselstrom, 1982] developed linear models for the TNDFS problems and solved them using mathematical programming approaches to maximise the total number of passengers. A set of routes is generated on a link-connected network, and the lines are then selected by assigning the frequency through linear programming.

[Ceder and Israeli, 1998] proposed a nonlinear mixed-integer program that considers both passenger and operator costs. First, a set of feasible routes connecting all nodes is generated. Next, a set covering problem is solved to find the minimal subsets of routes from which the most suitable subset is selected by applying a multi-objective analysis. Moreover, [Wan and Lo, 2003] examined modifying the routes and assigned frequencies and developed a linear mixed-integer programming model to minimise operating costs. However, this mathematical approach has limitations to large networks, and to be applied to an extensive scale network like London, it would need heuristics to solve the computation complexity. [Borndörfer et al., 2005] dealt with a line planning problem in public transport and formulated two multi-commodity flow models to minimise the total passenger travel time and the operating cost. The developed linear model was examined using a commercial LP solver to create the lines dynamically and then tested in the Potsdam city network in Germany. Last, [Iba, 2014] presents two integer linear programming models for the timetabling and vehicle scheduling problems and combines them in a bi-objective integrated model. Numerical experiments show that the proposed approach can solve scenarios with up to 50 bus lines.

[Lampkin and Saalmans, 1967] were the pioneers in utilising heuristics in the TNDFS. They developed a skeleton method to obtain the transit routes iteratively by extending the routes to improve the trip directedness and implemented a random greedy-based search method to obtain the frequencies. The latter was adjusted to minimise passenger trip times considering fleet size constraints and unlimited vehicle capacity. Later, [Silman et al., 1974] minimised the passenger's travel time and overcrowding discomfort. The method is split into two phases: first, a candidate set of routes is selected based on the previously mentioned skeleton method, and then at a second instance, the optimal assignment of frequencies is done under fleet size constraint using the gradient method. [Dubois et al., 1979] presented a TNDFS problem in three sub-problems: determining the set of street selection, selecting routes, and assigning the optimal frequency. A heuristic method was proposed for determining the substantial subset of streets to minimise the passenger's journey time subject to a budgetary constraint. Moreover, the set of routes was selected from the generated subset of roads. Eventually, the frequency is assigned using a gradient-based search heuristic to lower down the waiting time. The famous work of [Mandl, 1980] developed a two-stage approach: first, a feasible set of routes was created; then, heuristics were applied to improve the quality of the initial set of bus routes. Only in-vehicle travel costs were considered to evaluate route quality. In Mandl's pioneering work, a benchmark of 15 Swiss cities network was proposed. Mandl's

work is still vastly used for methodology efficiency comparisons. [Lee and Vuchic, 2005] tackled the TNDFSP with a different methodology, although more realistic considering variable transit demand under a given fixed total demand. Moreover, network and operational specifications were also included with the objective of minimising passenger travel time. Using a practical approach [Fusco et al., 2002] aimed to obtain the minimum overall system costs. Their heuristic approach is based on a genetic algorithm² to select sub-optimal routes or detours of the network. Apart from other literature, their work considers and introduces different criteria to generate routes simultaneously and a hierarchical transit network by easing the set of constraints to consider the town structure.

[Chakroborty and Wivedi, 2002] implement a different technique relying on minimal heuristics. The framework is based on the principles of genetic algorithms. Then, an evolutionary optimisation technique is used to determine a more optimal set of routes. Because they rely less on heuristics, these results show promise compared to others described in the literature.

[Zhao and Gan, 2003], [Zhao, 2006a], [Zhao and Zeng, 2007] try a mathematical approach to a generic network relying on as minimum heuristics as possible to solve large-scale network problems. The bi-objective cost function on the previous two works minimises passenger and operator costs, while the first minimise passenger cost. In all, the search is done with a stochastic global search scheme that combines simulated annealing, tabu, greedy, and bisection search methods. The applied methodology in published benchmark generic networks shows that it can improve results with reasonable computational resources. [Szeto and Jiang, 2014] propose a bi-level model that explicitly minimises the total number of passenger transfers in the objective function of the upper-level problem and incorporates strict capacity constraints to address the in-vehicle congestion in the lower-level problem. They employ a hybrid artificial bee colony (ABC) algorithm to determine route structures and a descent direction search method to determine an optimal frequency setting for a given route structure. This technique uses a similarity between how bees in nature look for food and how optimisation algorithms search for an optimum combinatorial optimisation problem. Results show that the algorithm can find optimal solutions and reduce computational time.

[Arbex and Cunha, 2015] formulated a multi-objective function to minimise passenger and operator costs by utilising an alternating objective genetic algorithm. For each iteration, the objective function changes until it reaches an optimum heuristic value.

Likewise, [Nikolić and Teodorović, 2014] aim for the same objective, applying, this time, a bee colony optimisation approach. [López-Ramos et al., 2017] integrated the TNDFSP in railway transit rapid systems with the dual objective of minimising passenger riding time and operator costs. The solution method comprises a corridor generation algorithm to extend the current network or create lines and then a line-splitting algorithm to assign the train frequency and deal with multiple line construction.

[Ngamchai and Lovell, 2003] optimise bus transit route design by manipulating genetic algorithms.

²Genetic algorithms are search algorithms based on concepts of natural selection and natural genetics and are used as general-purpose optimisation algorithms

To this end, seven genetic operators are introduced to incorporate unique service frequency settings for each route and reduce the search time. Besides, it also considers headways coordination to match passenger transfers at stations. It is showed on the published benchmark network by [Pattnaik et al., 1998] that the model is more efficient than binary-coded genetic algorithms. [Pattnaik et al., 1998] developed a two-phase model, where a set of candidate routes competing for the optimum solution is generated. Then the optimum set is selected by the genetic algorithm. This work was tested in Madras, India, where it has been considered a network benchmark for other models. [Zhao et al., 2015] employed the memetic algorithm, an evolutionary algorithm based on the genetic algorithm. The defined mathematical model minimises the passenger cost and unmet demand under route and capacity constraints. A try-and-error procedure is also introduced to verify the efficiency of the local search operator. [Jha, 2019] presented a multi-objective approach to TDNFSP by solving the problem in two stages. First, the route design problem is handled using an initial route set generation combined with a genetic algorithm (GA). Then, the frequency setting is dealt with with the multi-objective particle swarm optimisation (MOPSO). The depicted solutions improved trip directedness, reduced transfers and lowered the average travel time. The model was tested in Mandl's benchmark network. [Tom and Mohan, 2003] minimised the total system cost, which embeds the user and operator costs. The model has two phases. Firstly, a prominent candidate route set is generated by an algorithm. Secondly, a solution route set is selected with genetic algorithms. The integrated network design and frequency problem consider the frequency as a variable differing from other works in terms of the coding scheme. The method structure is similar to the works of [Pattnaik et al., 1998]. [Afandizadeh et al., 2013] further investigate the TDNFSP by also considering an evaluation procedure, depot assignment, a penalty for empty seats, and unmet demand to consider fleet constraints. The work was tested on Mandl's Swiss benchmark network with promising results compared to other assignments.

The works of [Buba and Lee, 2018], and [Zhao et al., 2015] minimise passenger cost and unmet demand by testing differential evolution algorithms on Mandl's Swiss network and other generic networks from the literature. The results show more efficient searches than other meta-heuristics, such as the previously mentioned genetic algorithm.

[Zhao, 2006b] defined a model that relies little on heuristics for a large-scale transit network. Instead, it uses a global search scheme based on simulated annealing and objective functions to minimise user cost and the unwillingness to make transfers. Last, [Fan et al., 2009] address the TDNFSP using an evolutionary multi-objective approach in a trade-off between the user and operator costs. The work structure has three phases: finding candidate solutions, generating feasible route sets, and a routine system for making smart neighbourhood moves.

In summary, the sub-problems in the TDNFSP study are already considered NP-hard problems due to their computational complexity [Guihaire and Hao, 2008]. According to [Szeto and Wu, 2011], at the first instance, exact approaches were taken. However, because of the NP-hard specification of a large-scale network, these methods are computationally complex. Many authors considered iterative processes to obtain near-optimal solutions using heuristics and meta-heuristics, given the difficulty of being solved optimally. The latter iterative approach is also an NP-hard,

multi-constrained combinatorial optimisation problem with a vast search space. The evaluation of candidate solutions route sets can be challenging and time-consuming, with many potential solutions rejected due to infeasibility [Fan et al., 2009], [Arbex and Cunha, 2015]. Table 2.1 gives an overview of the aforementioned papers that approach the TNDFSP.

Table 2.1: Summary of the TNDFSP Literature

AUTHORS	OBJECTIVE	SOLUTION METHOD
[Szeto and Wu, 2011]	Min Passenger Costs	GA, NS
[Wei Fan and Machemehl, 2004]	Min Passenger & Operator Costs	GA, NS
[Fan, 2006]	Min Passenger & Operator Costs	GA, SA
[Hasselstrom, 1982]	Min Passenger Costs	Mathematical
[Ceder and Israeli, 1998]	Min Passenger & Operator Costs	Mathematical
[Wan and Lo, 2003]	Min Operator Costs	Mathematical
[Borndörfer et al., 2005]	Min Passenger & Operator Costs	Mathematical
[Iba, 2014]	Min Passenger & Operator Costs	Mathematical
[Lampkin and Saalmans, 1967]	Min Passenger Costs	Heuristic, GS
[Silman et al., 1974]	Min Passenger Costs	Heuristic
[Dubois et al., 1979]	Min Passenger Costs	Heuristic
[Mandl, 1980]	Min Passenger Costs	Heuristic
[Lee and Vuchic, 2005]	Min Passenger Costs	Heuristic
[Fusco et al., 2002]	Min Passenger & Operator Costs	GA
[Chakroborty and Wivedi, 2002]	Min Passenger & Operator Costs	GA
[Zhao and Gan, 2003],	Min Passenger & Operator Costs	TS,SA,GS
[Zhao, 2006a],	Min Passenger & Operator Costs	TS,SA,GS
[Zhao and Zeng, 2007]	Min Passenger & Operator Costs	TS,SA,GS
[Szeto and Jiang, 2014]	Min Passenger Costs	ABC
[Arbex and Cunha, 2015]	Min Passenger & Operator Costs	GA
[López-Ramos et al., 2017]	Min Passenger & Operator Costs	Heuristic
[Ngamchai and Lovell, 2003]	Min Passenger Costs	GA
[Pattnaik et al., 1998]	Min Passenger & Operator Costs	GA
[Zhao et al., 2015]	Min Passenger Costs	MA
[Jha, 2019]	Min Passenger Costs	GA,MOPSO
[Tom and Mohan, 2003]	Min Passenger & Operator Costs	GA
[Tom and Mohan, 2003]	Min Passenger & Operator Costs	GA
[Afandizadeh et al., 2013]	Min Passenger Costs	GA
[Buba and Lee, 2018]	Min Passenger Costs	Meta-heuristics
[Zhao et al., 2015]	Min Passenger Costs	Meta-heuristics
[Zhao, 2006b]	Min Passenger Costs	SA
[Fan et al., 2009]	Min Passenger Costs	Meta-heuristics

The model presented herein is different from the existing literature TNDFSP models because it tackles the post-Covid-19 scenario and TfL specifications. The former is expressed as the reluctance of passengers towards travelling by public transport and the lessened expected demand. The latter concerns limitations towards the hourly operated travel distance to reduce operating costs. The TNDFP is taken into account using a simple heuristic that uses Dijkstra's Algorithm to create shortest paths tailored for high-demand OD pairs. Later these paths generate lines for

both OD pair directions and are updated to the pre-existing bus lines. At the same time, the bus network expands and the probability of having low-usage lines increases. Such lines will have a low frequency assigned after the complete model runs. The planner's knowledge can then be used to discard low-frequency lines manually, thus cutting operating costs. The TNFSP is solved with a two-staged model composed of two mixed-integer programs solved with an LP commercial solver, Gurobi. At the first stage, the passengers are sorted through the network to minimise their travel times, considering the crowdedness of each segment of the line and bus frequencies. The frequency is assigned optimally to the distributed flow at the second stage, considering the fleet size and operated kilometre constraints.

2.1 Contributions

The work of [Baaj, 1995] is the guideline for the construction of the Transit Network Design model. A heuristic algorithm that generates the lines based on the most demanding OD pairs is formulated. Furthermore, the passenger assignment model is directly retrieved from the work of [Spiess and Florian, 1989]. The assignment of passengers formulated as a MIP integrates all origins and all destinations while considering the growing crowdedness over the bus network. The model is further complemented with a traveller's transfer penalty.

Chapter 3

Solution Framework

The modelling framework developed to deal with the Transit Network Design and Frequency Setting Problem is presented in this section. Due to the complexity of the problem, each sub-problem is described individually. In each subsection, the description and assumptions are presented, followed by the formulation.

The first model is the Line Generation model (Section 3.1) based on the works of [Baaj, 1995]. The model is based on a Route Generation Algorithm (RGA) for the design of transit networks. The heuristic algorithm aims to generate additional lines for the existing bus network to directly serve the most demanding OD pairs, therefore, minimising travel times. The second model is the Passenger Assignment model (Section 3.2). It is adapted from the works of [Spiess and Florian, 1989] and complemented with a penalty for passengers transferring lines at a macro-level. The overall objective aims to minimise the travel times of passengers considering a probabilistic assignment based on the frequency of the bus lines and the crowding of each line. The third model is the Frequency Setting model (Section 3.3). The model is created from scratch to consider new variables due to the influence of Covid-19. These variables consist mainly in the future passenger's negative perception towards travelling in more crowded buses. The output frequency minimises travel times and optimises the bus fleet's deployment to meet the traveller's demand. Further analysis of the assigned frequencies can also be used to discard low-usage lines to cut operational costs.

The second and third models, namely the Passenger Assignment (PA) and the Frequency Setting (FS) models, function as a combined bi-level model. The output from the Passenger Assignment model is the input of the Frequency Setting model and vice-versa. The models iterate together to consider several objectives and constraints that would otherwise not converge to an optimum value.

In review, the model generates bus transit lines, distributes the bus passenger demand, and sets vehicle frequency on every bus line of the network. The general input data comprises a network graph $G=(S,A)$ containing the existing stops S , the direct connections between them A with the respective length and travel time, the transit demand OD matrix, bus company fleet size, and average bus capacity. Other parameters can also be adjusted, such as the convergence tolerance, perceived travel and waiting times, maximum distance travelled and others. Figure 3.1 depicts a

diagram with the solution framework. The solution method integrates the three models to solve

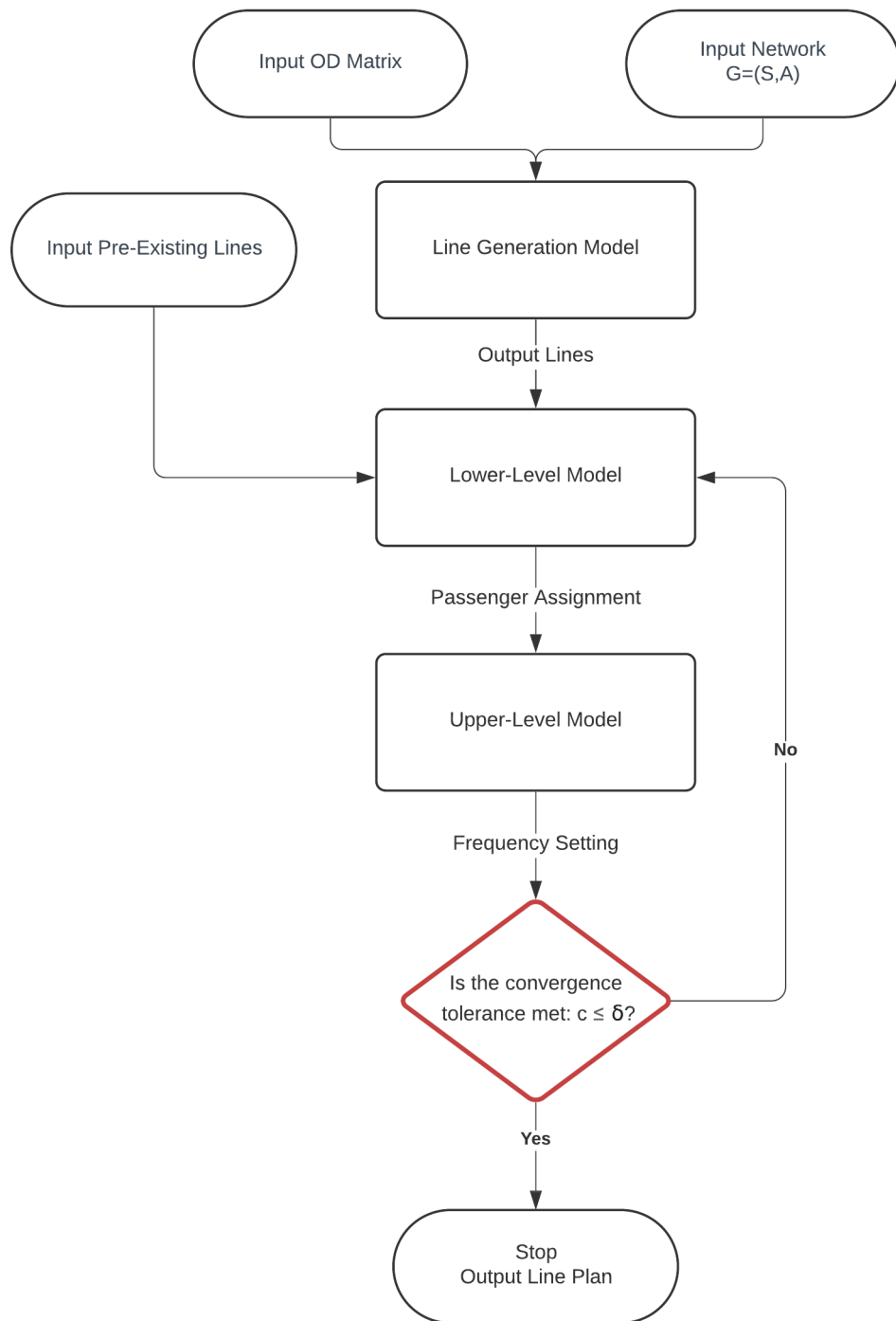


Figure 3.1: Solution Framework

the TNDPSP. The Line Generation model runs independently, while the Passenger Assignment and Frequency Setting iterate together to converge rapidly to the desired output. The structure of the algorithms starts with the LG model receiving the input data, which is used to generate

more direct bus lines. Note that the generated lines do not change afterwards. Next, the pre-existing lines and new lines are combined, and the algorithm proceeds to the bi-level model. A loop begins with the initial frequencies f_o of all lines set to the same value. The bi-level runs iteratively until the minimal convergence tolerance is reached. The process starts with the lower level pre-distributing the OD volume throughout the network, considering the frequency of the lines to board, the in-vehicle travel time, the usage of the line for crowdedness considerations, and an additional transfer penalty. The output from this model will serve to readjust the frequencies according to the in-vehicle crowding perception and boarding waiting times specified to meet passenger satisfaction. The new frequency is set and introduced in the lower-level model, and the process repeats. When the difference between the previous assignment and the current one is less than the convergence tolerance, the loop stops, and the output of the final bus frequencies is given.

3.1 Line Generation Model

The Line Generation model (LG) is inspired by the works of [Baaj, 1995]. The generation is complementary to the existing bus transit network in such a way as not to incur massive changes to the current composition and at the same time leave room to improve the connections. In the scenario where the disruption caused by Covid-19 in travel patterns led to inefficiencies of the service, the model fits in to create lines specified to the future travel demand. From this set of old and new lines, public transport companies may choose the line plan to be established for their service according to the volume of passengers served. A line plan comprises the bus line routes and their respective frequencies. A good line plan minimises costs and meets the passenger needs. Given the transit-demand matrix (OD_s matrix, $s \in S$, where S is the set of node pairs) corresponding to a selected period of operation, a description of the node network specifying for each node, its neighbouring nodes, and the in-vehicle travel times on all connecting street links (PTN), the maximum load factor allowed on any bus route, and the seating capacity of the fleet's buses, a route generation algorithm (RGA) determines the sets of routes that correspond to different trade-offs between the user and operator's costs.

RGA's overall structure is described in Figure 3.2. It starts by sorting the OD pairs of the matrix in the decreasing order of the number of trips to a new variable. Then, the first user input regarding the trade-off between the user and operator costs is considered by introducing parameter D_{min} . The value of D_{min} specifies the minimum percentage of trips that are to be satisfied directly by the generation of the new routes by the RGA. The greater this percentage, the more lines are generated and, therefore, the greater the operating costs, although the demand is served more directly and user costs are lessened.

Next, the algorithm sums the trips of the first sorted OD pairs until the accumulated sum, in percentage, is higher than or equal to D_{min} . The summed OD pairs are removed from the sorted OD and selected to generate lines. They are considered the most high-demanding OD pairs. For each high-demanding OD pair, it finds the shortest path using, for example, Dijkstra's algorithm from origin s to destination q , (s, q) and from destination q to source s , (q, s) . These paths are

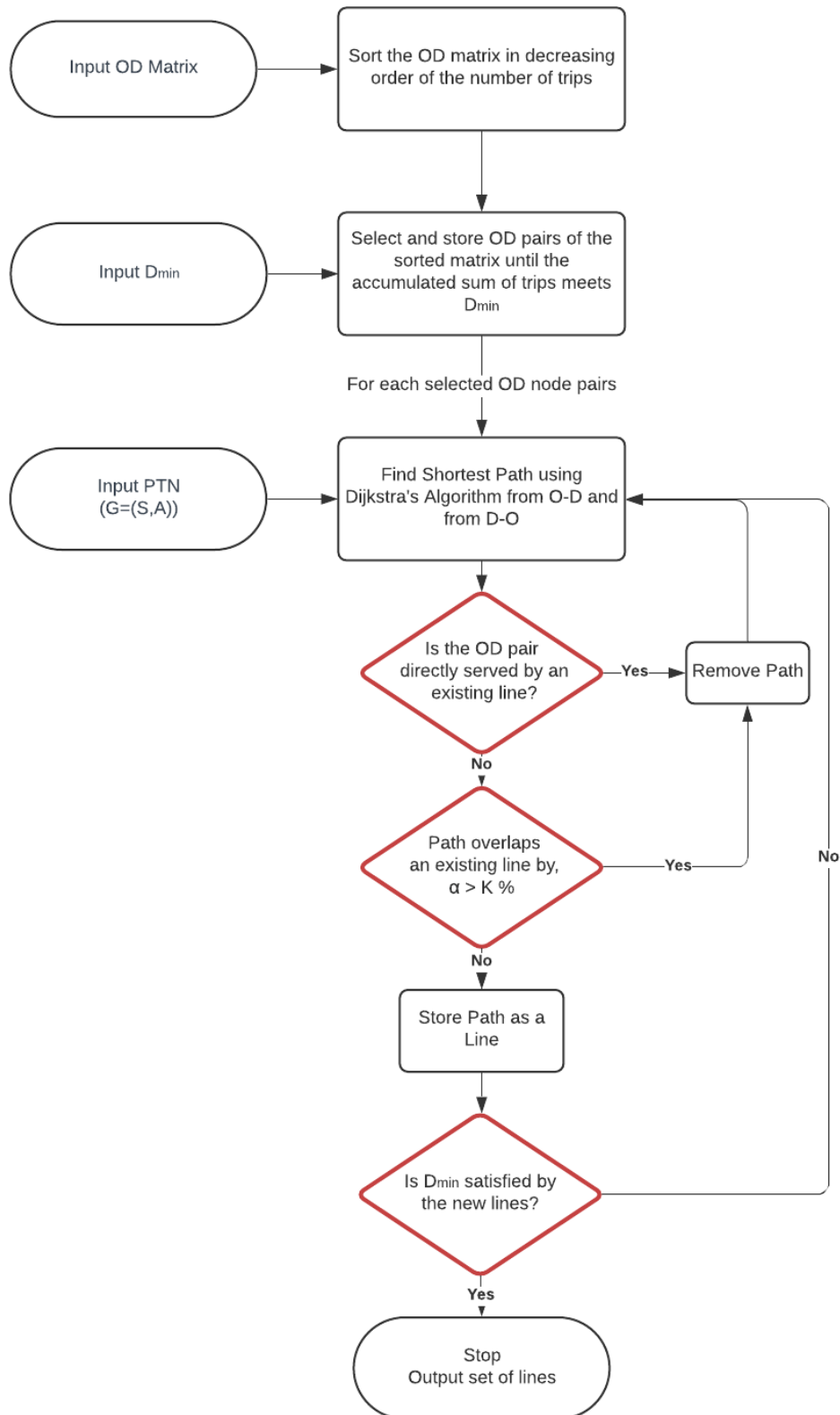


Figure 3.2: Adapted Route Generation Algorithm

concatenated and form a temporary line. The next steps filter the temporary line to check for overlapping. For instance, if both origin s and destination q are already served directly by the bus network, the temporary line is removed to avoid redundant lines. Further, overlapping filtration is applied. Here, the planner input is again requested for the parameter K indicating the maximum percentage of overlapping links compared to an existing line. If this condition is not met, then the temporary line is removed. Again, this is a trade-off between the operator and user's costs, as a low value of K will generate fewer lines.

The temporary line is stored as a new line of the bus network if the filtration steps do not remove it. After all high-demanding OD pairs are checked, the algorithm outputs the set of generated lines.

3.2 Lower-Level Model: Passenger Assignment Model

The Passenger Assignment Model is based on the works of [Spiess and Florian, 1989]. The output of this lower-level model is the distribution of the demanded volume of passengers through the network composed of arcs and stops. An arc a is a directed road connection between two bus stops. The arc contains information on the frequency of buses that travel the arc, the average travel time, and the respective length of the arc. The assignment is a probabilistic assignment that depends on the frequency of the bus lines. On this modulation, Spiess and Florian introduce a non-linear term on the objective function to consider the perceived travel cost due to congestion. This aspect is incorporated here but also considered and explored further at the micro-level on the Frequency Setting model.

One of the limitations of this model is that it does not include the passenger perception towards making a transfer to another bus, only the general waiting time. The made adaptation consists of a fixed penalty to the volume of transfer passengers, which contributes to passengers opting more for direct lines when such an option is available.

3.2.1 Assumptions for the Passenger Assignment Model

3.2.1.1 Trip Components

The formulation of a traveller's trip is segmented into trip components that can be represented by the arcs $a \in A$ in a network $G = (S, A)$. These arcs can be boarding arcs, alighting arcs, and in-vehicle travel arcs. The stops, S , vertices of the network, have incoming and outgoing arcs. The set of outgoing arcs at a stop s is denoted by A_s^+ , while the incoming arcs are expressed as A_s^- . It is important to denote that Spiess introduces the possibility of more than one stop at the exact physical location to allow the construction of a network where an arc corresponds to one and only one line (Figure 4.2).

3.2.1.2 Information of the Arcs

Each arc has a given travel time of t_a , a frequency $f_a \in [0, +\infty]$ and a waiting time $w_a \in [0, +\infty]$. Boarding and alighting arcs have a null value for travel time. Since boarding arcs imply waiting,

the frequency needs to be greater than zero, i.e. $f_a \in \mathbb{R}_+$. In contrast, the frequency and waiting time of in-vehicle travel arcs have a frequency $f_a = +\infty$ and waiting time $w_a = 0$. To summarize, three types of trip components are considered:

- Waiting for boarding ($t_a = 0$, $f_a \in \mathbb{R}_+$, $w_a \in \mathbb{R}_+$),
- In-vehicle travel ($t_a > 0$, $f_a = +\infty$, $w_a = 0$),
- Alighting ($t_a = 0$, $f_a = 0$, $w_a = 0$).

3.2.1.3 Probabilistic Assignment

As mentioned before the assignment of passengers depends on the frequency. This translates, in a probabilistic way, by considering on every stop a set of potential outgoing arcs A_i^+ . Let $x_{a,s} = 1$ if arc a is an outgoing arc of node s and $x_{a,s} = 0$ otherwise. The probability of choosing an arc over another is given by Equation 3.1:

$$P_{a,s} = \frac{f_a x_{a,s}}{\sum_{a' \in A} f_{a'} x_{a',s}}, \forall a \in A \quad (3.1)$$

Where the denominator represents the sum of all other outgoing arcs $x_{a',s}$ multiplied by the respective frequencies $f_{a'}$. Note that $P_{a,s} = 0$ if a is not an outgoing arc of s , because $x_{a,s} = 0$. Following this method the volume of passengers will be more densely distributed on high frequency lines that also have reduced waiting times. Then the passenger volume at node s can be given by:

$$v_s = \sum_{a \in A_s^-} v_a + \phi_s, \forall s \in S \quad (3.2)$$

Where ϕ_s is the demand at node s and v_a is the passenger volume of arc $a \in A_s^-$. Furthermore the expected waiting time w_s at node s is expressed by Equation 3.3:

$$w_s = \frac{\theta}{\sum_{a \in A} f_a x_{a,s}}, \forall s \in S \quad (3.3)$$

$\theta \geq 0$ is a coefficient that can take the value of $1/2$ if the arc is a high-frequency service, in which passengers arrive randomly at the station or a value of 1 for a low-frequency service. In the case $\theta = 1$, the distribution of inter-arrival times is an exponential distribution of headways with mean $\frac{1}{f_a}$ and an uniform passenger arrival rate.

3.2.1.4 Bus Capacity

As the planning concerns the aftermath of the pandemic, the bus capacity b^{cap} is the total capacity without considering social distancing restraints, and it is assumed to be a constant value for all the fleet; this is the average bus capacity. Nevertheless, the previously mentioned in-vehicle crowding perception negatively impacts travel time as buses overcrowd. For a network with a much-varying

fleet size, some model development would be desired to adjust the crowding function of the travel time in order to consider different space areas within the vehicle. The equivalent is valid for the Frequency Setting model, where another crowding function is considered.

3.2.1.5 Perceived Travel Cost Due to Congestion

To consider the in-vehicle congestion on the passenger's path selection, the riding travel time of an arc t_a can no longer be immutable, but rather a continuous non-decreasing function of the corresponding arc flow $t_a(v_a)$. The function adopted to update the travel time is retrieved from the Bureau of Public Roads (BPR):

$$\int_0^{v_a} d_a(x) dx = \left(\frac{v_a}{f_a b^{cap}} \right)^\beta \quad (3.4)$$

Where v_a is the passenger volume of trip component a , b^{cap} is the nominal capacity of the bus, and β a BPR function parameter that, in this case, takes the value of 2 and the function becomes a quadratic one.

3.2.1.6 Transfer Waiting Time and Associated Penalty

The transfer waiting time for a complex network like London's is hard to predict as it depends on at least two factors: the passenger arrival time at the transfer stop and the frequency of the interchange bus line. The arrival time is considered random as the generation of lines and frequency settings are not formulated to coordinate the transfers. Coordination means that in a transfer stop, two or more lines meet each other at the same time to reduce transfer waiting times. However, this would add much complexity to the model. Then the transfer waiting time t^{trans} considers only the frequency of the transfer bus line, and it has the same formulation as the waiting time to board at a stop t_s^{wait} (Subsection 3.3.1.2).

Some models consider walking times when transferring because of the distance between stops. In this model, for simplification, if a transfer is conducted, the walking time is omitted as transfers are to happen only in interchange stops. We aim to reduce the necessity of transfers to provide passengers with direct services, as in reality, uninterrupted connections are preferable [Stradling et al., 2007]. However, a journey might be incomplete without a transfer if there is no direct connection. In that case, the volume of transfers at a network level is accounted for and penalised by a fixed parameter β^{trans} . The additional time must be costly enough for passengers to prefer direct paths if such an alternative is available. The value is based on the work of [Yap et al., 2020] in The Hague, where a five-minute penalty was considered plausible.

In order to keep track of the volume of transfers, since there is not a trip component regarding transfer arcs and respective volume, this term is considered at a network level. The formulation can be given by the difference between the sum of the outgoing volume of all boarding arcs and the original demand, i.e., the number of boarding passengers of the OD matrix:

$$v^{transfer} = \sum_{a \in B} x_{a,s} \tilde{v}_{a,q} - \phi \quad (3.5)$$

Where $B \subseteq A$ is the set of boarding arcs and $\tilde{v}_{a,q}$ is the flow in arc a from all passengers traveling to destination q .

3.2.2 Objective Function for the Passenger Assignment Model

The optimisation model below allows the optimal strategy for all travellers from all origins to all destinations $q \in Q$. For this, it is defined a subset $S_q \subseteq S$ containing all the stops before destination q . Additional variables $\tilde{v}_{a,q}$ and $w_{s,q}$ indicating, respectively, the flow in arc a and the waiting time at stop s for passengers travelling to destination q are added.

$$MIN : \quad Z = \sum_{a \in A} t_a^0 v_a + \sum_{q \in Q} \sum_{s \in S} w_{s,q} + \sum_{a \in A} t_a^0 \left(\frac{v_a}{f_a b^{cap}} \right)^\beta + \beta^{trans} v^{transfer} \quad (3.6)$$

Subject to:

$$v_a = \sum_{q \in Q} \tilde{v}_{a,q} \quad \forall a \in A \quad (3.7)$$

$$\sum_{a \in A} \tilde{v}_{a,q} x_{a,s} - \sum_{a \in A} \tilde{v}_{a,q} y_{a,s} = \phi_{s,q}, \quad \forall s \in S_q, q \in Q \quad (3.8)$$

$$\tilde{v}_{a,q} x_{a,s} \leq f_a w_{s,q}, \quad \forall a \in A, s \in S, q \in Q \quad (3.9)$$

$$\tilde{v}_{a,q} \geq 0, \quad \forall a \in A, \forall q \in Q \quad (3.10)$$

$$\tilde{v}_{a,q} \in \mathbb{R}_+ \quad \forall a \in A, q \in Q, w_{s,q} \in \mathbb{R}_+ \quad \forall s \in S, q \in Q \quad (3.11)$$

$$v^{transfer} = \sum_{a \in B} x_{a,s} \tilde{v}_{a,q} - \phi \quad (3.12)$$

where,

t_a^0	is the average travel time of arc a without congestion
$\phi_{s,q}$	is the demand of passengers travelling from s to q
$x_{a,s}$	is a binary dummy variable that checks if arc a is an outgoing arc of s
$y_{a,s}$	is a binary dummy variable that checks if arc a is an incoming arc of s
ϕ	is the total demand
v_a	is the volume of passengers on arc a

The model formulated as a linear programme minimises the total arc travel time and waiting time at nodes. Constraint 3.8 is the passenger flow conservation constraint which ensures that the outgoing flow from all outgoing arcs from node s is equal to the incoming flow plus the passenger demand. Constraint 3.9 ensures that the passenger volume v_a in the outgoing arc a of node s is lower than or equal to the frequency of that arc multiplied by the total waiting time for all trips at node s . Note that if a is not an outgoing arc of node s , constraint 3.9 is satisfied because the left-hand-side is equal to zero. This constraint regards capacity limits in order for the model not to assign excess flow beyond an arc's capacity. Finally, constraint 3.10 ensures that the volume of passengers in an arc cannot be negative.

3.3 Upper-Level Model: Frequency Setting Model

The frequency setting model formulated as a mixed-integer linear program aims to attract more passengers by minimising passenger costs. It utilises the distribution of passengers and bus lines from the Passenger Assignment and Line Pool models output, respectively, as inputs. The model will adjust the frequencies of the routes to meet the demand more optimally. This adjustment ultimately interferes with the passengers' waiting time and in-vehicle travel time, which are addressed as passenger costs in this study. The waiting time is directly dependent on the frequency of a bus line at a given stop, as more frequent lines have a lower time headway between vehicles and, therefore, a lessened waiting time.

The in-vehicle crowding is indirectly related to the frequency, i.e., the frequency will determine how many buses operate in a given line in a given period. Consequently, if the bus capacity is given and we multiply it by the number of buses, this indicates the maximum passenger flow possible to transport over that period. The crowding on each bus is then determined by the difference between the sum of the assigned bus capacity and the total demand if the demand is evenly distributed across the time interval. If the number of buses is underestimated, the model will add more frequency to the highly required lines to reduce the crowding perception and ultimately reduce the in-vehicle travel time.

3.3.1 Assumptions for the Frequency Setting Model

3.3.1.1 In-Vehicle Travel Time and Crowding Time Penalty

The perceived in-vehicle travel time is perceptible as more lasting when there is crowding on-board. According to [Pel et al., 2014], passengers start to experience a disutility for crowding when the demand exceeds 50% to 70% of the bus seat capacity. In our scope, after Covid-19, people are likely to be reluctant to travel or relax in crowded public spaces. To consider such hesitation, the perceived in-vehicle travel time between two stops connected by an arc a of a given route r is given by the sum of the average road travel time $t_{r,a}^{ivt}$ multiplied by the in-vehicle crowding perception $\beta_{r,a}^{ivt}$. The crowding perception is formulated as a piecewise linear function of three conditionals (see Equation 3.13).

$$\beta_{r,a}^{ivt} = \begin{cases} M \times v_{r,a} & v_{r,a} > b^{cap} \\ \lambda - \frac{(\lambda-1) \times (b^{cap} - v_{r,a})}{b^{cap} - b^{sc}} & b^{sc} < v_{r,a} \leq b^{cap} \\ 1 & b^{sc} \leq v_{r,a} \end{cases} \quad (3.13)$$

with:

$$v_{r,a} = \frac{V_{r,a}}{f_r} \quad (3.14)$$

Where,

- $v_{r,a}$ is the in-vehicle load per bus that runs on arc a of route r
- $V_{r,a}$ is the passenger volume of arc a of route r over the time period T
- f_r is the frequency on route r
- b_r^{cap} is the bus total capacity
- b_r^{sc} is the bus seat capacity

Suppose the occupancy exceeds the total capacity b^{cap} . In that case, the multiplier takes a considerable value M , making it infeasible to board the bus considering the resulting increase in in-vehicle travel time. If not, a linear interpolation is used on the second conditional between the seating capacity b^{sc} and the standing bus capacity scenarios b^{cap} . The penalty takes the value of 1, i.e. no liability, if the number of passengers is less or equal than b^{sc} and a crowding parameter λ in the latter scenario where it is not possible to board more passengers. Equation 3.14 is expressed as the number of present passengers on every bus of an arc of a specific route. The distribution number is constant throughout all buses since the passenger demand is assumed to be homogeneously distributed across the time horizon. The latter is one hour as the frequencies are also defined per hour.

The beforehand piecewise linear (PWL) function is depicted in Figure 3.3.

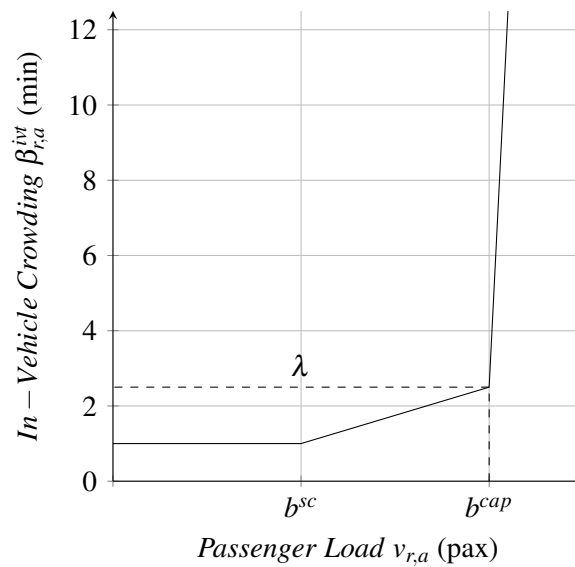


Figure 3.3: In-Vehicle Crowding Penalty Function

3.3.1.2 Waiting Time and Associated Penalty

The waiting time at a stop is a feature that resembles the passenger perception of the bus service's reliability and efficiency. Passengers are more attracted to using public transport if the expected waiting time is low and consistent. Nowadays, more travellers arrive at the stations closer to the departure time due to technology that provides real-time service data [Lüthi et al., 2007]. Regarding this information, the waiting time at a stop is based on two assumptions. The first considers a random arrival pattern if the stop serves a high-frequency service, i.e., it has short inter-arrival times, and one does not need to check the schedule before arriving. The second is that travellers will plan their arrival closer to the departure if they wish to go on a less frequent route. Regarding the former assumption's randomness, an average waiting time of half of the time headway between vehicles is used according to most benchmark literature on the matter. Meanwhile, the latter assumption considers a fixed waiting time ξ to attend all lines where the time-headway gap is longer. These two assumptions are depicted in Figure 3.4

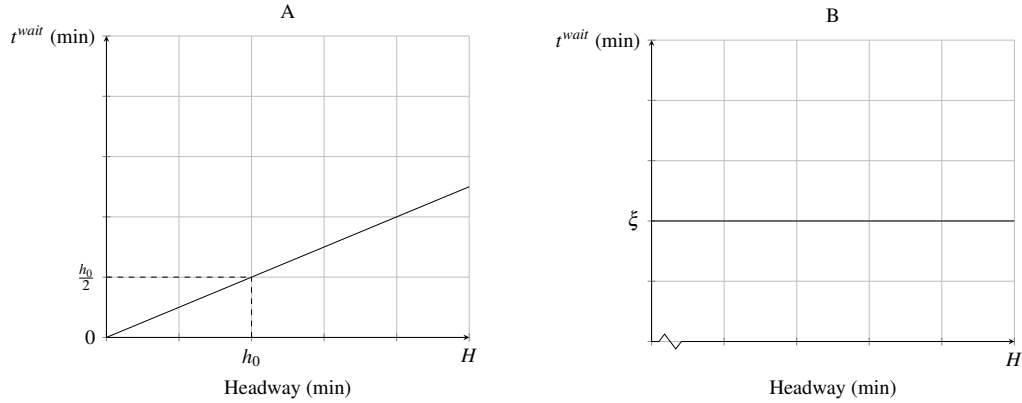


Figure 3.4: Passengers' Assumed Waiting Time *Versus* Vehicle Time-Headways. Graphic A: Waiting Time of a High Frequency Line. Graphic B: Waiting Time of a Low Frequency Line

The waiting time of a passenger at a given stop s , t_s^{wait} , is expressed as a piecewise linear function derived from the frequency of a route f_r (see Equation 3.15).

$$t_s^{wait} = \begin{cases} \xi & , f_r < \omega(veh/h) \\ \frac{1}{2f_r} & , f_r \geq \omega(veh/h) \end{cases} \quad (3.15)$$

Where,

ω is the service frequency that distinguishes a high from a low frequency line

The waiting time at a station is usually perceived as costlier compared to the in-vehicle time. There are many studies regarding the waiting time multiplier, resulting in quite a range of values. Many authors came up with different factors based on their research case and local surveys. On average, passengers with no access to real-time information perceived the wait time $0.83min$

(multiplier = 1.15) longer in the work of [Watkins et al., 2011] and 0.84min (multiplier = 1.21) more prolonged in the work of [Fan et al., 2016]. These studies have also shown that passengers perceived significantly less the cost with real-time information, around 30%. Due to the impossibility of researching the current perceived waiting time in London, we will adopt a median value, a multiplier of $\beta^{wait} = 1.2$. However, the parameter can be modelled and updated to a specific case study.

3.3.1.3 Reduced Travelled Length

In the wake of the financial strain that several transit operators face and the lessened demand, a restraint to the maximum hourly travelled distance was implemented in the model. Such a measure diminishes operating costs by limiting the frequencies and deploying fewer buses and drivers while reducing vehicle wear and tear. The length of each bi-direction line r is calculated by the sum of each arc's length $l_{r,a}$. Thus it is possible to calculate the distance per hour by multiplying the length of each line by the assigned frequency. The hourly covered distance of the network can then be given by:

$$L = \sum_r \sum_{a \in r} l_{r,a} f_r \quad (3.16)$$

3.3.2 Objective Function for the Frequency Setting Model

Now the objective function can be presented as Equation 3.17, below.

$$MIN : Z = \sum_{r \in R} \sum_{a \in r} (\beta_{r,a}^{ivt} t_{r,a}^{ivt} + \beta^{wait} t_{r,a}^{wait} g_{r,a}) v_{r,a} \quad (3.17)$$

For every line r in the set of lines R and for all arcs a in the lines r , the model is subjected to the following constraints:

$$\beta_{r,a}^{ivt} = \begin{cases} 1 & b^{sc} \leq v_{r,a} \\ \lambda - \frac{1.5 \times (b^{cap} - v_{r,a})}{b^{cap} - b^{sc}} & b^{sc} < v_{r,a} \leq b^{cap} \\ 100 \times v_{r,a} & v_{r,a} > b^{cap} \end{cases} \quad (3.18)$$

$$v_{r,a} = \frac{V_{r,a}}{f_r} \quad (3.19)$$

$$t_{r,a}^{wait} = \begin{cases} \xi & , f_r < \omega \\ \frac{1}{2f_r} & , f_r \geq \omega \end{cases} \quad (3.20)$$

$$N^{buses} = \sum_r \sum_{a \in r} \frac{t_{r,a}^{ivt}}{f_r} \quad (3.21)$$

$$N^{buses} \leq N^{MAX} \quad (3.22)$$

$$L = \sum_r \sum_{a \in r} l_{r,a} f_r \quad (3.23)$$

$$L \leq L^{MAX} \quad (3.24)$$

Where,

$t_{r,a}^{ivt}$	is the in-vehicle travel time of arc a of route r
$t_{r,a}^{wait}$	is the average waiting time before boarding a vehicle at arc a of route r
$\beta_{r,a}^{ivt}$	is the penalty associated with in-vehicle crowding
β^{wait}	is the fixed penalty associated with the perceived waiting time before boarding
$l_{r,a}$	is the length of an arc a belonging to route r
N^{buses}	is the number of deployed buses on the network
N^{MAX}	is the network's fleet size
L	is the travelled length on the network per hour
L^{MAX}	is the maximum span travel length of the network per hour
f_r	is the frequency of route r
B	is the set of boarding arcs

The formulation minimises the users' cost, which is implicitly related to the frequency of the lines. The equation goes through every line r and every arc a belonging to one and only line r . Like in the Passenger Assignment model, a journey is divided into trip components: boarding, riding, transferring, waiting and alighting. The formulation works on a network level as follows: the waiting time cost is employed if arc a is a boarding arc, the in-vehicle time cost is applied for all arcs in A , however, apart from the riding arcs, the travel times of boarding and alighting arcs are set to zero. Therefore only the riding arcs are penalised with the crowding perception.

In the case of a being an alighting arc, there is no associated cost. The first component regards the perceived in-vehicle travel time considering the crowding level of all buses and all arcs. The second component includes the waiting times of all passengers on arcs that are boarding arcs. Parameter $g_{r,a}$ is a binary parameter that takes the value of one if an arc is a waiting arc and zero otherwise. The last term, $v_{r,a}$ is the output of the passenger assignment model and corresponds to the volume of passengers on every arc. Constraint 3.18 is a piecewise linear function ensuring that the in-vehicle crowding perception penalty is updated accordingly. Constraint 3.19 is a linear constraint that calculates the number of passengers in a particular bus running in a specific arc of a line. Since each arc corresponds to a one and only line, if the total volume of the assigned passengers per arc $V_{r,a}$ is divided by the frequency of the operated buses of the line containing such an arc, the average number of travellers per arc and per bus ($v_{r,a}$) is obtained. The waiting time to board a bus, either if it is the first boarding or a transfer boarding, is given by the PWL function in constraint 3.20. Constraint 3.21 calculates the number of deployed buses per line r in regards to the round trip time: this is the sum of the average time to ride each arc of the line.

Furthermore, inequality 3.22 restraints the maximum number of deployed buses according to the fleet size N^{MAX} .

Finally, constraints 3.23 and 3.24 determine the travelled distance over the study period and limit them according to the maximum allowed travelled length and available resources from the operating company.

Chapter 4

Model Application

4.1 Case Study - Toy Network

The experiments with the bi-level model begin with a small 'toy' network, containing all of the considered trip components. Figure 4.1 illustrate the test network, while Figure 4.2 depicts the different trip components of the same network.

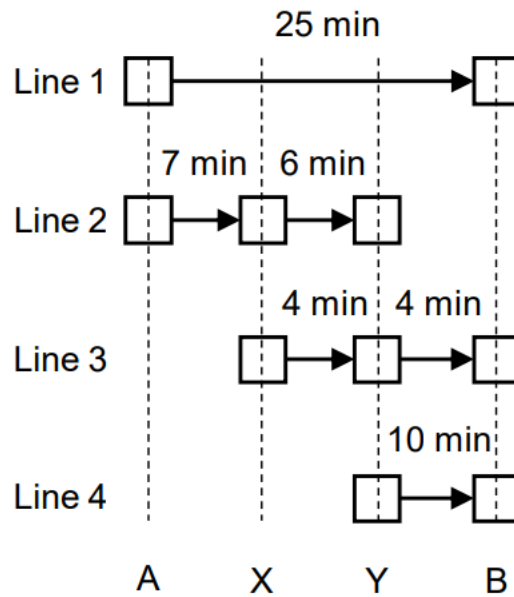


Figure 4.1: Example of a Test Network with Four Lines and Stops A, X, Y and B

In this example, there is a total of fourteen nodes (four stations and ten auxiliary nodes) and eighteen edges (six boarding arcs, six riding arcs, and six alighting arcs).

Below there is an example OD matrix presenting the number of passengers travelling from an origin stop to a destination stop (Table 4.1). The OD table also contains a demand with non-direct routes to destination vertices, which is relevant for testing how the model processes the transfers.

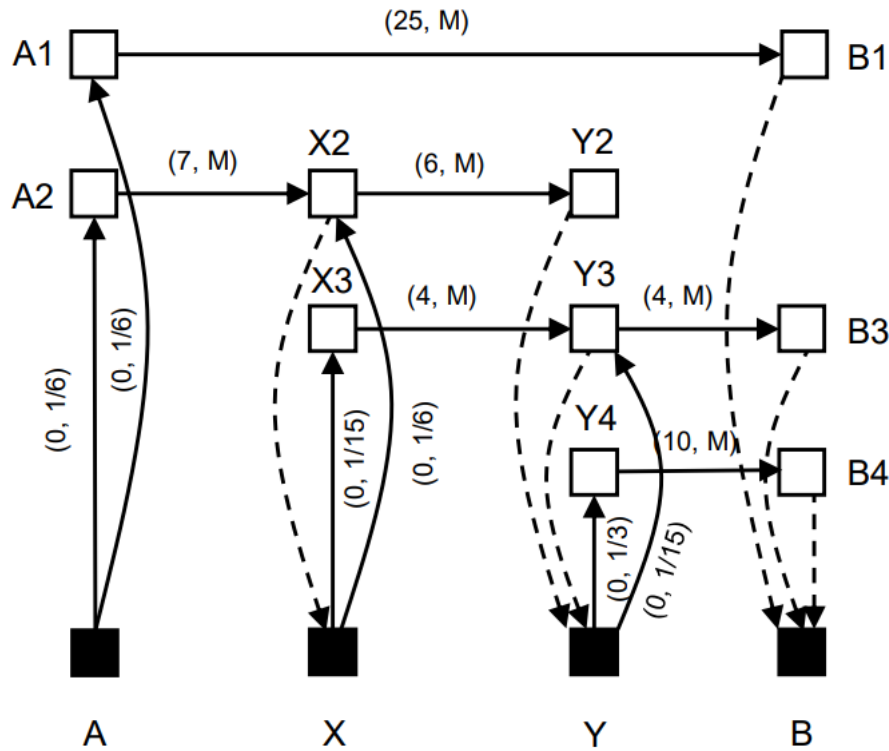


Figure 4.2: Representation of The Test Network with Different Trip Components, Travel Times and Frequencies

Table 4.1: Example OD Matrix for a Time Period of One Hour

O/D	A	X	Y	B
A	-	0	80	100
X	0	-	20	0
Y	0	0	-	0
B	0	0	0	-

4.1.1 Results Regarding the Toy Network

The experiments start by verifying the implementation of the new transfer penalty in the Passenger Assignment model. The experiment is conducted first in the lower-level with Gurobi, a mathematical solver in PYTHON 3.8 which was retrieved by means of an academic licence. The results of the different perceptions of the transfer penalty are depicted in Figure 4.3.

The introduction of the transfer penalty has reduced the number of transfers, which resembles the passenger's negative perception towards transfers when a direct route is available. In this specific network, as Figure 4.3 shows, the effect is more evident when the transfer penalty is higher than six minutes. It is expected that for a more extensive network, the transfer penalty will have an increased effect.

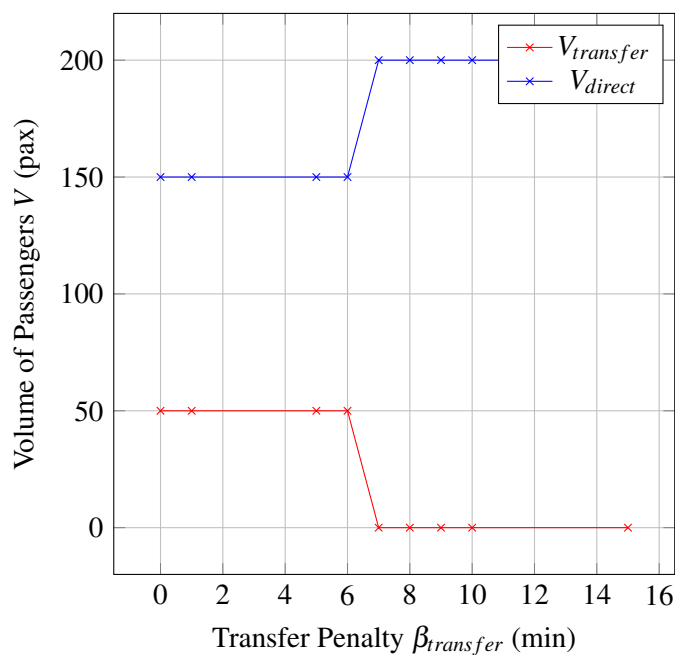


Figure 4.3: Volume of Direct and Transfer Passengers According to The Variation of The Transfer Penalty

Furthermore, the bi-level optimization model is tested with the following parameters (see Table 4.2):

Table 4.2: Parameters Considered for Running the Test Network of Figure 4.1

Parameters	Values	Units
b^{sc}	60	<i>seats</i>
b^{cap}	87	<i>seats</i>
λ	2.5	<i>min/min</i>
ξ	6	<i>min</i>
ω	10	<i>bus/h</i>
β^{wait}	1.2	<i>min/min</i>
β^{trans}	5	<i>min</i>

The test results, depicted in table 4.3, were obtained using different fleet sizes while keeping the other variables fixed. In this test, round trips are not considered for the bus assignment, which means that the frequency reflects the number of buses assigned to a route over one hour. The output also gives the total volume of passengers that used at least one arc belonging to a line. A number of iterations was completed until the tolerance convergence was less or equal to 1%.

Table 4.3: Output of The Bi-Level Optimization Model in the Example Network of Figure 4.1 with Different Fleet Sizes

Fleet Size	Line	Passengers	Frequency
6	1	46	0.77
	2	153	4
	3	3	0.03
	4	72	1.2
7	1	100	1.67
	2	100	5.3
	3	0	0
	4	0	0
10	1	100	1.67
	2	100	8.3
	3	0	0
	4	0	0
15	1	100	7.5
	2	100	7.5
	3	0	0
	4	0	0

The frequency assigned in the upper-level model is also optimised and validated by checking the passenger perception of the bus line. Under these circumstances, more vehicles will run on a more demanded line so that the demand is met and the vehicles are not overcrowded. As a consequence of assigning more buses, there is an increased chance of having more seated places. The waiting times are also generated according to the assigned frequency.

The running time of the bi-level model, for this network and others, depends on the size of the network, the size of OD demand and the relative tolerance for convergence. The chosen convergence alters the number of iterations the model will perform. For the example network, with a relatively low value of 1%, the convergence is under thirty seconds with three iterations.

The results depicted in Table 4.3 provide information on the number of passengers distributed on the existing four lines with the respective bus frequency. From the interpretation of these results it follows that, as the fleet size availability increases, the model converges to an optimum assignment of passengers, eliminating the number of sub-tours and unnecessary transfers. The frequency assigned also becomes more consolidated in order to reduce crowding. It is noted that the model gives more accurate results when the number of available buses provides more capacity than the OD demand requests. Therefore it is crucial to calibrate the fleet size and other parameters, such as the in-vehicle crowding perception, the waiting time perception, and the transfer penalty, to get more accurate results.

4.2 Case Study - Area of the London Network

The scope of the study is within the geographical area of Chelsea (within the Royal Borough of Kensington and Chelsea). The bus routes of this part of London have been more regularly a point of analysis on how to provide an optimal service. The borough of Kensington and Chelsea is one of the affluent areas in London, characterised by a high share of high-income residents. These residents opt more often for car usage, causing road traffic congestion and making it challenging to provide an attractive bus network with a reasonable cost recovery rate. Figure 4.4 depicts a map with the existing nineteen lines of the area in the study. TfL provided the supply data of this network. Considering the operational impacts of Covid-19, some parameters were also given by TfL (see Table 4.4). The first experiments for the complete model were done using a reduced OD matrix that contains the fifteen most demanded OD pairs. The matrix contains sensitive information that is not available through this dissertation.

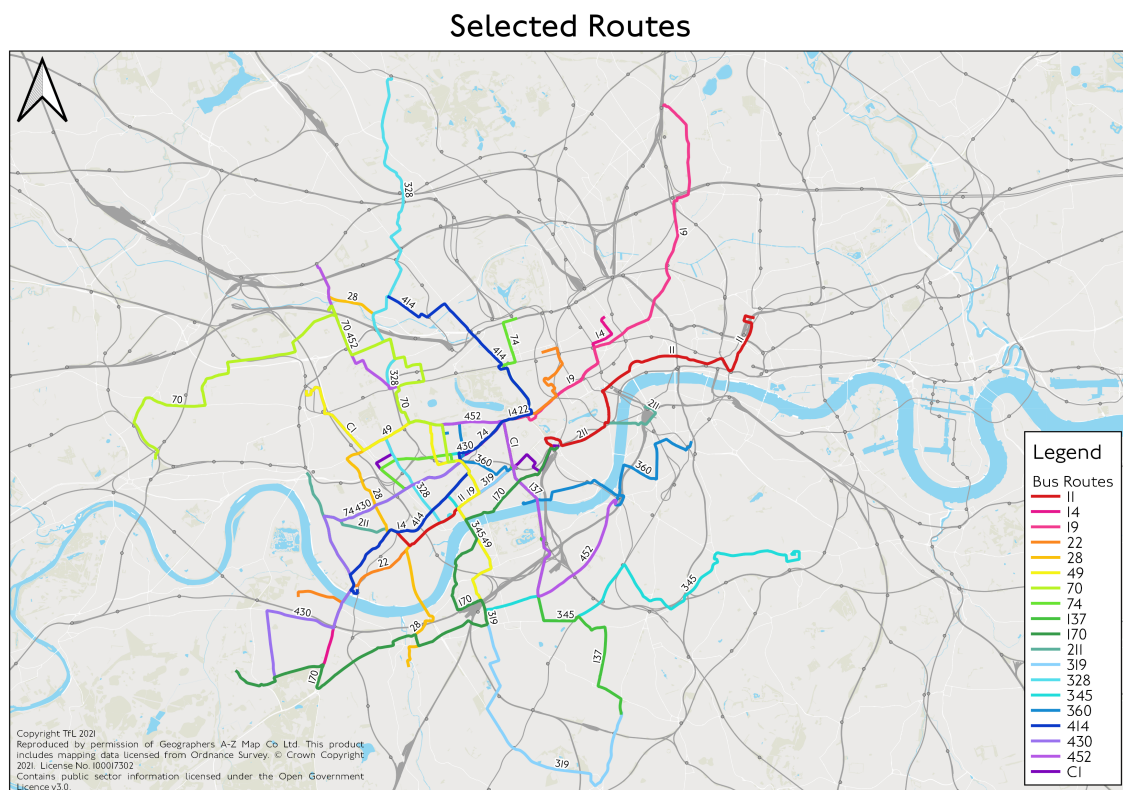


Figure 4.4: Map of the considered Bus lines within the Geographical Area of Chelsea

4.2.1 Results Regarding the London Network

The first experiments on the boroughs of Kensington and Chelsea of the London Network begin by running the LG model in Python 3.8. Computational times differ accordingly to the percentage of the total demand to be directly satisfied. In other words, it increases as the number of OD pairs considered to create lines increments. The OD matrix's input regards the complete area in the study

Table 4.4: Parameters Considered for Running The Bi-Level Model on the Chelsea Network of Figure 4.4

Parameters	Values	Units
b^{sc}	60	<i>seats</i>
b^{cap}	87	<i>seats</i>
λ	2.5	<i>min/min</i>
ξ	6	<i>min</i>
ω	10	<i>bus/h</i>
β^{wait}	2	<i>min/min</i>
β^{trans}	3.5	<i>min</i>
N^{MAX}	376	<i>buses</i>
L^{MAX}	2932	<i>km</i>

compromising thousands of OD pairs (7,077) at first. A small OD matrix comprising the fifteen most demanded OD pairs is also considered to demonstrate further the three models working together (LG, PA, FS) and to cut computational times. The latter OD matrix is additionally used to test different scenarios and the robustness of the three models. Table 4.5 depicts the percentage of D_{min} considered and the number of new generated lines.

Table 4.5: Number of Created Lines by the Line Generation model with Different Percentages of Demand to be Served by a Direct Connection

D_{min}	Overlapping %	Number of Additionally Generated Lines
10% : Complete OD Matrix	70	12
25% : Complete OD Matrix	70	24
50% : Complete OD Matrix	70	40
60% : Complete OD Matrix	70	34
75% : Complete OD Matrix	70	31
100% : Small OD Matrix	100	9

The results show that for different values of D_{min} , the size of the constructed lines differs drastically. One wants to have a limited number of lines to reduce network complexity in practice but wishes to serve the demand fast and directly. The trade-off between the number of lines and the network's connectivity is a topic that needs careful analysis. Nevertheless, in this context of this work, the aim is to generate extra lines that complement the existing network without adding too much complexity. In that sense, to consider the benefits of introducing the new lines with the existing connections, there will be fewer additional lines to add hereafter in further testings. In summary, the LG heuristic model is capable of producing results in reasonable computational time. However, the output number of lines is substantially large, which is not optimal. The introduction of new constraints such as the minimum number of stops for a path to be considered a line can help

reducing the generated lines. Further development, found in the work of [Baaj, 1995], can also help by introducing line expansion algorithms with different node (stop) insertion and selection strategies to apply after the shortest paths algorithms. By creating more extensive lines, more OD pairs have the likelihood to be served by a single line.

Now the testing of the PA and the FS models takes place. For this, three-line configurations to be optimised are considered. These are the existing lines, the generated lines from the LG model and a combination of both existing and generated lines. The output presented in Table 4.6 concerns the optimisation of the existing lines with the small OD matrix aforementioned.

Table 4.6: Optimisation of the Existing Lines of Chelsea's Network with A Small OD Matrix

Lines	Frequency	Volume of Boarding Passengers
11	0	0
14	0	0
19	9	100
22	0	0
28	14.7	150.6
49	8.9	23.6
70	8.7	93.6
74	0	0
137	12.3	137.6
170	0	0
211	5	20
319	15	136.5
328	13	137
345	9	100
360	19.6	302
414	9.18	73.6
430	0	0
452	0	0
C1	0	0

The assignment employed 279 from a total of 376 buses, with a summed frequency of 124 buses per hour. As shown in the results, lines with more affluence are prone to have higher frequencies. In contrast, lines with a null value represent lines that might be discarded as they serve little to no demand. Besides not meeting the demand, such lines do not generate revenue and shall be reallocated or discarded in further analysis to cut users and operators' costs.

As the FS model was limited to one hour, the results are obtained when running the first iteration. The total computational time for both models was seven hours. The running was conducted in a computer with 8GB of RAM and a Ryzen5 with 2.1GHz processor. Though the OD matrix is small, a solution, even if not wholly optimal, was reached in a total of nineteen lines comprising a vast network. Including the needed auxiliary edges and vertices to run, the network comprised over 1800 stops and 3800 arcs. The computational times show that the model still needs improvements

to deal with more data. When trying to run larger OD matrices, memory errors would result.

The following test network is generated with the LG model to serve the twenty most demanded pairs. A broader number of OD pairs is considered to create a more diverse network without guaranteeing that all lines generated will be fully used. The results of the optimisation are depicted in table 4.7 after running the PA and FS models with the small OD matrix.

Table 4.7: Optimisation of the Generated Network Lines with a Small OD Matrix

Lines	Frequency	Volume of Boarding Passengers
L1 Auxiliary	178	373
L2 Auxiliary	0	0
L3 Auxiliary	104	167
L4 Auxiliary	41.85	17
L5 Auxiliary	84	105
L6 Auxiliary	86	68
L7 Auxiliary	102.8	201
L8 Auxiliary	66.6	64
L9 Auxiliary	89.7	202

The optimisation for the generated network was complete within two iterations after setting the tolerance convergence of $\delta = 0.1$ between iterations. The running time was four and half minutes, which is a considerable difference compared to the previous computation using the existing lines and to the fact that a practically optimal solution was obtained this time. The number of used buses was 243, and the summed frequencies were 753 buses per hour. The latter is reflected by the reduced length of the generated lines. A round trip time is completed much rapidly, and one bus can do multiple round trip times over the same period. For this assignment, line *L2 Auxiliary* may be discarded.

Finally, the optimisation of the combined existing and generated routes is presented in Table 4.8. The results show that passengers used both the existing and the newly created lines, in terms of the number of boardings 37% of the total demand boarded existing lines and 63% the generated ones. Even though the generated lines are tailored to serve the most demanding twenty OD pairs, some existing lines are still in use, which is a vital indicator that the model can combine the networks and provide a more optimal solution. Once more, lines that serve practically no demand shall be discarded to avoid unnecessary operating costs. The number of used buses was 264, and the summed frequency was 362. Regarding computational times, the model solved one iteration in 8h30, had limited the FS model to run for one hour. The convergence of the FS model depends on the number of linear variables and constraints. For this study, the FS model stabilised the convergence after one hour, with a 60% near-optimum solution.

Table 4.8: Optimisation of the Combined Existing and Generated Network Lines with a Small OD Matrix

Lines	Frequency	Volume of Boarding Passengers
11	0	0
14	0	0
19	0	0
22	0	0
28	14.4	77.7
49	0	0
70	7.7	49.2
74	0	0
137	16	87.6
170	0	0
211	0	0
319	12.2	37
328	8.4	50
345	0	0
360	14.4	111.1
414	9.7	55.6
430	0	0
452	0	0
C1	0	0
L1 Auxiliary	98.6	286
L2 Auxiliary	27	78.9
L3 Auxiliary	30.8	55.6
L4 Auxiliary	18.8	14.2
L5 Auxiliary	27.3	49.2
L6 Auxiliary	26.3	34
L7 Auxiliary	24	114.4
L8 Auxiliary	28.5	32
L9 Auxiliary	33.4	134.7

Chapter 5

Concluding Remarks

The planning of a bus network is a continuous optimisation process that requires attention in every stage in order to achieve a robust network. Because the trade-off between operating and user costs changes the possibility of having the best solution, planners and public transport companies must embrace constant adaptations to new scenarios and prepare to mitigate disruptions, such as those caused by the impacts of Covid-19. Concerning the effects of the pandemic, replanning the bus service through modelling is essential to provide operators with an evidence-based tool that can deliver more efficient solutions [Gkiotsalitis and Cats, 2020]. The outputs shall address the current problems, as many companies are still struggling to provide efficient and robust service to the community and making it rentable [Tirachini, 2020]. The developed TNDFS model optimises the networks in the aftermath of the pandemic crisis and the ensuing service disruptions. Attending to minimising the passenger cost, this is the total travel time, several variables to regain passenger's trust in public transport are considered. Special consideration is given to the reluctance passengers feel towards gathering in crowded vehicles, heavily penalised regarding the user-perceived travel time. Further research on this subject, whether through studies or surveys, will help improve the realism of these penalties and better accommodate in-vehicle passengers shortly.

The TNDFS model's structure comprises a Line Generation heuristic and two Mixed-Integer Programmes that assign the passenger demand onto the network and set the optimal bus frequency. All models are NP-hard, being the MIP's the ones that consume extra computational resources. The model is tested on a small-sized test network with promising results regarding the implemented variables and constraints and further tried in the boroughs of Kensington and Chelsea, an area within the London bus network. Several try-outs of the model on the latter network show that the model lacks tractability for this level of extension of the networks. In order to overcome computational infeasibility, a small OD matrix was used for testings. The reproducibility for more prominent representations requires other optimisation tools.

The LG model is a simplified and adapted version of the works of [Baaj, 1995] that is capable of handling more extensive sets of network data. Further research into this model is necessary to make the lines broader, reduce redundancy as well as the number of lines. A few guidelines for improvement would include introducing more constraints for a determined path to be sufficiently

'good' to generate a line, implementing an expanding route algorithm to avoid short lines, and adopting a final analysis procedure to guarantee that those additional requirements are met.

The PA model is retrieved from [Spiess and Florian, 1989] and updated with a transfer penalty as, in reality, passengers opt for direct connections when such are available. A probabilistic assignment is made according to the frequencies of the bus lines. The travel time is considered as a function of the crowding of a routeing arc and the respective predicted travel time. The model assigns passengers through the network, and the resultant volume is used as input for the FS model. The efficiency of this model lacks extensive representations due to the necessity of creating arcs and additional stops at the exact physical locations for each arc to correspond to a single line only. Such a factor makes the programme run through millions of linear variables on London's network and possibly run out of RAM. For future research at this stage, heuristic algorithms might be the best option to reduce computational times and possibly to reach a near-optimum distribution of passengers.

The FS model is constructed from scratch to optimise the bus frequencies for the distributed demand output of the PA model. The model minimises travel and waiting times, which are directly dependent on the frequency. Constraints to the fleet size and the hourly operated distance are also considered to cut operational costs. The running of this mathematically formulated model is more efficient than the PA model, but the optimisation itself takes longer. The waiting time for the output of the upper-level model is the primary constraint for this model, and the second is the convergence of the optimal solution. In order to overcome the latter, limiting the optimisation time makes it possible to retrieve a near-optimum solution.

In summary, the PA and FS models are formulated mathematically to provide optimal results for the current specifications. As a result, computational times and loss of memory are a setback for the tractability of wider networks that contain a considerable number of bus lines and OD pairs. To resolve the shortcoming of memory, one might consider ways of storing the variables generated through optimisation. With respect to computational times, further optimisation of the script is needed, especially on the FS model created from scratch that may need some adjustments.

Appendix A

Code Script

The code script used to run testings of the models is given in this section. The code is programmed in Python 3.8 and Gurobi, a commercial LP optimization solver retrieved by means of an academic licence to Python. Subsection [A.1](#) presents the code of the Line Generation algorithm, and subsection [A.2](#) exhibits the code of the combined Passenger Assignment and Frequency Setting models.

A.1 Line Generation Script

```
1 # -*- coding: utf-8 -*-
2 """
3 Line Generation Model
4 Created on Tue Jul 20 17:18:03 2021
5
6 @author: Manuel Filgueiras
7 """
8 # =====
9 #LIBRARIES
10 import os
11 import sys
12 import gurobipy as gp
13 from gurobipy import GRB
14 import pandas as pd
15 import numpy as np
16 import os
17 import ast
18 import csv
19 import math as math
20 import statistics
21 import xlwt
22 from xlwt import Workbook
23 import xlswriter
24 import openpyxl
25 import time
```

```

26 import re
27 import operator
28 import ast
29 import numpy as np
30 from collections import defaultdict, deque
31 import copy
32 start_time = time.time()
33
34 # =====
35 #FUNCTIONS FOR THE SHORTEST PATH AND DIJKSTRA ALGORITHMS
36 class Graph(object):
37     def __init__(self):
38         self.nodes = set()
39         self.edges = defaultdict(list)
40         self.distances = {}
41
42     def add_node(self, value):
43         self.nodes.add(value)
44
45     def add_edge(self, from_node, to_node, distance):
46         self.edges[from_node].append(to_node)
47         self.distances[(from_node, to_node)] = distance
48
49 def dijkstra(graph, initial):
50     visited = {initial: 0}
51     path = {}
52     nodes = set(graph.nodes)
53     while nodes:
54         min_node = None
55         for node in nodes:
56             if node in visited:
57                 if min_node is None:
58                     min_node = node
59                 elif visited[node] < visited[min_node]:
60                     min_node = node
61         if min_node is None:
62             break
63         nodes.remove(min_node)
64         current_weight = visited[min_node]
65         for edge in graph.edges[min_node]:
66             try:
67                 weight = current_weight + graph.distances[(min_node, edge)]
68             except:
69                 continue
70             if edge not in visited or weight < visited[edge]:
71                 visited[edge] = weight
72                 path[edge] = min_node
73     return visited, path
74

```

```

75
76 def shortest_path(graph, origin, destination):
77     visited, paths = dijkstra(graph, origin)
78     full_path = deque()
79     _destination = paths[destination]
80     while _destination != origin:
81         full_path.appendleft(_destination)
82         _destination = paths[_destination]
83     full_path.appendleft(origin)
84     full_path.append(destination)
85     return visited[destination], list(full_path)
86
87 # =====
88 #USER INPUT
89 #Insert OD matrix as dict:
90 od_matrix = pd.read_excel(r'Demand Top 20.xlsx')
91 od_matrix = od_matrix.set_index(['fromnodename', 'tonodename'])['Demand'].T.to_dict
92     ()
93 #Insert Dmin value in %
94 D_min=100
95 #read excel files and store as Dataframes
96 df1 = pd.read_excel(r'BusLines.xlsx', sheet_name = 'Links')
97 df2 = pd.read_excel(r'BusLines.xlsx', sheet_name = 'Stops')
98 #Insert Max overlap between two lines in %:
99 overlap_value=100
100 # =====
101 # ARRANGEMENT OF INPUT DATA
102 nodes1=[]
103 for i in range(len(df2)):
104     nodes1.append(df2.iat[i,1])
105 nodes1=list(dict.fromkeys(nodes1))
106 link0=[]
107 link1=[]
108 d=[]
109 c=[]
110 links=[]
111 for i in range(len(df1)):
112     df1.iat[i,4]=df1.iat[i,4]/60
113     for j in range(len(df2)):
114         if df1.iat[i,0]==df2.iat[j,0]:
115             df1.iat[i,0]=df2.iat[j,1]
116
117         if df1.iat[i,1]==df2.iat[j,0]:
118             df1.iat[i,1]=df2.iat[j,1]
119 for i in range(len(df1)):
120     df1.iat[i,0]= str(df1.iat[i,0]).replace(" ", "")
121     df1.iat[i,1]= str(df1.iat[i,1]).replace(" ", "")
122

```

```

123 for i in range(len(df1)):
124     link0.append(str(df1.iat[i,0]))
125     link1.append(str(df1.iat[i,1]))
126     d.append(df1.iat[i,3])
127     c.append(df1.iat[i,4])
128     links.append((str(df1.iat[i,0]),str(df1.iat[i,1])))
129 ptn = pd.DataFrame({'link0': link0, 'link1': link1, 'distance': d, 'time': c})
130 edges = {i: 0 for i in links}
131 distances = {i: 0 for i in links}
132 i=-1
133 for a in links:
134     i+=1
135     edges[a]=c[i]
136     distances[a] = d[i]
137 nodes=link0+link1+nodes1
138 nodes=list(dict.fromkeys(nodes))
139 sorted_od = dict(sorted(od_matrix.items(),key=operator.itemgetter(1),reverse=True))
140 dict_items = sorted_od.items()
141 sorted_od1=list(sorted_od)
142 total_demand=sum(sorted_od.values())
143 soma=0
144 keys_to_remove=[]
145 for key in sorted_od1:
146     if soma>= 0.01*D_min*total_demand:
147         break
148     soma+=sorted_od[key]
149     keys_to_remove.append(key)
150
151 for element in keys_to_remove:
152     sorted_od.pop(element)
153 M={i:0 for i in keys_to_remove}
154 for element in keys_to_remove:
155     M[element]=od_matrix[element]
156 print("--- %s seconds ---" % (time.time() - start_time))
157
158 # =====
159 # SHORTEST PATH DEBUG
160
161 graph = Graph()
162 routesToCalculate=M
163 all_passengers = sum(routesToCalculate.values())
164 for i in nodes:
165     graph.add_node(i)
166 for j in edges:
167     cost = edges.get(j)
168     fromNode = j[0]
169     toNode = j[1]
170     graph.add_edge(fromNode, toNode, cost)
171 paxdict={}

```

```

172 first=[]
173 second=[]
174
175
176 df_first = pd.DataFrame()
177 df_second = pd.DataFrame()
178 iteration=0
179 output_lines_dict={}
180 routee=1
181 #calculate shortest paths
182 for route in routesToCalculate:
183     output_lines={}
184     pax=routesToCalculate.get(route)
185     try:
186         ok, path = shortest_path(graph, route[0], route[1])
187         fastest_path_1=path
188         ok, path = shortest_path(graph, route[1], route[0])
189         fastest_path_2=path
190         join =fastest_path_1
191         join.extend(fastest_path_2)
192         output_lines={(str(routee)+'aux'):join}
193         #Conditional Requirements: OD already served?
194         if iteration==0:
195             output_lines_dict.update(output_lines)
196             iteration=1
197             routee+=1
198         else:
199             nao=0
200             for line in list(output_lines_dict.keys()):
201                 if route[0] in output_lines_dict[line]:
202                     if route[1] in output_lines_dict[line]:
203                         nao=1
204                         break
205             if nao ==0:
206                 output_lines_dict.update(output_lines)
207                 routee+=1
208
209     except KeyError:
210         print('not included: missing links')
211     except:
212         print('wrong')
213 #Overlapping Conditional
214 lines_to_remove=[]
215 for linha in output_lines_dict:
216     output_auxiliar=output_lines_dict[linha]
217     output_auxiliar=list(dict.fromkeys(output_auxiliar))
218     count = len(output_auxiliar)
219     removida={}
220     removida=output_lines_dict.copy()

```

```

221     del removida[linha]
222     for i in removida:
223         intersection = len([value for value in output_auxiliar if value in list(
            removida[i])])
224         overlap= (intersection/count)*100
225         if overlap >=overlap_value:
226             lines_to_remove.append(linha)
227 lines_to_remove=list(dict.fromkeys(lines_to_remove))
228 for i in lines_to_remove:
229     try:
230         del output_lines_dict[i]
231     except KeyError:
232         pass
233 # =====
234 # EXPORT THE OUTPUT LINES TO EXCEL
235
236 print(len(output_lines_dict))
237 routeNumber = 1
238 for i in output_lines_dict:
239     elementos=list(output_lines_dict[i])
240     for index in range(len(elementos)-1):
241         df_first = df_first.append({'ROUTE':str(routeNumber)+'aux', 'DIRECTION': 0,
            'STOPSEQUENCE':0, 'STOPCODE':elementos[index], 'STOPNAME':0,
242             'STOPCODE_NEXT':elementos[index+1], 'LINK':0,
243             'SCHEDULEDDISTANCE (m)': distances.get((elementos[index],
            elementos[index+1])),
244             'SCHEDULEDTIME (sec)': edges.get((elementos[index],
            elementos[index+1]))}, ignore_index=True)
245
246     routeNumber +=1
247
248
249
250
251 column_names = ["ROUTE", "DIRECTION", "STOPSEQUENCE", "STOPCODE", "STOPNAME", "
            STOPCODE_NEXT", "LINK", "SCHEDULEDDISTANCE (m)", "SCHEDULEDTIME (sec)"]
252
253 df_first = df_first.reindex(columns=column_names)
254
255 for i in range(len(df_first)):
256     if df_first.iat[i,3]==df_first.iat[i,5]:
257         df_first.iat[i,7]=0
258         df_first.iat[i,8]=0
259
260 with pd.ExcelWriter(r'20_OD_Auxiliary_routes.xlsx') as writer:
261     df_first.to_excel(writer)
262 # =====

```

A.2 Passenger Assignment and Frequency Setting Script

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Jul  2 15:25:57 2021
4
5 @author: Manuel Filgueiras
6 """
7 # =====
8 #LIBRARIES
9 import os
10 import sys
11 import gurobipy as gp
12 from gurobipy import GRB
13 import pandas as pd
14 import numpy as np
15 import os
16 import ast
17 import csv
18 import math as math
19 import statistics
20 import xlwt
21 from xlwt import Workbook
22 importxlsxwriter
23 import openpyxl
24 import time
25 import re
26 import random
27 import secrets
28 import csv
29 start_time = time.time()
30
31 # =====
32 #USER INPUT PART1
33
34 #Insert Raw Excel file {Route, Direction, StopSequence, Stopcode, Stopname,
35     Stopcode_next, Link, Distance, Time}
36 df1 = pd.read_excel(r'BusLines.xlsx', sheet_name = 'Routes')
37 df_primary = pd.read_excel(r'Auxiliary_routes.xlsx')
38 #Insert OD Matrix as a data frame
39 data = pd.read_excel(r'Demand Top 20.xlsx')
40
41 #Insert Bus Seat Capacity:
42 bus_seat_capacity = 60
43 #Insert Bus Capacity:
44 bus_capacity = 87
45 #Insert In-vehicle Crowding Perception:
46 crowding_perception = 2.5

```

```

46 #Insert Max Total Frequency of the network:
47 max_freq = 10000000
48 #Insert the transfer penalty in min:
49 transfer_penalty = 5
50 #Insert Waiting factor:
51 waiting_multiplier = 1.2
52 #Insert avg waiting time for low-frequency line:
53 csi= 6
54 #Insert max frequency per line:
55 max_freq_line = 200
56 #Insert convergance relative tolerance:
57 tolerance = 0.1
58 #Insert Maximum number of deployed buses:
59 max_number_buses=376
60 # Insert Maximum Meters covered per hour:
61 max_meters_hour=2932000
62 #Initial frequency assigned to all boarding arcs
63 freq_i=1/6
64 # =====
65 # ADJUSTS THE DATA\eliminate unwanted stops of the excel file
66
67 route = []
68 direction = []
69 algo=[]
70 for i in range(len(df1)):
71     df1.iat[i,4]= df1.iat[i,4].replace(" ", "")
72
73 for i in range(len(df1)-1):
74     df1.iat[i,8]=df1.iat[i,8]/60
75     j = i+1
76     route = df1.iat[i, 0]
77     routel = df1.iat[j, 0]
78     direction = df1.iat[i, 1]
79     directionl = df1.iat[j, 1]
80     if route == routel:
81         if direction != directionl:
82             if df1.iat[i, 5] != df1.iat[j, 3]:
83                 df1.iat[i, 5] = df1.iat[j, 3]
84                 df1.iat[i, 6] = 0
85                 df1.iat[i, 7] = 0
86                 df1.iat[i, 8] = 0
87         elif route != routel:
88             df1.iat[i, 5] = df1.iat[i, 3]
89             df1.iat[i, 6] = 0
90             df1.iat[i, 7] = 0
91             df1.iat[i, 8] = 0
92     df1.iat[-1, 5] = df1.iat[-1, 3]
93     df1.iat[-1, 6] = 0
94     df1.iat[-1, 7] = 0

```

```

95     df1.iat[-1, 8] = 0
96
97 for i in range(len(df1)-1):
98     j = i+1
99     route = df1.iat[i, 0]
100    route1 = df1.iat[j, 0]
101    if route!=route1:
102        df1.iat[i,5]=df1.iat[i,4]
103        df1.iat[i,3]=df1.iat[i,4]
104    else:
105        df1.iat[i,3]=df1.iat[i,4]
106        df1.iat[i,5]=df1.iat[j,4]
107
108 df1.iat[-1,3]=df1.iat[-1,4]
109 df1.iat[-1,5]=df1.iat[-1,4]
110 #join the two bus networks
111 df1=df1.append(df_primary, ignore_index=True)
112 # =====
113 #VARIABLES for the passenger assignment
114
115 #list of all of arcs
116 A = []
117 #list of all stops (including auxiliary)
118 I = []
119 boarding_arc = []
120 riding_arc = []
121 alighting_arc = []
122 stops = []
123 auxiliary_stops = []
124 for a in range(len(df1)):
125     boarding_arc.append((str(df1.iat[a, 3]),str(df1.iat[a, 3])+'Riding'+str(df1.iat
126     [a,0])+'d'+str(df1.iat[a,1])))
127
128     alighting_arc.append((str(df1.iat[a, 3])+'Riding'+str(df1.iat[a,0])+'d'+str(df1
129     .iat[a,1]), str(df1.iat[a, 3])))
130
131     riding_arc.append((str(df1.iat[a, 3])+'Riding'+str(df1.iat[a,0])+'d'+str(df1.
132     iat[a,1]),str(df1.iat[a, 5])+'Riding'+str(df1.iat[a,0])+'d'+str(df1.iat[a,1])))
133
134     stops.append(str(df1.iat[a, 3]))
135
136     auxiliary_stops.append((str(df1.iat[a, 3])+'Riding'+str(df1.iat[a,0])+'d'+str(
137     df1.iat[a,1])))
138
139 boarding_arc=list(dict.fromkeys(boarding_arc))
140 ridingding_arc=list(dict.fromkeys(riding_arc))
141 alighting_arc=list(dict.fromkeys(alighting_arc))
142 stops=list(dict.fromkeys(stops))
143 auxiliary_stops=list(dict.fromkeys(auxiliary_stops))

```

```

140 A.extend(boarding_arc)
141 A.extend(riding_arc)
142 A.extend(alighting_arc)
143 A_removed=list(dict.fromkeys(A))
144 I = stops+auxiliary_stops
145 I = list(dict.fromkeys(I))
146 def lines(route):
147     boarding=[]
148     alighting=[]
149     riding=[]
150     for i in range(len(df1)):
151         if df1.iat[i,0]==route:
152             boarding.append((str(df1.iat[i, 3]),str(df1.iat[i, 3])+'Riding'+str(df1
153             .iat[i,0])+'d'+str(df1.iat[i,1])))
154             alighting.append((str(df1.iat[i, 3])+'Riding'+str(df1.iat[i,0])+'d'+str
155             (df1.iat[i,1]), str(df1.iat[i, 3])))
156             riding.append((str(df1.iat[i, 3])+'Riding'+str(df1.iat[i,0])+'d'+str(
157             df1.iat[i,1]),str(df1.iat[i, 5])+'Riding'+str(df1.iat[i,0])+'d'+str(df1.iat[i
158             ,1])))
159             arcs=boarding+alighting+riding
160             arcs=list(dict.fromkeys(arcs))
161             return(arcs)
162 # =====
163 #USER INPUT PART2
164
165 #Insert here as many variables as the number of lines (preferable in order of
166 appearance on the Raw excel file):
167
168 l11 = tuple(lines(11)); l14 = tuple(lines(14)); l19 = tuple(lines(19)); l22 = tuple
169 (lines(22)); l28 = tuple(lines(28)); l49 = tuple(lines(49)); l70 = tuple(lines
170 (70)); l74 = tuple(lines(74)); l137 = tuple(lines(137)); l170 = tuple(lines
171 (170)); l211 = tuple(lines(211)); l319 = tuple(lines(319)); l328 = tuple(lines
172 (328)); l345 = tuple(lines(345)); l360 = tuple(lines(360)); l414 = tuple(lines
173 (414)); l430 = tuple(lines(430)); l452 = tuple(lines(452)); lC1 = tuple(lines('
174 C1')); l1aux = tuple(lines('1aux')); l2aux = tuple(lines('2aux')); l3aux = tuple
175 (lines('3aux')); l4aux = tuple(lines('4aux')); l5aux = tuple(lines('5aux'));
176 l6aux = tuple(lines('6aux')); l7aux = tuple(lines('7aux')); l8aux = tuple(lines
177 ('8aux')); l9aux = tuple(lines('9aux'))
178
179 #Add them here:
180
181 L=(l11,l14,l19,l22,l28,l49,l70,l74,l137,l170,l211,l319,l328,l345,l360,l414,l430,
182     l452,lC1,l1aux,l2aux,l3aux,l4aux,l5aux,l6aux,l7aux,l8aux,l9aux)
183
184 # =====
185 # ADJUSTS THE DATA
186
187 link0 = []
188 for a in range(len(A)):
189     link0.append(A[a][0])
190
191 link1 = []
192 for a in range(len(A)):
193     link1.append(A[a][1])

```

```

174 c = []
175 for a in range(len(boarding_arc)):
176     c.append(0)
177 for a in range(len(riding_arc)):
178     c.append(df1.iat[a, 8])
179 for a in range(len(alighting_arc)):
180     c.append(0)
181 f = []
182 for a in range(len(riding_arc)):
183     f.append(freq_i)
184 for a in range(len(riding_arc), len(A)):
185     f.append(100000)
186 d=[]
187 for a in range(len(boarding_arc)):
188     d.append(0)
189 for a in range(len(riding_arc)):
190     d.append(df1.iat[a, 7])
191
192 for a in range(len(alighting_arc)):
193     d.append(0)
194 df2 = pd.DataFrame({'link0': link0, 'link1': link1, 'ca': c, 'fa': f, 'distance': d
195                     })
196 # =====
197 # CONCATENATE OD MATRIX
198 destinations=[]
199 for i in range(len(data)):
200     destinations.append(data.iat[i,1])
201 destinations=list(dict.fromkeys(destinations))
202 R=destinations
203 matrix=np.random.randint(0,1,size=(len(I),len(R)))
204 df3=pd.DataFrame(matrix, index=I,columns=R)
205 for j in range(len(data)):
206     df3.loc[data.iat[j,0],data.iat[j,1]]=data.iat[j,2]
207 with pd.ExcelWriter('./input.xlsx') as writer:
208     df2.to_excel(writer, sheet_name='input')
209     df3.to_excel(writer, sheet_name='matrix')
210 # =====
211 #VARIABLES FOR THE PA PART2
212 M = 10000000
213 c = {i: 0 for i in A}
214 f = {i: 0 for i in A}
215 d = {i: 0 for i in A}
216 a_plus = {(a, i): 0 for a in A for i in I}
217 a_minus = {(a, i): 0 for a in A for i in I}
218 g = {(i, r): 0 for i in I for r in R}
219 index = {r: 0 for r in R}
220 C = {i: 87 for i in A}
221 for r in R:
222     j = -1

```



```

270 volume_out = model.addVars(I, vtype=gp.GRB.CONTINUOUS,
271                             lb=0, name='volume_out')
272 v_transfer = model.addVar(vtype=gp.GRB.CONTINUOUS, name='v_transfer')
273
274
275 # Constraints
276 model.addConstrs(v[a]==sum(vr[a[0],a[1],r] for r in R) for a in A)
277
278
279 for r in R:
280     if r in I:
281         Iq=[]
282         Iq.extend(I)
283         Iq.remove(r)
284         for i in Iq:
285             model.addConstr(sum(vr[a[0],a[1],r]*a_plus[a,i] for a in A)-sum(vr[
286 a[0],a[1],r]*a_minus[a,i] for a in A) == g[i,r])
287 model.addConstrs(volume_in[i]==sum(g[i,r]+sum(vr[a[0],a[1],r]*a_minus[a,i] for
288 a in A) for r in R) for i in I)
289 model.addConstrs(volume_out[i]==sum(v[a]*a_plus[a,i] for a in A) for i in I)
290
291 model.addConstrs(vr[a[0],a[1],r]*a_plus[a,i]<=f[a]*w[i,r] for a in A for i in I
292 for r in R)
293 model.addConstrs(vr[a[0],a[1],r]>=0 for a in A for r in R)
294 model.addConstr(v_transfer == sum(volume_out[i] for i in I) - sum(g[i, r] for i
295 in I for r in R))
296
297
298 # Objective Function
299
300 obj = sum(c[a]*(v[a]+(1/(C[a]*f[a]))*v[a]*v[a]) for a in A)+sum(sum(w[i,r] for
301 i in I) for r in R) + transfer_penalty * v_transfer
302 model.setObjective(obj, GRB.MINIMIZE)
303
304
305 # Results
306 model.params.NonConvex = 2
307 model.optimize()
308 if model.status == GRB.OPTIMAL:
309     print(model.status, 'optimal')
310     print('Obj: %g' % model.objVal)
311     model.write("out.attr")
312     model.printAttr('X')
313     model.printStats()
314
315 # Output Volume

```

```

314     vals_volume = {k: v.X for k, v in v.items()}
315     print(vals_volume)
316
317 else:
318     print(model.status, 'Passenger Assignment not optimal')
319     model.printStats()
320     model.getVars()
321
322 print("--- %s seconds ---" % (time.time() - start_time))
323
324
325 # -----
326 #FREQUENCY SETTING MODEL
327 T=1
328 minimum_waiting_time = 60/(2*max_freq_line)
329 import gurobipy as gp
330 from gurobipy import GRB
331 m = gp.Model()
332 m.setParam('NonConvex', 2)
333
334 #indices
335 i = [L[line][arc] for line in range(0, len(L)) for arc in range(0, len(L[line]))]
336 i=list(dict.fromkeys(i))
337 #add Vars
338 bivt = m.addVars(i, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name='bivt')
339
340 t_wait= m.addVars(i, lb=minimum_waiting_time, vtype=GRB.CONTINUOUS, name='twait')
341
342 g2 = m.addVars(i, vtype=GRB.BINARY, name='g2')
343
344 Nr = m.addVars(L, vtype=GRB.CONTINUOUS)
345
346 fr = m.addVars(L, vtype=GRB.CONTINUOUS)
347
348 fr_inverted = m.addVars(L, vtype=GRB.CONTINUOUS)
349
350 z = m.addVars(i, vtype=GRB.CONTINUOUS, name='flow_in_bus')
351
352 fr_all_links = m.addVars(i, vtype=GRB.CONTINUOUS, name='frequency_all_links')
353
354 #auxiliary binary variables:
355
356 g3= m.addVars(i, vtype=GRB.BINARY, name='g3')
357
358 g4= m.addVars(i, vtype=GRB.BINARY, name='g4')
359
360 g5= m.addVars(i, vtype=GRB.BINARY, name='g5')

```

```

361
362     bivtz= m.addVars(i, vtype=GRB.CONTINUOUS, name= 'bivtz')
363
364     t_waitz= m.addVars(i, vtype=GRB.CONTINUOUS, name= 'twaitz')
365
366     #number of available buses:
367
368     n_buses= m.addVar(vtype=GRB.INTEGER, name='n_buses')
369
370     n_freq=m.addVar(vtype=GRB.INTEGER, name='n_freq')
371
372     x=m.addVars(L,vtype=GRB.CONTINUOUS)
373
374     x_inverted=m.addVars(L,vtype=GRB.CONTINUOUS)
375
376     meters_covered=m.addVar(vtype=GRB.INTEGER, name='meters_covered')
377
378     #Objective
379     obj= sum(bivtz[a] * t_ivt[a] for a in i) + sum(waiting_multiplier * t_waitz[a]
380           * g5[a] for a in i) + sum(waiting_multiplier * csi * g3[a] * vals_volume[a] for
381           a in i)
382     m.setObjective(obj,GRB.MINIMIZE)
383
384     #Constraints
385     m.addConstr(n_freq == sum(Nr[l] for l in L))
386
387     m.addConstr(n_buses== sum(c[arc]*(x_inverted[l]) for l in L for arc in l))
388
389     m.addConstr(meters_covered == sum(d[arc]*fr[l] for l in L for arc in l))
390
391     m.addConstr(meters_covered <= max_meters_hour)
392
393     m.addConstr(n_buses <= max_number_buses)
394     for l in L:
395         for arc in l:
396
397             m.addConstr(fr_all_links[arc] == fr[l]/60)
398
399             m.addConstr(Nr[l] <= max_freq)
400
401             m.addConstr(Nr[l] == fr[l]*T)
402
403             m.addConstr(fr[l] * fr_inverted[l] == 1)
404
405             m.addConstr(x[l] == 60 * fr_inverted[l])
406
407             m.addConstr(x[l] * x_inverted[l] == 1)

```

```

408     m.addConstr(t_wait[arc] == 60 * fr_inverted[l]*0.5)
409
410     m.addConstr((g3[arc] == 0) >> (t_wait[arc] <= csi))
411
412     m.addConstr((g3[arc] == 1) >> (t_wait[arc] >= csi+0.1))
413
414     m.addConstr((g4[arc] == 0) >> (t_wait[arc] >= csi+0.1))
415
416     m.addConstr((g4[arc] == 1) >> (t_wait[arc] <= csi))
417
418     m.addConstr((g5[arc] == g4[arc] * g2[arc]))
419
420     m.addConstr(g2[arc] == 1 if arc in boarding_arc else g2[arc] == 0)
421
422     #Input volume from the Passenger Assignment
423     for arcs in vals_volume.keys():
424         if arcs==arc:
425             m.addConstr(z[arc] == vals_volume[arcs] * fr_inverted[l])
426
427             m.addGenConstrPWL(z[arc], bivt[arc], [0, bus_seat_capacity,
bus_capacity, 1000], [1, 1, crowding_perception, 100000])
428
429             m.addConstr(bivtz[arc] == vals_volume[arcs] * bivt[arc])
430
431             m.addConstr(t_waitz[arc] == vals_volume[arcs] * t_wait[arc])
432
433     m.optimize()
434
435     # =====
436     # PRINT STATS AND RESULTS
437     # =====
438     if m.status == GRB.OPTIMAL: #check if the solver is capable of finding an
optimal solution
439         print(m.status,'optimal')
440         print('Obj: %g' % m.objVal)
441         m.printStats()
442         m.printAttr('X')
443
444
445     #Adjust frequency of travel and alightning arcs to infinity for the
iterations to proceed
446     f={k: v.X for k,v in fr_all_links.items()}
447     q = f.keys()
448
449     for i in q:
450         if i not in boarding_arc:
451             f[i] =100000
452

```

```

453     print('vals_volume: ', vals_volume) # The passenger volume distributed
through the network after the assignment
454     print('v_transfer: ', v_transfer) # The number of transfers at a network
level
455     print('Nr: ',Nr) # The frequency of every line
456     model.write("out.attr")
457     m.write("out1.attr")
458
459
460     var_names = []
461     var_values = []
462
463     for var in model.getVars():
464         if var.X > 0:
465             var_names.append(str(var.varName))
466             var_values.append(var.X)
467     for var in m.getVars():
468         if var.X > 0:
469             var_names.append(str(var.varName))
470             var_values.append(var.X)
471
472     # Write to csv
473     with open('Bi_level_results.csv', 'w') as myfile:
474         wr = csv.writer(myfile, quoting=csv.QUOTE_ALL)
475         wr.writerow(zip(var_names, var_values))
476
477
478     else:
479         print(m.status,'not optimal')
480         m.printStats()
481         m.getVars()
482     #Convergence Parameters
483
484
485     print("--- %s seconds ---" % (time.time() - start_time))
486     # avg = statistics.mean(vals_volume.values())
487     # #Change here the convergence parameters: rel_tol (relative tolerance) -> the
one to adjust
488     # if lastavg is not None and math.isclose(avg, lastavg, rel_tol=tolerance,
abs_tol =0 ):
489         #     break
490     # lastavg = avg
491     i=i+1 # Only needed if we run 1 iteration; else comment out

```


References

- [Iba, 2014] (2014). An integrated approach for timetabling and vehicle scheduling problems to analyze the trade-off between level of service and operating costs of transit networks. *Transportation Research Part B: Methodological*, 70:35–46.
- [Jha, 2019] (2019). A multi-objective meta-heuristic approach for transit network design and frequency setting problem in a bus transit system. *Computers and Industrial Engineering*, 130:166–186.
- [Afandizadeh et al., 2013] Afandizadeh, S., Khaksar, H., and Kalantari, N. (2013). Bus fleet optimization using genetic algorithm a case study of Mashhad. Technical Report 1.
- [Aloi et al., 2020] Aloi, A., Alonso, B., Benavente, J., Cordera, R., Echániz, E., González, F., Ladisa, C., Lezama-Romanelli, R., López-Parra, Á., Mazzei, V., Perrucci, L., Prieto-Quintana, D., Rodríguez, A., and Sañudo, R. (2020). Effects of the COVID-19 Lockdown on Urban Mobility: Empirical Evidence from the City of Santander (Spain). *Sustainability*, 12(9):3870.
- [Arbex and Cunha, 2015] Arbex, R. O. and Cunha, C. B. (2015). Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm. *Transportation Research Part B: Methodological*, 81:355–376.
- [Baaj, 1995] Baaj, M. H. (1995). Hybrid route generation heuristic algorithm for the design of transit networks. 3(1):31–50.
- [Borndörfer et al., 2005] Borndörfer, R., Grötschel, M., and Pfetsch, M. E. (2005). A Path-Based Model for Line Planning in Public Transport A Path-Based Model for Line Planning in Public Transport *. Technical report.
- [Buba and Lee, 2018] Buba, A. T. and Lee, L. S. (2018). A differential evolution for simultaneous transit network design and frequency setting problem. *Expert Systems with Applications*, 106:277–289.
- [Bucsky, 2020] Bucsky, P. (2020). Modal share changes due to COVID-19: The case of Budapest. *Transportation Research Interdisciplinary Perspectives*, 8:100141.
- [Carteni et al., 2021] Carteni, A., Francesco, L. D., and Martino, M. (2021). The role of transport accessibility within the spread of the Coronavirus pandemic in Italy. *Safety Science*, 133:104999.
- [Ceder and Israeli, 1998] Ceder, A. and Israeli, Y. (1998). User and Operator Perspectives in Transit Network Design. *Transportation Research Record: Journal of the Transportation Research Board*, 1623(1):3–7.

- [Chakroborty and Wivedi, 2002] Chakroborty, P. and Wivedi, T. (2002). Optimal route network design for transit systems using genetic algorithms. *Engineering Optimization*, 34(1):83–100.
- [Cui et al., 2021] Cui, Q., He, L., Liu, Y., Zheng, Y., Wei, W., Yang, B., and Zhou, M. (2021). The impacts of COVID-19 pandemic on China’s transport sectors based on the CGE model coupled with a decomposition analysis approach. *Transport Policy*, 103:103–115.
- [Dubois et al., 1979] Dubois, D., Bel, G., and Llibre, M. (1979). A set of methods in transportation network synthesis and analysis. *Journal of the Operational Research Society*, 30(9):797–808.
- [Fan et al., 2009] Fan, L., Mumford, C. L., and Evans, D. (2009). A simple multi-objective optimization algorithm for the Urban transit routing problem. In *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, pages 1–7.
- [Fan, 2006] Fan, Wei Machemehl, R. B. (2006). Optimal Transit Route Network Design Problem with Variable Transit Demand: Genetic Algorithm Approach. *Journal of Transportation Engineering*, 132(1):40–51.
- [Fan et al., 2016] Fan, Y., Guthrie, A., and Levinson, D. (2016). Waiting time perceptions at transit stops and stations: Effects of basic amenities, gender, and security. *Transportation Research Part A: Policy and Practice*, 88:251–264.
- [Fusco et al., 2002] Fusco, G., Università, M. P., Studi, D., Tre, R., Gori, S., and Petrelli, M. (2002). A heuristic transit network design algorithm for medium size towns Origin-Destination Trip Demand Matrix Estimation from Different Data Sources View project Design of the Delivery Scheme of Freight in Urban Areas View project A HEURISTIC TRANSIT NETWORK DESIGN ALGORITHM FOR MEDIUM SIZE TOWNS. Technical report.
- [Ghosh et al., 2020] Ghosh, A., Nundy, S., Ghosh, S., and Mallick, T. K. (2020). Study of COVID-19 pandemic in London (UK) from urban context. *Cities*, 106:102928.
- [Gkiotsalitis and Cats, 2020] Gkiotsalitis, K. and Cats, O. (2020). Public transport planning adaption under the COVID-19 pandemic crisis: literature review of research needs and directions. *Transport Reviews*, pages 1–19.
- [Guihaire and Hao, 2008] Guihaire, V. and Hao, J. K. (2008). Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42(10):1251–1273.
- [Hasselstrom, 1982] Hasselstrom, D. (1982). PUBLIC TRANSPORTATION PLANNING: A MATHEMATICAL PROGRAMMING APPROACH.
- [Ibarra-Rojas et al., 2015] Ibarra-Rojas, O. J., Delgado, F., Giesen, R., and Muñoz, J. C. (2015). Planning, operation, and control of bus transport systems: A literature review.
- [Lampkin and Saalmans, 1967] Lampkin, W. and Saalmans, P. D. (1967). The Design of Routes, Service Frequencies, and Schedules for a Municipal Bus Undertaking: A Case Study. *Journal of the Operational Research Society*, 18(4):375–397.
- [Lee and Vuchic, 2005] Lee, Y.-J. and Vuchic, V. R. (2005). Transit Network Design with Variable Demand. *Journal of Transportation Engineering*, 131(1):1–10.

- [López-Ramos et al., 2017] López-Ramos, F., Codina, E., Marín, Á., and Guarnaschelli, A. (2017). Integrated approach to network design and frequency setting problem in railway rapid transit systems. *Computers and Operations Research*, 80:128–146.
- [Lüthi et al., 2007] Lüthi, M. ., Weidmann, U. ., and Nash, A. (2007). Passenger arrival rates at public transport stations.
- [Mandl, 1980] Mandl, C. E. (1980). Evaluation and optimization of urban public transportation networks. *European Journal of Operational Research*, 5(6):396–404.
- [Meng et al., 2018] Meng, M., Rau, A., and Mahardhika, H. (2018). Public transport travel time perception: Effects of socioeconomic characteristics, trip characteristics and facility usage. *Transportation Research Part A: Policy and Practice*, 114:24–37.
- [Ngamchai and Lovell, 2003] Ngamchai, S. and Lovell, D. J. (2003). Optimal Time Transfer in Bus Transit Route Network Design Using a Genetic Algorithm. *Journal of Transportation Engineering*, 129(5):510–521.
- [Nikolić and Teodorović, 2014] Nikolić, M. and Teodorović, D. (2014). A simultaneous transit network design and frequency setting: Computing with bees. *Expert Systems with Applications*, 41(16):7200–7209.
- [Pattnaik et al., 1998] Pattnaik, S. B., Mohan, S., and Tom, V. M. (1998). Urban Bus Transit Route Network Design Using Genetic Algorithm. *Journal of Transportation Engineering*, 124(4):368–375.
- [Pel et al., 2014] Pel, A. J., Bel, N. H., and Pieters, M. (2014). Including passengers’ response to crowding in the Dutch national train passenger assignment model. *Transportation Research Part A: Policy and Practice*, 66(1):111–126.
- [Przybylowski et al., 2021] Przybylowski, A., Stelmak, S., and Suchanek, M. (2021). Mobility Behaviour in View of the Impact of the COVID-19 Pandemic—Public Transport Users in Gdansk Case Study. *Sustainability*, 13(1):364.
- [Silman et al., 1974] Silman, L. A., Barzily, Z., and Passy, U. (1974). Planning the route system for urban buses. *Computers and Operations Research*, 1(2):201–211.
- [Spiess and Florian, 1989] Spiess, H. and Florian, M. (1989). Optimal strategies: A new assignment model for transit networks. *Transportation Research Part B*, 23(2):83–102.
- [Stradling et al., 2007] Stradling, S., Carreno, M., Rye, T., and Noble, A. (2007). Passenger perceptions and the ideal urban bus journey experience. *Transport Policy*, 14(4):283–292.
- [Szeto and Jiang, 2014] Szeto, W. Y. and Jiang, Y. (2014). Transit route and frequency design: Bi-level modeling and hybrid artificial bee colony algorithm approach. *Transportation Research Part B: Methodological*, 67:235–263.
- [Szeto and Wu, 2011] Szeto, W. Y. and Wu, Y. (2011). A simultaneous bus route design and frequency setting problem for Tin Shui Wai, Hong Kong. *European Journal of Operational Research*, 209(2):141–155.
- [TfL, 2019] TfL (2019). Annual Report and Statement of Accounts 2019/20. Technical report.

- [TfL, 2020] TfL (2020). All London buses now taking payments as capacity limits are safely increased - Transport for London.
- [Tirachini, 2020] Tirachini, A. (2020). The Journal of Public Transportation is published by the Center for Urban Transportation Research at the University of South Florida Journal of Public Transportation | scholarcommons. *Journal of Public Transportation*, 22(1):1–21.
- [Tom and Mohan, 2003] Tom, V. M. and Mohan, S. (2003). Transit route network design using frequency coded genetic algorithm. *Journal of Transportation Engineering*, 129(2):186–195.
- [UITP, 2020] UITP (2020). PUBLIC TRANSPORT AUTHORITIES AND COVID-19 IMPACT AND RESPONSE TO A PANDEMIC. Technical report.
- [UITPb, 2020] UITPb (2020). KNOWLEDGE BRIEF. Technical report.
- [Wan and Lo, 2003] Wan, Q. K. and Lo, H. K. (2003). A Mixed Integer Formulation for Multiple-Route Transit Network Design. Technical report.
- [Watkins et al., 2011] Watkins, K. E., Ferris, B., Borning, A., Rutherford, G. S., and Layton, D. (2011). Where Is My Bus? Impact of mobile real-time information on the perceived and actual wait time of transit riders. *Transportation Research Part A: Policy and Practice*, 45(8):839–848.
- [Wei Fan and Machemehl, 2004] Wei Fan, A. and Machemehl, R. B. (2004). Optimal Transit Route Network Design Problem: Algorithms, Implementations, and Numerical Results 6. Performing Organization Code Unclassified. Technical report.
- [Wielechowski et al., 2020] Wielechowski, M., Czech, K., and Grzęda, Ł. (2020). Decline in Mobility: Public Transport in Poland in the time of the COVID-19 Pandemic. *Economies*, 8(4):78.
- [Wilbur et al., 2008] Wilbur, M., Ayman, A., Ouyang, A., Poon, V., Kabir, R., Vadali, A., Pugliese, P., Freudberg, D., Laszka, A., and Dubey, A. (2008). Impact of COVID-19 on Public Transit Accessibility and Ridership. Technical report.
- [Yap et al., 2020] Yap, M., Cats, O., and van Arem, B. (2020). Crowding valuation in urban tram and bus transportation based on smart card data. *Transportmetrica A: Transport Science*, 16(1):23–42.
- [Zhao, 2006a] Zhao, F. (2006a). Large-Scale Transit Network Optimization by Minimizing User Cost and Transfers. *Journal of Public Transportation*, 9(2):107–129.
- [Zhao, 2006b] Zhao, F. (2006b). Large-Scale Transit Network Optimization by Minimizing User Cost and Transfers. *Journal of Public Transportation*, 9(2):107–129.
- [Zhao and Gan, 2003] Zhao, F. and Gan, A. (2003). Optimization of Transit Network To Minimize Transfers. page 153.
- [Zhao and Zeng, 2007] Zhao, F. and Zeng, X. (2007). Optimization of User and Operator Cost for Large-Scale Transit Network. *Journal of Transportation Engineering*, 133(4):240–251.
- [Zhao et al., 2015] Zhao, H., Xu, W., and Jiang, R. (2015). The Memetic algorithm for the optimization of urban transit network. *Expert Systems with Applications*, 42(7):3760–3773.

- [Zhou et al., 2020] Zhou, P., Yang, X. L., Wang, X. G., Hu, B., Zhang, L., Zhang, W., Si, H. R., Zhu, Y., Li, B., Huang, C. L., Chen, H. D., Chen, J., Luo, Y., Guo, H., Jiang, R. D., Liu, M. Q., Chen, Y., Shen, X. R., Wang, X., Zheng, X. S., Zhao, K., Chen, Q. J., Deng, F., Liu, L. L., Yan, B., Zhan, F. X., Wang, Y. Y., Xiao, G. F., and Shi, Z. L. (2020). A pneumonia outbreak associated with a new coronavirus of probable bat origin. *Nature*, 579:270–273.