Dynamic request assignment in aerial ride sharing operations

AE5310 - Thesis Control and Operations

K. Nikolakopoulos

Dynamic request assignment in aerial ride sharing operations

AE5310 - Thesis Control and Operations

by

K. Nikolakopoulos

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 23 April 2021.

Student number:	4348974
Project duration:	March 2020 – April 2021
Thesis committee:	Chair: Dr. M.D. Pavel
	Daily supervisor: Dr. A. Bombelli
	Member: Ir. P.C. Rolling

TU Delft

Acknowledgements

The completion of this thesis would not be possible without the support of my supervisor A. Bombelli. I am grateful for all the help and support you gave me throughout this project and made this day possible. A big thanks to my friends who morally supported me during the project and particularly at times where it was not easy for me. Lastly, I want to thank my parents who made my studies possible and therefore I want to dedicate this thesis to them.

K. Nikolakopoulos
Delft, April 2021

Contents

List of Figures

List of Tables

List of Abbreviations

ALNS	Adaptive Large Neighborhood Search
ATC	Air Traffic Control
B&B	Branch and Bound
B&C	Branch and cut
B&P	Branch and Price
B&PC	Branch and Price and Cut
D&D	Dynamic and Deterministic
D&S	Dynamic and Stochastic
DARP	Dial A Ride Problem
DDT	Desired Delivery time
DOC	Delivery Oriented Customers
DP	Dynamic Programming
DPT	Desired Pick up time
DRT	Directed ride time
DSV	Desired Service Time
ERT	Excess Ride Time
FAM	Fleet Assignment Model
FAP	Fleet Assignment Problem
FCFS	First-Come-First-Serve
GA	Genetic Algorithm
GDP	Gross Domestic Product
LNS	Large Neighborhood Search
LOD	Level of Dissatisfaction
LS	Local Search
MRT	Maximum Ride time limitation
O-D	Origing - Destination
OF	Objective Function
P2P	Point to Point
PDP	Pick-up and Delivery Problem
PDPTW	Pick-up and Delivery Problem with Time Windows

POC	Pick-up Oriented Customers
S&D	Static and Deterministic
S&S	Static and Stochastic
SA	Simulated annealing
SARP	Share-A-Ride Problem
SF	Safety Factor
SOC	State of Charge
TS	Tabu Search
TSN	Time Space Network
TW	Time Window
UAE	United Arab Emirates
UAM	Urban Air Mobility
UMS	Urban Mobility Services
USA	United States of America
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VTOL	Vertical Take-Off and Landing

Introduction

In response to traffic congestions commercial companies such as Uber will operate a small fleet of vehicles, by 2023 in the US, to accommodate aerial ride-sharing operations. Such a way of transportation reduces drastically transportation times, reducing ground congestion issues. However, before a real implementation there are challenges that need to be addressed. These can be categorized between operational and performance related. Operational related challenges include air traffic safety, noise limitations in urban areas, accessibility to pick up and delivery locations and the relevant associated costs. Performance limitations concern mainly reliability and durability of batteries.

The focus of this thesis is to accommodate demand during real-time operations. This is done similarly to ground transportation companies such as Uber and Bolt where new requests are received instantly, and part of the pre-planned routes need to be re-adjusted, to accommodate these requests. It is therefore necessary to develop an algorithm that provides a sub-optimal solution, in order to re-plan these routes in a limited computational time. This is done in order to be used in real-life operations.

This thesis report is organized as follows: In Part I, the scientific paper is presented and part II contains the relevant Literature Study that supports the research. Finally, in Part III, the customer data, optimized customer solution and routes are presented to the reader for all instances.

I

Scientific Paper

Solving the dynamic request assignment problem in aerial ride sharing operations: A metaheuristic approach

Konstantinos Nikolakopoulos,*

Delft University of Technology, Delft, The Netherlands

Abstract

The concept of Urban Air Mobility (UAM) services was created mainly in response to traffic congestions. In this research we focus on UAM services such as those provided by Uber Elevate. We therefore present a framework to solve the Urban Air Mobility Problem with Time Windows (UAMP-TW) under dynamic demand, using an Adaptive Large Neighborhood Search (ALNS) algorithm. The objective of this study is to maximize the operational profit and consider customer satisfaction. Satisfaction is measured by two factors: (1) deviation from desired departure time to actual departure time and (2) deviation from nominal trip duration to actual trip duration. In our analysis we aim to determine a relationship between customers and their contribution towards profit. We address this by running simulation instances that cover three operational scenarios: a morning and evening commuter transportation case (scenarios 1 and 2) and the an occurrence of an event at a specific location (scenario 3). Multiple simulation runs indicated stability, for all three instances, due to low variation of the profit from the mean. A sensitivity analysis on the customers' time-window lengths, satisfaction factors and types concluded that customers with higher time-window lengths are more profitable since it is easier to share-rides with other users. The analysis also showed that when the satisfaction factors have a higher weight in the deviation from the departure time than the trip duration, the overall customer satisfaction is increased together with the profit and the percentage of customers who share rides. Scenario 1 has a higher rate of rebalancing empty vehicles because most requests are generated in the suburbs while the depot is located downtown. This leads to a lower vehicle deployment. In scenarios 2 and 3, most requests are generated downtown and thus more vehicles are deployed. Under dynamic demand, the algorithm has an acceptance rate of new requests of about 90% while a penalty is given to customers who cancel a ride. Analysis showed that customers are rejected if an empty vehicle has to rebalance to their location unless they are premium. In terms of the computational efficiency the algorithm is able to handle between 40-50 requests simultaneously.

1 Introduction

Inadequate mass transit options, obstacles in the roads and development in areas where transit and road systems are 'poor' are the main reasons for traffic congestions. As a result, millions of hours are wasted every day on roads worldwide [Holden and Goel, 2016]. Apart from travel time uncertainties and increased probability of accidents, traffic congestions impact the environment and economy. In many regions across the world, vehicle emissions is the main source of air pollution. In this case the pollutants of vehicle emissions include carbon monoxide and dioxide, nitrogen oxides and hydrocarbons [Zhang and Batterman, 2013]. The increase of traffic congestions increase pollutant emissions, causing lower air quality and health related problems to people living near large road networks. It is estimated that the cost of congestion in the USA is approximately $ 121 billion per year, which corresponds to 1 % of its Gross Domestic Product [Alonso-Mora et al., 2017]. To put this number further into perspective, more than 5.5 billion hours are lost while in traffic and about 3 billion gallons of fuel is used. According to Uber [Holden and Goel, 2016] an average resident in San Francisco yearly spends on average 230 hours traveling between home and work. The population is expected to further increase in future decades [U.N D.E.S.A, 2019], and therefore there will be an increased need of transportation services. For the aforementioned reasons, Urban Mobility Services (UMS) would require to meet future transportation demand in a more sustainable way. This can be achieved either by investing in heavy-infrastructure concepts such as roads, tunnels, etc. or by new means. With the advance of technology, researchers are looking into new concepts of transportation such as Hyperloop, Urban Air Mobility services, smart roads, super trains and elevated cycle paths [Toesland, 2018].

Urban Air Mobility (UAM) services are effective, safe and a sustainable way of transporting passengers or goods in large metroplex areas. Vehicles will use vertiports to take-off and land. Operations within a

*MSc Student, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

UAM service will vary among news gathering, traffic studies, package deliveries, transportation services and other [Thipphavong et al., 2018]. This research focuses on transportation services alone, such as Uber elevate. According to a study performed by Uber [Holden and Goel, 2016], such services are expected to be fully operational by a small fleet of vehicles (e-VTOL aircrafts) by 2023 in the United States. A big advantage of UAM compared to other means of transportation is that route durations are significantly decreased. An example would be that a 2 h 12 min ride could be performed in 15 min [Holden and Goel, 2016]. Such travel reduction will significantly reduce traffic congestions, making travel within a metroplex area more efficient. Additionally, vehicles are fully electric resulting to zero emission pollutants.

At the same time the challenges of such services needs to be addressed before a real implementation can take place. These can be categorized as operational and performance related. Operational challenges concern real time scheduling of the fleet, air traffic safety, accessibility to pick up and drop off locations, noise limitations in urban areas, charging rates for customers and costs associated with the construction of vertiports. Vehicle performance challenges are associated mainly with battery reliability and durability.

Most recent research in this field relates to departure and arrival sequencing, trajectory management, conflict resolutions, air traffic control, optimum arrival times and concept of operations [Thipphavong et al., 2018, Bosson and Lauderdale, 2018, Mue, 2017, Vascik and Hansman, 2017]. This research is therefore primarily focused on the operations of such services. The objective is to schedule real-time (dynamic) flights of vehicles in metroplex areas, while maximizing operational profit by considering customer satisfaction. As it is known from similar transportation problems, some requests are received instantly. Therefore, a part of the pre-planned routes has to be adjusted, in order to accommodate these passengers, only if they have a positive contribution towards the profit. This will be achieved by developing a meta-heuristic algorithm, which is able to provide sub-optimal solutions in a limited computational time. The problem will be applied to different time horizons which are instance-dependant and allow simulation with high and low forecasted demand peaks.

Since this type of problem is not operation yet, the customer data and vertiport locations, will be based upon previous published works. The model can be verified, by comparing the results with an exact approach, compare the results with previous research or a sensitivity analysis. For the purpose of this study, the final approach is chosen. As there is no real-life data available, validation is not possible.

The structure of the remaining paper is the following: In section 2, the relevant academic literature is reviewed. In section 3, the problem is defined and the set of operational constraints together with assumptions are described. Section 4, describes the construction of the algorithm and discuses in detail the main inputs of the model and how this data is constructed. Section 5, discusses in detail the elements of the chosen meta-heuristic algorithm and the order of events taken to solve the UAMP. The results of the different instances are then described and analyzed in section 6, while the conclusion and recommendations for future work are provided in section 7.

2 Literature review

The UAMP can be seen as a combination of both ground and air transportation services with, however, several fundamental differences. This shows the importance of designing new operational models, specifically designed for UAMP, that are based on a new set of constraints and assumptions. Nevertheless, the UAMP is classified under the vehicle routing problem (VRP) thus making it non-deterministic (NP-hard). Similar transportation problems are the Pick-up and delivery problem (PDP), Dial a ride problem (DARP) and Share a ride problem (SARP). For convention, the referral to those problems will be done by their acronyms.

The PDP is a variation of the VRP where a set of routes needs to satisfy several transportation requests of goods [Savelsbergh and Sol, 1995]. The problem consists of a central [Ropke and Pisinger, 2006] or multiple depots [Jewpanya et al., 2016]. Vehicles visit the *pick-up* and *delivery* nodes to pick-up and drop off the parcels respectively. According to [Savelsbergh and Sol, 1995] requests may have a set of origin but a single destination, a single origin but a set of destination, multiple origins and destinations and finally different starting and ending locations. In most cases requests have a specified time-window which informs the range at which these requests can be picked up or be delivered. This is known as the pick up and delivery problem with time-windows (PDPTW) [Ropke and Pisinger, 2006]. The mathematical formulation of any PDP variation, can be done either via a connection network approach [Savelsbergh and Sol, 1995] or a time space network approach [Mahmoudi and Zhou, 2016].

The DARP is a variation of the PDP where vehicles are transporting passengers instead of goods; more particularly elderly and disabled people. Users can share rides, as they are transported in the same vehicles (mini-buses). It is often the case that the same user will request the service twice in a day. This is because there are outbound requests (from home to destination) and inbound requests (from destination to home)

[Cordeau and Laporte, 2007]. Furthermore, DARP usually works with time-windows , which means that users do not specify an exact pick-up/delivery time of the day but rather a time interval period. They are only allowed to specify either their pick-up or delivery time-window and then the operator is responsible to assign the remaining time-window to that customer. Since in DARP users are sharing rides, the main issue that occurs is that customers do not want to take large detours to arrive at their destinations even if it could be beneficial for the system's performance. Hence a maximum ride time constraint is specified for each transport request.

The SARP is a combination of the DARP and PDP where people and parcels share the same vehicle (taxis) [Li2, 2014]. Since customer satisfaction has an impact on the service level, people requests have priority over parcels. By combining people and parcels together, the expected time of the trip will increase as more time is required to pick up the parcel. An important parameter is that all passengers need to be satisfied, while parcel accommodation is optimized. Since people and parcels require different types of service a different price is associated to them. If a taxi stops too many times or exceeds the maximum ride time significantly, it is assumed that the passenger will refuse to pay [Li2, 2014]. There are differences between the service requirements of people and parcels. For example, the pick-up/delivery time-windows of passengers are more critical. Detours of original routes can be too time consuming thus not acceptable by customers. Combinations of parcel pick-ups might not be possible due to vehicle capacity. Finally, the cost-benefit relationship between people transportation and parcels delivery is different. In the dynamic scenario of SARP passengers are accommodated at the time of their call. These customers provide a pick up location and a desired pick-up/delivery time window. In the case where more than 1 taxi is available the one closest to the customer will be re-directed [Li2, 2014]. Thus, it is necessary to have a monitor that observes the states of the taxi in terms of capacity, location, loads etc. If possible, parcels are added in the route. Nevertheless, an interesting parameter in the formulation of the SARP [Li2, 2014] is the discount factor, which is applied when passengers exceed their direct delivery time. This factor can be found in the objective function, and compares the extra ride time of passengers with the direct delivery time.

When looking at the solution approaches to solve transportation problems, these are defined between exact and approximate. When the nature of the problem is dynamic, most researches typically tend to choose approximate solutions as they provide a good quality solution (sub-optimal) in a limited computational time. The most used algorithms for a transportation problem are the Tabu search (TS), Adaptive large neighborhood search (ALNS) and Genetic algorithms (GA).

[Cordeau and Laporte, 2003] have incorporated advance TS concepts with diversification strategies to solve the DARP. A diversification strategy forces the solution search in unexplored areas of the feasible regions. This is achieved by penalizing frequently made moves and temporarily accepting infeasible solutions. Such a method is proven to provide effective and efficient results. A parallel TS implementation for the dynamic DARP is presented in [Attanasio et al., 2004], where a high percentage of users is satisfied. [Gendreau et al., 2006] suggested a TS in a VRP where new requests occur in real time and specify a pick-up/delivery location. The neighborhood structure was based on ejection chains and results depicted good quality solutions in a real-time application.

[Ropke and Pisinger, 2006] solved the PDPTW using an ALNS algorithm. *Shaw removal heuristics, random and worst removal heuristics* were used for the destroy functions whereas *basic greedy and regret heuristics* were used for the repair one. The reason for choosing all of the above mentioned heuristics methods is that for one instance type one heuristic method can be more suited than another. The ALNS proved to be applicable to most problems while being effective and efficient due to its robust and fast construction heuristics.

[Barkaoui and Gendreau, 2013] solved the dynamic VRP with GA. The problem included real time information with the use of variable travel times. These variables were updated with using a dynamic traffic simulation. The outcome showed a significant cost reduction when incorporating the dynamic VRP with real time data. The algorithm showed good performance and was able to compute the solution in a quick time. [Taniguchi and Shimamoto, 2004] solved the dynamic VRP with time-windows using adaptive evolutionary algorithms. A comparison between the adaptive and the hybrid algorithm was made to compare the quality of the solutions and the robustness of the algorithms. As a result, the adaptive algorithm showed better performance than the hand-tuning ones.

The objective function (OF) of such problems is typically varying among minimizing duration, completion time, travel times, route length, customer dissatisfaction, vehicle numbers, and costs or maximizing vehicle utilization, profit [Savelsbergh and Sol, 1995]. In a dynamic environment it is not always clear what OF to use. This is because several routes in the next time point might be re-adjusted as a new request can appear or a customer cancels their request. [Savelsbergh and Sol, 1995] indicate that dynamic problems should emphasize on metrics which affect the near future more than the entire time horizon.

3 Problem Description

In section 3.1, the problem is described together with the research objectives. Section 3.2, proposes the mathematical formulation, describes the list of constraints and defines the related parameters. For a synopsis of the parameters readers are referred to appendix C.

3.1 Problem statement

In order to decide what type of fleet to use and how to dispatch the vehicles, commercial companies such as Uber would have to make tactical and strategic decisions [Shihab and Wei, 2019]. This is done in order to satisfy the forecasted demand, while maximizing profit margins. Similarly to other transportation problems, this network will consist of a central depot where the vehicles start and finish their route. The depot is situated in the downtown area as [Vascik and Hansman, 2017] have suggested.

Every customer has fixed associated information when reserving the service, which are fed as input to the optimization algorithm. These include (a) **Origin**: Pick-up location, (b) **Destination**: Delivery location, (c) **Time Window (TW)**: Time interval with which the customer, specifies the earliest and the latest time at which they desire to be picked-up or delivered according to his orientation. It is assumed that the early TW corresponds to the customer's ideal time to start/end their journey, (d) **Customer orientation**: Pick-up (POC) or delivery oriented customers (DOC). POC, refers to the customers who want to be picked up at a particular TW while DOC to the customers who want to be delivered at a particular TW, (e) **Customer type**: Standard and premium customers. Standard customers pay a lower rate fee but are willing to share a ride with other customers. Premium customers pay a higher rate fee but travel alone and have a higher priority, (f) ETW factor: Factor associated with the deviation between the desired departure/ arrival time and the actual one, and (g) **Duration factor**: Factor associated with the deviation between the minimum and actual trip duration.

Once the customer's trip is completed, the following information is recorded: (a) **Initial charge**: Fixed fare per km from origin to destination and variable fare per h, depending on the time spent on the vehicle, (b) **Satisfaction**: Measures the quality of service based on the ETW and duration factors, (c) **Eligibility of discount**: If the customer satisfaction is lower than a given threshold, a discount is given to the customer, and (d) **Final charge**: If the customer is eligible for discount, that value is subtracted from the initial charge.

An important constraint of the problem, is that all pre-booked customers must be satisfied, while dynamic customers are accommodated only if they have a positive contribution to the operational profit. In short, the model tries to maximize the profit of the operations while avoiding large detours since they might contribute to higher discount.

3.2 Problem Formulation

The set of customers C is defined such that $C = C^s \cup C^d$. The first subset refers to the static customers, while the second one to the dynamic ones. Static customers, are the ones who have booked a trip in advance and their cardinality is known while dynamic are unexpected customers and their cardinality is unknown. They appear in the system at the time of booking, consistently with on-demand transport services. When an unexpected customer j has been assessed and accepted by the network both static and dynamic sets are updated. This is done in such a way that $C^s = C^s \cup \{j\}$ and $C^d = C^d \backslash \{j\}$. Each set is split into two more subsets such that $C^s = C^s_p \cup C^s_d$ and $C_d = C^d_p \cup C^d_d$. The first subset refers to the customers who have a desired pick up time window (POC), thus cannot start the journey before a particular time, while the latter refers to the customers who have a desired delivery time window (DOC), thus they cannot be delivered at their destination after a certain time. This finalizes the customer set to: $C = C^s_{p,s} \cup C^s_{p,p} \cup C^s_{d,s} \cup C^s_{d,p} \cup C^d_{p,s} \cup C^d_{p,p} \cup C^d_{d,s} \cup C^d_{d,p}$.

Every customer request i, belongs in the customer set C such that $i \in C$. As specified in section 3.1, for every customer i there is an associated time window (e_i, l_i) which is constrained to be either a pick up or a delivery one. For every request i, there is an associated pick up and delivery node as explained in the following paragraphs. R_i is the associated actual travel time of that customer to go from the pick up node to the associated delivery node. Additionally, every request i has a processing time (p_i), that includes the embarking time at the pick up node and the disembarking time at the delivery node. Customer satisfaction (s_i) is calculated using eqs. (1) and (2), while the meaning the variables is located in table 17. For dynamic customers, there is an additional factor that measures satisfaction which has a higher weight than the other two and describes whether or not the customer has been accomodated. This is shown by eq. (3) and eq. (4). The extra term ϕ_i, is a binary value [0,1]. In case $\phi_i = 0$, the customer is discarded since he has not been accepted to be accommodated.

$$s_i^{POC} = \alpha_i \frac{\min(D_i, e_i)}{\max(D_i, e_i)} + \beta_i \frac{t_{i,i+\sigma}}{R_i} \quad (1)$$

$$s_i^{DOC} = \alpha_i \frac{\min(A_{i+\sigma}, e_i)}{\max(A_{i+\sigma}, e_i)} + \beta_i \frac{t_{i,i+\sigma}}{R_i} \quad (2)$$

$$s_{i_D}^{POC} = \alpha_i \frac{\min(D_i, e_i)}{\max(D_i, e_i)} + \beta_i \frac{t_{i,i+\sigma}}{R_i} + \epsilon_i \phi_i \quad (3)$$

$$s_{i_D}^{DOC} = \alpha_i \frac{\min(A_{i+\sigma}, e_i)}{\max(A_{i+\sigma}, e_i)} + \beta_i \frac{t_{i,i+\sigma}}{R_i} + \epsilon_i \phi_i \quad (4)$$

Customer satisfaction is used to calculate the associated customer discount (d_i). For customers i a discount is given according to table 1, where the discount is expressed as a percentage of the initial fare. The initial fare is calculated using eq. (5) and the final fare using eq. (6).

$$C_{i_{\text{initial}}} = d_{i,i+\sigma}\gamma_i + R_i\delta_i \quad (5)$$

$$C_{i_{\text{final}}} = C_{i_{\text{initial}}} - d_i C_{i_{\text{initial}}} \quad (6)$$

Table 1: Discount distribution

Satisfaction (s_i) [%]	Discount (d_i) [%]
95-100	0
85-95	5
70-85	10
0-70	20

Concerning the route structure, the Urban Air Mobility Problem (UAMP) is defined on a directed graph $G = (N, A)$, where N is the set of *nodes* and A is the set of *arcs*. N is the union of 4 different node sets such that N $= O_d \cup N_p \cup N_d \cup D_d$. Node type, description and range can be found in table 2. All types of nodes are located in vertiports. The number of nodes however is irrelevant to the number of vertiports.

Table 2: Node Information

Node type	Description	Enumeration (Range)	Cardinality
O_d	Origin depot	0	1
N_p	Pick-up nodes	1 to σ	σ
N_d	Delivery nodes	$\sigma + 1$ to 2σ	σ
D_d	Destination depot	$2\sigma + 1$	1
N	Set of nodes	0 to $2\sigma + 2$	$2\sigma + 2$

Even though the origin and destination depot have a different enumeration from a graph perspective, they both represent the same physical depot, located at the same vertiport. Additionally, having different node numbering for the depots, allows to model for unused vehicles whose route is a zero cost from O_d to D_d. Both the pick up and delivery nodes have the same cardinality $|N_p| = |N_d| = \sigma$ where σ simply represents the number of customer requests. As seen in similar problem formulations [Cordeau and Laporte, 2007, Savelsbergh and Sol, 1995], O_d is set to 0 and D_d is set to $2\sigma + 1$. Thus, the overall cardinality of nodes of N is $|N| = 2\sigma + 2$.

Pick up and delivery nodes are defined in such a way that the pick-up/ delivery node pair has the following form $(i, i + \sigma)$. Each node $i \in N_p$ is characterized by an associated TW $[e_i, l_i]$ which represents the earliest and latest time at which the service can start. The time window constraint is satisfied by the following inequality $e_i < B_i < l_i$, where B_i represents the start of service at that node. The same conditions are imposed to the delivery nodes N_d. For POC, the TW at the pick up node is given by the information customer i provided while the TW for their associated delivery node is calculated and defined as $e_{i+\sigma} = e_i + t_{i,i+\sigma}$ and $l_{i+\sigma} = e_i + xt_{i,i+\sigma}$, where x defines the maximum ride time factor. The time window constraint for the delivery node is then satisfied using the following inequality $e_{i+\sigma} < B_{i+\sigma} < l_{i+\sigma}$. For DOC, the delivery node TW is given by the provided information while the pick-up node TW is calculated $e_i = e_{i+\sigma} - xt_{i,i+\sigma}$ and $l_i = l_{i+\sigma} - t_{i,i+\sigma}$.

A, is the set of all feasible arcs between the nodes in the directed graph. Arc characterisation for entering and exiting nodes can be found in table 3.

Table 3: Arcs information

Node type	Entering arcs	Exiting arcs
O_d	-	$\sigma + 1$
N_p	σ or $\sigma - 1$	σ or $\sigma - 1$
N_d	σ or $\sigma - 1$	σ or $\sigma - 1$
D_d	$\sigma + 1$	-

For the first node (O_d) there are no entering arcs as the vehicle starts its route from the depot. The exiting arcs ($\sigma + 1$) lead towards either a pick-up node or node D_d, where the latter corresponds to an unused vehicle. Every pick-up node N_p has an entering condition (σ) when it originates from the depot or ($\sigma - 1$) when it originates from another pick-up node. Every delivery node N_d has the same entering and exiting conditions where (σ) is associated with pick-up nodes and ($\sigma - 1$) is associated with delivery nodes. The destination depot has an entering arc of ($\sigma + 1$) coming from a delivery node. No exiting edges are associated with that node since the route is ended at that node.

Every problem that belongs in the vehicle routing problem (VRP) family is composed of several constraints. Therefore, the UAMP consists of designing k vehicles routes on G such that:

(i) Every route starts and ends[1] at the depot,

(ii) The origin and destination depot correspond to the same physical location,

(iii) At the central depot customer requests can also be present,

(iv) For every request i arcs $a_i, a_{i+\sigma}$ belong to the same route and node $N_{i+\sigma}$ is visited after node N_i,

(v) Every request i cannot be served outside of its associated time window $[e_i, l_i]$

(vi) If customer's i satisfaction is lower than a given threshold a discount is given,

(vii) Customers that are classified as premium have priority and do not share rides with other users,

(viii) All static (pre-booked) customers must be accommodated while dynamic customers are only accommodated if they have a positive contribution to the objective function,

(ix) For every vehicle k the number of passengers on board cannot exceed the maximum vehicle capacity Q_k,

(x) At any moment in time, the battery level of vehicle k cannot be less than 10% of its maximum capacity due to safety regulations,

(xi) All vertiports are equipped with high voltage charging stations,

(xii) Charging of vehicles is allowed at pick-up nodes if there is a waiting time of at least 5 min and on delivery nodes the vehicle is set to charge for at least 10 min if there are no customers on board,

(xiii) If with the current charge a vehicle is not able to reach any destination, it is set to charge fully,

(xiv) Charging of vehicles with customers on board is not permitted,

(xv) The overall objective is to maximize the profit of the operations.

A simple example of an instance is observed in fig. 1. This instance takes into account 4 vertiports such that $V_p = [1, 2, 3, 4]$. Both depots are located in $V_p = 1$. Overall, 5 customers are associated with that instance, therefore $\sigma = 5$. From the aforementioned node enumeration the pick up nodes are set such that $N_p = [1, 2, 3, 4, 5]$ and the delivery nodes such that $N_d = [6, 7, 8, 9, 10]$. The origin depot is set that $O_d = 0$ and the destination depot $D_d = 11$ $(2\sigma + 1)$. Two different vehicles are dispatched where their routes are shown by the different colors r_1 (black route) and r_2 (red route). The route of $r_1 = [0, 1, 6, 2, 3, 7, 8, 4, 9, 11]$ and of $r_2 = [0, 5, 10, 11]$.

[1]The vehicle ends its journey at the depot only at the end of operations (00:00)

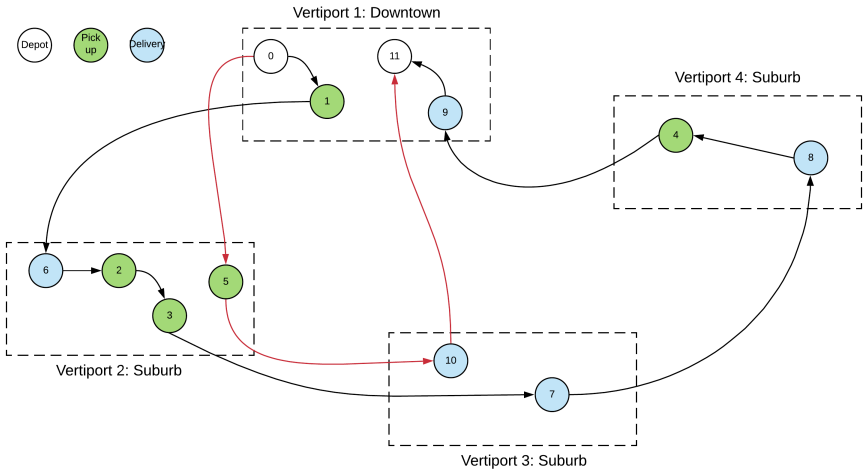

Figure 1: route example

4 Input parameters

A schematic drawing that contains the components of the operational model can be found in fig. 2. The initial stages to construct the network data are **demand distribution**, **vertiport design and distribution** and finally **fleet characteristics/ information**. Once these inputs are constructed an initial solution is generated. This solution is then optimized using an ALNS algorithm where the final results are obtained. Concerning the design and implementation of both the initial solution and ALNS please refer to section 5.

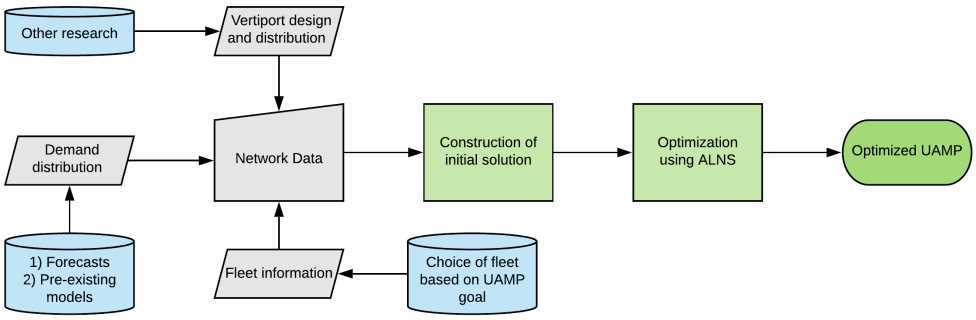

Figure 2: Components of operational model

The network data contains all the relevant information for the design of the UAM network. An important note is to understand that all the inputs to the model are related to each other. This is because, vertiport distancing is affected by the range of the vehicles whereas vertiport capacity defines the maximum number of vehicles that such a network can support. Moreover, demand satisfaction is affected by the number of available seats. Having more vehicles than the required demand, can cause loss of profit since not all the vehicles will be utilized. Additionally, demand directly affects the number and capacity of vertiports that are spread in the metropolitan area. All aforementioned factors have an impact on the strategic design of the UAM model since it directly impacts operating cost, profit margin and customer satisfaction.

4.1 Vertiport design and distribution

Vertiports are locations where vehicles can take-off and land. As suggested by [Bosson and Lauderdale, 2018], potential locations are hospital helipads, building roofs and municipal airports. At first, the UAM network should be incorporated in large metropolitan areas due to their feasible infrastructure and economic market. Some of these locations include Dubai, Sao Paolo, Los Angeles, Houston and Dallas-Forth Worth [Holden and Goel, 2016, Moon, 2017, Bosson and Lauderdale, 2018, Vascik and Hansman, 2017].

Different authors have performed research on vertiport distribution and have suggested a set of potential locations in Los Angeles [Vascik and Hansman, 2017] and Dallas [Bosson and Lauderdale, 2018]. The disadvan-

tage of using this data for this research is that it is specifically applied to certain locations. The approach taken by [Patterson et al., 2018] was found more appropriate for the focus of this study. It suggests that the vertiport distribution follows a symmetrical hexagonal pattern which allows to simulate any large metroplex city such as Huston, London, Dallas, etc. In total there are 7 vertiports where the one with coordinates at (0,0) represents the downtown area and the central depot while the remaining 6 the suburb areas. Unfortunately, this approach is limited to large coastal cities such as Los Angeles and New York city.

Figure 3, shows the vertiport distribution for this research. The vertiports are constructed using radial coordinates ($x = R\cos\frac{360}{n}$ and $y = R\sin\frac{360}{n}$) where R is the radius and n the number of vertiports based on (x,y)=(0,0). In table 4, the reader can get information concerning the distancing of the vertiports from the central coordinates, the types, together with the total vehicle and charging capacity. This vertiport layout will remain unchanged for all instances.

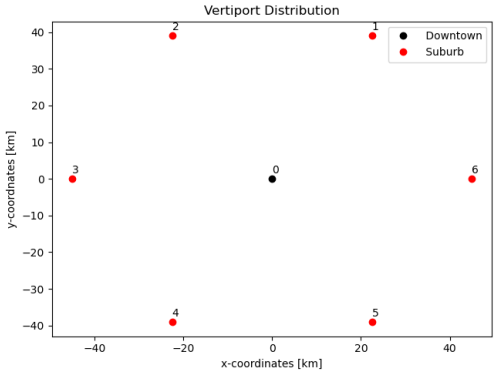

Figure 3: Vertiport distribution

Table 4: Vertiport specifications

Classification	Enumeration	Number	Radius [km]	Capacity	Charging capacity
Downtown	0	1	0	20	20
Suburb	1 to 6	6	45	20	20

Vertiport proximity defines the neighboring vertiports (distance proximity) in relation to each other. For *vertiport 0*: {0, 1, 2, 3, 4, 5, 6}, *vertiport 1*: {1, 0, 2, 6}, *vertiport 2*: {2, 0, 3, 1}, *vertiport 3*: {3, 0, 2, 4}, *vertiport 4*: {4, 0, 3, 5}, *vertiport 5*: {5, 0, 4, 6} and *vertiport 6*: {6, 0, 5, 1}.

4.2 Fleet characteristics

In recent years there have been many companies that develop e-VTOL vehicles including *Joby Aviation, E-Hang, Toyota, Lilium and Airbus* [Vascik and Hansman, 2017]. The specifications of every e-VTOL varies, impacting directly the operational constraints of the vehicles [Shihab and Wei, 2019]. This is because, every vehicle has a different range, cruise speed, seating capacity and charging capabilities. From an extensive literature review, it was observed that cruise speed vary from 96 km h^{-1} to 300 km h^{-1}, seating capacity from 2 to 5 passengers excluding pilots, range from 30 km to 300 km and finally battery capacity from 14.4 kW h to 63 kW h. The main reason for such variations among vehicle specifications is the different mission profiles that these vehicles are have, as explained in [Patterson et al., 2018]. The specification of potential e-VTOLs associated with this research as shown in table 5.

Table 5: List of potential e-VTOLs

	E-Hang 184	Kitty Hawk Cora	Lilium Jet	Joby S4
Passenger capacity	2	2	5	4
Cruise speed (km h^{-1})	130	180	252	322
Range (km)	30-40	160	300	240
Battery capacity (kW h)	14.4	63	38	N/A
Cruise Power (kW)	34.6	63	28	N/A

As mentioned in section 3, this research focuses on ride-sharing applications, meaning a desired seating capacity of 4-5 passengers. Additionally, vertiports can have a distance of more than 100 km apart from each other. It is therefore essential that the vehicle can perform a few routes without recharging. For this reason, the *Lilium Jet* is chosen for this research which is planned to be introduced by 2025. Concerning the battery consumption per flight, a linear relationship is assumed [Bacchini and Cestino, 2019] and is also independent of the number of passengers per flight leg. These values are shown in table 6.

Table 6: Mission profile times and required energy

Mission phase	Time [s]	Energy rate [$\cdot P_{\text{cruise}}$]
Embarking	180	0
Taxi-out	30	0.1
Take-off	30	3
Climb	60	2
Cruise	t_{ij}	1
Descent	60	2
Land	30	3
Taxi-in	30	0.1
Disembarking	180	0

Recharging of e-VTOLs takes places at the vertiports. It is assumed that all vertiports are equipped with high voltage charging stations [Shihab and Wei, 2019] and due to emerging battery technologies, charging times are estimated at 30 minutes [Vascik and Hansman, 2017]. Uber elevate [Holden and Goel, 2016] performed a study to calculate the cost per km, where it was estimated to be $ 1.02 (0.86 EUR).

4.3 Demand Distribution

One of the most important inputs in the UAM model is the Origin-Destination (O-D) market demand, representing the number of passengers willing to travel within the vertiports. A typical day of operations is categorized between morning/ evening and high/ low peak hours. During the morning hours, passengers commute from home to their destination (office, school, recreational activities), whereas in the evening the opposite behavior is observed [Shihab and Wei, 2019]. As a result, during the morning hours, the vehicles should be deployed near the suburb areas where typically residential areas are located.

High peak hours are the periods of the day where demand reaches its peak values. During real time operations a variation in the demand is associated with fluctuations in the prices (dynamic pricing) for that service as incorporated in [Shihab and Wei, 2019] work. This is inspired from Uber [Gurley, 2014] where prices vary at different times of the day. For this research, a simple fare model is chosen instead, as the aim is not to determine the 'ideal' charge rate but rather to identify the impact of incorporating different passengers types in the model. Therefore, fare prices are dependent on the distance and time spent in the vehicle as seen in Yook et al. [Yook and Heaslip, 2015] work.

Passenger transfers take place among vertiports and not from door to door. The latter concept could be applicable in the long term, where vehicles will be able to take off and land from private residencies [Holden and Goel, 2016]. Hence, customers who request an aerial ride would have to get to the nearest vertiport by themselves. This assumptions ensures that the distancing and the time required for passengers to arrive at the origin vertiport, is not taken into account and that customer requests are generated at the vertiports. Additionally, it is assumed that the start of daily operations will be at 06:00 and the end of the operations at 00:00. This means that between 00:00-06:00, customers will not be accommodated.

To identify the customer types, we looked at the demographics of Uber [McGrath, 2017] in USA. Data shows the age groups, income and urban context of using the service presented in appendix A. NASA Air mobility [Mobility, 2018] conducted a study to observe how these values will vary for UAM, where it was noted that the demand the first couple of years will be much lower compared to other services.

4.3.1 Static demand

These requests are created stochastically as shown in [Kohlman and Patterson, 2018] work. It is a probability curve (bimodal distribution) which consists of the sum of three normal distributions that are normalized to 1. This allows to model for high (8am, 4pm) and low (12pm) peak hours as seen in fig. 4. The curve is created for every vertiport and is generated by multiplying its "demand weight" (instance dependent and describes the popularity of the vertiport) with the curve. Requests are generated by comparing a random number (0-1) to

the curve. If the random number is lower than the curve's value a request is generated. The process is repeated for every time step.

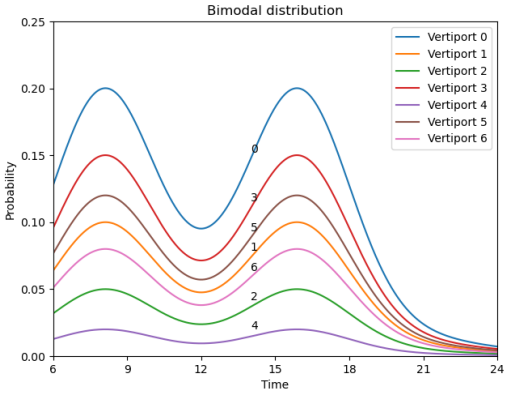

Figure 4: Example of Bimodal distribution

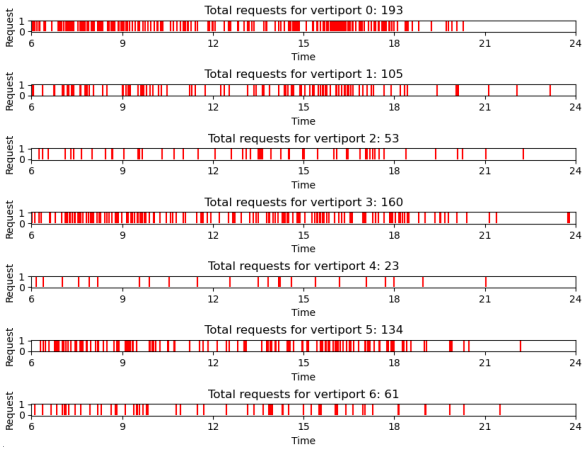

Figure 5: Number of requests

This method ensures that requests are generated at a particular vertiport, however it does not assign a destination, a time-window and customer classification. In order to assign a destination to the customer, a random number is generated between (0,1) where intervals represent the associated vertiport and they are instance dependant. The same process applies to assigning time-windows to the customers and classifying their status and orientation. Looking at the demographics (appendix A), in most instances premium customers will be approximately 30% of the total demand.

4.3.2 Dynamic demand

Dynamic demand refers to uncertainty associated with respect to the static demand. It can have the form of a request cancellation or a new customer appearing. An illustration of dynamic demand changes throughout a day of operations can be observed in fig. 6. Demand is updated every Δ time-steps and requests can be generated or cancelled for the time interval $[n\Delta, (n+2)\Delta]$, where $n = 0, 1, 2, ..., i$ and represents time progression. The reason why we chose to re-evaluate the demand every two time-steps is because we are interested in its effect for a short time period. Since $\Delta = 0.5h$ we are looking at new requests for the upcoming hour. In the figure 'grey' circles represent the static requests where the colorful ones represent dynamic changes. An 'X' refers to a cancellation of a request, while a colorful request represent an appearance of a new request for that particular time interval. During the first time interval $[t_0, t_2]$, 3 new requests have appeared while no customer has cancelled. In the second time interval though $[t_1, t_3]$, it is observed that 2 customers have cancelled and requests a service. What's important to understand in this case is the overlapping of events. Even-though a customer has cancelled in the time interval $[t_1, t_2]$, it only becomes known at $t = t_1$. This is done similarly to [Psaraftis et al., 2016], as it better simulates customer behavioral patterns.

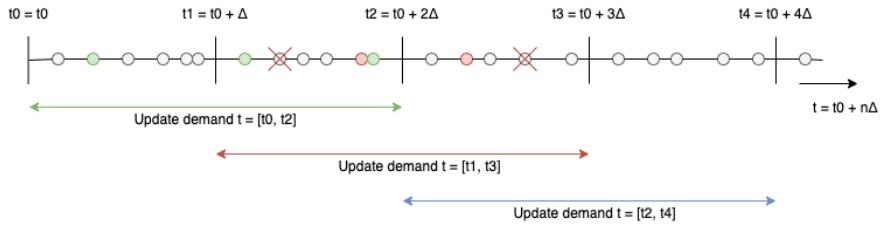

Figure 6: Demand evolution over time

Demand updates can be of the form; (1) no changes, (2) only cancellations, (3) only new requests, (4) both cancellations and new requests. From an operational perspective, as seen in [Ritzinger et al., 2016], cancellations are the least likely form of demand change to occur.

New requests, are assumed to be pick-up oriented only and are generated similarly to the static ones. From the bimodal distribution curve, the origin of the requests is determined together with the associated desired departure time. The assignment of destination, time-window and customer classification is done in exactly the same way as the static case. The question that rises when considering dynamic demand, has to do with

with how many new requests are cancelled or appeared. This is instance specific however its determined by looking at the time-interval of the day that the update happens. During high peak hours, more 'new' requests/ cancellations are likely to appear while at low peak hours the opposite behavior is observed.

5 Solution methodology

Firstly, a brief description of how the ALNS works is presented in section 5.1. The construction of the initial solution is described in section 5.2 while the main aspects and parameters that construct the ALNS are described in detail from section 5.3 to section 5.6. The pseudo-codes for all heuristics are given in appendix B.

5.1 ALNS description

As explained in section 2, there is a number of meta heuristic algorithms which are appropriate to solve this problem. The chosen algorithm is ALNS as it is able to explore large parts of the solution space (typically up to 40%) and is able to adapt to the characteristics of each instance while avoiding to get trapped in the local minima. This makes the algorithm very robust and suitable for different instances. It was firstly introduced by [Ropke and Pisinger, 2006] and for a thorough description of the algorithm readers are referred to [Potvin and Gendreau, 2019]. In general ALNS is based on the Large neighborhood Search (LNS) algorithm, where multiple destroy and insertion heuristics are used. Each heuristic has an associated weight that is adjusted dynamically as the search advances so that it adapts to each instance [Potvin and Gendreau, 2019]. Even though, the methodology of ALNS is the same in various problems its implementation is problem specific. Therefore, before describing its implementation it is essential to introduce further notations and concepts.

R is defined as the sets of routes which compose the solution to a specific instance. Each route $r \in R$ at the beginning of the operations (06:00) starts at the origin depot (*Node*: 0) and returns to the destination depot (*Node*: $2\sigma + 1$) only at the end of its daily operations (00:00). In the meantime, the vehicle is visiting nodes in order to satisfy customer requests. During the full day of operations each route (r) is characterised by an even number of nodes and there must be at least 4 nodes in the route. If the number of nodes is 4 that means that a single customer is accommodated by that vehicle. Each node $i \in r$ has time and vehicle related information. *Time related information* include: (1) A_i: Arrival time at node i, (2) B_i: Start of service at node i, (3) W_i: Waiting time at node i and (4) D_i: Departure time from node i. *Vehicle related information* include: (1) BLa_i: Arrival battery level at node i, (2) BLd_i: Departure battery level from node i, (3) BLc_{s_i}: Start of charge time at node i and (4) BLc_{e_i}: End of charge time at node i.

Additionally to the route related information, in section 3.2, other relevant node information is provided. The one with particular interest is the associated TW of every node $[e_i, l_i]$. With this information, $B_i = \max(e_i, A_i)$ and $W_i = B_i - A_i$. For node 0, the A_0 is set to 0 as the vehicle arrived at the depot at midnight (00:00). $D_0 = t_0$, where t_0 represents the start of operations. For the remaining of the route, we define i and j as 2 consecutive nodes. In this case, $A_j = D_i + t_{ij}$ and $D_j = B_j + p_j$ where p_j is the processing time and corresponds to passenger embarking or disembarking.

The goal of ALNS is to determine a feasible sub-optimal solution. This is done by allowing the algorithm to explore infeasible solutions, as it might lead to better final solution. An infeasible solution is defined as the one where any route has caused violation of the constraints. Infeasibility of constraints can be one of the following: (1) *Premium infeasibility*: When a premium customer shares a ride with either a standard or another premium customer, (2) *Vehicle capacity infeasibility*: Can only occur during a ride-sharing operation and the number of customers on board is larger than the vehicle capacity, (3) *Time window infeasibility*: Occurs when a customer is being accommodated outside of his desired time window, (4) *Maximum ride time infeasibility*: Occurs during ride sharing operations and concerns customers that spends more time on the vehicle than a certain threshold, (5) *Battery level infeasibility*: Occurs when the vehicle arrives at a node with less than 10% of its total battery capacity.

From the above information, there are two different types of solutions; **Feasible** where violation of constraints doesn't occur and **Infeasible** where at least one constraint has been violated. As ALNS is an iterative process, both solutions are accepted and this is because as seen in [Li et al., 2016] work, when accepting infeasible solutions the algorithm tends to determine a better final solution. However, when accepting an infeasible solution a penalty is added to the objective function as portrayed in eq. (7). In the equation T, P, B, R, V are respectively the violations for Time window, premium customers, battery levels, maximum ride time and vehicle capacity whereas $\epsilon_T, \epsilon_P, \epsilon_B, \epsilon_R, \epsilon_V$ are the associated weights. The values of these weights depend on how much we want to penalize the objective function in terms of feasibility.

$$J^* = J - (\epsilon_T T + \epsilon_P P + \epsilon_B B + \epsilon_R R + \epsilon_V V) \tag{7}$$

The pseudo code of the ALNS can be seen in line 1. The procedure is straight-forward. The algorithm starts with an input as initial solution (described in section 5.2) and then a part of the solution is destroyed using removal operators (described in section 5.3). The number of requests that are removed in every iteration are dependant on the parameter of ξ and satisfy $0.1n \leq q \leq \xi n$, where n represents the number of requests that are served by all vehicles. One has to note that the larger the value of ξ, the slower the algorithm becomes and the lower the value of ξ the higher the chance that the algorithm will be trapped in the local minima. In most research papers such as the one of [Ropke and Pisinger, 2006], when $\xi = 0.4$ it gives good quality solutions. These requests are then reinserted into the route using the insertion operators (described in section 5.4). The working principle for all insertion operators is the same; iterate over the removed customers (q), iterate over all vehicle routes ($r \in R$) and iterate over every flight leg of each vehicle. Depending on the function of the insertion operator, the 'best customer' is added in their best position. To determine the best position, the objective function of every combination must be calculated, which means that all time and vehicle related parameters need to be calculated too. This procedure is repeated until all customers have been inserted. For example, in an instance of 100 customers where 15% of the solution is destroyed (15 customers are removed) there are $C(100, 15) = \frac{100!}{15!85!} = 2.5 \cdot 10^{17}$ combinations of inserting these customers. As one expects, these many combinations per iteration affect dramatically the speed of the algorithm. It is therefore fundamentally important to define the size of the neighborhood and reduce unnecessary calculations.

To reduce unnecessary calculations, the concept of *memoization* is introduced. This is an optimization technique which stores the result of expensive functions and returns it when the same input is given [Norvig, 1991]. For this problem, this method is extremely beneficial when the same route is to be evaluated at different iterations. To reduce the size of the neighborhood (number of combinations) the following techniques were applied:

1. If a delivery node ($N_{i+sigma}$) is placed before its associated pick up node (N_i), the route is discharged as such a combination will only lead to infeasible routes,
2. From an operational point of view the First-In-First-Out (FIFO) method was introduced. This simply translates that any customer who boards in the vehicle has to be delivered first, discharging all other cases.
3. Premium customers have priority and thus need to be placed in the routes first. According to constraint (vii)[2] any combination that includes ride-sharing operation among premium customers is also discharged.
4. If a pick up node is inserted at some point in the route and its ETW is larger than the LTW of a following pick up node, the route is discharged, as it leads to Time Window infeasibility,
5. During ride sharing operations, if there are 2 or more consecutive pick up nodes and the travel time is higher than the maximum ride time of that customer, that route is also discharged,
6. Finally, during ride sharing operations, vertiports have an assigned property of 'proximity' which is defined in section 4.1. This method ensures that if a pick up occurs at a particular vertiport and another customer is to be picked-up at another vertiport, both the pick up and delivery nodes will be in the proximity of the first customer.

For the static problem the solution is straight forward. Once the demand is known for the time interval of interest, the algorithm solves it and determines a sub-optimal feasible solution (routes, associated time and vehicle information). However, when the nature of the problem is dynamic the solution methodology changes. This is because a part of the pre-planned routes (static solution) need to be re-evaluated once dynamic customers appear. This is achieved by using a time-horizon model inspired from [Bongiovanni et al., 2019]. A example of the time-horizon used in this model is shown in fig. 7. The 'blue' dots represent the static (pre-booked) requests which are known at $T = 0$. Each rolling-horizon has a seperate color to indicate its length set at $[nL, (n+2)L]$, where $n = 0, 1, ..., i$. In the aforementioned figure the following elements are shown: *Rolling horizon - 1*, 2 new requests appeared and 1 is cancelled, *Rolling horizon - 2*: 2 new requests appeared and 1 is cancelled and finally *rolling horizon - 3*: 4 new requests appeared and none was cancelled.

[2]Customers that are classified as premium have priority and do not share rides with other users

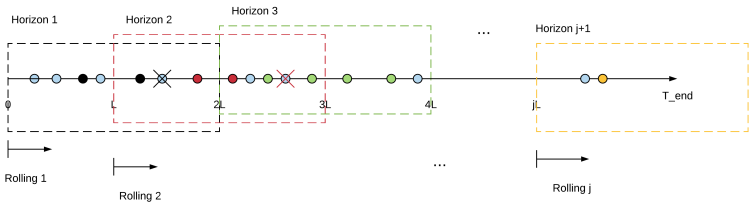

Figure 7: Example of rolling horizon with cancellations and extra requests

During execution the ALNS is first determining the set of sub-optimal feasible routes for the static requests. At the beginning of each planned interval phase, the algorithm determines whether or not there has been a change in the demand for that specific time interval. In cases where there hasn't been any change the algorithm proceeds and evaluates the next interval. If however there has been a change in the demand, it first determines whether or not there has been cancellations or appearance of new requests. In the case where there are only cancellations, the algorithm re-computes the routes excluding these customers. In case new requests show up, the algorithm re-computes the routes and finds a new objective function. If the value of the objective function is better or equal to the one with the static requests, these customers are being accepted in the solution and accommodated. In all other cases customers are rejected. In case where both cancellations and new requests appear, the algorithm re-evaluates the static solution with the cancelled requests. It then determines a new objective function that includes both cancellations and new requests. The process is then repeated for all planning intervals. An illustrative flowchart of this process is depicted in fig. 8. An important note while solving the dynamic problem, when new requests have appeared they are served according to the *First-Come-First-Serve* (FCFS) principle, meaning that if they have been accepted to the solution in one time interval, they have to be part of the solution in the next interval even if that will lead to rejecting a more profitable customer.

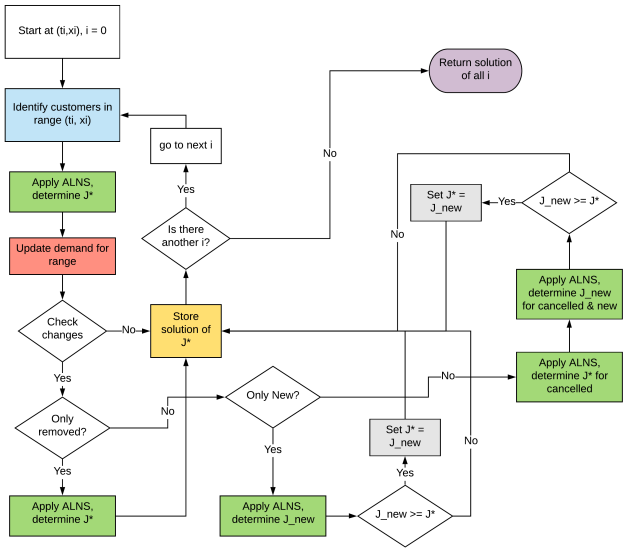

Figure 8: Optimization flowchart for dynamic requests

In the case where a new customer has been found profitable and inserted into the route the planned initial route is then modified. An example of this case can be observed in figs. 9 and 10.

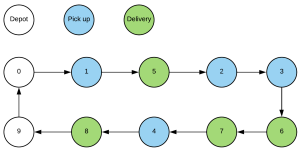

Figure 9: Initial planned route

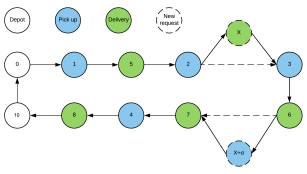

Figure 10: Modified route after dynamic customer insertion

5.2 Initial solution

The initial solution (s_0) is constructed by randomly assigning requests to routes. To avoid inserting all requests in one route, the number of customers in the route is identified. Then, the number of vehicles to use is determined by simply dividing the total number of unassigned requests by the number of customers per route. A request i, is then randomly assigned to a vehicle by inserting the associated pick up and delivery node sequentially at the end of the partially constructed route. This procedure is repeated until the maximum number of customers per route is reached and then another vehicle is deployed. Its important to note that the initial solution does not take into account ride-sharing operations as it inserts pick-up nodes deliberately after delivery nodes. This procedure ensures that constraints (i)[3] and (iv)[4] are satisfied while all other constrains may be violated. The pseudo-code for the construction of the initial solution can be found in line 2. The reason for constructing a partially infeasible initial solution is because it can lead to better results as seen in the work of [Cordeau and Laporte, 2003, Ropke and Pisinger, 2006]

5.3 Removal operators

In this section the removal operators (heuristics) are described. Three different removal heuristics are used in this research. All of them take as input a given solution and an integer parameter q which describes the number of requests to be removed. For the *Shaw* and *Worst* removal, an added parameter p is added which is used to determine the degree of randomization in the chosen heuristic. Technically, removal heuristics can turn a feasible solution into an infeasible solution. The inferior heuristics are taken and adopted for this problem from [Ropke and Pisinger, 2006].

5.3.1 Worst removal

The worst removal heuristic, removes q requests for the current solution (s) whose removal corresponds to the **lowest** decrease in the objective function J^*. This is an iterative process until all q requests are removed. The general idea behind this heuristic is to remove requests that have the lowest contribution to the profits and later re-assign them to a different position in the same or different route. For every request i that is served by a given vehicle k in a solution s, a profit function is defined such that $\text{profit}(i, s) = J^*(s) - J^*_{-i}(s)$ where $J^*_{-i}(s)$ is the profit of the solution by completely removing request i. The randomization parameter p is used to avoid circumstances where the same requests are removed over and over.

5.3.2 Shaw removal

The Shaw removal, or else similarity removal, was firstly introduced by Shaw in [Ropke and Pisinger, 2006]. The idea behind this heuristic is to remove requests that are similar to each other. This makes it easier to shuffle and explore new solution spaces while perhaps creating new better solutions. By removing non-similar requests, it is harder to re-insert them in the current solution as these requests will probably be inserted at either their original or a bad position. Since every problem is different, the algorithm is modified and adapted to fit the needs of the UAMP scheduling problem. The similarity of two requests (i, j) is constructed using the *relatedness measure* $R(i, j)$. Low values of $R(i, j)$ depict higher similarity among requests. For this problem, the relatedness measure consists of the following terms: distance, start service time and passenger classification. Every term is weighted using the following weights α, β, γ. The relatedness between two requests is given using eq. (8).

$$R(i,j) = \alpha \left(d_{i,j} + d_{i+\sigma,j+\sigma}\right) + \beta \left(|\tau_i - \tau_j| + |\tau_{i+\sigma} + \tau_{j+\sigma}|\right) + \gamma |T_i - T_j| \tag{8}$$

It is assumed that all terms in the aforementioned equation are normalized such that they can take values from the interval $[0, 1]$. At first, the heuristic chooses a random request to be removed. At the next stage, the relatedness of this request with the remaining requests is computed using eq. (8). The resulted array L is then sorted in descending order. A random number y is generated that takes values from [0,1] and p creates some randomness in the selection approach. The selected request is then removed from the resulted array. It is an iterative process until all q requests have been removed. The higher the value of p, the most similar requests will be removed, while low values of p introduce more randomness in the process. For a better understanding of the Shaw removal algorithm, readers are referred to [Ropke and Pisinger, 2006].

[3] Every route starts and ends at the depot

[4] For every request i arcs $a_i, a_{i+\sigma}$ belong to the same route and node $N_{i+\sigma}$ is visited after node N_i

5.3.3 Random removal

The random removal heuristic is a variation of the Shaw removal where $p = 1$. It randomly removes q requests from the solution s. According to [Ropke and Pisinger, 2006], implementation of the random removal is done in order to run faster than the Shaw removal heuristic.

5.4 Insertion operators

Once the requests have been removed from the solution s, they need to be re-inserted into the routes. This is done using insertion heuristics inspired by [Ropke and Pisinger, 2006]. Generally, insertion heuristics can be categorized as sequential or parallel. Since the routes are already partially filled, parallel heuristics are used for this problem. For a better understanding concerning the differences between sequential and parallel heuristics readers are referred to [Potvin and Rousseau, 1993].

5.4.1 Basic greedy heuristic

The basic greedy heuristic is the simplest construction heuristic. The idea behind it is to place every request at its best position. It performs i iterations until all q removed requests have been inserted into the routes. This is done by checking the value of objective function at every k position of all possible combinations. Since this is a profit maximization problem, we are interested in placing request i where $p_i = \max_k \in K \left\{\Delta J^*_{i,k}\right\}$. If a request cannot be inserted to a route then $p_i = -\infty$. The inserted request is the one that maximizes eq. (9).

$$\max_{i \in U} p_i \tag{9}$$

A disadvantage of using this approach is the fact that it focuses primarily on requests that correspond to the highest profit, therefore it makes it difficult to tackle requests that are harder to re-position since less combinations exists.

5.4.2 Regret heuristics

K-Regret heuristics improve the performance of the basic greedy heuristic as they incorporate some sort of look ahead information when deciding what request to insert to the solution. For this problem, a 2-regret heuristic is incorporated, meaning that we are interested in inserting request i, where the difference between the first and second best positions is the highest. This is done by incorporating a variable $x_{ik} \in \{1, ..., m\}$ for which request i has the k^{th} highest insertion profit such that $\Delta J^*_{i,x_{ik}} \geq J_{i,x_{ik'}}$. The regret value $c^* = J^*_{i,x_{i1}} - J^*_{i,x_{i2}}$. In every iteration, the request i that satisfies eq. (10) is chosen.

$$\max_{i \in U} c^*_i = \max_{i \in U} \left\{ \sum_{j=1}^{k} \left(J^*_{i,x_{i1}} - J^*_{i,x_{ij}} \right) \right\} \tag{10}$$

5.4.3 Tabu greedy heuristic

The tabu greedy heuristic has the same working principle with the greedy heuristic with, however, one important difference. A 'tabu-list' keeps track of all the insertion moves that have occurred in the previous iterations. In order to explore a larger part of the solution space, the tabu-greedy algorithm inserts moves that haven't been inserted in the past even if they are not the most profitable ones.

5.5 Weight adjustment for adaptiveness

In LNS metaheuristics, one could use one removal and insertion operator, however in this paper we use all of them. According to Shaw [Ropke and Pisinger, 2006], this alternation among heuristics will create a more robust solution. It is therefore important to identify which removal and insertion heuristics will be used at every iteration. This is done by using the roulette wheel selection principle. An important side note, is the fact that insertion and removal heuristics are chosen independently of each other. Having k heuristics that have a weight w_i, such that $i \in \{1, 2, ...k\}$ heuristic j is chosen with probability $\frac{w_j}{\sum_{i=1}^{k} w_i}$.

These weights are automatically adjusted at segments (fixed number of iterations) using statistics from previous iterations. This is done by keeping track of the performance of the heuristics at every iteration. The higher the score, the more successful the heuristic has been. At the beginning of every segment the score of the heuristics is reset to zero. Scores of heuristics grow in the following way [Ropke and Pisinger, 2006]:

- σ_1: Last remove and insertion heuristic resulted in a new global best solution,

- σ_2: Last remove and insertion heuristic resulted in both new visited solution and $J(s') > J(s)$, where $J(s')$ and $J(s)$ correspond to current and previous accepted solution respectively,
- σ_3: Last remove and insertion heuristic resulted in new visited solution but $J(s') < J(s)$ where the solution has been accepted.

The reason for rewarding heuristics that do not improve the solution, is because they diversify the search thus looking at a larger solution space. This is simply done by keeping track of which solutions have been visited in the past by assigning a key to each solution and storing them in a table.

Since in every iteration 1 removal and 1 insertion heuristic are used, it is not known which one of the two resulted in a better/ new solution and therefore both of them are rewarded. At the end of every segment, the weights are adjusted and calculated based on the scores of each heuristic. If w_{ij} is the weight of the heuristic i in segment j, the weights for the following segment are calculated using eq. (11). At the first segment, all heuristics have equal weights.

$$w_{i,j+1} = w_{ij}\,(1-r) + r\frac{\pi_i}{\theta_i} \tag{11}$$

In the above mentioned equation, r is the reaction factor which control how fast the adjustments of the weights will occur based on the effectiveness of the heuristics. If the value of r is set to zero, the weights are not adjusted. π_i corresponds to the score of the heuristic while θ_i counts how many times this heuristic has been used in each segment. An illustration example of a simple instance for 5000 iterations and 20 segments is shown in fig. 11.

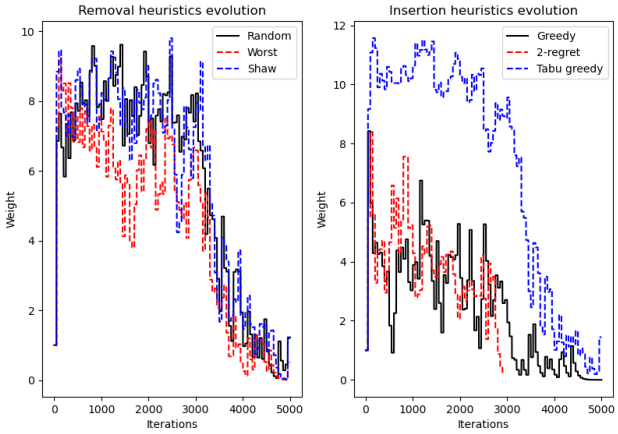

Figure 11: Removal and insertion heuristic weight adaptiveness

5.6 Accept/ Reject criteria

As explained in section 5.1, there are different ways to escape the local minima. This is done by accepting solutions that are worse in order to explore a bigger solution space. The chosen acceptance/ reject criteria for this problem is taken from simulated annealing. Simulated annealing was chosen because its preferred over other methods when trying to determine an approximate global optimum than a precise local optimum in a limited computational time. The algorithm works by accepting solution s' given the current solution s using eq. (12).

$$p_{\text{accept}} = e^{\frac{f(s')-f(s)}{T}} \tag{12}$$

In the above mentioned equation $T > 0$ and corresponds to the temperature. It starts with an initial value of T_{start} and at every iteration it is decreased by $T = Tc$, where c is the cooling rate and can take values such that $0 < c < 1$. Determining both the temperature and the cooling rate, is instance specific. However, according to [Ropke and Pisinger, 2006] we want to accept a w percent worse solution than the current solution with probability 0.5. This can be done by inspecting the initial solutions. The formula then to calculate the initial temperature at a given instance is derived using eq. (13). The effect of SA can be seen in fig. 12, where worse solutions have been accepted to find an overall better solution.

$$T = \frac{(1+w)J^*_{\text{initial}} - J^*_{\text{initial}}}{\ln 0.5} \tag{13}$$

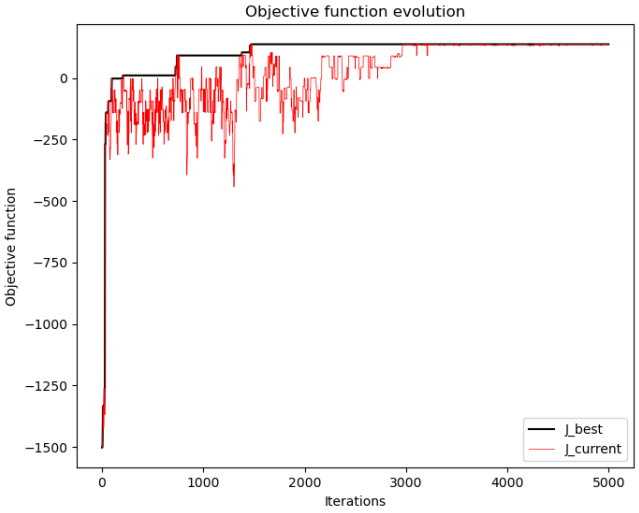

Figure 12: Objective function evolution

6 Experiments and analysis

This section presents the findings of this research. In section 6.1, the reader can find a thorough description of the instances and how they were generated. The purpose of section 6.2 is to describe which parameters of the ALNS tend to give better solutions and which heuristics perform best for this problem. The instances are first analyzed without any dynamic demand updates. The reason for that is to determine the robustness of the algorithm and determine which customers contribute the most to the profit. These results are presented in sections 6.3 to 6.5 respectively. The dynamic problem is described in section 6.6 and identifies the effect of rejecting, accepting customers into the solution and what is the impact towards the profit when some customers cancel. The model was developed in Python 3.8 using an iOS machine with 8 GB RAM and a quad-core intel Core i7

6.1 Description of instances

In section 2 we classified the nature of the UAMP and found common grounds with other transportation models such as the DARP and the SARP. However, the combinations of grouped pick-up with delivery nodes, together with the unique sets of constraints including the FIFO make it difficult to work with existing or adapted instances. Therefore we generate our own instances using the information presented in section 4. The generated instances simulate different parts of the day and have different purposes. For the remaining of this section we will refer to these as instance 1, 2 and 3.

Instance 1 defines morning transportation and customers commuting mainly from the suburb areas towards downtown, where most offices and schools are located in large metroplex areas [Vascik and Hansman, 2017]. The starting time of the day t_0 was set at 06:30 am while the ending time was at 12:00 noon. The δ_t, which defines the time-step at which we check whether a request is generated is set at $\delta_t = 0.03h$. The weights to generate customer requests for vertiports 0 to 6 were $[0.05, 0.1, 0.08, 0.11, 0.02, 0.12, 0.08]$. The weights to assign destination were $[4, 1, 1, 1, 1, 1, 1]$ giving a higher priority towards the downtown vertiport. From table 16, approximately 30% of the customers are considered premium. In total 70 requests were generated.

Instance 2 defines evening transportation and customers commuting mainly from downtown to the suburbs, where most residential areas are located. The starting time of the day t_0 was set at 15:30 am while the ending time was at 21:00. Similarly with instance 1, δ_t was also set at 0.03h, in order to be able to compare the two instances. The weights to generate customer requests for vertiports 0-6 were $[0.33, 0.09, 0.08, 0.08, 0.08, 0.08, 0.08]$ respectively. The weights to assign destination were $[3, 5, 5, 5, 5, 5, 5]$ which gives the same probability for all suburb vertiports. 30% of the customers are also considered premium. In total 70 requests were generated.

Instance 3 defines evening transportation but towards a major event in one of the suburb areas. The starting and ending times of the day were the same as instance 2 (t_0, t_{end}=15:30,21:00). The weights to generate customer requests for vertiports 0-6 were $[0.33, 0.09, 0.1, 0.1, 0.1, 0.1, 0.1]$ and to assign destination $[4, 15, 2, 2, 2, 2, 2]$. In difference with instance 2, it can be observed that the weights to generate requests are higher for instance 3. This is because, when an event takes place it is more likely that more people will want to use the service, thus

the number of requests are expected to further increase. δ_t was also set at 0.03h and 30% of the customers are considered premium. In total 79 requests were generated.

In order to determine how customers contribute to the profit, each of the aforementioned instances will be modified in the same way. Suppose $X\_original$, corresponds to the original instance where $X = 1, 2, 3$. Each instance will be modified against (1) customer satisfaction factors, (2) Time-window length and (3) customer classification - fully premium or standard. X_0505_factor, corresponds to the case when the customer satisfaction factors are set for every customer at $\alpha_i, \beta_i = (0.5, 0.5)$. $X\_0505\_05\_tw$, $X\_0505\_02\_tw$ corresponds to the same satisfaction factors, but also the time-window length of every customer is respectively $+0.5h, +0.2h$. $X\_0505\_05\_tw\_std$, $X\_0505\_05\_tw\_pre$ has the same criteria as the above mentioned modifications but in the first case all passengers are considered standard whereas in the second case all passengers are considered premium. Therefore for $\alpha_i, \beta_i = (0.5, 0.5)$ the following instances exist: X_0505_factor, $X\_0505\_05\_tw$, $X\_0505\_02\_tw$, $X\_0505\_05\_tw\_std$, $X\_0505\_05\_tw\_pre$, $X\_0505\_02\_tw\_std$, $X\_0505\_02\_tw\_pre$. The same procedure is repeated for $\alpha_i, \beta_i = (1, 0)$ and $\alpha_i, \beta_i = (0, 1)$.

6.2 Parameter tuning and heuristic performance

In order to tune the parameters that affect the performance of ALNS, the first step is to determine the ones which have an impact on it. To do so, we look at the different modules of the ALNS and assess them. Concerning the removal operators, the value of ξ determines the number of requests to be removed per iteration. Parameters α, β, γ, p are found in the Shaw removal (described in section 5.3.2) and parameter p_{worst} is found in the worst removal (section 5.3.1). Random removal and insertion operators do not have any parameters that affect the performance of the ALNS. Weight adjustment is controlled by parameters $\sigma_1, \sigma_2, \sigma_3, r$ and finally accepting or rejecting a solution parameters c and w are used.

Since this is an ad-hoc iterative process, initially the parameters were set as the ones seen in [Ropke and Pisinger, 2006], where $\alpha, \beta, \gamma, p, p_{worst}, w, c, \sigma_1, \sigma_2, \sigma_3, r, \xi = (9, 3, 2, 6, 3, 0.05, 0.999, 33, 9, 13, 0.1, 0.4)$. The initial parameter that was modified was ξ. We tested the heuristics for ξ ranging from 0.1 to 0.5 with a step size of 0.1. The results are depicted in table 7. Nominally, the average gap of the ALNS solution is compared with the one of an exact approach. The one with the lowest average gap gives the nominal ξ value. In this research, a different approach is followed because an exact approach is not formulated. We are therefore interested in the value of ξ that returns the highest profit. Since the algorithm determines sub-optimal solutions, the indicated profit will always be less than the optimal one. Thus the higher the profit, the small the average gap will be.

Table 7: Parameter ξ against profit for all instances

ξ	0.1	0.2	0.3	0.4	0.5
Profit - Instance 1	973.7	1025.7	1068.9	1109	1064.4
Profit - Instance 2	980.5	1004.3	1056.2	1097	1040.3
Profit - Instance 3	1300.2	1330.2	1450.3	1484.6	1469.3

The rest of the parameters are determined in the following way. We start the process with an initial guess and tune one parameter at a time. Although it is a time-consuming process, the finalized parameters are shown in table 8.

Table 8: ALNS parameters

Parameters	α	β	γ	p	p_{worst}	w	c	σ_1	σ_2	σ_3	r	ξ
Initial	9	3	2	6	3	0.05	0.999	33	9	13	0.1	0.4
Final	9	3	5	6	5	0.35	0.9999	33	15	20	0.5	0.4

Concerning the heuristics' performance, from the removal heuristics random and Shaw removal had the best scores while worst removal did not perform very well. The reason for that, is because worst removal, has the tendency of removing similar solutions over and over again even when including some randomness. When the greedy heuristic insertion is used, the solution tends to go to a previous value thus getting trapped in a local minima. The ones with the best performance were the 2-regret and tabu-greedy heuristic. The reason why the 2-regret performs better than the greedy heuristic is because it can improve its performance since it incorporates some sort of look ahead information when deciding which request to insert to the solution. Tabu-greedy helps to diversify the search as it tends to accept unique solutions which can lead to a better overall solution. These claims are also supported by fig. 11.

6.3 Algorithm robustness and computational times

An important step is to determine the stability of the algorithm, in other words how well it performs under different seeds. To do so, each instance is run 100 times at different seeds for the static case. A histogram of the profit is shown in figs. 13 to 15 for each instance respectively.

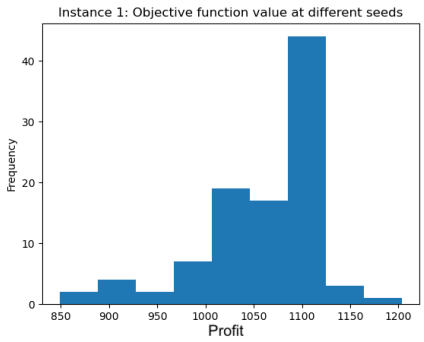

Figure 13: Instance 1

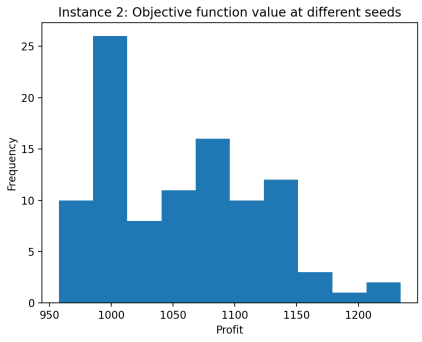

Figure 14: Instance 2

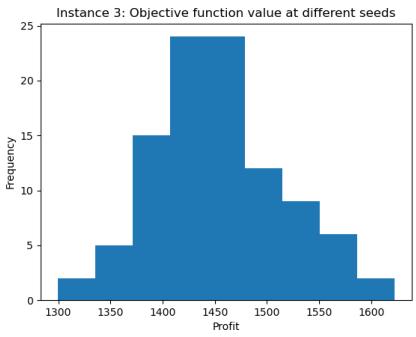

Figure 15: Instance 3

To further analyze the findings of the figures, table 9 is created. μ corresponds to the mean profit value, σ to the standard deviation, $\sigma_{\bar{x}}$ to the standard error, CV to the coefficient of variation, Skew to the skewness of the graph and K to the Kurtosis. A low CV (< 1) depicts low spread of the values with respect to the mean. This indicates that the algorithm is stable under the different instances, since it tends to find similar solutions.

Table 9: Algorithm stability

Instance	μ	σ	$\sigma_{\bar{x}}$	CV	$Skew$	K	min	max	Range
Instance 1	1063.5	66.2	6.6	0.06	-1.04	0.94	850	1203	353
Instance 2	1057.1	63.8	6.4	0.06	0.41	-0.51	962	1304	341
Instance 3	1453.3	62.9	6.3	0.04	0.3	0.21	1301	1650	349

In table 10, the profit is shown for the different instances per rolling horizon. Additionally, we can observe how many requests the algorithm is solving simultaneously and what are the computational times. The problem is solved every $\delta_{plan} = 0.5h$ time steps. When looking at the cumulative profit at every rolling horizon, the results are clear. In some cases customers have a negative contribution to the profit, but since they are 'static' customers they must be accommodated. As such, in the dynamic case customer who have a negative contribution to the profit should be rejected. Finally, it is observed that even if in some cases the number of requests that the algorithm is trying to solve are the same, the computational times vary up to 1 minute. This happens due to the type of customers that the algorithm is trying to accommodate. The more premium customers are present, the lower the computational times will be, due to the reduced number of combinations in the construction heuristics.

Table 10: Instances objective function and computational time

Instance	RH	1	2	3	4	5	6	7	8	9	10	11	12	Total
1	Requests	8	13	12	7	3	7	3	8	4	4	1	0	70
	Time [s]	54.8	130.4	120.7	45.1	17.9	45.4	19.1	55.6	26.2	29.2	5.2	0	495.3
	Profit	-20.6	297.5	647.4	769.3	827.9	1008.9	1001.7	1106.1	1025.7	1142.5	1109	1109	1109
2	Requests	14	15	13	6	2	7	3	7	2	0	1	0	70
	Time [s]	143.6	183.7	145.9	40.4	10.6	50.4	20.4	50.4	13.9	0	5	0	664.5
	Profit	147.5	491.9	797.8	874.3	899.3	952.6	938.2	1045.1	1078.2	1078.2	1097.9	1097.9	1097.9
3	Requests	14	16	17	8	3	7	3	7	2	0	1	0	78
	Time [s]	193.6	265.45	223.27	69.1	20.4	48.12	19.97	48.47	11.02	0	5.44	0	904.9
	Profit	366.8	703.93	1184.41	1194.9	1240.4	1378.93	1414.1	1498.3	1518.9	1518.9	1484.6	1484.6	1484.6

In terms of computational times the following observations were made:

1. **Removal Heuristics:** Worst and Shaw removal are the slowest removal heuristics. This is because Worst removal has to remove all requests, recompute all routes without these requests and then re-evaluate the objective function to then remove the 'worst' customers. However, the algorithm tends to get faster with

time due to memoization that was implemented. Average times for worst removal vary for the number of requests but are approximately $0.22 - 0.71$ seconds. Shaw removal is the slowest removal heuristic since every time its chosen it needs to re-evaluate the relatedness matrix for all requests. Average times are $0.44 - 1.09$ seconds per iterations depending on the total number of requests.

2. **Insertion Heuristics:** All insertion heuristics have similar performance with 2-regret being a slight slower than greedy and tabu-greedy. All insertion heuristics tend to become faster as the iterations progress due to memoization, which means that they do not have to recompute all the route information if they have been found in the past. Average times can be between 0.34 - 6.2 seconds per iteration.

The algorithm needs to be able to provide a sub-optimal solution much faster than δ_{plan}, in order to give the user an immediate response on whether or not they will be accommodated. As $\delta_{plan} = 0.5h$, the computational time of the algorithm for all instances per rolling horizon varies between 5.2 to 265.5 seconds depending on the number of requests.

6.4 Sensitivity analysis

In order to draw conclusions about how passengers contribute to the profit, each instance was modified. This is done by performing a sensitivity analysis about the time window length, customer satisfaction factors, and customer classifications. The KPI's of particular interest in this analysis are: profit, customer satisfaction, percentage of passengers sharing rides and the total number of vehicles used. One should expect variations in the aforementioned values at every sub-instance. The plots figs. 13 to 15, seem to indicate a stable behavior of the algorithm for all of the instances. As such, due to the computational limitations it was decided to run each sub-instance once at the same seed for 1000 iterations. The findings are presented in table 11.

Table 11: Variation of instances

Instances	Profit			$\bar{s_i}$			Shared rides [%]			Total vehicles		
Instance	x=1	x=2	x=3	x=1	x=2	x=3	x=1	x=2	x=3	x=1	x=2	x=3
$X\_original$	1109.0	1097.9	1484.6	0.96	0.97	0.95	40.0	31.4	50	12	20	20
X_0505_factor	1111.1	1014.9	1490.8	0.96	0.97	0.94	37.2	40	50	14	20	20
$X\_0505\_05\_tw$	1270.9	1449.9	1538.7	0.96	0.97	0.95	44.3	44.3	50	12	20	20
$X\_0505\_05\_tw\_std$	1329.4	1424.5	1975.4	0.94	0.96	0.93	68.6	71.4	74.4	10	20	19
$X\_0505\_05\_tw\_pre$	1704.3	1693.7	1286.9	0.99	0.99	0.99	0	0	0	14	20	20
$X\_0505\_02\_tw$	831.2	787.1	1309.8	0.97	0.96	0.96	28.6	32.9	43	14	20	20
$X\_0505\_02\_tw\_std$	734.1	1022.4	1377.4	0.95	0.96	0.94	44.3	54.3	60.3	14	20	20
$X\_0505\_02\_tw\_pre$	1754.2	1509.2	1202.4	0.99	0.99	0.99	0	0	0	17	20	20
X_10_factor	1130.2	1066.9	1493.2	0.98	0.99	0.99	41.4	50	47.4	13	20	20
$X\_10\_05\_tw$	1256.9	1428.8	1655.2	0.97	0.99	0.98	51.4	50	52	12	20	20
$X\_10\_05\_tw\_std$	1531.1	1634.8	2223.9	0.98	0.99	0.99	75.7	77.1	70.5	11	20	20
$X\_10\_05\_tw\_pre$	1658.4	1693.7	1286.9	0.99	0.99	0.99	0	0	0	14	20	20
$X\_10\_02\_tw$	916.5	839.6	1406.8	0.99	0.98	0.99	32.9	34.3	48.7	15	20	20
$X\_10\_02\_tw\_std$	893.8	1257.6	1795.8	0.99	0.99	0.99	47.2	55.7	66.6	14	20	20
$X\_10\_02\_tw\_pre$	1800.1	1509.2	1250.1	0.99	0.99	0.99	0	0	0	17	20	20
X_01_factor	993.4	967.9	1261.2	0.94	0.97	0.93	31.4	28.6	39.8	13	20	20
$X\_01\_05\_tw$	1202.1	1174. 2	1500.8	0.94	0.96	0.94	38.6	31.4	42.3	12	20	20
$X\_01\_05\_tw\_std$	1240.2	1321.6	1717.6	0.92	0.93	0.91	57.2	50	57.7	11	20	20
$X\_01\_05\_tw\_pre$	1754.2	1693.7	1299.6	0.99	0.99	0.99	0	0	0	14	20	20
$X\_01\_02\_tw$	762.8	742.2	1009.8	0.96	0.94	0.95	20	25	33.3	15	20	20
$X\_01\_02\_tw\_std$	651.6	873.5	1414.8	0.93	0.93	0.94	32.9	41.4	57.7	15	20	20
$X\_01\_02\_tw\_pre$	1800.1	1509.2	1250.1	0.99	0.99	0.99	0	0	0	16	20	20

From the aforementioned table the following can be concluded:

- **Customer satisfaction factors**: The original instances have averaged $\alpha_i, \beta_i = (0.49, 0.51)$ respectively. This is the reason why, when only $\alpha_i, \beta_i = (0.5, 0.5)$, small changes are observed: (a) in the profit, (b) customers sharing rides and (c) overall customer satisfaction. However, when α_i, β_i are at the two extremes (1,0) or (0,1) respectively there is an immediate effect on the KPIs. In the first case, $\alpha_i, \beta_i = (1, 0)$ satisfaction is near an optimal value (0.99). Because of that, the profit generated by customers also increases and allows for more ride-sharing operations, thus reducing the number of vehicles. When $\alpha_i, \beta_i = (0, 1)$ satisfaction reaches the lowest, resulting in profit decrease. This outcome is expected and is logical because a higher priority is given to the deviation of the desired departure or arrival time against

the actual time. This allows for a higher percentage of ride-sharing operations since the 'trip duration' factor has no impact on the objective function. However, this mainly impacts 'standard customers' who share rides. Therefore, in order to maximize profit, standard passengers should have as high α_i as possible and as low β_i as possible.

- **Time-window length**: The time-window length, is one of the parameters that affects profit, satisfaction and shared-rides the most. Initially, all instances have an average time-window length varying between 0.38-0.44 hours. When increasing the time-window to 0.5 hours, an increase in the profit is observed. Satisfaction rates are in most cases stable and near optimal but can however decrease by 0.01-0.02 units. Since a discount is given to customers whose satisfaction is less than 95%, the impact on the profit can be considered as negligible. Customers who have an overall higher time window are more 'flexible', meaning that they can share rides easier, which in turn has a positive effect on the profit. This is because the more customers who share the same vehicle the lower the cost will be. When the time-window interval is set for all customers at 0.2h, profit is drastically reduced together with the number of people sharing rides. An interesting finding is that the behavior of the system is not the same for all instances. For example, in instance 1 when the time-window is set at 0.5 hours and all customers are standard, the profit increases in both cases. However, when the time-window length is set at 0.2 hours and all customers are standard the profit decreases. This is not the case for instances 2 and 3 where the profit increases in all cases. Such a finding requires further analysis but it is believed that it is due to the distribution of the requests among vertiports.

- **Passenger classification**: Finally, passenger classification depends on their status (only standard or only premium). The outcome is straight forward: when only standard passengers are present, more customers are sharing rides irrespective of the time-window length. This will have a positive contribution to the profit. When only premium customers are present there is a lower vehicle utilization. One would expect that this can have a negative profit contribution, but since these customers are charged at a much higher rate, their impact is positive. In instance 1 and 2, when only premium customers are present, the profit is the highest.

All in all, the following passenger combinations will lead to higher profits.

- **Standard customers**: For standard customers, the ideal scenario is that they have as high α_i as possible and as low β_i as possible. Additionally, a time-window length of at least 0.4 hours is ideal since it corresponds to higher ride sharing, without impacting as much customer satisfaction.

- **Premium customers**: Since premium customer do not share any rides, their satisfaction criteria barely affect the overall profit and satisfaction. This is because they always travel from their origin to their destination. A higher time-window length indicates that less vehicles will be used, which is desired.

In table 11, some anomalies were noticed that need further analysis and clarification. These concern, how the profit decreases for $1\_factor\_02\_tw\_std$ but increase for $2\_factor\_02\_tw\_std$ and $3\_factor\_02\_tw\_std$. At first it seems that there is a turning point for the time-window length, which has a positive or a negative contribution to the overall profit. To test that, we modify instance 1 even further and try to determine that turning point. The same is performed for the other instances as well. The findings are presented in table 12.

Table 12: Time window length turning point

Instance	TW length
1	0.27
2	0.05
3	0.09

From the table it can be observed that instance 1 has a much higher turning point than 2 and 3. Particularly instance 2 can be seen as there is no turning point for the profit due to the length of the TW interval set at 3 minutes. To get a better understanding of why this occurs, it is essential to look at the demand distribution over the vertiports.

Instance 1 simulates morning transportation which means that most requests are originated at the suburb vertiports and most requests have as destination the downtown area. Since the central depot is located in the downtown area, it means that the vehicle has to arrive at these locations as early as possible. The cost associated with transporting the vehicle from the downtown area towards the suburb areas does not affect

passenger pricing. This means that less vehicles will be deployed as they have less travel cost distance. This directly impacts profits on ride sharing. The lower the TW length interval, the more vehicles will be deployed towards these areas and thus the cost of rebalancing empty vehicles increases. However, when the TW length is above 0.27 hours, the more people share rides and thus less empty vehicles are rebalanced.

Instance 2 simulates evening transportation which means that most requests are originated at the downtown vertiport and have as a destination the vertiports in the suburb areas. As the central depot is located in the downtown area, more vehicles are deployed and higher profit is generated because there are fewer vehicle rebalances compared to the first instance. As a result, there is not a 'precise' TW length turning point as a new vehicle can be deployed while avoiding to rebalance an empty vehicle which will lead to higher cost.

Instance 3 has a mixed behavior between instance 1 and 2. This is because most requests are generated at the downtown vertiport, due to the time of the day while the desired destination is a suburb vertiport, where an event occurs. Since not all requests are generated at the downtown vertiport there is a higher TW length turning point than instance 2 but much lower than instance 1. Similarly, the cost of rebalancing empty vehicles is reduced due to the number of requests being present at the downtown vertiport. The distribution of requests of all instances can be observed in figs. 16 to 18 respectively.

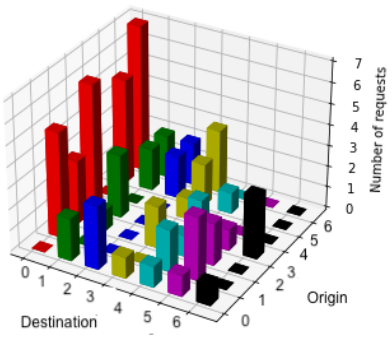

Figure 16: Instance 1

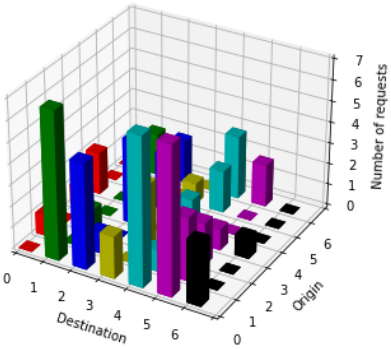

Figure 17: Instance 2

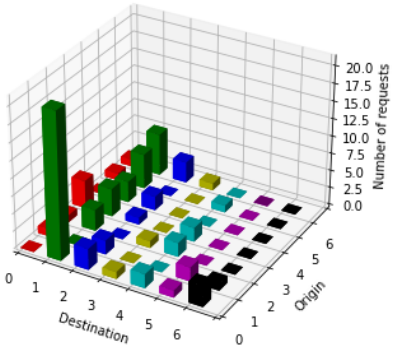

Figure 18: Instance 3

In a real-life situation, customers are unlikely to have large time-windows especially during the morning hours. This is because there is a rush to get to the office and schools. As a result, it can be expected that during morning hours of transportation, the time-window length will be as low as possible, minimizing the number of people who can share rides and therefore the profit. Thus a constraining parameter to the model from a customer perspective will be the time-window length. However, this can be improved if the company increases the number of depots in the suburb areas, or by rebalancing them early in the morning towards the suburbs. By doing so, even if the time-window lengths are reduced, there is an increased probability of people sharing rides and reducing vehicle costs.

6.5 Ride-sharing analysis

This section provides the findings of the ride sharing analysis. To do this, we use the original instances $X\_original$ for $X = [1, 2, 3]$ and its time-window variations for $\alpha_i, \beta_i = (0.5, 0.5)$. The findings are presented in table 13. The table shows for every instance the total percentage of ride-sharing, then an analysis is performed to evaluate where the ride-sharing comes from. Requests who ride-share can have common origin/ destination, both origin and destination, are considered transit or other. Transit means, that a vehicle picks up multiple requests who have the same origin, delivers the customers at their destination but then has at least 1 or more requests to deliver. Lastly, "other" refers to requests who do not have a common origin or destination but are requests that are in the same vertiport proximity. This case works particularly well for delivery-oriented customers.

Table 13: Ride-sharign analysis per instance

Instance	Ride sharing [%]	Origin [%]	Destination [%]	Both [%]	Transit [%]	Other [%]
1_*original*	40	46.4	7.2	14.3	17.8	14.3
1_0505_05_*tw*	44.3	48.2	14.8	13.5	11.3	12.2
1_0505_05_*tw_std*	68.6	30.4	4.5	26.1	19.5	19.5
1_0505_02_*tw*	28.6	50	10	10	20	10
1_0505_02_*tw_std*	44.3	37.2	16.2	20.5	9.2	16.9
2_*original*	31.4	0	36.3	45.5	13.6	4.5
2_0505_05_*tw*	44.3	12.9	22.5	29.3	19.1	16.1
2_0505_05_*tw_std*	71.4	12	22	32	20	14
2_0505_02_*tw*	32.9	0	43.4	34.7	13.1	8.7
2_0505_02_*tw_std*	54.3	8.1	29.7	37.8	10.8	13.5
3_*original*	50	20.5	10.2	41.1	10.3	17.9
3_0505_05_*tw*	50	10.5	21.1	47.3	10.5	10.5
3_0505_05_*tw_std*	74.4	13.5	20.3	44.1	16.9	5.1
3_0505_02_*tw*	43	11.7	32.3	38.3	14.7	2.9
3_0505_02_*tw_std*	60.3	12.7	14.8	51.1	12.7	8.5

Instance 1, has the highest contribution of ride-sharing when requests are originated at the same vertiport. The remaining 'ride-sharing options' are evenly distributed. When the TW length is increased a higher number of people are sharing rides but also the number of people sharing rides with common destination also doubles. When all customers are standard, the number of people sharing rides with common origin and destination doubles, while the percentage of people sharing rides with common origin decreases. When the TW length is at 0.2h, the highest contribution still remains the people sharing rides with common origin.

On the contrary, instance 2 and 3 have similar patterns. Most of the requests sharing rides are customers who have common origin and destination, or just common destination. Noting from table 11, that the profit for the first instance decreases when the TW length is at 0.2h while for instances 2 and 3 increases, this can also be connected with how the algorithm interprets and uses ride-sharing operations. Highest profits are observed when customers have a common origin and destination since the profit will be doubled compared to the cost. However, that's not always the case, and this is simply because in the first instance sharing rides with people of common origin gives a higher diversification to the vehicles, thus are spread more evenly across the vertiports. In instance 2 and 3, that's not really necessary since more requests are generated at the downtown vertiport where the depot is also located. For illustration purposes, the average percentage for instance 1,2,3 and their contribution is depicted in figs. 19 to 21 respectively.

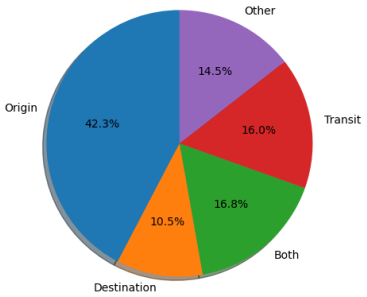

Figure 19: Instance 1

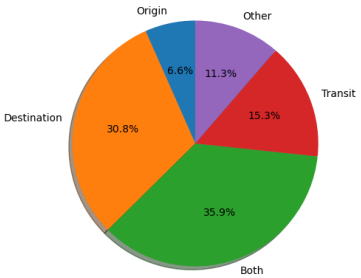

Figure 20: Instance 2

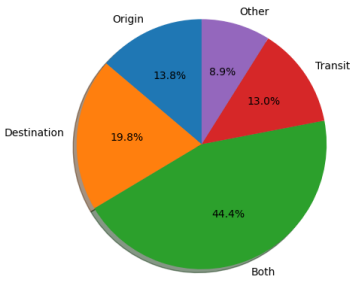

Figure 21: Instance 3

6.6 Dynamic customers

The last stage of this research is to see how the algorithm behaves under dynamic demand. For every instance demand is updated randomly causing customer cancellations and new customers appearing, as described in section 4.3.2. The finding are shown in table 14.

Table 14: Instances under dynamic demand

Instance	Requests	Profit Static	Cancellations	New customers	Accepted customers	Profit dynamic
Instance 1	70	1109	3	15	93.8	1758.3
Instance 2	70	1097.9	2	20	90	1708.9
Instance 3	78	1484.6	0	5	100	1691.8

As it can be seen from the aforementioned table, the algorithm does not always accept all the new requests. To determine the reason behind that, it is necessary to look at the routes up to that point and where the new customers originate from, as well as what is their destination. After doing so, it was concluded that the algorithm tends to mainly accommodate new standard customers who are willing to share a ride. Premium passengers are fully accommodated in most cases if they originate at vertiport 0, which also happens to be the depot. It was also found that sometimes, when customers cancel, the effect on the objective function is positive. This is due to the fact, that some "static" customers can be located in the suburb areas and there are no vehicles nearby that can accommodate them. This means, that an empty vehicle must rebalance from the downtown area just to accommodate these customers. However, as seen in similar operational concepts such as the one of Uber [Holden and Goel, 2016], when customers cancel they have to pay a "cancellation fee". For this problem, a cancellation fee is applied in the following way: The customer must pay 10% of their nominal price if they decide to cancel the ride within the current time interval.

Increasing the dynamic passengers number, introduces higher uncertainty in the number of cancellations and new requests. The goal of increasing the number of dynamic passengers is to evaluate the effect on the profit and the acceptance rate of new requests. The results are shown in table 15.

Table 15: Increased demand uncertainty

Instance	Requests	Profit Static	Cancellations	New customers	Accepted customers	Profit dynamic
Instance 1	70	1109	8	17	70.6	1517.9
Instance 2	70	1197.9	7	24	87.5	1634.3
Instance 3	78	1484.6	5	25	100	2324.2

For instance 1, it is observed that there is a 166% increase in the number of cancellations and a 13% increase in the number of new requests. In most cancellations the profit decreases, however when customer 63 cancels, the profit increases. This is because, in the original static route an empty vehicle had to rebalance and pick up that customer who was also standard. The computational times per rolling horizon were doubled of the ones with only static demand. This means that the algorithm needs at most 10 minutes to generate the new routes. In real-life operations this creates a serious limitation to the problem. This is because when the demand is updated at $t_1 = 8$, a customer who requests a ride between $08:00 - 08:10$, will not be able to be accommodated since the algorithm does not provide the solution quick enough. A similar behavior is observed for instances 2 and 3, however higher acceptance rates are observed. This because most of these requests are generated at the depot, and thus there is a higher vehicle availability for these customers without having to rebalance empty vehicles.

Another major limitation of this study concerns the customers who cancel rides. In this case, since we are looking at the short-term horizon of the demand, all customers who cancel have to pay a cancellation fee. However, during real-time operations a customer might cancel a ride way ahead of time. In this case, since the solving horizon is limited, these customers should not be penalized. This is because the system will be able to recover when determining sub-optimal routes in the upcoming rolling horizon. For future research, it is recommended to assume that all vertiports will be considered as depots and observe the impact it will have on the profit as well as the acceptance rate of new customer requests.

7 Conclusion

This research focused on the dynamic problem of aerial ride sharing operations. The objective was to determine a computationally efficient algorithm providing a sub-optimal solution. From the analysis, the algorithm has indicated to be stable while a small deviation from the mean profit exists when run under different seeds. This is due to the low coefficient of variation as presented in table 9.

There are many factors impacting on the profitability, namely: (1) satisfaction factors, (2) type of customers, (3) origin and destination, (4) vehicle status and (5) time-window length. The larger the time-window length, the more it allows for standard customers to share rides and thus reduce the operational costs of a vehicle.

Larger time-window lengths can increase the profit margins by at least 14% depending on the instance. We are therefore interested in standard customers with large time-windows. Concerning the satisfaction factors, the more weight is given towards the deviation of the desired departure/arrival time with the respect to the actual one, the higher the profit will be. This result is expected because we are minimizing the overall flight duration factor, thus allowing more ride-sharing options among customers. Customers who are situated in the depot are more likely to occupy a new vehicle, simply because it is more cost effective than rebalancing an empty vehicle.

Ride-sharing operations among customers is also an instance-specific subject. The basic idea is that customers who have common origin and destination are accommodated at the same time from the same vehicle, however this is dependent on the time-window length. The highest ride-sharing among customers with common origin and destination (44%) was noted in instance 3 where more than the other instances. An interesting observation that was determined from the ride-sharing analysis was the fact that the algorithm can sometimes choose customers to share-rides via transit which corresponds to about 15% for all instances. It was noted though that this mostly occurs for customers who have a lower trip duration factor than the ones who do not.

In terms of computational times, the algorithm does not perform as seen in other literature papers, such as the one from [Ropke and Pisinger, 2006]. Even though some methods were examined that boosted the computational efficiency, the algorithm is still not able to handle more than 50 to 60 customers per rolling horizon window. The reason behind this variation depends can be attributed to the customer type. Premium customers have less combinations than standard customers and therefore are less computationally expensive.

This research is a first step towards a dynamic model for the UAM problem. However, there are several recommendations that following research can consider. Firstly, a different meta-heuristic approach, such as Tabu-search could be considered. By comparing the results of the different meta-heuristic approaches to an exact approach, it will be interesting to observe the result deviation and which meta-heuristic algorithm tends to produce a better solution. Furthermore, the computational efficiency of the algorithm sets a good precedent for further research. The model was developed in python 3.8 using an iOS machine with 8 GB RAM and a quad-core intel Core i7. For future work, in order to consider more requests per rolling horizon, it is suggested to utilize a lower level software language such as C++ which is much faster than Python. If Python is used again, it is suggested to investigate different Python optimizations such as Numpy vectorization, Cython and parallelization.

Concerning the recommendations of the ALNS, it would be interesting to include more k-regret heuristics (3, 4, etc) as shown in [Ropke and Pisinger, 2006] and evaluate the effect they will have on the objective function. Furthermore, the shortest-path removal heuristic could also be used to remove all the requests from the shortest routes. Additionally, as seen in [Li et al., 2016], the algorithm should also incorporate and accept a higher number of unfeasible solutions, since these tend to result in an overall better objective function. The simulated annealing stopping criteria used in this paper performed as expected since worse accepted solutions lead to an overall better objective function. According to [Potvin and Gendreau, 2019], there are other stopping criteria which could perform more effectively, since there is less randomization into the selection criteria. The insertion heuristics can be either approximate solutions such as the ones used in this research or exact solutions which can be more computationally expensive [Potvin and Gendreau, 2019]. However, these exact algorithms can be relaxed, increasing their computational efficiency but reducing their solution quality.

References

[Li2, 2014] (2014). The Share-A-Ride Problem: People and parcels sharing taxis. *European Journal of Operational Research*, 238(1):31–40.

[Mue, 2017] (2017). Enabling airspace integration for high-density on-demand mobility operations. *17th AIAA Aviation Technology, Integration, and Operations Conference, 2017*, (June):1–24.

[Alonso-Mora et al., 2017] Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., and Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences of the United States of America*, 114(3):462–467.

[Attanasio et al., 2004] Attanasio, A., Cordeau, J. F., Ghiani, G., and Laporte, G. (2004). Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387.

[Bacchini and Cestino, 2019] Bacchini, A. and Cestino, E. (2019). Electric VTOL configurations comparison. *Aerospace*, 6(3).

[Barkaoui and Gendreau, 2013] Barkaoui, M. and Gendreau, M. (2013). An adaptive evolutionary approach for real-time vehicle routing and dispatching. *Computers and Operations Research*, 40(7):1766–1776.

[Bongiovanni et al., 2019] Bongiovanni, C., Kaspi, M., Geroliminitis, N., and Jean-Francois, C. (2019). A Learning Large Neighborhood Search for The Dynamic Electric Autonomous Dial-A-Ride Problem. *8th Symposium of the European Association for Research in Transportation*, pages 1–8.

[Bosson and Lauderdale, 2018] Bosson, C. S. and Lauderdale, T. A. (2018). Simulation evaluations of an autonomous urban air mobility network management and separation service. *2018 Aviation Technology, Integration, and Operations Conference*.

[Cordeau and Laporte, 2003] Cordeau, J. F. and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594.

[Cordeau and Laporte, 2007] Cordeau, J. F. and Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153(1):29–46.

[Gendreau et al., 2006] Gendreau, M., Guertin, F., Potvin, J. Y., and Séguin, R. (2006). Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies*, 14(3):157–174.

[Gurley, 2014] Gurley, B. (2014). A deeper look at uber's dynamic pricing model.

[Holden and Goel, 2016] Holden, J. and Goel, N. (2016). Fast-Forwarding to a Future of On-Demand Urban Air Transportation. pages 1–98.

[Jewpanya et al., 2016] Jewpanya, P., Chen, Y.-w., and Yu, V. F. (2016). The pickup and delivery multi-depot vehicle routing problem.

[Kohlman and Patterson, 2018] Kohlman, L. W. and Patterson, M. D. (2018). System-level urban air mobility transportation modeling and determination of energy-related constraints. *2018 Aviation Technology, Integration, and Operations Conference*, pages 1–38.

[Li et al., 2016] Li, B., Krushinsky, D., Van Woensel, T., and Reijers, H. A. (2016). An adaptive large neighborhood search heuristic for the share-a-ride problem. *Computers and Operations Research*, 66:170–180.

[Mahmoudi and Zhou, 2016] Mahmoudi, M. and Zhou, X. (2016). Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state-space-time network representations. *Transportation Research Part B: Methodological*, 89:19–42.

[McGrath, 2017] McGrath, F. (2017). The demographics of uber's us users.

[Mobility, 2018] Mobility, N. U. A. (2018). Urban air mobility (uam) market study.

[Moon, 2017] Moon, M. (2017). Dubai tests a passenger drone for its flying taxi service. Access date: 09.03.2020.

[Norvig, 1991] Norvig, P. (1991). Techniques for automatic memoization with applications to context-free parsing. *Comput. Linguist.*, 17(1):91–98.

[Patterson et al., 2018] Patterson, M. D., Antcliff, K. R., and Kohlman, L. W. (2018). A proposed approach to studying urban air mobility missions including an initial exploration of mission requirements. *Annual Forum Proceedings - AHS International*, 2018-May.

[Potvin and Gendreau, 2019] Potvin, J.-Y. and Gendreau, M. (2019). *Handbook of Metaheuristics Associate Series Editor*.

[Potvin and Rousseau, 1993] Potvin, J. Y. and Rousseau, J. M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340.

[Psaraftis et al., 2016] Psaraftis, H., Wen, M., and Kontovas, C. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31.

[Ritzinger et al., 2016] Ritzinger, U., Puchinger, J., and Hartl, R. F. (2016). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231.

[Ropke and Pisinger, 2006] Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

[Savelsbergh and Sol, 1995] Savelsbergh, M. and Sol, M. (1995). The general pick-up and delivery problem. (1):17 – 29.

[Shihab and Wei, 2019] Shihab, S. A. M. and Wei, P. (2019). By Schedule or On-Demand ? - A Hybrid Operational Concept for Urban Air Mobility Services. pages 1–13.

[Taniguchi and Shimamoto, 2004] Taniguchi, E. and Shimamoto, H. (2004). Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. *Transportation Research Part C: Emerging Technologies*, 12(3-4 SPEC.ISS.):235–250.

[Thipphavong et al., 2018] Thipphavong, D. P., Apaza, R. D., Barmore, B. E., Battiste, V., Belcastro, C. M., Burian, B. K., Dao, Q. V., Feary, M. S., Go, S., Goodrich, K. H., Homola, J. R., Idris, H. R., Kopardekar, P. H., Lachter, J. B., Neogi, N. A., Ng, H. K., Oseguera-Lohr, R. M., Patterson, M. D., and Verma, S. A. (2018). Urban air mobility airspace integration concepts and considerations. *2018 Aviation Technology, Integration, and Operations Conference*.

[Toesland, 2018] Toesland, F. (2018). Five futuristic modes of transport transforming travel.

[U.N D.E.S.A, 2019] U.N D.E.S.A (2019). *World Population Prospects 2019*. Number 141.

[Vascik and Hansman, 2017] Vascik, P. D. and Hansman, John, R. (2017). Constraint identification in on demand mobility for aviation through an exploratory case study of Los Angeles. *17th AIAA Aviation Technology, Integration, and Operations Conference, 2017*, (June 2017):1–25.

[Yook and Heaslip, 2015] Yook, D. and Heaslip, K. (2015). Effective modeling for a distance-based fare structure with a time-expanded network. *Journal of Public Transportation*, 18(1):1–13.

[Zhang and Batterman, 2013] Zhang, K. and Batterman, S. (2013). Air pollution and health risks due to vehicle traffic. *Science of the Total Environment*, 450-451:307–316.

A Uber Demographics

Table 16: Demographics of Uber US users

Age		Income		Urban context	
Group	(%)	Tier	(%)	Area	(%)
<24	37	Bottom 25 %	22	Urban	46
25-34	28	Mid 50 %	44	Suburban	48
35-44	17	Top 25 %	27	Rural	6
45-54	12	Not stated	7		
>55	6				

B Pseudo-codes

Data: initial solution (routes, time stamps, battery levels, vehicle occupation)
Result: Optimized solution (routes, time stamps, battery levels, vehicle occupation)
Initialization ;
$s_b \leftarrow s_0$;
nextIt $\leftarrow$ True ;
while *nextIt is True* **do**
  q = customers to remove ;
  $S' \leftarrow S$;
  Apply removal operator ;
  Apply insertion operator ;
  if *J*(S') >* J_b^* **then**
    $S \leftarrow S'$, $S_b \leftarrow S'$ **else**
      if *J*(S') > J(S)* **then**
        S $\leftarrow$ S'
      end
      else
        $S \leftarrow S'$ with probability p
      end
    end
  end
  if *termination condition met* **then**
    nextIt $\leftarrow$ False
  end
end

Algorithm 1: ALNS

Data: Unassigned customer, distance matrix, time matrix, battery consumption matrix, cost matrix
Result: routes, time stamps, vehicle battery levels, customers on board
Initialization ;
c = number of customers per route ;
r = number of routes ;
for *i in range(r)* **do**
  for *j in range(c)* **do**
    customer = random(unassigned requests) ;
    r[i] = customer(pick up node) + customer(delivery node) ;
    remove customer from unassigned requests;
  end
  compute time stamps, vehicle battery levels, customers on board ;
end

Algorithm 2: Initial solution

Data: Function worst_removal($s \in \{solutions\}$, $q \in N$, $p \in R_+$)
Result: removed requests q
Initialization ;
while *q > 0* **do**
  Array: L = All requests i, sorted in descending order profit(i, s) ;
  y = random number in interval $[0, 1)$;
  request to remove: $r = L\,[y^p|L|]$;
  remove request r from solution s ;
  q = q - 1 ;
end

Algorithm 3: Worst removal

Data: Function Shaw_removal($s \in \{solutions\}$, $q \in N$, $p \in R_+$)
Result: removed requests q
Initialization ;
request: r = randomly selected request from S ;
set of requests to remove: D =$\{r\}$
;
while $|D| < q$ **do**
1 | r = a randomly selected request from D ;
| Array: L = array containing all request from s not in D ;
| sort L such that $i < j \rightarrow R(r, L_i) < R(r, L_j)$;
| y = random number in interval $[0, 1)$
| $D = D \cup \{L\,[y^p|L|]\}$;
end
remove requests in D from s ;

Algorithm 4: Shaw removal

2 C Problem parameters

Table 17: Parameters for the UAMP

Parameter	Description
σ	Number of requests
$[e_i, l_i]$	Early and late TW to visit node i
d_{ij}	Distance between node i and j
t_{ij}	Time between node i and j
D_i	Depart time from node i
A_i	Arrival time at node i
W_i	Waiting time at node i
τ_i	Start of service at node i
c	Cost parameter per km
n	Vertiport index such that $n \in V$
vc_n	Vehicle capacity at vertiport n
vcc_n	Vehicle charging capacity at vertiport n
k	Vehicle index such that $k \in K$
Q_k	Capacity of vehicle k
BLa_i^k	Arrival battery level of vehicle k at node i
BLd_i^k	Departure battery level of vehicle k at node i
p_i	Processing time of customer i (embark, disembark)
α_i	Time window deviation factor of customer i
β_i	Trip duration factor of customer i
γ_i	Distance related fee for customer type i (per km)
δ_i	Time related fee for customer type i (per h)
s_i	Satisfaction of customer i
R_i	Ride time of customer i
L_i	Distance travelled of customer i
c_i	Charge of customer i
d_i	Discount of customer i

II

Literature Study
previously graded under AE4020

1 Introduction

1.1. Background information

Inadequate mass transit options, obstacles in the roads and development in areas where transit and road systems are poor are the main reasons for traffic congestions. As a result million of hours are wasted every day on the road worldwide [27]. Apart from travel time uncertainties and increased accidents probabilities, traffic congestions impact our society in the following ways:

- **Environment**: In many regions across the world, vehicle emissions are the main source of air pollution. Such pollutants include carbon monoxide and dioxide, nitrogen oxides and hydrocarbons [70]. With the increase of traffic congestions in the past years it is estimated that pollutant emissions will increase, causing lower air quality and health related problems to people living near large road networks.
- **Economy**: It is estimated that the cost of congestion in the USA is approximately $ 121 billion per year which corresponds to 1 % of its Gross Domestic Product (GDP) [6]. To put this number further in perspective more than 5.5 billion of hours are lost while in traffic and about 3 billion gallons of fuel are burnt. Uber [27] states that an average resident in San Francisco spends on average 230 hours traveling between home and work.

Nevertheless, the population is expected to further increase the next decades [66] impacting even more current transportation services. Urban Mobility Services (UMS) would require to sustain future transportation demands in a more sustainable way. This can be achieved either by investing in heavy-infrastructure approaches (roads, tunnels, etc.) or by other means. With the advance of technology, researchers are looking into new ways of transportation such as Hyperloop, Urban Air Mobility (UAM) services, smart roads, super trains and elevated cycle paths [64].

UAM services are effective, safe and a sustainable way of transporting passengers or goods in large metroplex areas. Vehicles will use vertiports to take-off and land. Operations within a UAM service will vary between news gathering, traffic studies, package deliveries, transportation services and other [63]. The focus of this research concerns transportation service such as Uber elevate. Such services are expected to be fully operational by a small fleet of e-VTOL aircrafts by 2023 in United States of America (USA) [27]. A big advantage of UAM is that route durations are significantly decreased. For instance a 2 h 12 min ride could be performed in 15 min [27]. Such travel reduction will relieve traffic congestions and make travel within a metroplex area more efficient and effective. In addition, Vertical Take-Off and Landing (VTOL) vehicles will be fully electric meaning that they produce zero emission pollutants.

At the same time the challenges of such a service need to be faced before a real implementation. These can be categorized in the followings [27]:

- **Operational**: Real time scheduling of the fleet, air traffic safety, accessibility to elevated pick-up and delivery points and noise limitations in urban areas.
- **e-VTOL Performance**: Challenges associated with battery reliability and durability.
- **Costs**: profit margins and costs of such a service and the initial costs associated with the construction of vertiports; these are areas where the aircrafts will be able to take off and land.

1.2. Motivation and problem statement

Recent literature concerning UAM services has been evolving around departure and arrival sequencing, trajectory management, conflict resolutions, Air Traffic Control (ATC), optimum arrival times and concept of operations [3, 14, 63, 67].

Furthermore, there has been a lot of research in the scheduling of Vehicle Routing Problem (VRP) where different mathematical approaches and solutions to the problem are proposed including heuristics, dynamic programming and branch and bound methods [22, 29, 52, 56].

Other research is focused on ride-sharing applications where the waiting times and the delays of customers are minimized [6, 7]. UAM service can be seen as a combination of both ground and air transportation services, however, with several fundamental differences. This shows the importance of creating new operational models specifically design for UAM services that are based on new set of constraints and demand data [60]. Concerning the passenger demand, most transportation problems are using strategies that are designed based on trip request models and passenger mobility data [41, 47].

In order to decide what type of fleet to use and how to dispatch e-VTOLs, commercial companies such as Uber would have to make tactical and strategic decisions [60]. This is done to satisfy the forecasted demand, while maximizing profit margins. In most ground transportation problems, customer requests are happening real-time (dynamically). Since a UAM service is similar in this aspect to a ground transportation system, the following question arises:

"What kind of modelling approach should be implemented in a UAM network, and which optimisation algorithms can take dynamic customer requests into account effectively?"

1.3. Research objectives

The aim of this research is to schedule real-time (dynamic) flights of e-VTOLs in metroplex areas. This is because new requests are received instantly and part of the pre-planned routes should be re-adjusted to accommodate them. Therefore, it is necessary to develop an algorithm that is capable of providing good-quality solutions in a limited computational time. This will be apllied to a certain time-horizon (6 h) to simulate a specific period of the day with high and low forecasted demand peaks.

This is an optimization problem where the main objective is to maximize profit (revenue - cost) by taking into account customer dissatisfaction, where a discount is given when exceeding maximum ride time. Thus the research objective is defined as follows:

'Maximize profit considering customer dissatisfaction by creating an optimization algorithm for a dynamic UAM network.'

Since UAM is not operational yet, there is no actual data concerning **demand** and **vertiport locations**. Therefore, the model will be based on a fabricated set of data. These include demand forecasts and potential locations of vertiports, which can directly impact the results of the model. A sensitivity analysis should be performed to explore the model's limits by varying the number of vehicles, customer types and demand distributions.

The model can be validated by comparing this algorithm with the work of other researches, or by comparing it with an exact solution to the problem. The latter is more applicable, very little research exists publicly regarding UAM. It allows a direct comparison between the developed algorithm and an exact solution, to determine its precision.

1.3.1. Research questions

To get a better understanding of the topic, this literature study tackles the following questions which are necessary to develop the UAM model. Table 1.1 indicates the location of the answers in this report.

1. What are the operational constraints and cost parameters related to aerial ride sharing applications?
2. Which modelling framework should be used to investigate the problem?
    - What are the differences and similarities between the investigated models?
    - How are they applicable in a UAM network?

3. Which algorithms are appropriate for such application to provide an effective and efficient solution?
    - Is the solution satisfactory from an optimization perspective?
    - Is the solution computed fast enough for a dynamic request assignment?
4. Should vehicle rebalancing be incorporated in the model and if so which strategy is the most applicable?
5. What are the criteria of rejecting new customer requests and in what order should they be served?

Table 1.1: Location of research questions answers in the report

Literature question	Answer
1	Chapter 2
2	Sections 3.1 and 3.2
3	Chapter 4
4	Section 3.4
5	Section 3.3

Nevertheless, it is important to divide the research objective into research questions that will be answered once the model is developed. These questions include:

1. What is the effect on profits when considering different types of passengers?
    - The focus is to see how direct vs. connecting passengers affect the profits.
2. What is the impact of the number of vehicles when maximizing the profits of a UAM network?
3. What is the number of vehicles and/or customers that the model can incorporate while solving the problem in a low computational time?

1.4. Report structure

The report is structured in the following way. Chapter 2 discusses the concepts of UAM and creates an analysis on the operational constraints and parameters that are essential in aerial ride-sharing applications. Chapter 3 provides an overview of modelling approaches of transportation problems which are applicable to a UAM network. Chapter 4 provides information for the solution approaches used in transportation problems. These include includes exact and approximate solutions (heuristics) to determine which one would be the most applicable for this thesis. Finally, a conclusion of all the findings is shown in chapter 5.

2

Urban Air Mobility

This chapter is devoted to a Urban Air Mobility (UAM) network. Firstly, section 2.1 looks at the main feature that such a network should have and their interactions. The mission profile is discussed in section 2.2 and each variant of the model is discussed in sections 2.3 to 2.5, where the operational constraints of the network are discussed.

2.1. Network characteristics

A UAM network should provide on-demand services in a metroplex area. Such a network is affected by the location and capacity of vertiports, demand forecasts and vehicle specifications. This raises the following questions regarding the network's sub-models:

- **Network model:**
  - For which time period this model will take place?
- **Vehicle model:**
  - What kind of vehicles should be used for such a service?
  - Will the model incorporate homogeneous or heterogeneous fleet?
  - What is the number of vehicles that the model will encounter?
- **Vertiport model:**
  - How many vertiports should be used and what should be the distancing between these vertiports?
  - What is the capacity of each vertiport?
- **Demand model:**
  - What kind of demand forecasts should be used?
  - Will the model incorporate high and low peak hours?
  - What are the commuting trends of passengers in different hours?
  - How will the demand model be verified since data are fabricated?

For a UAM network there is an interaction between the aforementioned sub-models. For example, vertiport distancing is affected by the range of the vehicles whereas vertiport capacity defines the maximum number of vehicles that such a network can support. Moreover, demand satisfaction is affected by the number of available seats. Having more vehicles than the available demand capacity, can cause loss of profit since not all the vehicles will be utilized. Additionally, demand directly affects the number and capacity of vertiports that are spread in the metropolitan area. All of these factors have an impact on the overall UAM model in terms of operating cost, profit margin and customer satisfaction. The aforementioned are illustrated in fig. 2.1.

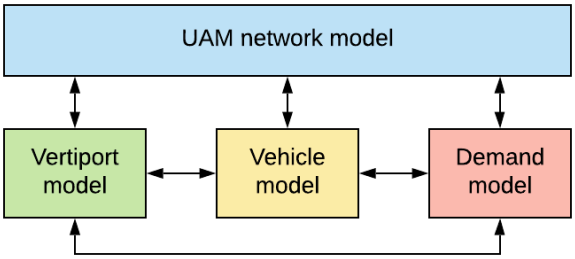

Figure 2.1: Major UAM model components and interaction

2.2. Mission profile

The general mission profile of a UAM network consists of four distinct operations as seen in fig. 2.2. These are as follow:

1. **Customer request:** Customer requests a vehicle to be transported from vertiport i to j.
2. **Customer first-mile travel:** If no vehicle is available at the vertiport, a vehicle is dispatched to pick up the customer. This phase also includes passenger embarking and taxiing-out from the vertiport.
3. **Flight:** This phase includes take-off, climb, cruise and landing of the vehicle.
4. **Customer last-mile travel:** This is the final phase and includes taxiing-in and customer disembarking. Here the vehicle either refuels/recharges or prepares for the next mission.

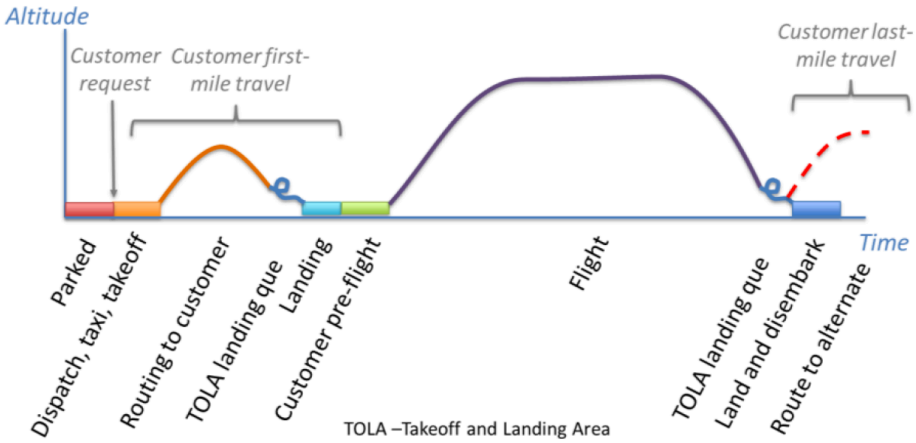

Figure 2.2: E-VTOL and customer operation profile [67]

Patterson et al. [46] further analyzes the mission profiles of UAM into three distinct categories. The first mission, **'long-range, small payload'**, represents the daily commuting to work and back. It consists of a single person with little luggage commuting distances up to 140 km in a day. The second mission, **short-range, large payload**, involves several people travelling with large amounts of luggage over short distances. These distances account for up to 70 km during the day. The last mission is defined as **'most constraining mission'**, and is a combination of the two aforementioned missions. It includes long-range missions while allowing large payloads. This configuration is an example of large groups of people travelling over long distances.

The mission profile of UAM transportation gives useful information to compute the flight time and the required charge to perform the mission. Table 2.1 depicts the required times and energy for each phase of the mission profile [38]. The total time of the mission is obtained by adding all the components. The cruise time t_{ij} is obtained using the distance between the departure (i) and arrival vertiport (j) d_{ij} and the cruise speed of the vehicle (V_{cruise}):

$$t_{ij} = \frac{d_{ij}}{V_{\text{cruise}}} \tag{2.1}$$

Similarly the energy consumption during each phase is obtained by multiplying the time spent at each phase with the power requirement, expressed as a factor of the cruise power. The total energy consumption is the sum of all phases. These values can be obtained in vehicle specification manuals or scientific reports such as Bacchini et al. [10].

An important constraint imposed by Uber [27] is that customers are eligible for a aerial ride if the estimated route duration is at least 40 % less than its corresponding ground transportation trip.

Table 2.1: Mission profile times and required energy

Mission phase	Time [s]	Energy rate [$\cdot P_{\text{cruise}}$]
Embarking	180	0
Taxi-out	30	0.1
Take-off	30	3
Climb	60	2
Cruise	t_{ij}	1
Land	30	3
Taxi-in	30	0.1
Disembarking	180	0

2.3. Vehicle model

The vehicle model defines the type and properties of the vehicles to be used for this research. The first step is to clarify the type of vehicle that is most applicable. As mentioned in section 1.1, transportation needs to be sustainable and most recent research is focused on e-Vertical Take-Off and Landing (VTOL) aircraft [7, 67]. This is because they have reduced noise levels and zero-emissions [27]. Additionally, a homogeneous fleet (identical vehicle characteristics) is chosen as done by Shihab et al. [60] and Kohlman et al. [38].

In recent years a lot of companies are developing e-VTOL vehicles, some of which include *Joby Aviation*, *E-Hang*, *Toyota* and *Airbus* [67]. e-VTOL specifications directly impact the operational constraints of the vehicles [60]. These include vehicle capacity, cruise speed, range and battery specifications. Depending on the mission profile (urban or sub-urban) of each vehicle, e-VTOLs have a varying capacity of 2-5 passengers excluding pilots. Cruise speed varies from 96 km h^{-1} to 330 km h^{-1}, while range varies from 30 km to 300 km. Table 2.2 shows a list of potential e-VTOLs for a UAM network. Battery specifications are discussed in more depth in section 2.3.1.

Table 2.2: List of potential e-VTOLs

	E-Hang 184	Kitty Hawk Cora	Lilium Jet	Joby S4
Passenger capacity	2	2	5	4
Cruise speed (km h^{-1})	130	180	252	322
Range (km)	30-40	160	300	240
Battery capacity (kWh)	14.4	63	38	N/A
Cruise Power (kW)	34.6	63	28	N/A

2.3.1. Battery specifications

The distances between vertiports are unequal as mentioned by Bosson et al. [14] and Vascik et al. [67]. Thus it is important to take into account the battery's State of Charge (SOC). It describes the state of the battery at the end of each flight operation. It can be modelled as a simple linear function as described by Kleinbekman et al. [36]. At first, the total electric power demand is computed as shown in eq. (2.2). The Safety Factor (SF) is assumed to be 1.5 [36] to encounter for weather conditions and emergency situations. Both the rotor η_p and mechanical efficiencies η_e are constants and depend on the chosen e-VTOL. The required power P_r can

be taken from table 2.2 and each flight state from table 2.1. The battery's current at all the flight states $I(F)$ is depicted by eq. (2.3).

$$P_d(F) = \text{SF} \cdot \frac{1}{\eta_p} \frac{1}{\eta_e} P_r(F) \tag{2.2}$$

$$I(F) = \frac{P_d(F)}{V_n} \tag{2.3}$$

The flight states F are numbered 1 to 6 and represent taxi-out, take-off, climb, cruise, landing and taxi-in such that $1 \leq F \leq 6$ where each number corresponds to the state of the flight phase. Each flight lasts for a certain period of time with a start and end period denoted by t_0^F and t_f^F respectively. V_n is the nominal battery voltage; a constant for a specific eVTOL. The SOC at each flight phase is calculated given eq. (2.4) [48] where Q is the battery's capacity. The total SOC is given by eq. (2.5) where it sums the SOC at each flight state. To avoid deep discharge [36] assumes empty battery if the SOC of a battery reaches 10 % where the vehicle must recharge.

$$SOC(F) = \frac{I(F) \cdot (t_f^F - t_0^F)}{3600 \cdot Q} \tag{2.4}$$

$$SOC = \sum_{F=1}^{6} SOC(F) \tag{2.5}$$

Recharging of e-VTOLs takes places at the vertiports. It is assumed that all vertiports are equipped with high voltage charging stations [60] and due to emerging battery technologies, charging times are estimated at 30 minutes [67]. It is important from an operational perspective to keep a minimum turnaround time of the vehicles. This means, that if an e-VTOL has a sufficient charge and new requests show up, the vehicle will continue its journey.

2.3.2. Costs

Operating costs in e-VTOLs are split between direct and indirect costs. Direct costs include pilot training and vehicle maintenance. According to Uber [27] the maintenance costs will account for 22 % of the direct cost baseline and pilot costs up to 36 %. In case of autonomous vehicles (long-term project), the operating costs will decrease. On the other hand, indirect costs, will account for non-vehicle specific costs such as; payment, registration and permit fees, infrastructure and insurance. It is estimated that near term the cost price per kilometer will be approximately $ 1.02, and in the long term $ 0.8.

2.4. Vertiport model

Vertiports (Nodes) are locations in a metroplex area where VTOLs can take-off and land. Their concept is similar to airports, however smaller in size and capacity. Potential locations of vertiports are hospital helipads, building roofs and municipal airports [14]. A UAM network should be first incorporated in large metropolitan areas as they feature feasible infrastructure with viable economical markets. Such areas area observed in table 2.3 [14, 27, 44, 67].

Table 2.3: List of feasible infrastructure areas for a UAM network

Area	Country
Dubai	United Arab Emirates (UAE)
Sao Paolo	Brasil
San Franscisco	United States of America (USA)
Los Angeles	USA
Huston	USA
Dallas-Fort Worth	USA

Defining the number of vertiports, their capacity and distancing among each other are the main elements composing the vertiport model. Bosson et al. [14] suggested a possible distribution of vertiports in Dallas-Forth Worth without defining their capacity. The model consists of 20 vertiports portrayed as black dots in fig. 2.3 where the span of distances ranges between 5 km for the shortest distance and 130 km for the longest. It covers a total of 24 100 km^2 in the metropolitan area. Distances between each vertiports are considered Euclidean; straight lines in a metric space.

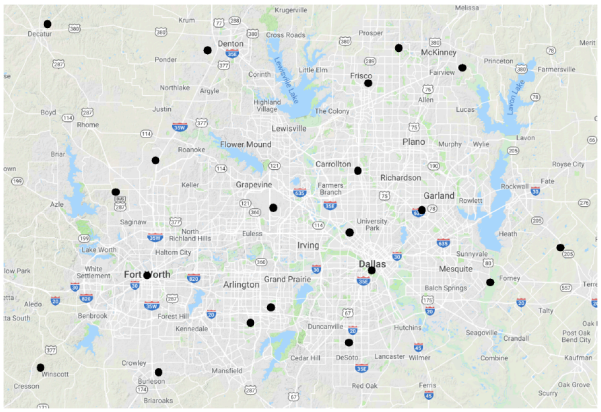

Figure 2.3: Possible location of vertiports in Dallas [14]

Shihab et al. [60] designed a fabricated vertiport model consisting of 3 fully connected vertiports where their distancing was assumed to be from 96 km to 128 km long inspired by Bosson et al. [14]. 2 of the vertiports represent neighborhood areas while the last one the downtown area. 20 e-VTOLs are incorporated in the model meaning that each vertiport can have of a capacity up to 7 vehicles.

Kohlman et al. [38] suggested a possible network distribution of vertiports in Houston, Texas. The model consists of 7 vertiports and assumes equal distancing among all of them. Vertiport 1, located in the downtown area is equipped with 8 take-off/landing pads and zero charging stations. Vertiport 2-7 located in sub-urban areas are equipped with 4 take-off/landing pads and 8 charging stations each. The distancing between the vertiports is referred as L_m and is equal to 35 km. The vertiport model follows an hexagon pattern inspired by Patterson et al. [46] as seen in fig. 2.4. The hexagon pattern allows for a simulation of the downtown area and the suburbs for most cities. However, this approach is limited to large coastal cities such as Los Angeles and New York city [46].

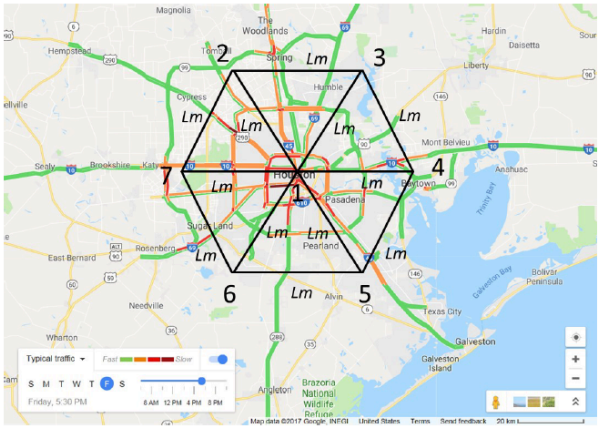

Figure 2.4: Location of vertiports in Huston [38]

Both Kohlmand et al. [38] and Shihab et al. [60] incorporate in their models fewer vertiports with similar capacity than Bosson et al. [14]. This is because [14] defines a set of possible vertiport locations in specifically Dallas-Forth. For this model, the adoption of 13-14 vertiports in a hexagon pattern with some variation seems a good approach to the problem. For instance, there could be 2 vertiports located in the downtown area as seen in [14] with their distancing varying from 3 km to 5 km with the constraint that movements between these vertiports are not permitted. A benefit of such addition to the network is related to demand satisfaction. For example, if one vertiport in the downtown has reached its maximum capacity passengers can be re-directed to the nearest one. The remaining 12 vertiports will be located in an hexagon pattern with their distancing varying from 30 km to 60 km to represent 'suburb' and 'outer suburb' areas. The vertiport capacity in the downtown areas should be higher than the ones in the suburbs as mentioned in Kohlman et al. [38], however all vertiports should be assumed to have the charging capabilities as discussed in Shihab et al. [60].

The choice of the distancing is depending on the mission profile discussed in section 2.1.

2.5. Demand model

Demand represents the amount of passengers in a given Origing - Destination (O-D) market [60]. As stated in section 2.1, demand drives the operational decisions of the UAM as it is the main source of revenue. The factors driving a demand model are illustrated by fig. 2.5 and are categorized as the following [54]:

- Trip fare
- Travel time including number of connections and frequency
- Quality of service
- Passenger characteristics; age, income

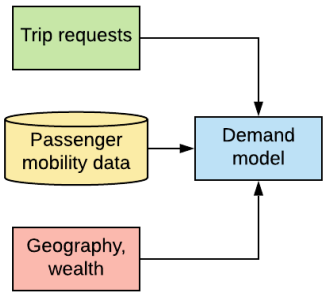

Figure 2.5: Factors driving demand model

A day of operations could be distinguished between **morning**, **evening** and **high**, **low** peak hours. During the morning hours passengers commute mainly from home to work, schools, recreational activities and church whereas during the evening periods the opposite behavior is observed [60]. Figure 2.6 show the behavior of passengers during morning and evening hours. Vertiports 1 & 2 correspond to neighborhood areas, while vertiport 3 to the downtown area. This implies that during morning hours, e-VTOLs should be deployed in the suburbs of the network to accommodate as much demand as possible. Little requests are observed from the downtown region to the suburbs, whereas at evening hours the opposite is observed.

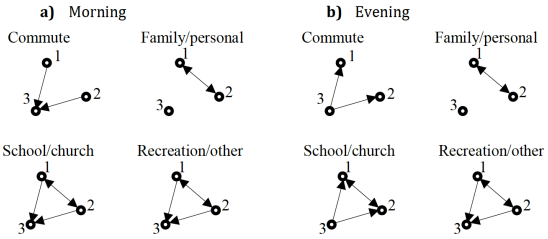

Figure 2.6: O-D demand market a) morning vs b) evening [60]

High peak hours or else rush hours, are the periods of the day where demand reaches its peak values. During real time operations a variation in the demand is associated with fluctuations in the prices (dynamic pricing) for that service [60]. This is inspired from Uber [25] where at different times of the day, prices vary. Such concept is applicable to UAM as incorporated in [60], however outside of the score of this thesis. A simple fare model can be incorporated based on the travelled distance and time spent in the vehicle as seen in Yook et al. [69] work.

In the near term e-VTOL vehicles are constrained to transfer passengers between 2 vertiports and not door to door. In the long term, UAM services could take-off and land in private residents [27]. This translates that a customer who request an aerial ride would have to get to the nearest vertiport by himself. Thus, it is reasonable to assume that all demand will be originated at each vertiport. This assumption ensures that the UAM model will not notify passengers to which vertiport they should go, but requests are generated at every vertiport.

The demographics of Uber [40] in the USA show the age groups, income and urban context of using this service. The data is shown in table 2.4 and describes that 65 % of all users are the in the ages groups of $< 24-34$ years old, while 94 % uses the service in urban and suburban environments. NASA Air mobility [42] conducted a study to observe how these values will vary for UAM. It was noted that the demand the first couple of years will be much lower however the demographics shown in the table will have a great variety.

Table 2.4: Demographics of Uber US users

Age		Income		Urban context	
Group	(%)	Tier	(%)	Area	(%)
<24	37	Bottom 25 %	22	Urban	46
25-34	28	Mid 50 %	44	Suburban	48
35-44	17	Top 25 %	27	Rural	6
45-54	12	Not stated	7		
>55	6				

An interesting concept which could be used in a UAM and is adopted by Uber [27] is the classification of passengers. For the scope of this research these passengers could be classified between **'direct'** and **'transfer'**. Direct passengers are those who pay an additional fare and travel directly from vertiport i to j. Transfer passengers are the ones who pay a standard fare and accept to share a ride with other or unexpected customers. From the demographics it reasonable to assume that 22 % of the total UAM demand will be categorized as direct passengers.

As seen in section 2.4 in fig. 2.3 Bosson et al. [14] suggest a network of 20 vertiports locations. In fig. 2.7, a network connecting the downtown area of Dallas to the remaining 19 vertiports is observed. Figure 2.8, shows a possible complete network in Dallas-Forth Worth, where it consists of 190 different routes among all vertiports. The driving factors of the connectivity and frequency of routes is based on the demand model.

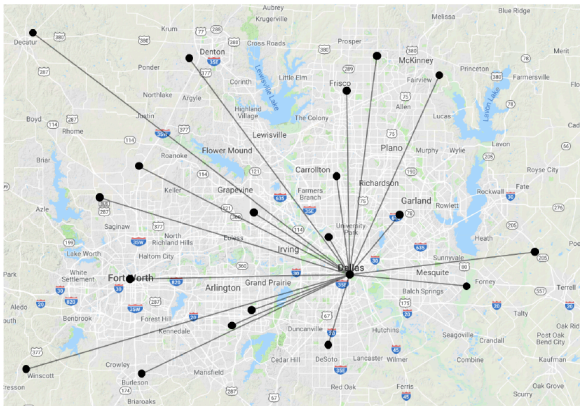

Figure 2.7: Possible connectivity from Dallas downtown [14]

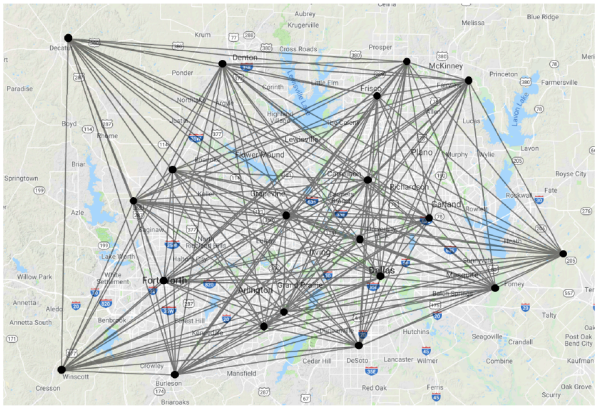

Figure 2.8: Possible route network in Dallas [14]

Shihab et al. [60] propose a forecasted demand model that is based on past studies for helicopter charter services, wealth and census data. The model is given in figs. 2.9 and 2.10. Since the proposed model consists of 3 vertiports (2 neighborhoods and 1 downtown area), there are 6 distinct O-D markets each one with variations in their demand values. The terms V1-2, simply defines the demand from vertiport 1 to 2.

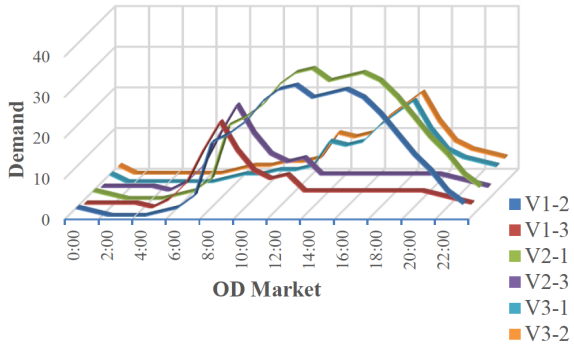

Figure 2.9: Demand model for different O-D markets [60]

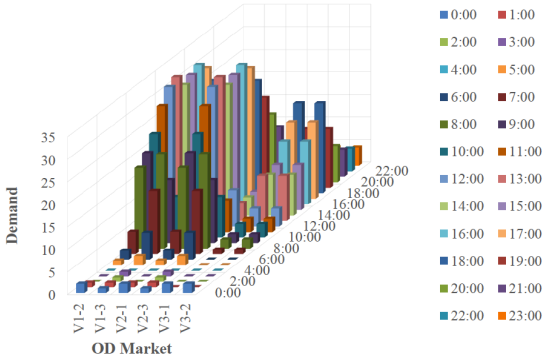

Figure 2.10: Hourly column demand chart for different O-D markets [60]

Kohlman et al. [38] propose a demand model that generates customer requests stochastically at every vertiport. It is a probability curve (bimodal distribution) which consists the sum of 3 normal distributions that are normalized to 1. This allows to model for high (8am, 4pm) and low (12pm) peak hours as seen in fig. 2.11. An additional parameter referred as M_4 is used to scale the curve, where $M_4 = 8500$. The curve for every vertiport is generated by multiplying its weight with the curve. From fig. 2.4, vertiport 1 has a weight of 2 while vertiport 2-7 have an assigned weight of 1. Requests are generated with comparing a random number (0-1) to the curve. If the random number is lower than the curve's value a request is generated. The process is repeated for every time step (10 s).

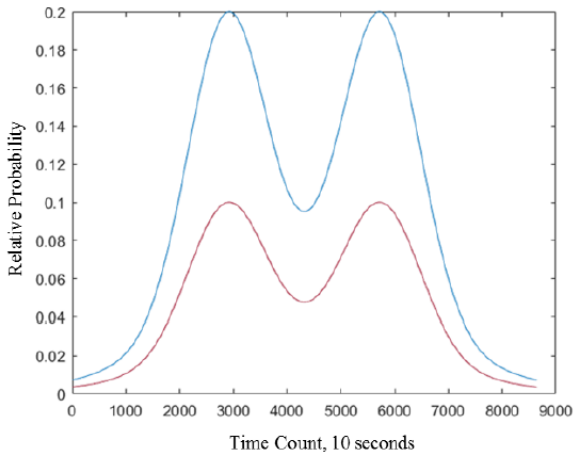

Figure 2.11: Demand probability function for a day of operations [38]

In both Shihab et al. [60] and Kohlman et al. [38], demand models there is a peak period around 8am and 4pm. Since these values are forecasted, and directly affect the credibility of the results, [60] suggests to vary the normal distribution from 5-50 %. Since standard deviation depicts the accuracy of the forecasted results, by varying it we can simulate the accuracy of the forecasted demand model. The demand model proposed by [38] was also used by [36] whose paper focused on arrival scheduling for on demand UAM. Even though both demand model is applicable for an entire day of operations, [60] suggests a small scheduling horizon of 3-6 hours for real time trip requests.

3

Transportation models

This chapter focuses on the modelling approaches for Urban Air Mobility (UAM). At first in section 3.1 the problem is categorized in a family of similar problems, where different ground and air transportation models are reviewed. Section 3.2 discusses the most suitable modelling apporach for this particular problem and lists a set of possible objective functions. Finally section 3.3 and section 3.4 discuss the strategies for inserting customers dynamically and rebalancing empty vehicles.

3.1. Problem categorization

Finding a set of nominal routes for a specific fleet in UAM is a combinatorial optimization problem. Such problems belong in the family of Vehicle Routing Problem (VRP) and are considered non-deterministic (NP-hard). The concept of VRP was first introduced by Dantzig et al. [21] and is defined as *"the problem of designing optimal delivery or collection routes from one or several depots to a number of geographically scattered cities or customers, subject to side constraints"* [51]. Depots are referred as the geographical locations where vehicles start and end their routes during operations.

The problem can be categorized between **'static'** or **'dynamic'** and **'deterministic'** or **'stochastic'**. As a result it can be Static and Deterministic (S&D), Static and Stochastic (S&S) or Dynamic and Deterministic (D&D), Dynamic and Stochastic (D&S) [51]. A problem is considered *'static'* if the demand distribution (customer requests for UAM) are known before the construction of the route network whereas in a *dynamic* problem a part of the requests are known. The 'unknown' requests becomes available during operations. A *deterministic* problem means that information such as potential location of users, users showing up on time, exact duration of operation is known with certainty ahead of time. In a *stochastic* problem such information is unknown even if they are assumed according to a probability distribution.

A VRP is applicable in any form of transportation such road, maritime and air. Due to its wide range of applications there are many variants to the problem some of which include:

- **VRP with time windows**: Locations where people/ goods have to be picked-up or delivered within a certain time period.
- **Capacitated VRP**: Vehicles have a certain capacity while transporting passengers/goods.
- **Open VRP**: Vehicle's return to depots are not constrained.
- **VRP with multiple trips**: A vehicle can execute more than 1 route.

In general Toth et al. [65] describe the VRP as a graph (road network) using arcs (connecting links) and vertices (nodes) such that $G = (V, A)$. Arcs are classified as *directed* or *undirected* depending on the interaction among them. Each arc has an associated cost, travel time and distance. In VRP, nodes represent customers (located in different areas) and depots (vehicle's starting/ending points) whereas edges represent the roads among them. The modelling approaches for a VRP can vary between **'vehicle flow formulations'**, **'commodity flow formulations'** and **'set partitioning problem'** [65]. The VRP design its routes in such a way that:

- All locations are visited at least once by a vehicle.
- Vehicles start and end their route at the depot
- Side constraints according to the problem formulation are satisfied. These can include vehicle capacity and range, time related constraints etc.

3.1.1. Time Windows

Time Window (TW) are widely used in transportation models as they represents a time interval in which customers specify the earliest and the latest time that the service can takes place [55]. Thus, they are very relevant to a UAM network. TW are denoted as $[e_i, l_i]$ where e_i corresponds to the earliest time and l_i to the latest one. TW can be distinguished between *explicit* and *implicit*. Explicit TW, take into account the earliest time point, e_i. If the vehicle arrives at that node at time A_i where $A_i < e_i$, then the vehicle waits at that node until it can depart. The vehicle can depart from that node at a departure time $D_i = max\{A_i, e_i\}$. Implicit TW are used to control customer dissatisfaction. In most transportation services customers specify either a desired pick-up ($\hat{A}_{i-}$) or delivery ($\hat{D}_{i+}$) time.

A half open TW is defined by satisfying $A_{i-} \leq \hat{A}_{i-}$ and $D_{i+} \geq \hat{D}_{i+}$. These inequalities mean that the actual arrival time (A_{i-}) has to be no more than the desired one. Similarly, the actual departure time (D_{i+}) has to be greater than the desired one. On demand responsive situations, customers require a service as soon as possible. Savelsbergh et al. [55] propose that these clients will be scheduled as last in the vehicles. To avoid this situation and prevent the customer for an indefinite waiting closed pick up windows are introduced $[0, l_i]$ where l_i corresponds to an input to the system. These define a maximum deviation from the desired and the actual time of pick-up or delivery. An illustration of a time windowed operations is shown in fig. 3.1. The first part of the figure shows Desired Pick up time (DPT) related customers while the latter the Desired Delivery time (DDT) ones.

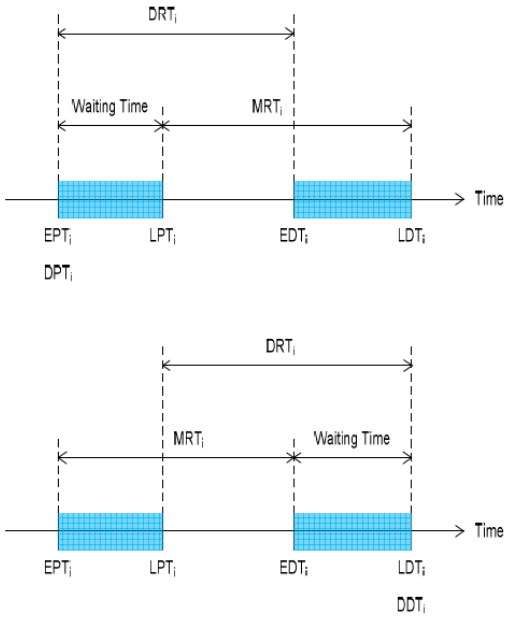

Figure 3.1: Time Window customers according to DPT and DDT [29]

A common problem associated with a TW is that they do not ensure the Maximum Ride time limitation (MRT) constraints. For example if a customer has a pick-up time window of [10:00, 10:15] and a delivery window of [10:30,10:45], then the maximum ride time is 30 minutes. If the pick-up would occur at 10:00am and the delivery at 10:45am, this would violate the maximum ride time constraint. Therefore, shrinking for example the pick-up to [10:15,10:15] enforces the maximum ride time constraint.

3.1.2. Relevant models

To set up a UAM model it is essential to understand the characteristics of problems belonging in the same family. This part of the report focuses on identifying variants of the VRP applied both in ground and air transportation. The most used models in these areas are the Pick-up and Delivery Problem (PDP), Dial A Ride Problem (DARP), Share-A-Ride Problem (SARP) and Fleet Assignment Problem (FAP). By understanding the similarities and differences of these models it will be easier to identify a modelling apporach for UAM.

Pick-up and delivery problem

The PDP is a variation of the VRP where a set of routes needs to satisfy several transportation requests of goods [55]. The problem consists of a central [52] or multiple depots [31]. Vehicles visit the *pick-up* and *delivery* nodes to pick-up and drop off the parcels respectively. An illustration of the PDP network is shown in fig. 3.2. There are many variations to the problem where [55]:

- requests have a set of origins but a single destination,
- requests have a single origin but a set of destinations,
- requests have multiple sets of origins and destinations,
- vehicles having different starting and end locations,
- Requests evolve on real-time of operations.

The last point, refers to demand responsive situations and is irrelevant of the scheduling horizon length. Such problems increase the complexity as to account for new requests either 1 pre-planned route will change or a new vehicle will be dispatched as seen in Yeun et al. [1]. This indicates that the meaning depots can sometimes becomes null as drivers may sleep at the last location of their visit or that the first location of the next day of operations [55].

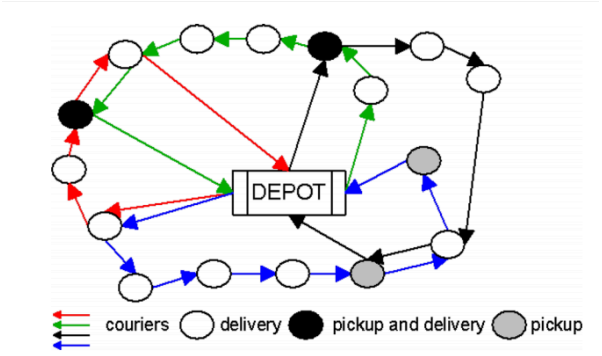

Figure 3.2: Illustration of a single depot PDP network [61]

Savelsbergh et al. [55] formulates a general PDP in the following way. At first there is a set of transportation requests defined as N. Each request i is defined such that $i \in N$, and each request i has a load of q_i such that $q_i \in N$. These requests are transported from pick-up nodes N^+ to delivery nodes N^-. Loads that are being picked-up have a positive quantity whereas loads that are being delivered have a negative quantity. This is defined mathematically as $q_j = \sum_{j \in N^+} q_j = -\sum_{j \in N^-} q_j$. The set of all nodes V includes all the pick-up and delivery nodes thus $V = N^+ \cup N^-$. Apart from the requests, there is a set of vehicles denoted by M where each vehicle k belongs in that set such that $k \in M$. Furthermore, each vehicle has a specific capacity denoted by Q_k such that $Q_k \in N$. In many cases, there are many locations where the vehicles starts and end their journey represented by k^+ and k^- respectively. The start and end of all locations is defined by $M^+ := \{k^+ | k \in M\}$ and $M^- := \{k^- | k \in M\}$. Finally the set of all vehicle routes is given by W and is defined as $W := M^+ \cup M^-$. Lastly, visiting from i to j is done in such a way that $i, j \in V \cup U$ with a travel distance d_{ij}, travel time t_{ij} and travel cost c_{ij}.

Dial-A-ride problem

The DARP is a variation of the PDP where vehicles instead of goods are transporting passengers; more particularly elderly and disabled people. Users can share rides, as they are transported in the same vehicles (mini-buses). It is often the case that the same user will request the service twice in a day. This is because there are outbound requests (from home to destination) and inbound requests (from destination to home) [19]. Furthermore, DARP usually works with time windows which means that users do not specify an exact pick-up/delivery time of the day but rather a time interval period. They are only allowed to specify either their pick-up or delivery time window and then the operator is responsible to assign the remaining time window to that customer.

Since in DARP users are sharing rides, the main issue that occurs is that customers don't want to take large detours to arrive at their destinations even if it could be beneficial for the system performance. Hence a maximum ride time constraint is specified for each transport request. In this problem, time windows could be beneficial for the operator since they can schedule the routes to accommodate as many passenger as possible.

A formulation of the DARP model is given below proposed by Cordeau [17]. The network takes places in a directed graph $G = (V, A)$ where V represents the sets of all nodes and A the sets of all arcs. As discussed above, customers have pick-up nodes (P), delivery nodes (D), the start location $(\{0\})$ and end location $(\{2n+1\})$ of the vehicles. Therefore $V = \{\{0, 2n+1\}, P, D\}$, where $P = \{1, ..., n\}$ and $D = \{n+1, ..., 2n\}$. Furthermore a request is coupled with a pick-up and a delivery service $(i, n+i)$ such that $i \in P$ and $n+i \in D$. For every vertex $v_i \in V$ there is an associated load q_i where at the starting and ending locations is equal to zero, thus $q_0 = q_{2n+1} = 0$. The load at the pick up nodes should always be greater or equal to zero since the vehicle pick up customers such that $q_i = 0$ for $i = 1, ...n$ and $q_i = -q_{i-n}$ for the delivery nodes $(i = n+1, ..., 2n)$. This is the same characteristics as observed in the PDP with the difference that parcels are replaced by people. Each vehicle has a certain seat capacity defined Q_k and a maximal route duration $k \in K$ denoted by T_k. Traveling in the arcs (i, j) there is an associated cost c_{ij} and a travel time t_{ij}. Arcs are defined in the following way: $A = \{(i, j) : i = 0, j \in P \; or \; i, j \in P \cup D, i \neq j \; and \; i \neq n+j, \; or \; i \in D, j = 2n+1\}$.

Share-A-ride problem

The SARP is a combination of the DARP and PDP where people and parcels share the same vehicle (taxis) [2]. Thus, the following situation can arise from such a problem:

- a taxi carrying people only
- a taxi carrying parcels only
- a taxi carrying a combination of people and parcels

Since customer satisfaction has an impact on the service level, people requests have priority over parcels. By combining people and parcels together, the expected time of the trip will increase as a more time is required to pick up the parcel. According to a survey from uber [27], trip times within urban areas last about 14 minutes, thus only small time deviations are allowed by passengers.

Kamar et al. [34] describe the reasons why people prefer single rides than sharing with other people. These reasons are related to personal security and time convenience which thus makes people reject trips with strangers. For that reason, Li et al. [2] models SARP by taking into account 1 passenger per vehicle and trying to accommodate as much parcel as possible. An important parameter is that all passengers need to be satisfied, while parcel satisfaction is optimized. Since people and parcels have different types of service a different price is associated with them. If a taxi either stops too many times or exceeds the maximum ride time significantly it is assumed that the passenger will refuse to pay [2]. An illustration of SARP is shown by fig. 3.3 where the triangle points represent people and the circular points represent parcels. SARP assumes that a parcel can be served by at most one vehicle if served at all [2].

There are differences in the requirements between people and parcels. For example, the pick-up/delivery time windows of passengers are more critical. Detours of original routes can be too time consuming thus not acceptable by customers. Combinations of parcel pick-ups might not be possible due to vehicle capacity. Finally, the cost/benefits between people and parcels are different.

In terms of modelling perspective, every service request (people and parcels) is associated with a pick-up node and a delivery node. Therefore, we have a total set of requests σ which has (n) passenger requests and

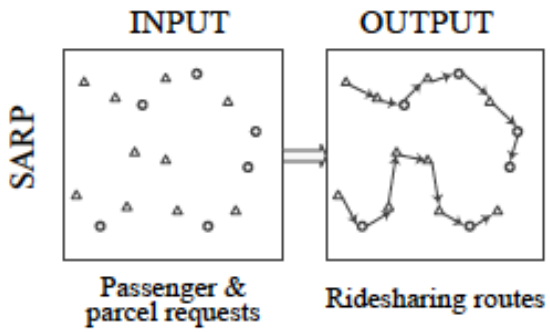

Figure 3.3: Example of a SARP network [2]

(m) parcel requests such that $n, m \in \sigma$. The model is defined in an undirected graph [2] $G = (V, E)$ where V is the set of nodes and E is the set of edges. As explained earlier there is a set of nodes for customers (V^p) and a set of nodes for parcels (V^f). Each set of nodes has a specific pick-up and delivery point such that $V^p = V_p^p \cup V_d^p$ and $V^f = V_p^f \cup V_d^f$. Additionally, there are starting and ending locations for taxis similar to the one described in DARP model. Therefore $V = \{V^p, V^f, \{0\}, \{2\sigma + 1\}\}$. In each transerse arc $(i, j) \in E$ there is an associated travel time (t_{ij}) and distance (d_{ij}). To avoid repetition, the rest of the problem is formulated in a similar way as the DARP in terms of the loads, vehicle sets, time windows and vehicle capacities.

Nevertheless, an interesting parameter in the formulation of the SARP by Li et al. [2] is the discount factor which is applied when passengers exceed their direct delivery time. This factor (γ_4) as denoted in the model, is located in the objective function and compares to the extra ride time of passengers with the direct delivery time.

Similarly, with the previous models SARP can be categorized as in static and dynamic. In the dynamic scenario passengers are accepted at the time of their call. These customers have a desired pick-up/delivery time window for and the associated locations. In the case where more than 1 taxi is available the one closest to the customer will be re-directed [2]. Thus, it is necessary to have a monitor that observes the states of the taxi in terms of capacity, location, loads etc. If feasible parcels are inserted in the route. In most cases, parcels are already known and thus are available for a dynamic insertion [2].

Fleet assignment problem

The FAP is a variation of the VRP applied in the airline industry where aircrafts are assigned to the scheduled flights. With the recent advances in technology researchers are integrating the FAP together with the scheduling problem since these two have the highest contributions on the revenues [59]. The problem's formulation is typically a mixed integer program which is based on the flight network of the airline. The network can be constructed using an arc based formulations where arcs represent connections or a Time Space Network (TSN) approach where arcs represent flight legs. Sherali et al. [59] define a flight leg as an airport to airport segment starting at a specific time connecting a flight. Both approaches are similar in terms of executing the main constraints however due to their different interpretations the mathematical formulations are different.

A connection network structure was first introduced by Abara [4] and nodes represent time points of the flights departures or arrivals. Additionally a source and sink node are created to account for the beginning and end day effects. Arcs are categorized between *flight connection*, *terminating* and *originating*. Flight connection arcs connect the departure with the arrival nodes, terminating arcs connect the arrival nodes with the sink node and finally originating arcs connect the departure nodes with the source node. This approach is similar to PDP, DARP and SARP formulations with however some fundamental differences due to the nature of the problem. An illustration of a connection network structure representing 12 feasible connection in a station is given by fig. 3.4.

In his model Abara defines a set of flight legs (L), a set of fleet (F) and finally a set of stations denoted by (S). A flight leg is associated with indexes i and j such that $(i, j \in F)$. When the indexes $(i, j = 0)$ it denotes the originating and terminating arcs respectively. Each aircraft (f) belongs in the set of fleet such that $f \in F$ and each station $(s \in S)$. It is defined as a profit maximization problem by taking the difference between revenues and costs.

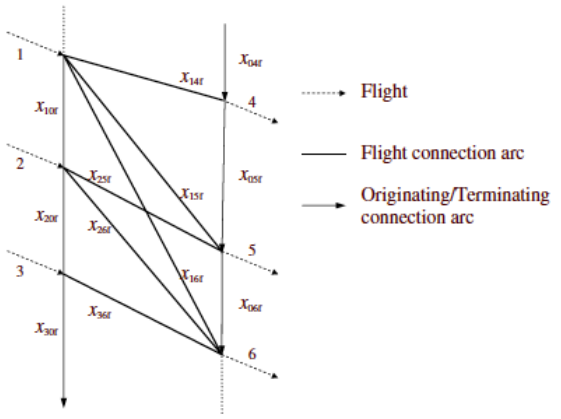

Figure 3.4: Representation of a connection network structure [59]

On the other hand a TSN approach is focused on flight leg representation. This allows the model to decide for the connections while respecting the feasibility in both time and space [59]. By doing so, there is a higher freedom for connections while the number of decision variables is reduced since the number of flight legs is less than the number of connections. Arcs are categorized between *ground*, *flight* and *overnight*. Ground arcs represent the aircraft staying at the same location for a specific time period, flight arcs represent the flight legs and overnight arcs are used to connect the last with the first events of the day. They are essential for the continuity of the fleet during a day of operations. A TSN is superimposed as as set of networks for every aircraft. A node in TSN is defined as an arrival or departure of an aircraft at every airport. Each node has an associated time to it. An illustration of a TSN is shown fig. 3.5, where it considers two stations for two different aircrafts specified as type 1 and 2. Flight arcs are depicted using the black line and black dotted line depending on the aircraft while nodes are representation using black dots.

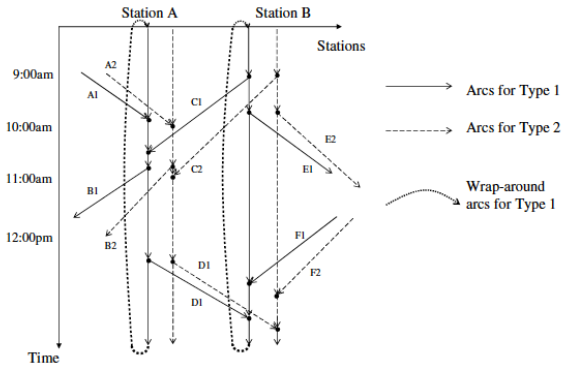

Figure 3.5: Time space network representation [59]

3.2. Modelling approach

The UAM network will be constructed using graph theory similar to the models described in section 3.1.2. Nodes will will consist of only vertiports while Arcs will be *directed* and represent flight paths. The main question that arises is which modelling approach should be used for a UAM network; a connection network structure or a TSN approach. Most transportation problems discussed above use a connection network structure even if the nature of the problem is dynamic. This is done by assigning a new decision variable that takes time into account, however the state of the vehicles is not shown. Since e-VTOLs will make decisions during real-time of operations a TSN approach is preferred. These decisions are:

- e-VTOLs flying from one vertiport to another while transporting passengers. In this state, vehicles either reschedule to a new route considering new requests or fly as initially planned.
- Empty e-VTOLs are rebalanced from one vertiport to another to serve a potential demand there.

- e-VTOLs are idle at the vertiport to fully recharge
- e-VTOLs decisions are delayed for the next time instance meaning they are idled at a vertiport without operating.

Shihab et al. [60], constructed a UAM model with a TSN approach for that reason. The time interval was set to 30 min due to the minimum time it took to go from one vertiport to another. The movement of vehicles occurred with 2 different types of arcs; flight and ground. e-VTOLs that are staying idle or recharging are represented by ground arcs. On the contrary flight arcs represent the flow from traversing from one vertiport to another. This is a similar approach as seen in the FAP [59] however modified for a UAM network.

Mahmoudi et al. [39], modelled the PDP with time windows using TSN. The paper used a multi-commodity network flow model that was based on the integration of vehicles which were carrying states within a TSN. Such approach helped to optimize simultaneously the passengers to vehicle assignment and the turnaround routing times. The usage of a 3D TSN helped to enumerate transportation states at any moment in time together with the vehicles paths in both space and time. Kliewer et al. [37] used a TSN approach to solve the multiple depot bus scheduling problem. The usage of TSN lead to a important size reduction when compared to a connection based network, flow or set partitioning model. Such modelling approach enabled to solve the problem with large scale of instances and thousands of scheduled trips.

Nevertheless concepts from the above described models can still be incorporated in UAM. For instance, the discount factor expressed in the SARP, can be used in our model as a compensation for the customers that detour and are delayed over a certain period of time. In addition, the same model suggested that people have priority over parcels and thus this could imply that direct passengers have priority over indirect. Finally TW could be incorporated from the DARP model in such a way that customers will specify a desired TW for pick-up or delivery.

3.2.1. Objective function

A major criteria in transportation problems is the Objective Function (OF) which depicts what the mathematical model is trying to minimize or maximize. Depending on the problem formulation there is a large variation of OF. The most frequently used ones are listed below [55]:

1. **Minimize Duration**: This OF consists of minimizing the total time that a vehicle requires to execute the route. It includes a) travel and b) waiting times c) loading and unloading of passengers/ goods.
2. **Minimize Completion time**: Route completion time is defined as time it takes to complete a specific service at the last location. In case the vehicle start time is fixed this OF is considered very similar with minimizing the duration of the trip.
3. **Minimize travel time**: This OF strictly relates to minimize the time spent on the vehicle.
4. **Minimize route length**: This OF relates to minimizing the traveling distance.
5. **Minimize customer dissatisfaction**: This OF is measured by taking into account the time deviations between the actual and desired pick-up or delivery time of the customers. Such a parameter can be modelling using either linear or non linear functions.
6. **Minimize vehicle number**: Minimizing number of vehicle directly relates to cost minimization. However such OF can have an impact on customer satisfaction services as less vehicles could lead to higher waiting times.
7. **Maximize profit**: Lastly, profit maximization is considered by taking the difference between revenues and costs. In profit maximization functions it is easier to reject customer requests if its determined that their profit contribution is negligible or not contribute much to the profit of the company when it comes to transporting unfavorable customers.

In dynamic environments it is not always clear what OF to use. This is because several routes in the next time point might be re-adjusted as a new request can appear or a customer ends his request. This indicates that the first 4 OF have no clear meaning or purpose. Savelsbergh et al. [55] indicates that dynamic problems should emphasize on metrics which affect the near future more than the entire time horizon.

A profit maximization problem is dependant on the revenues thus the fares that the passengers are charged with. In the case of UAM it was determined in section 2.5 that fares vary dynamically depending on demand or a simple fare model can be integrated for the purpose of this study.

3.2.2. Constraints

In optimization, constrains are divided in 2 categories; *hard* and *soft* constraints. The first, relates to the of metrics that must be satisfied while the latter relates to penalization of the OF if these conditions are not met. Depending on the nature of the problem and its formulation constraints are adjusted in every problem. The most applicable ones are defined below.

Demand related constraints

Such constraints limit the passenger flow from vertiport i to j. They ensure that the number of available seats in a given route are determined by the product of the capacity of the vehicles and the number of dispatched vehicles in the given arc at a particular time point. Such constraints are used in airline models as seen in [12] and in a UAM model [60].

Vehicle and vertiport capacity

Vehicle capacity constraints can be categorized as capacitated or uncapacitated in a dynamic formulation [51]. Uncapacitated vehicles are referred to as the ones that their capacity is infinite, thus can accommodate all passengers, however such a scenario is not realistic. In most problem formulations such as Li et al. [2] and Shihab et al. [60] vehicles are capacitated. The vertiport capacity is related to the fleet assignment problem [59]. Once a vertiport reaches its capacity limit, a vehicle would either reject a customer request or land in a nearby vertiport.

Energy constraints

Energy constraints guarantee that the vehicle has enough energy to transverse arc (i, j). Shihab et al. [60] enforced that all of the operating e-VTOLs are fully charged. Bongiovanni et al. [13] modelled the DARP with e-VTOLs and thus considered in their constraints battery management, detours to charging stations and charging times of the vehicles.

Time related constraints

Time related constraints are used in dynamic situations and they are normally implied as soft TW or no TW constraints. A hard TW constraint is difficult to satisfy with certainty unless there are infinite number of vehicles or customers are rejected. On the contrary, soft TW can be either one sided or two-sided and a penalization in the objective function is given if the TW is violated. The rest of time related constraints are incorporated for improving the quality of service. These constraints include a mixture of TW such as the inner TW is soft while the outer is hard etc [51]. Other time related constraints include maximum ride time limitations or defining a maximum route length [2]. Even though time related constraint increase the complexity of the problem, they also limit the solution space and thus they can be beneficial.

3.3. Dynamic customer requests

Passengers who request an immediate transport between 2 vertiports within a time point without being part of the static problem are considered as 'unexpected customers'. This is a particular case of a dynamic system behavior because their request will have to be accepted or rejected in a reasonable amount of time. The main reasons behind this are the following:

- eVTOLs will not be stationary at vertiports for a long time waiting for customers as we would like to achieve as low turnaround time as possible;
- Customer satisfaction should remain as high as possible. Adjusting unexpected customers into a route will cause delays and deteriorate customer satisfaction.

To model dynamic customer requests Coslovich et al. [20] suggests the following approach. A set of customers C is split into 2 subsets C^s and C^d such that $C = C^s \cup C^d$. The first subset refers to the static customers, the ones who have booked a trip in advance while the latter to the unexpected customers whose elements remain unknown.

The static set C^s is further split into 2 different sets C_P^s and C_D^s such that $C^s = C_P^s \cup C_D^s$. The first subset, refers to customers who have a desired pick-up time window. This includes groups who do not want to start their trip earlier than a specific time (Pick-up Oriented Customers (POC)). Subsequently, the latter subset refer to customers who have a desired delivery time window. This includes groups than want to be in their destination no later than a specific time point (Delivery Oriented Customers (DOC)). To put this in context, it could refer to students leaving and going to school as they have to be there at a specific time while they know the time that school starts. Since some unexpected customers will be able to be accepted in the operational model, it is important to measure the Level of Dissatisfaction (LOD) of the customers who required a service in advance. To do that, a linear model is used as described by eq. (3.1)-eq. (3.3). The idea behind it, is by using weight factor coefficients $\alpha_{p,d} = 0.5$ and $\beta = 0.5$ to determine the deviation between the desired pick-up or delivery time against the actual times, while taking into account the minimum time it takes to go from location i to j. For the case of the unexpected customers no time deviations are accounted since the requests are unknown.

$$LOD_i^{POC} = \alpha_p \cdot DSV_i + \beta \cdot ERT_i = \alpha_p(t_i^p - \hat{t}_i{}^p) + \beta(t_i^d - t_i^p - d(p_i, d_i)) \forall i \in C_P^s \tag{3.1}$$

$$LOD_i^{DOC} = \alpha_d \cdot DSV_i + \beta \cdot ERT_i = \alpha_d(\hat{t}_i{}^d - t_i^d) + \beta(t_i^d - t_i^p - d(p_i, d_i)) \forall i \in C_D^s \tag{3.2}$$

$$LOD_i^x = \beta \cdot ERT_i = \beta(t_i^d - t_i^p - d(p_i, d_i)) \forall i \in C^D \tag{3.3}$$

The ultimate objective is to account for as many unexpected passengers as possible as this could result in a higher profit. According to Coslovich et al. [20], the following rules/ constraints are applied to unexpected customers:

- Unexpected customers are served using the First-Come-First-Serve (FCFS) rule.
- LOD of static customers should be minimized.
- The deviations of Desired Service Time (DSV) of each customer should not exceed a specific upper bound.
- the deviations of Excess Ride Time (ERT) of each customer should not exceed a specific upper bound.
- Finally, the ERT of the unexpected customers shouldn't exceed a given upper bound.

When an unexpected customer j has been accepted by the network, the sets seen before are being updated by removing the customer j from the C^d subset and adding him to the C^s subset. Mathematically this translates to: $C_P^s = C_P^s \cup \{j\}$ and $C^d = C^d \backslash \{j\}$. By accepting at least one customer, either a pre-planned route will change or a new vehicle will be dispatched. A change of the pre-planned route can be visualized by fig. 3.6.

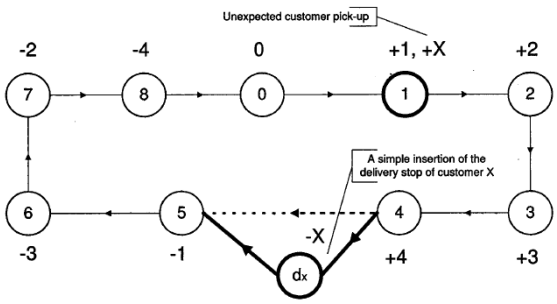

Figure 3.6: Insertion of a customer in a pre-existing route [20]

Customer satisfaction can be evaluated using average pick-up and delivery delays as shown in eqs. (3.4) and (3.5) [29].

$$\text{Average pickup delay time} = \frac{\sum_{i \in P}(\text{Actual pickup time} - \text{expected pickup time})}{\text{number of customers}} \tag{3.4}$$

$$\text{Average delivery delay time} = \frac{\sum_{i \in D}(\text{Actual delivery time} - \text{expected delivery time})}{\text{number of customers}} \tag{3.5}$$

3.3.1. Customer rejection

Customer rejections are usually undesirable but in a dynamic context could be inevitable. This is due to limitations of the resources (vehicle capacity) or with the existence of hard TW. Thus customer rejections are related to the formulation of time related constraints. Most of literature on dynamic VRP does not allow customer rejections [51]. In a UAM model new requests are inserted at random time periods and customer rejection is allowed if:

- Vehicles reach their maximum capacity.
- It is not profitable to re-adjust the route or rebalance a vehicle to pick up a new customer.
- Vehicles that transport 'premium' customers are not able to ride-share with other customers.

3.4. Vehicle Rebalancing

A key problem that arises with on demand transportation is that sometimes vehicles are imbalanced. For example, during morning transportation hours, there is a higher amount of demand from the sub-urban areas towards the downtown as explained in section 2.5. This can have an impact on the distribution of the e-VTOLs as it limits their availability in areas with potential demand in the next time steps. Vehicle re-balancing has an impact on the operating costs of the network as they vehicle flies empty to accommodate a potential demand in another vertiport. This is taken into account in the OF of the UAM model.

Iglesias et al. [30] modelled a fleet of autonomous vehicles using a time-expanded network and computed the optimal rebalancing strategies using preemptive re-positioning. This method outperformed state-of-the-art rebalancing strategies as it reduced waiting times by 89.6 %. In the model it was assumed that perfect information is available for future customer requests and the rebalancing occurred to empty vehicles from locations that had a surplus number of vehicles and reaching the capacity of the depot. Idling vehicles were considered as a special case and this is because they were not rebalanced but instead recharged or waited until the next time point.

Wallar et al. [68] developed a model for rebalancing idle vehicles in ride sharing applications. The model was split in 2 sub-problems i) to estimate real-time demand for every region in manhattan and ii) to optimize the assignment of idle vehicles using integer linear programming. They used a pro-active rebalancing strategy over the operated area to match the demand distribution. This adaptation decreased the waiting times of customers as the probability of having an available taxi in a nearby area was higher. For the exact implementation of the model please refer to Wallar's paper. The results of simply rebalancing idle vehicles depicted a decrease in average waiting time by 17 % while the travel delay was decreased by 86 %.

Lastly, Pavone et al. [47] solved the rebalancing problem using a robotic approach. A fluid model was used for both customers and vehicles. The rebalancing policy operated under stochastic customer demand. The policy works by allowing every 'stop' to reach a certain equilibrium with vehicles. If the equilibrium is exceeded and there are no waiting customers the vehicles are rebalanced which was solved using a linear program. For a deeper understanding of the algorithms methodology readers are referred to [47].

It is evident that rebalancing strategies are minimizing waiting times and trip delays. Such adaption in the UAM model can be beneficial especially when taking into account dynamic customers. All authors use a different rebalancing strategy which performs in a similar manner. Thus it is reasonable to assume that the most suited rebalancing strategy depends on the formulation of the problem.

4

Solution approaches

In this chapter we present different solutions approaches to transportation problems to understand which one is the most appropriate for this application. In section 4.1 exact solutions are being analysed and in section 4.2 approximate solutions (heuristics).

4.1. Exact solutions

Exact solutions find an optimal solution to the problem. They are mainly inclined for static and deterministic problems but several authors have attempted to use them in a dynamic context such as [56]. The most common exact algorithms are Branch and Bound (B&B) and Dynamic Programming (DP).

4.1.1. Branch and bound

B&B algorithms are particularly useful for determining an optimal solution. Its applications are particularly noted in integer, nonlinear programming but also problems such Vehicle Routing Problem (VRP) and quadratic assignment problems. These algorithms work by branching the solution set into subsets and determine the best possible solution. They are normally classified as:

- **Branch and cut (B&C)**: In this case, cutting planes are added to the problems tree. The additional cuts cause LP-relaxation of the problem which assists in finding integer solutions and verify the problems optimality.
- **Branch and Price (B&P)**: This approach focuses on the 'column generation' instead of generating cuts to relax the LP in the B&B procedure. Column generation is particularly suited to tackle large scale instancing problems. This is because the problem is split in the *'master'* problem and the *sub-problem'*. The master problem has an excluded set of columns to reduce the computational intensity. By solving the 'pricing sub-problems' columns are generated and added to the master problem to improve the problem relaxation.
- **Branch and Price and Cut (B&PC)**: This approach is a combination of the B&C and B&P where the algorithms take advantage of both methods. This is done at first by the column generation technique and by improving the LP-relaxation using cutting planes.

The first B&B algorithm to address the Dial A Ride Problem (DARP) was addressed by Cordeau et al. [17] and applied valid inequalities including *subtour elimination* and *generalized order constraints* as cuts for a 3-index formulation. A 2-index formulation of the Pick-up and Delivery Problem (PDP) was introduced by Ropke et al. [53] and incorporated Cordeau [17] cuts and new cases of valid inequalities. These included *strengthened capacity*, *strengthened infeasible path* and *fork constraints*. Shihab et al. [60], address the Urban Air Mobility (UAM) scheduling problem as a mixed integer quadratic programming using a 3-index formulation and proposes B&B methods to solve it.

Chen et al. [15] solved a dynamic VRP with hard time windows using column generation algorithms. The customer requests were obtained randomly over time within the specified Time Window (TW). The Objective Function (OF) was to minimize the total travel length and the algorithm was generating single vehicle routes

(columns) over time. This was done with the utilization of existing columns and by solving the sub-problem at every decision time step using the generated columns up to that point.

Christiansen et al. [16] solved the capacitated VRP with a stochastic demand. It was formulated using a set-partitioning approach and was proven that the sub-problem can be solved using dynamic programming.

4.1.2. Dynamic programming

DP is a mathematical technique used to make a sequence of interrelated decisions and determine their optimal combination. In DP there is not a particular mathematical formulation, but it adjusts to every problem differently.

The characteristics of DP according to [26] are the following. The problem is divided into 'stages' (column of nodes) with a 'policy decision' (node links) at every stage. Every stage has a associate state (node) which can either be finite or infinite. States are related to the conditions of the system. The policy decision ensures the transformation of the current state with the beginning of the next stage, according to a probability distribution. The goal is to determine an optimal policy at each stage for all states which is independent of the policy decisions adopted in previous stages. The solution starts by determining the optimal policy for the last stage until it finds a solution for the initial stage. This is achieved using a 'recursive relationship. DP can be categorized as deterministic or stochastic. Deterministic DP is when the state at the next stage is completely determined by the state and policy decision at the current state. On the contrary a probabilistic DP uses a probability distribution to determine what the next stage will be.

Schmid et al. [56] solved the dynamic ambulance relocation and dispatching problem using approximate DP. By comparing the model with real life data it was shown that the algorithm provided high quality solutions. [50] solved the DARP problem by considering single vehicles and many to many immediate requests with DP. The paper first focused on the static case and then consider a dynamic environment. For the dynamic problem the algorithm was optimizing only over the known inputs and didn't take into account future customers. Even though the results converged to an optimal solution, the computational times were higher than a heuristic based method.

Finally, Secomandi et al. [57] solved the dynamic VRP where customer demands was unknown using neuro-DP. The paper compared 2 different types of neuro-DP algorithms referred as 'optimistic approximate policy' and 'roll-out policy'. The first approach improved the nearest-neighbor policy performance by 2.3 % while the latter showed better quality results.

4.2. Approximate solutions

In this section we look at another approach to solve scheduling transportation problems. Meta-heuristic methods are effective in solving problems with large instances and finding satisfactory solutions while have low computational times.The most famous meta-heuristics algorithms used in transportation scheduling problems are Simulated annealing (SA), Tabu Search (TS), Large Neighborhood Search (LNS) and Genetic Algorithm (GA).

4.2.1. Simulated annealing

SA is one of the simplest meta-heuristics to solve a combinational optimization problem while providing good quality solutions [26]. The algorithms simulates the annealing process of materials and was first introduced by [35]. It is a Local Search (LS) algorithm that looks for the best solution by moving from a current trial solution to a intermediate neighbor of this solution. This is done according to an acceptable probability that deteriorates with the search evolution. The selection process for the intermediate neighbor in a maximization problem is as follows [26]:

- If the new solution (Z_n) is higher than the current (Z_c) then that candidate is always accepted.
- Otherwise the candidate is accepted with probability e^x. Where $x = \frac{Z_n - Z_c}{T}$

For a minimization problem, the order of Z_n, Z_c is reversed. T is a parameter called 'current temperature' and changes throughout the search while simulating the cooling process of metals [5]. For a deeper understanding of the working principles of the SA please refer to [26, 49]. Nevertheless, the outline of a basic SA algorithm consists of the following steps [26]:

1. **Initialization:** Starts with any possible feasible initial solution.

2. **Iterations:** The selection rule is used to selected next trial solutions.

3. **Check the temperature schedule:** When the desired number of iterations is performed at T, the value of T is decreased to perform iterations at the next value.

4. **Stopping rule:** When the desired number of iterations has been reached at the smallest T value iterations are stopped. The best trial solution is accepted as the final one.

Afifi et al. [5] used a SA algorithm to solve the VRP with TW and synchronization constraints. The different instances that were incorporated in that particularly study showed that the SA algorithm was faster and outperformed the existing solution approaches. It is often the case that SA is begin combined with other meta-heuristic algorithms to obtain better quality solutions. For instance, [45] solves the PDP with TW using a combination of TS and SA referred as 'reactive TS'. The results proven to be accurate as they returned solutions within 1 % of the optimal value in a fraction of time in comparison to other algorithms.

4.2.2. Tabu search

TS is one of the most popular and used meta-heuristic algorithms. It was firstly proposed by [24] and similarly to SA it is also a LS method. A tabu list ensures that the solution is not 'stuck' in the local minima. It records the recent history of events and the moves which are forbidden to be executed in the following iterations. When the tabu list is full, its oldest member is deleted allowing flexibility in the upcoming iterations [26]. It consists at moving from a solution to the best neighbor at every iteration even if this causes deterioration to the OF [26]. TS can have more advance concepts as described in [49]. However, the outline of a basic TS algorithm is as follows [26]:

1. **Initialization:** Start with any possible feasible initial solution.

2. **Iterations:** A local strategy is used to define a move into the local neighborhood of the current solution. Any move from the tabu list is eliminated unless it is a better solution that the best trial solution so far. The remaining moves determine which one will provide the best solution. This solution is accepted regardless if its better or worse than the previous solution. The tabu list is updated to avoid going back to previous solutions.

3. **Stopping rule:** Stopping criteria can be defined as i) a fixed number of iterations, ii) fixed amount of CPU time, iii) fixed number of continuous iterations without improving the value of the objective function or iv) combinations from the above. The best solution from all iterations is used as the final one.

TS has been used widely in transportation problems, especially when the nature of the problem is dynamic [43]. For example, [18] has incorporated advance TS concepts with diversification strategies to solve the DARP. A diversification strategy forces the solution search in unexplored areas of the feasible regions. This is achieved by penalizing frequently made moves and temporarily accepting infeasible solutions. Such method is proven to provide effective and efficient results. A parallel TS implementation for the dynamic DARP is presented [8] where a high percentage of users is satisfied.

An improved TS method to solve the VRP is proposed by [32]. This was achieved by using mutating operators that affect the degree of the algorithm's performance. Furthermore, the neighborhood search was altered by considering random samples which significantly reduce the computational times. By adding randomness to the problem the tabu list can be shortened as it is easier to escape local minima. However, this can come at a cost of not finding a very optimal solution which is fixed by adding probabilities to activate tabu criteria. The improved TS showed that the quality of the solution compared to other algorithms was good aside of the size of the problem. It also showed stability in the results and fast computational times.

Finally, Gendreau et al. [23] suggested a TS in a VRP where new requests occur in real time and specify a pick-up/delivery location. The neighborhood structure was based on ejection chains and results depicted good quality solutions in a real-time application.

4.2.3. Large neighborhood search

LNS is classified as one of the most used meta-heuristic algorithms to solve transportation scheduling problems. It was first introduced by [58] and the initial solution is improved by 'destroying' and 'repairing' the current solution [49]. LNS algorithms are so popular because they observe possible solutions in very large neighborhoods, thus it is easier to find local optima of high quality. Therefore such algorithms can return better solutions. The working principle behind LNS is the **'destroy'** and **'repair'** function. The destroy functions works by destructing parts of the current solution stochastically so that different parts of the solution are destroyed at every iteration. For instance, a destroy function in a transportation problem could remove part of the customers that are being accommodated. The main parameter in the destroy function is the *'degree of destruction'* as it shows to what extend the part of the solution is destroyed. The repair function on the contrary rebuilds parts of the destructed solution by inserting the removed passengers using greedy heuristics. Thus, a destroy function makes the solution infeasible while the repair function returns it to a feasible one. For a deeper understanding of the destroy and repair functions in LNS readers are referred to [49].

The algorithm starts with an initial solution which is assumed as the best global solution. The destroy and the repair function are used to create a new solution which is then evaluated. It is then determined whether this new solution should be accepted or rejected. The 'accept' function is imposed in different ways where the simplest one accepted only improved solutions (hill-climber). Lately a lot of researchers have been involved with acceptance criteria inspired from SA [49]. Other acceptance criteria that perform well are record-to-record travel and threshold accepting. If the current solution is better than the previous solution, it replaces it. This process is done until the best solution is found. The basic outline of LNS consists of the following steps [49]:

1. **Initialization:** Starts with any possible feasible initial solution
2. **Iterations:** Using the destroy at first and then the repair function a new solution is obtained which is evaluated. If the solution is accepted and is better than the previous solution it replaces the old one.
3. **Stopping rule:** Stopping criteria can be defined as i) a fixed number of iterations, ii) fixed amount of CPU time, iii) fixed number of continuous iterations without improving the value of the OF or iv) combinations from the above. The best solution from all iterations is used as the final one.

A variant of LNS that is used to deliver high quality solutions when comparing to other methods is Adaptive Large Neighborhood Search (ALNS). It was introduced by [52] and allows multiple destroy and repair functions to operate at the same time (multiple neighborhood searches). Weights are assigned in both function and control the rate that they occur. These weights are adjusted dynamically meaning that the algorithm adapts to the problem's instance. The weight (ρ^-, ρ^+) for each function are the same at the initial phase but they gradually change using the 'roulette wheel principle' as seen in eq. (4.1).

$$\phi_j^- = \frac{\rho_j^-}{\sum_{k=1}^{|\Omega^-|} \rho_k^-} \tag{4.1}$$

At the end of every iteration a score ψ is given for both functions as seen eq. (4.2). In order to avoid rejecting solutions the ω values are set up in such a way that $\omega_1 > \omega_2 > \omega_3 > \omega_4$ where ω_4 is assumed to have a very low value [49].

$$\psi = \max \begin{cases} \omega_1 & \text{new global solution} \\ \omega_2 & \text{new solution better than current solution} \\ \omega_3 & \text{new solution accepted} \\ \omega_4 & \text{new solution rejected} \end{cases} \tag{4.2}$$

The weights are updated at the end of every segment using eq. (4.3) where λ is a decay parameter controlling the sensitivity of the weights. The basic outline of the ALNS is the same as LNS with including the extra parameters discussed above.

$$\rho_a^- = \lambda \rho_a^- + (1-\lambda)\psi, \quad \rho_b^+ = \lambda \rho_b^+ + (1-\lambda)\psi \tag{4.3}$$

As mentioned earlier [52] solved the PDP with TW using ALNS algorithms. *Shaw removal heuristics, random and worst removal heuristics* were used for the destroy functions whereas *basic greedy and regret heuristics* were used for the repair one. The reason for choosing all of the above mentioned heuristics methods

is that for one instance type one heuristic method can be more suited than another. The ALNS proved to be applicable to most problems while being effective and efficient due to its robust and fast construction heuristics.

Azi et al. [9] used ALNS algorithms for the VRP with multiple routes where new customer requests occurred dynamically. Multiple instances for future customer requests were considered on whether or not these requests should be included in the solution. It was demonstrated that using operators in customers, routes and workday levels is more beneficial than just using customer based operators.

4.2.4. Genetic algorithms

GA is another meta-heuristic approach which has a different concept from LS algorithms. Such algorithms are proven to be effective for exploring various parts of the feasible region and evolve towards the best feasible solution [26]. It was first introduced by [28] and are based on the theory of evolution, thus it is a bias process of natural selection. This is achieved by using 'genetic operators' such as selections, mutations and crossovers. The process begins with an initial population set referred as 'genes'. Each gene has a unique 'chromosome' which records a set of potential solutions. A fitness function is generated that produces a fitness score (solution quality) for each gene. A high fitness score is associated with a high probability of genes to be selected in the following phase called 'selection phase'. During this phase the genes with the highest score are being chosen as 'parents' to pass their genes in order to reproduce. Selection techniques vary between *Roulette wheel selection*, *rank* and *tournament selection*. The selected parents try to reach a randomly generated crossover point by exchanging genes while trying to achieve the 'crossover point'. The gene exchange can cause 'mutation' such that parts of the chromosomes change their probability to ensure that the population is not stuck in local minima. This procedure happens until the newly produced genes do not have a significant difference. When this occurs, it is said that the population has converged to a potential set of acceptable solutions. For a more detailed explanation of the GA concepts please refer to [26]. Nevertheless, the outline of the algorithm is as follows:

1. **Initialization:** Starts with an initial population which belongs to a feasible set of trial solutions. These can be generated randomly and the value of the OF (fitness score) is evaluated for each member.

2. **Iterations:** A bias random process is used towards the members with the highest fitness score that pair up to create the parents. The set of parents create a new set of feasible solutions that are mutated. When the mutation leads to infeasible solution sets it is referred as 'miscarriage'. Once a feasible set of solution is selected, a new population (iteration) occurs.

3. **Stopping rule:** Stopping criteria is defined as i) fixed number of iterations, ii) fixed amount of CPU time, iii) fixed number of continuous iterations without improvement in the OF or iv) combinations of the above. The best trial solution is chosen as the final value.

Barkaoui et al. [11] solved the dynamic VRP with GA. The problem included real time information with the use of variable travel times. These variables were updated with using a dynamic traffic simulation. The outcome showed a significant cost reduction when incorporating the dynamic VRP with real time data. The algorithm showed good performance and was able to compute the solution in a quick time.

Taniguchi et al. [62] solved the dynamic VRP with time windows using adaptive evolutionary algorithms. A comparison between the adaptive and the hybrid algorithm was made to compare the quality of the solutions and the robustness of the algorithms. As a result, the adaptive algorithm showed better performance than the hand-tuning ones.

Jorgensen et al. [33] solved the DARP using GA and routing heuristics. The algorithm was based on cluster-first and route-second approach and it was alternating between assigning customers to vehicles and solving the routing problem independently. While comparing the method to different meta-heuristics it was proven to provide good quality results at an efficient amount of time.

5

Conclusion

This literature study provides an extensive overview for the required tools to develop a dynamic request assignment in Urban Air Mobility (UAM). A UAM network can be seen as a combination of both ground and air transportation services with fundamental differences. This expresses the importance to develop operational models specifically designed for UAM. In this study, the dynamic request assignment is addressed under the profit maximization perspective. More particularly, the objective is to maximize the profit, consider customer satisfaction and accomodate dynamic customers when possible. The approach for this problem is i) to understand the operational constraints applied in UAM, ii) the relevant models to understand the similarities and differences and iii) finally to explore the solutions algorithms that are capable of providing efficient and effective solutions.

A UAM model is an interrelated model consisting of the vehicle, vertiport and demand model. For the application of this study the most suitable mission profiles were the short-range, large payload and long-range, large payload. The vehicle model should incorporate a homogeneous fleet of e-Vertical Take-Off and Landing (VTOL)s where the number of vehicles is dependant on the vertiport and demand model. For the vertiport model a hexagon pattern is found appropriate due to its easy adaptation in large metroplex areas. The model should consists of 7-8 or 13-14 vertiports depending on the mission profile that is chosen. 2 vertiports located in the downtown area while the rest will be located in a hexagon pattern around them. All vertiports should be equipped with charging stations while their capacity will vary according to the locations of the areas. Lastly, the demand model can be incorporated as a bimodal distribution to simulate high and low peak hours. Since the nature of the problem is dynamic, a small scheduling horizon of 3-6 hours is found appropriate.

The UAM scheduling problem was classified under the Vehicle Routing Problem (VRP) family. A connection network structure was found more appropriate Time Space Network (TSN) approach since we are interesting in modelling the exact times of customers. Dynamic customer requests should be served in a First-Come-First-Serve (FCFS) method and customer rejection is allowed if these customers have a negative contribution towards the profit.

Concerning the solution methods, both exact and approximate solutions were investigated. Approximate solutions (meta-heuristics) are found to be more applicable in this research since they provide a sub-optimal solution with low computational times. All meta-heuristic algorithms fit the problem's description however research suggested that combining elements between them can provide an overall better solution.

III

Supporting work

1

Appendix 1: Customer solutions under dynamic demand

Optimized customer solution under dynamic demand for instance 1:

cust_idx	origin	destination	etw_pu	etw	ltw	etw_factor	duration_factor	cust_type	premium	new_cust	cancelled	accomodated	B_i	D_i	W_i	A_i+s	B_i+s	L_i	s_i	c_il	d_i	c_if	k	premium_violation	tw_violation
1	3	0	7	7	7.3	0.3	0.7	0	0	0	0	1	7.0	7.0	0.0	7.3	7.3	0.2	1.0	58.0	0.0	58.0	7	0	0
2	0	1	7	7	7.3	0.7	0.3	0	0	0	0	1	7.0	7.0	0.0	7.3	7.3	0.2	1.0	59.1	0.0	59.1	5	0	0
3	3	1	7.1	7.1	7.2	0.9	0.1	0	1	0	0	1	7.1	7.1	0.0	7.5	7.5	0.4	1.0	134.6	0.0	134.6	4	0	0
4	3	0	7.2	7.2	7.8	0.6	0.4	0	1	0	0	1	7.5	7.5	0.3	7.7	7.7	0.2	1.0	78.4	0.0	78.4	3	0	0
5	2	3	7.2	7.2	7.4	0	1	0	0	0	0	1	7.2	7.2	0.0	7.5	7.5	0.2	1.0	59.1	0.0	59.1	3	0	0
6	0	2	7.3	7.3	7.9	0.9	0.1	0	0	0	0	1	7.6	7.6	0.3	7.8	7.8	0.3	1.0	59.4	0.0	59.4	1	0	0
7	6	0	7.3	7.3	7.8	0.5	0.5	0	0	0	0	1	7.3	7.3	0.0	7.6	7.6	0.3	1.0	58.3	0.0	58.3	1	0	0
8	0	3	7.4	7.4	7.7	0.2	0.8	0	0	0	0	1	7.4	7.4	0.0	7.7	7.7	0.2	1.0	58.0	0.0	58.0	7	0	0
9	3	1	7.5	7.5	8.1	0.3	0.7	0	0	0	0	1	7.8	7.8	0.3	8.2	8.2	0.4	1.0	99.8	0.0	99.8	8	0	0
10	4	3	7.5	7.5	7.7	0.1	0.9	0	1	0	0	1	7.5	7.5	0.0	7.8	7.8	0.2	1.0	79.9	0.0	79.9	8	0	0
11	5	0	7.5	7.5	8	0.2	0.8	0	0	0	0	1	7.7	7.7	0.2	8.0	8.0	0.3	0.9	59.8	3.0	56.8	6	0	0
12	6	3	7.5	7.5	8.1	0.4	0.6	0	0	0	0	1	7.7	7.7	0.2	8.4	8.4	0.6	0.8	121.1	12.1	109.0	9	0	0
13	3	6	7.5	7.5	8	0.9	0.1	0	0	0	0	1	7.8	7.8	0.3	8.4	8.4	0.6	0.9	121.1	6.1	115.0	8	0	0
14	5	2	7.5	7.5	7.9	0.3	0.7	0	0	0	0	1	7.7	7.7	0.2	8.2	8.3	0.5	0.9	118.9	5.9	113.0	6	0	0
15	0	2	7.6	7.6	7.9	0.5	0.5	0	0	0	0	1	7.7	7.7	0.1	8.0	8.0	0.2	1.0	59.1	0.0	59.1	3	0	0
16	5	0	7.7	7.7	8.3	0.9	0.1	0	0	0	0	1	7.7	7.7	0.0	8.0	8.0	0.3	1.0	59.8	0.0	59.8	6	0	0
17	6	2	7.7	7.7	8.1	0.8	0.2	0	0	0	0	1	7.7	7.7	0.0	8.1	8.1	0.4	1.0	99.8	0.0	99.8	9	0	0
18	3	6	7.8	7.8	8.3	0.3	0.7	0	1	0	0	1	7.8	7.8	0.0	8.2	8.2	0.4	1.0	154.0	0.0	154.0	7	0	0
19	5	1	7.9	7.9	8.1	0.9	0.1	0	0	0	0	1	7.9	7.9	0.0	8.3	8.3	0.4	1.0	101.4	0.0	101.4	10	0	0
20	1	0	7.9	7.9	8.1	0.2	0.8	0	0	0	0	1	7.9	7.9	0.0	8.2	8.2	0.2	1.0	59.1	0.0	59.1	4	0	0
21	6	0	7.9	7.9	8.1	0.4	0.6	0	0	0	0	1	7.9	7.9	0.0	8.2	8.2	0.2	1.0	58.0	0.0	58.0	11	0	0
22	6	0	8	8	8.5	0.8	0.2	0	0	0	0	1	8.4	8.4	0.4	8.7	8.7	0.3	1.0	58.3	0.0	58.3	11	0	0
23	2	3	8	8	8.5	0.3	0.7	0	0	0	0	1	8.3	8.3	0.3	8.6	8.6	0.3	0.9	59.8	3.0	56.8	3	0	0
24	6	3	8.1	8.1	8.7	0.3	0.7	0	1	0	0	1	8.4	8.4	0.3	8.8	8.8	0.4	1.0	154.0	0.0	154.0	4	0	0
25	5	2	8.1	8.1	8.8	0.7	0.3	0	0	0	0	1	8.5	8.5	0.4	9.1	9.1	0.6	0.9	123.5	6.2	117.3	12	0	0
26	1	0	8.2	8.2	8.5	0.4	0.6	0	1	0	1	0													
27	0	5	8.2	8.2	8.8	0.7	0.3	0	0	0	0	1	8.2	8.2	0.0	8.5	8.5	0.3	1.0	59.8	0.0	59.8	12	0	0
28	2	4	8.2	8.2	8.6	0.4	0.6	0	0	0	0	1	8.3	8.3	0.1	8.8	8.8	0.5	0.8	105.9	10.6	95.3	3	0	0
29	3	0	8.2	8.2	8.9	0.5	0.5	0	0	0	0	1	8.8	8.8	0.6	9.1	9.1	0.3	0.9	58.3	2.9	55.4	2	0	0
30	3	0	8.3	8.3	8.9	0.1	0.9	0	0	0	0	1	8.8	8.8	0.5	9.1	9.1	0.3	1.0	58.3	0.0	58.3	2	0	0
31	5	3	8.4	8.4	8.5	0.3	0.7	0	1	0	0	1	8.4	8.4	0.0	8.8	8.8	0.4	1.0	134.6	0.0	134.6	2	0	0
32	6	3	8.4	8.4	8.6	0.8	0.2	0	0	0	0	1	8.4	8.4	0.0	8.9	8.9	0.5	1.0	116.3	0.0	116.3	11	0	0
33	3	4	8.4	8.4	8.8	0.4	0.6	0	0	0	0	1	8.5	8.6	0.1	8.8	8.8	0.3	0.9	59.8	3.0	56.8	3	0	0
34	3	4	8.5	8.5	8.8	0.7	0.3	0	0	0	0	1	8.7	8.7	0.2	8.9	8.9	0.2	1.0	59.1	0.0	59.1	9	0	0
35	6	1	8.6	8.6	9.2	0.5	0.5	0	0	0	0	1	8.6	8.6	0.0	8.9	8.9	0.2	1.0	59.1	0.0	59.1	13	0	0
36	1	5	8.7	8.7	9.4	0.5	0.5	0	1	0	0	1	8.9	8.9	0.2	9.3	9.3	0.4	1.0	137.2	0.0	137.2	7	0	0
37	6	0	8.8	8.8	9.5	0.8	0.2	0	0	0	0	1	8.8	8.9	0.0	9.1	9.1	0.2	1.0	58.0	0.0	58.0	8	0	0
38	2	5	8.8	8.8	9.3	0.9	0.1	0	0	0	0	1	8.8	8.8	0.0	9.2	9.2	0.4	1.0	116.0	0.0	116.0	1	0	0
39	6	1	8.9	8.9	9.5	0.3	0.7	0	1	0	0	1	8.9	8.9	0.0	9.2	9.2	0.2	1.0	79.9	0.0	79.9	15	0	0
40	1	4	8.9	8.9	9.1	0.4	0.6	0	1	0	0	1	8.9	8.9	0.0	9.3	9.3	0.4	1.0	157.0	0.0	157.0	5	0	0
41	0	6	9	9	9.3	0	1	0	0	0	1	0													
42	5	0	9.2	9.2	9.9	0.4	0.6	0	0	0	0	1	9.2	9.3	0.0	9.5	9.5	0.3	1.0	59.4	0.0	59.4	1	0	0
43	2	0	9.4	9.4	9.9	0.8	0.2	0	1	0	1	0													
44	3	0	9.5	9.5	10	0.1	0.9	0	1	0	0	1	9.5	9.5	0.0	9.8	9.8	0.2	1.0	78.4	0.0	78.4	16	0	0
45	1	4	9.6	9.6	9.7	0.3	0.7	0	0	0	0	1	9.6	9.6	0.0	10.0	10.0	0.4	1.0	116.0	0.0	116.0	10	0	0
46	3	5	9.6	9.6	10	0.1	0.9	0	0	0	0	1	9.6	9.6	0.0	10.0	10.0	0.4	1.0	99.4	0.0	99.4	4	0	0
47	1	0	9.7	9.7	10.1	0.8	0.2	0	0	0	0	1	10.0	10.0	0.3	10.3	10.3	0.2	1.0	59.1	0.0	59.1	16	0	0
48	2	1	9.8	9.8	10.3	0	1	0	1	0	1	0													
49	5	1	9.8	9.8	10.2	0.6	0.4	0	0	0	1	0													
50	3	0	9.9	9.9	10.1	1	0	0	0	0	0	1	9.9	9.9	0.0	10.2	10.2	0.3	1.0	58.3	0.0	58.3	11	0	0
51	0	1	10	10	10.7	0.6	0.4	0	0	0	0	1	10.4	10.4	0.4	10.6	10.6	0.3	1.0	59.4	0.0	59.4	3	0	0
52	2	0	10	10	10.7	0.3	0.7	0	0	0	0	1	10.0	10.0	0.0	10.3	10.3	0.2	1.0	59.1	0.0	59.1	6	0	0
53	1	0	10.1	10.1	10.6	0.6	0.4	0	0	0	0	1	10.4	10.4	0.3	10.7	10.7	0.2	1.0	59.1	0.0	59.1	11	0	0
54	5	3	10.6	10.6	10.7	0.5	0.5	0	0	0	0	1	10.6	10.6	0.0	11.0	11.0	0.4	1.0	99.4	0.0	99.4	4	0	0
55	1	5	10.7	10.7	11.1	0.7	0.3	0	0	0	0	1	10.7	10.7	0.0	11.1	11.1	0.4	1.0	101.4	0.0	101.4	13	0	0
56	6	2	10.7	10.7	11.3	0.5	0.5	0	0	0	0	1	11.1	11.1	0.4	11.6	11.6	0.5	0.9	104.1	5.2	98.9	1	0	0
57	0	2	10.7	10.7	10.8	0.9	0.1	0	0	0	0	1	10.7	10.7	0.0	11.0	11.0	0.2	1.0	59.1	0.0	59.1	14	0	0
58	0	4	10.8	10.8	11.1	0.8	0.2	0	0	0	0	1	10.8	10.8	0.0	11.1	11.1	0.2	1.0	59.1	0.0	59.1	6	0	0
59	1	0	10.9	10.9	11.5	0.3	0.7	0	1	0	0	1	10.9	10.9	0.0	11.2	11.2	0.2	1.0	79.9	0.0	79.9	7	0	0
60	5	0	10.9	10.9	11.4	0.9	0.1	0	1	0	0	1	11.3	11.4	0.4	11.6	11.6	0.2	1.0	79.9	0.0	79.9	12	0	0
61	6	0	10.9	10.9	11.3	0.9	0.1	0	0	0	0	1	10.9	10.9	0.0	11.4	11.4	0.5	1.0	65.3	0.0	65.3	1	0	0
62	3	1	11.1	11.1	11.2	0.6	0.4	0	0	0	0	1	11.1	11.1	0.0	11.6	11.6	0.5	0.9	103.8	5.2	98.6	4	0	0
63	5	4	11.1	11.1	11.2	0.7	0.3	0	0	0	1	0													
64	6	0	11.1	11.1	11.6	0.4	0.6	0	0	0	0	1	11.1	11.1	0.0	11.4	11.4	0.3	1.0	58.7	0.0	58.7	1	0	0
65	2	0	11.4	11.4	12.1	0.3	0.7	0	0	0	0	1	11.6	11.6	0.2	11.9	11.9	0.2	1.0	59.1	0.0	59.1	1	0	0
66	3	6	11.5	11.5	12	0.4	0.6	0	0	0	0	1	11.5	11.5	0.0	11.9	11.9	0.4	1.0	113.7	0.0	113.7	2	0	0
67	1	5	11.7	11.7	11.9	0.6	0.4	0	1	0	0	1	11.7	11.7	0.0	12.1	12.1	0.4	1.0	137.2	0.0	137.2	3	0	0
68	2	5	11.8	11.8	12.4	0.4	0.6	0	0	0	1	0													
69	5	0	11.8	11.8	12.3	0.9	0.1	0	0	0	0	1	11.8	11.8	0.0	12.1	12.1	0.2	1.0	59.1	0.0	59.1	13	0	0
70	6	0	12	12	12.4	0.4	0.6	0	0	0	1	0													
71	5	1	8.3	8.3	8.8	0.5	0.5	0	0	0	0	1	8.5	8.5	0.2	8.9	8.9	0.4	1.0	102.1	0.0	102.1	12	0	0
72	6	1	8.6	8.6	9.1	0.5	0.5	0	1	0	0	1	8.6	8.6	0.0	8.9	8.9	0.2	1.0	79.9	0.0	79.9	7	0	0
73	0	6	9.4	9.4	9.9	0.5	0.5	0	0	0	0	1	9.5	9.5	0.1	9.8	9.8	0.3	1.0	58.3	0.0	58.3	1	0	0
74	3	1	9.8	9.8	10.3	0.5	0.5	0	0	0	0	1	9.9	9.9	0.1	10.4	10.4	0.5	0.9	103.8	5.2	98.6	11	0	0
75	5	1	10.3	10.3	10.8	0.5	0.5	0	1	0	0	1	10.3	10.3	0.0	10.7	10.7	0.4	1.0	137.2	0.0	137.2	7	0	0
76	0	1	9.5	9.5	10	0.5	0.5	0	1	0	0	1	9.8	9.8	0.3	10.0	10.0	0.2	1.0	79.9	0.0	79.9	16	0	0
77	4	1	10.1	10.1	10.6	0.5	0.5	0	0	0	0	1	10.1	10.1	0.0	10.6	10.6	0.5	0.9	118.5	5.9	112.6	3	0	0
78	1	6	10	10	10.5	0.5	0.5	0	0	0	0	1	10.0	10.0	0.0	10.3	10.3	0.2	1.0	59.1	0.0	59.1	15	0	0
79	3	4	10.2	10.2	10.7	0.5	0.5	0	0	0	0	1	10.2	10.2	0.0	10.5	10.5	0.2	1.0	59.1	0.0	59.1	8	0	0
80	2	5	10.9	10.9	11.4	0.5	0.5	0	1	0	0	1	11.0	11.0	0.1	11.4	11.4	0.4	1.0	157.0	0.0	157.0	14	0	0
81	2	1	11	11	11.5	0.5	0.5	0	0	0	0	1	11.4	11.4	0.4	11.6	11.6	0.3	1.0	58.3	0.0	58.3	4	0	0
82	1	4	11	11	11.5	0.5	0.5	0	0	1	0	0													
83	2	4	12.2	12.2	12.7	0.5	0.5	0	0	1	0	0													
84	3	1	12.4	12.4	12.9	0.5	0.5	0	0	1	0	0													
85	0	6	12	12	12.5	0.5	0.5	0	0	1	0	0													
86	0	1	12	12	12.5	0.5	0.5	0	0	1	0	0													
87	4	1	12.8	12.8	13.3	0.5	0.5	0	0	1	0	0													

Optimized customer solution under dynamic demand for instance 2:

cust_idx	origin	destination	etw_pu	etw	ltw	etw_factor	duration_factor	cust_type	premium	new_cust	cancelled	accomodated	B_i	D_i	W_i	A_i+s	B_i+s	L_i	s_i	c_il	d_i	c_if	k	premium_violation	tw_violation
1	3	2	16	16	16.3	0.3	0.7	0	0	0	0	1.0	16.0	16.0	0.0	16.3	16.3	0.2	1.0	59.1	0.0	59.097442	8	0	0
2	0	1	16	16	16.3	0.7	0.3	0	0	0	0	1.0	16.0	16.0	0.0	16.3	16.3	0.2	1.0	59.1	0.0	59.097442	1	0	0
3	0	4	16.1	16.1	16.2	0.9	0.1	0	1	0	0	1.0	16.1	16.1	0.0	16.4	16.4	0.2	1.0	79.9	0.0	79.9161882	5	0	0
4	0	1	16.1	16.1	16.7	0.6	0.4	0	1	0	0	1.0	16.1	16.1	0.0	16.4	16.4	0.2	1.0	79.9	0.0	79.9161882	3	0	0
5	3	4	16.1	16.1	16.3	0	1	0	0	0	0	1.0	16.1	16.1	0.0	16.4	16.4	0.2	1.0	59.1	0.0	59.097442	16	0	0
6	0	2	16.2	16.2	16.8	0.9	0.1	0	0	0	0	1.0	16.5	16.5	0.3	17.0	17.0	0.4	0.9	66.1	3.3	62.7925699	6	0	0
7	3	0	16.2	16.2	16.7	0.5	0.5	0	0	0	0	1.0	16.5	16.5	0.3	16.7	16.7	0.3	1.0	58.3	0.0	58.34	14	0	0
8	2	3	16.2	16.2	16.5	0.2	0.8	0	0	0	0	1.0	16.2	16.2	0.0	16.5	16.5	0.3	1.0	59.4	0.0	59.447442	14	0	0
9	0	4	16.3	16.3	16.9	0.3	0.7	0	0	0	0	1.0	16.8	16.8	0.5	17.1	17.1	0.3	0.9	59.8	3.0	56.8075699	14	0	0
10	0	5	16.3	16.3	16.5	0.1	0.9	0	1	0	0	1.0	16.3	16.3	0.0	16.6	16.6	0.2	1.0	79.9	0.0	79.9161882	9	0	0
11	6	1	16.3	16.3	16.8	0.2	0.8	0	0	0	0	1.0	16.7	16.7	0.4	17.2	17.2	0.5	0.6	67.4	13.5	53.9259536	7	0	0
12	0	5	16.4	16.4	17	0.4	0.6	0	0	0	0	1.0	16.4	16.4	0.0	16.7	16.7	0.2	1.0	59.1	0.0	59.097442	10	0	0
13	0	6	16.4	16.4	16.9	0.9	0.1	0	0	0	0	1.0	16.4	16.4	0.0	16.7	16.7	0.2	1.0	58.0	0.0	57.99	7	0	0
14	4	3	16.4	16.4	16.8	0.3	0.7	0	0	0	0	1.0	16.4	16.4	0.0	16.7	16.7	0.2	1.0	59.1	0.0	59.097442	5	0	0
15	0	2	16.5	16.5	16.8	0.5	0.5	0	0	0	0	1.0	16.5	16.5	0.0	17.0	17.0	0.4	0.8	66.1	6.6	59.4876978	6	0	0
16	4	0	16.5	16.5	17.1	0.9	0.1	0	0	0	0	1.0	17.1	17.1	0.6	17.3	17.4	0.3	1.0	59.4	0.0	59.447442	14	0	0
17	5	3	16.5	16.5	16.9	0.8	0.2	0	0	0	0	1.0	16.6	16.6	0.1	17.0	17.0	0.4	1.0	99.8	0.0	99.7954202	9	0	0
18	6	5	16.5	16.5	17	0.3	0.7	0	1	0	0	1.0	16.9	16.9	0.4	17.2	17.2	0.2	1.0	79.9	0.0	79.9161882	10	0	0
19	3	4	16.5	16.5	16.7	0.9	0.1	0	0	0	0	1.0	16.5	16.5	0.0	16.8	16.8	0.2	1.0	59.1	0.0	59.097442	17	0	0
20	5	2	16.5	16.5	16.7	0.2	0.8	0	0	0	0	1.0	16.6	16.6	0.1	17.2	17.2	0.6	0.7	123.1	12.3	110.76138	9	0	0
21	0	3	16.6	16.6	16.8	0.4	0.6	0	0	0	0	1.0	16.6	16.6	0.0	16.9	16.9	0.2	1.0	58.0	0.0	57.99	15	0	0
22	0	2	16.7	16.7	17.2	0.8	0.2	0	0	0	0	1.0	16.7	16.7	0.0	17.0	17.0	0.3	1.0	59.8	0.0	59.797442	6	0	0
23	6	4	16.7	16.7	17.2	0.3	0.7	0	0	0	0	1.0	16.7	16.7	0.0	17.1	17.1	0.4	1.0	99.8	0.0	99.7954202	3	0	0
24	0	5	16.8	16.8	17.4	0.3	0.7	0	1	0	0	1.0	16.8	16.8	0.0	17.1	17.1	0.2	1.0	79.9	0.0	79.9161882	4	0	0
25	0	4	16.8	16.8	17.5	0.7	0.3	0	0	0	0	1.0	16.8	16.8	0.0	17.1	17.1	0.3	1.0	59.4	0.0	59.447442	14	0	0
26	3	2	16.8	16.8	17.1	0.4	0.6	0	1	0	0	1.0	16.8	16.8	0.0	17.1	17.1	0.2	1.0	79.9	0.0	79.9161882	5	0	0
27	5	4	16.9	16.9	17.5	0.7	0.3	0	0	0	0	1.0	17.1	17.1	0.2	17.3	17.3	0.2	1.0	58.0	0.0	57.99	4	0	0
28	1	5	16.9	16.9	17.3	0.4	0.6	0	0	0	0	1.0	16.9	16.9	0.0	17.3	17.3	0.4	1.0	101.4	0.0	101.351111	1	0	0
29	6	2	16.9	16.9	17.6	0.5	0.5	0	0	0	0	1.0	16.9	16.9	0.0	17.4	17.4	0.5	0.9	103.8	5.2	98.573999	7	0	0
30	6	1	17	17	17.6	0.1	0.9	0	0	0	0	1.0	17.0	17.0	0.0	17.3	17.3	0.2	1.0	59.1	0.0	59.097442	8	0	0
31	2	5	17	17	17.1	0.3	0.7	0	1	0	0	1.0	17.0	17.0	0.0	17.4	17.4	0.4	1.0	157.0	0.0	156.952376	6	0	0
32	6	4	17.1	17.1	17.3	0.8	0.2	0	0	0	1	0.0													
33	1	5	17.2	17.2	17.6	0.4	0.6	0	0	0	0	1.0	17.4	17.4	0.2	17.8	17.8	0.4	1.0	101.4	0.0	101.351111	8	0	0
34	0	5	17.2	17.2	17.5	0.7	0.3	0	0	0	0	1.0	17.2	17.2	0.0	17.7	17.7	0.4	0.9	66.1	3.3	62.7925699	18	0	0
35	0	3	17.3	17.3	17.9	0.5	0.5	0	0	0	0	1.0	17.3	17.3	0.0	17.6	17.6	0.2	1.0	58.0	0.0	57.99	13	0	0
36	3	6	17.3	17.3	18	0.5	0.5	0	1	0	0	1.0	17.6	17.6	0.3	18.0	18.0	0.4	1.0	154.0	0.0	153.951429	13	0	0
37	0	2	17.3	17.3	18	0.8	0.2	0	0	0	0	1.0	18.0	18.0	0.7	18.3	18.3	0.3	1.0	59.4	0.0	59.447442	10	0	0
38	0	5	17.4	17.4	17.9	0.9	0.1	0	0	0	0	1.0	17.4	17.4	0.0	17.7	17.7	0.3	1.0	59.4	0.0	59.447442	18	0	0
39	0	4	17.4	17.4	18	0.3	0.7	0	1	0	0	1.0	17.4	17.4	0.0	17.7	17.7	0.2	1.0	79.9	0.0	79.9161882	11	0	0
40	5	4	17.4	17.4	17.6	0.4	0.6	0	1	0	0	1.0	17.5	17.5	0.1	17.7	17.7	0.2	1.0	78.4	0.0	78.4157143	10	0	0
41	6	5	17.4	17.4	17.7	0	1	0	0	0	0	1.0	17.4	17.4	0.0	17.7	17.7	0.2	1.0	59.1	0.0	59.097442	19	0	0
42	4	0	17.4	17.4	18.1	0.4	0.6	0	0	0	0	1.0	17.7	17.7	0.3	18.0	18.0	0.3	1.0	59.4	0.0	59.447442	10	0	0
43	0	1	17.5	17.5	18	0.8	0.2	0	1	0	1	0.0													
44	0	1	17.7	17.7	18.2	0.1	0.9	0	1	0	0	1.0	17.7	17.7	0.0	18.0	18.0	0.2	1.0	79.9	0.0	79.9161882	23	0	0
45	1	5	17.7	17.7	17.8	0.3	0.7	0	0	0	0	1.0	17.8	17.8	0.1	18.2	18.2	0.4	1.0	101.7	0.0	101.701111	22	0	0
46	0	6	17.7	17.7	18.1	0.1	0.9	0	0	0	0	1.0	17.7	17.7	0.0	18.0	18.0	0.2	1.0	58.0	0.0	57.99	20	0	0
47	2	3	17.8	17.8	18.2	0.8	0.2	0	0	0	0	1.0	17.8	17.8	0.0	18.1	18.1	0.2	1.0	59.1	0.0	59.097442	5	0	0
48	4	2	17.8	17.8	18.3	0	1	0	1	0	0	1.0	17.8	17.8	0.0	18.2	18.2	0.4	1.0	137.2	0.0	137.165714	2	0	0
49	0	4	18	18	18.4	0.6	0.4	0	0	0	1	0.0													
50	2	1	18.4	18.4	18.6	1	0	0	0	0	0	1.0	18.4	18.4	0.0	18.7	18.7	0.2	1.0	58.0	0.0	57.99	2	0	0
51	0	1	18.6	18.6	19.3	0.6	0.4	0	0	0	0	1.0	18.6	18.6	0.0	19.0	19.0	0.3	0.9	62.6	3.1	59.4675699	21	0	0
52	3	2	18.6	18.6	19.3	0.3	0.7	0	0	0	1	0.0													
53	0	1	18.7	18.7	19.2	0.6	0.4	0	0	0	0	1.0	18.7	18.7	0.0	19.0	19.0	0.3	1.0	59.4	0.0	59.447442	21	0	0
54	0	5	18.7	18.7	18.8	0.5	0.5	0	0	0	0	1.0	18.7	18.7	0.0	19.0	19.0	0.2	1.0	59.1	0.0	59.097442	13	0	0
55	2	5	18.8	18.8	19.2	0.7	0.3	0	0	0	0	1.0	19.2	19.2	0.4	19.6	19.6	0.4	1.0	116.3	0.0	116.3048884	5	0	0
56	0	4	18.8	18.8	19.4	0.5	0.5	0	0	0	0	1.0	18.8	18.8	0.0	19.1	19.1	0.2	1.0	59.1	0.0	59.097442	25	0	0
57	3	2	18.9	18.9	19	0.9	0.1	0	0	0	0	1.0	18.9	18.9	0.0	19.2	19.2	0.3	1.0	59.4	0.0	59.447442	5	0	0
58	0	4	19	19	19.3	0.8	0.2	0	0	0	0	1.0	19.0	19.0	0.0	19.3	19.3	0.2	1.0	59.1	0.0	59.097442	2	0	0
59	1	2	19.1	19.1	19.7	0.3	0.7	0	1	0	1	0.0													
60	0	1	19.2	19.2	19.7	0.9	0.1	0	1	0	0	1.0	19.2	19.2	0.0	19.5	19.5	0.2	1.0	79.9	0.0	79.9161882	24	0	0
61	5	2	19.6	19.6	20	0.9	0.1	0	0	0	0	1.0	19.6	19.6	0.0	20.1	20.1	0.5	1.0	118.5	0.0	118.544884	8	0	0
62	1	4	19.7	19.7	19.8	0.6	0.4	0	0	0	0	1.0	19.7	19.7	0.0	20.1	20.1	0.4	1.0	116.0	0.0	115.954884	4	0	0
63	0	5	19.7	19.7	19.8	0.7	0.3	0	0	0	0	1.0	19.8	19.8	0.1	20.1	20.1	0.2	1.0	59.1	0.0	59.097442	2	0	0
64	0	2	19.8	19.8	20.3	0.4	0.6	0	0	0	0	1.0	19.9	19.9	0.1	20.1	20.1	0.3	1.0	59.4	0.0	59.447442	8	0	0
65	1	0	19.9	19.9	20.6	0.3	0.7	0	0	0	0	1.0	19.9	19.9	0.0	20.2	20.2	0.2	1.0	59.1	0.0	59.097442	3	0	0
66	0	6	19.9	19.9	20.4	0.4	0.6	0	0	0	0	1.0	20.2	20.2	0.3	20.4	20.4	0.2	1.0	58.0	0.0	57.99	3	0	0
67	6	4	19.9	19.9	20.1	0.6	0.4	0	1	0	0	1.0	19.9	19.9	0.0	20.3	20.3	0.4	1.0	134.6	0.0	134.583692	20	0	0
68	3	5	20.1	20.1	20.7	0.4	0.6	0	0	0	1	0.0													
69	2	3	20.4	20.4	20.9	0.9	0.1	0	0	0	0	1.0	20.4	20.4	0.0	20.7	20.7	0.2	1.0	59.1	0.0	59.097442	9	0	0
70	6	2	21	21	21.4	0.4	0.6	0	0	0	1	0.0													
71	2	1	16	16	16.5	0.5	0.5	0	0	0	0	1.0	16.3	16.3	0.3	16.5	16.5	0.2	1.0	58.0	0.0	57.99	8	0	0
72	0	4	16.3	16.3	16.8	0.5	0.5	0	1	0	0	1.0	16.3	16.3	0.0	16.6	16.6	0.2	1.0	79.9	0.0	79.9161882	2	0	0
73	3	4	16.9	16.9	17.4	0.5	0.5	0	0	0	0	1.0	16.9	16.9	0.0	17.2	17.2	0.2	1.0	59.1	0.0	59.097442	15	0	0
74	6	4	16.2	16.2	16.7	0.5	0.5	0	0	0	0	1.0	16.6	16.6	0.4	17.1	17.1	0.5	0.9	102.5	5.1	97.3912591	3	0	0
75	1	0	16.5	16.5	17	0.5	0.5	0	0	1	0	0.0													
76	0	4	16.5	16.5	17	0.5	0.5	0	0	1	0	0.0													
77	0	1	17.4	17.4	17.9	0.5	0.5	0	0	0	0	1.0	17.6	17.6	0.2	17.9	17.9	0.3	1.0	59.4	0.0	59.447442	14	0	0
78	0	4	17	17	17.5	0.5	0.5	0	0	0	0	1.0	17.0	17.0	0.0	17.3	17.3	0.2	1.0	59.1	0.0	59.097442	12	0	0
79	0	1	17.5	17.5	18	0.5	0.5	0	0	0	0	1.0	17.5	17.5	0.0	17.8	17.8	0.3	1.0	59.4	0.0	59.447442	22	0	0
80	5	1	18.3	18.3	18.8	0.5	0.5	0	0	0	0	1.0	18.3	18.3	0.0	18.7	18.7	0.4	1.0	101.4	0.0	101.351111	3	0	0
81	4	5	17.6	17.6	18.1	0.5	0.5	0	1	0	0	1.0	17.6	17.6	0.0	17.9	17.9	0.2	1.0	78.4	0.0	78.4157143	3	0	0
82	4	1	18.1	18.1	18.6	0.5	0.5	0	0	0	0	1.0	18.1	18.1	0.0	18.7	18.7	0.5	0.9	120.1	6.00403657	114.076695	4	0	0
83	0	1	18.4	18.4	18.9	0.5	0.5	0	0	0	0	1.0	18.4	18.4	0.0	18.7	18.7	0.3	1.0	59.4	0	59.447442	4	0	0
84	5	6	18.4	18.4	18.9	0.5	0.5	0	1	0	0	1.0	18.4	18.4	0.0	18.7	18.7	0.2	1.0	79.9	0	79.9161882	6	0	0
85	4	5	19.1	19.1	19.6	0.5	0.5	0	1	0	0	1.0	19.1	19.1	0.0	19.4	19.4	0.2	1.0	78.4	0	78.4157143	11	0	0
86	3	1	19	19	19.5	0.5	0.5	0	0	0	0	1.0	19.0	19.0	0.0	19.4	19.4	0.4	1.0	99.4	0	99.4454202	7	0	0
87	6	1	19.5	19.5	20	0.5	0.5	0	1	0	0	1.0	19.5	19.5	0.0	19.8	19.8	0.2	1.0	79.9	0	79.9161882	6	0	0
88	4	1	19.5	19.5	20	0.5	0.5	0	0	1	0	0													
89	0	5	20.4	20.4	20.9	0.5	0.5	0	0	1	0	0													
90	1	6	20.2	20.2	20.7	0.5	0.5	0	0	1	0	0													
91	0	1	21	21	21.5	0.5	0.5	0	0	0	0	1	21	21.01	0	21.25611865	21.2661186	0.24611865	0.999762018	59.09744201	0	59.097442	9	0	0
92	0	2	21.2	21.2	21.7	0.5	0.5	0	0	0	0	1	21.2	21.21	0	21.45611865	21.4661186	0.24611865	0.999764262	59.09744201	0	59.097442	3	0	0
93	2	3	21.1	21.1	21.6	0.5	0.5	0	1	0	0	1	21.1	21.11	0	21.35611865	21.3661186	0.24611865	0.999763145	79.91618824	0	79.9161882	8	0	0
94	4	1	21.4	21.4	21.9	0.5	0.5	0	1	0	0	1	21.4	21.41	0	21.8382373	21.8482373	0.4282373	0.999766464	156.9523765	0	156.952376	12	0	0

Optimized customer solution under dynamic demand for instance 3:

cust_idx	origin	destination	etw_pu	etw	ltw	etw_factor	duration_factor	cust_type	premium	new_cust	cancelled	accomodated	B_i	D_i	W_j	A_i+s	B_i+s	I_i	s_i	c_i1	d_i	c_if	k	premium_violation	tw_violation
1	3	1	16	16	16.3	0.3	0.7	0	0	0	0.0	1.0	16.0	16.0	0.0	16.6	16.6	0.6	0.7	107.6	10.7612104	96.8508938	6	0	0
2	0	1	16	16	16.3	0.7	0.3	0	0	0	1.0	0.0													
3	0	1	16.1	16.1	16.2	0.9	0.1	0	1	0	0.0	1.0	16.1	16.1	0.0	16.4	16.4	0.2	1.0	79.9	0	79.9161882	5	0	0
4	0	1	16.1	16.1	16.7	0.6	0.4	0	1	0	0.0	1.0	16.1	16.1	0.0	16.4	16.4	0.2	1.0	79.9	0	79.9161882	4	0	0
5	3	1	16.1	16.1	16.3	0	1	0	0	0	0.0	1.0	16.1	16.1	0.0	16.6	16.6	0.5	0.7	104.5	10.4462104	94.0158938	6	0	0
6	0	1	16.2	16.2	16.8	0.9	0.1	0	0	0	0.0	1.0	16.2	16.2	0.0	16.6	16.6	0.3	1.0	62.6	0	62.597442	11	0	0
7	3	0	16.2	16.2	16.7	0.5	0.5	0	0	0	0.0	1.0	16.6	16.6	0.4	16.8	16.8	0.2	1.0	58.0	0	57.99	10	0	0
8	2	1	16.2	16.2	16.5	0.2	0.8	0	0	0	0.0	1.0	16.4	16.4	0.2	16.6	16.7	0.3	0.9	59.0	2.952	56.088	6	0	0
9	0	1	16.3	16.3	16.9	0.3	0.7	0	0	0	0.0	1.0	16.8	16.8	0.5	17.1	17.1	0.2	1.0	59.1	0	59.097442	10	0	0
10	0	3	16.3	16.3	16.5	0.1	0.9	0	1	0	0.0	1.0	16.3	16.3	0.0	16.6	16.6	0.2	1.0	78.4	0	78.4157143	10	0	0
11	6	1	16.3	16.3	16.8	0.2	0.8	0	0	0	0.0	1.0	16.7	16.7	0.4	16.9	16.9	0.2	1.0	59.1	0	59.097442	3	0	0
12	0	4	16.4	16.4	17	0.4	0.6	0	0	0	0.0	1.0	16.4	16.4	0.0	16.7	16.7	0.2	1.0	59.1	0	59.097442	13	0	0
13	0	6	16.4	16.4	16.9	0.9	0.1	0	0	0	0.0	1.0	16.4	16.4	0.0	16.7	16.7	0.2	1.0	58.0	0	57.99	3	0	0
14	4	1	16.4	16.4	16.8	0.3	0.7	0	0	0	0.0	1.0	16.6	16.6	0.2	17.0	17.0	0.4	1.0	116.3	0	116.304884	7	0	0
15	0	1	16.5	16.5	16.8	0.5	0.5	0	0	0	0.0	1.0	16.8	16.8	0.3	17.0	17.0	0.3	1.0	59.4	0	59.447442	16	0	0
16	3	0	16.5	16.5	17.1	0.9	0.1	0	0	0	0.0	1.0	16.5	16.5	0.0	16.8	16.8	0.3	1.0	58.3	0	58.34	16	0	0
17	4	1	16.5	16.5	16.9	0.8	0.2	0	0	0	0.0	1.0	16.8	16.8	0.3	17.3	17.3	0.5	1.0	118.9	0	118.894884	21	0	0
18	5	4	16.5	16.5	17	0.3	0.7	0	1	0	0.0	1.0	16.5	16.5	0.0	16.8	16.8	0.2	1.0	78.4	0	78.4157143	21	0	0
19	6	1	16.5	16.5	16.7	0.9	0.1	0	0	0	0.0	1.0	16.5	16.5	0.0	16.8	16.8	0.2	1.0	59.1	0	59.097442	18	0	0
20	3	1	16.5	16.5	16.7	0.2	0.8	0	0	0	0.0	1.0	16.5	16.5	0.0	17.0	17.0	0.5	0.8	104.1	10.4112104	93.7008938	16	0	0
21	5	1	16.5	16.5	16.7	0.4	0.6	0	0	0	0.0	1.0	16.5	16.5	0.0	17.2	17.2	0.6	0.8	110.6	11.0614153	99.5527374	24	0	0
22	0	1	16.6	16.6	17.1	0.8	0.2	0	0	0	0.0	1.0	17.0	17.0	0.4	17.3	17.3	0.3	1.0	59.8	0	59.797442	21	0	0
23	0	2	16.7	16.7	17.2	0.3	0.7	0	0	0	0.0	1.0	16.7	16.7	0.0	17.0	17.0	0.2	1.0	59.1	0	59.097442	19	0	0
24	6	2	16.7	16.7	17.3	0.3	0.7	0	1	0	0.0	1.0	16.7	16.7	0.0	17.1	17.1	0.4	1.0	134.6	0	134.583692	12	0	0
25	0	1	16.8	16.8	17.5	0.7	0.3	0	0	0	0.0	1.0	17.0	17.0	0.2	17.3	17.3	0.3	1.0	59.8	0	59.797442	21	0	0
26	0	1	16.8	16.8	17.1	0.4	0.6	0	1	0	1.0	0.0													
27	3	4	16.8	16.8	17.4	0.7	0.3	0	0	0	0.0	1.0	16.8	16.8	0.0	17.1	17.1	0.2	1.0	59.1	0	59.097442	17	0	0
28	5	1	16.9	16.9	17.3	0.4	0.6	0	0	0	0.0	1.0	16.9	16.9	0.0	17.7	17.7	0.7	0.7	114.1	11.4114153	102.702737	23	0	0
29	1	2	16.9	16.9	17.6	0.5	0.5	0	0	0	0.0	1.0	17.6	17.6	0.7	17.8	17.8	0.3	0.9	58.7	2.9345	55.7555	12	0	0
30	6	1	16.9	16.9	17.5	0.1	0.9	0	0	0	0.0	1.0	16.9	16.9	0.0	17.2	17.2	0.3	1.0	59.4	0	59.447442	24	0	0
31	6	2	17	17	17.1	0.3	0.7	0	1	0	0.0	1.0	17.0	17.0	0.0	17.4	17.4	0.4	1.0	134.6	0	134.583692	8	0	0
32	2	3	17	17	17.2	0.8	0.2	0	0	0	0.0	1.0	17.0	17.0	0.0	17.3	17.3	0.2	1.0	59.1	0	59.097442	19	0	0
33	4	2	17	17	17.4	0.4	0.6	0	0	0	0.0	1.0	17.0	17.0	0.0	17.5	17.5	0.5	0.9	105.6	5.27891527	100.29939	13	0	0
34	6	1	17.1	17.1	17.4	0.7	0.3	0	0	0	0.0	1.0	17.4	17.4	0.3	17.7	17.7	0.3	0.9	61.3	3.06666447	58.2666248	23	0	0
35	5	1	17.1	17.1	17.7	0.5	0.5	0	0	0	0.0	1.0	17.1	17.1	0.0	17.7	17.7	0.6	0.8	107.5	10.7464153	96.7177374	23	0	0
36	1	5	17.2	17.2	17.9	0.5	0.5	0	1	0	0.0	1.0	17.6	17.6	0.4	18.0	18.0	0.4	1.0	137.2	0	137.165714	21	0	0
37	0	1	17.2	17.2	17.9	0.8	0.2	0	0	0	0.0	1.0	17.7	17.7	0.5	17.9	18.0	0.3	1.0	59.8	0	59.797442	17	0	0
38	2	4	17.2	17.2	17.7	0.9	0.1	0	0	0	0.0	1.0	17.4	17.4	0.2	17.9	17.9	0.5	1.0	105.6	0	105.578305	8	0	0
39	0	1	17.3	17.3	17.9	0.3	0.7	0	1	0	0.0	1.0	17.3	17.3	0.0	17.6	17.6	0.2	1.0	79.9	0	79.9161882	14	0	0
40	3	2	17.3	17.3	17.5	0.4	0.6	0	1	0	0.0	1.0	17.3	17.3	0.0	17.6	17.6	0.2	1.0	79.9	0	79.9161882	19	0	0
41	0	6	17.3	17.3	17.6	0	1	0	0	0	0.0	1.0	17.3	17.3	0.0	17.6	17.6	0.2	1.0	58.0	0	57.99	2	0	0
42	0	1	17.4	17.4	18.1	0.4	0.6	0	0	0	0.0	1.0	17.7	17.7	0.3	17.9	17.9	0.3	0.9	59.8	2.9898721	56.8075699	17	0	0
43	0	1	17.4	17.4	17.9	0.8	0.2	0	1	0	0.0	1.0	17.7	17.7	0.3	17.9	17.9	0.2	1.0	79.9	0	79.9161882	22	0	0
44	5	0	17.4	17.4	17.9	0.1	0.9	0	1	0	0.0	1.0	17.4	17.4	0.0	17.7	17.7	0.2	1.0	79.9	0	79.9161882	22	0	0
45	6	2	17.4	17.4	17.5	0.3	0.7	0	0	0	0.0	1.0	17.4	17.4	0.0	17.9	17.9	0.5	0.8	104.5	10.4462104	94.0158938	23	0	0
46	3	4	17.4	17.4	17.8	0.1	0.9	0	0	0	0.0	1.0	17.7	17.7	0.3	17.9	17.9	0.3	1.0	59.4	0	59.447442	8	0	0
47	4	1	17.4	17.4	17.8	0.8	0.2	0	0	0	0.0	1.0	17.4	17.4	0.0	17.9	17.9	0.5	1.0	118.9	0	118.894884	17	0	0
48	0	1	17.5	17.5	18	0	1	0	1	0	0.0	1.0	17.5	17.5	0.0	17.8	17.8	0.2	1.0	79.9	0	79.9161882	20	0	0
49	6	1	17.6	17.6	18	0.6	0.4	0	0	0	0.0	1.0	17.6	17.6	0.0	17.9	17.9	0.2	1.0	59.1	0	59.097442	1	0	0
50	0	1	17.7	17.7	17.9	1	0	0	0	0	0.0	1.0	17.7	17.7	0.0	18.2	18.2	0.5	1.0	66.8	0	66.797442	25	0	0
51	1	0	17.7	17.7	18.4	0.6	0.4	0	0	0	0.0	1.0	18.2	18.2	0.5	18.5	18.5	0.3	1.0	59.4	0	59.447442	25	0	0
52	0	1	17.7	17.7	18.4	0.3	0.7	0	0	0	0.0	1.0	17.9	17.9	0.2	18.2	18.2	0.3	0.9	60.5	3.0248721	57.4725699	25	0	0
53	2	0	17.8	17.8	18.3	0.6	0.4	0	0	0	0.0	1.0	17.8	17.8	0.0	18.1	18.1	0.3	1.0	59.4	0	59.447442	4	0	0
54	4	2	17.8	17.8	17.9	0.5	0.5	0	0	0	0.0	1.0	17.8	17.8	0.0	18.2	18.2	0.4	1.0	101.4	0	101.351111	15	0	0
55	6	3	17.9	17.9	18.3	0.7	0.3	0	0	0	0.0	1.0	17.9	17.9	0.0	18.3	18.3	0.4	1.0	113.7	0	113.74	2	0	0
56	0	2	18	18	18.6	0.5	0.5	0	0	0	0.0	1.0	18.6	18.6	0.6	18.8	18.8	0.2	1.0	59.1	0	59.097442	13	0	0
57	5	1	18.2	18.2	18.3	0.9	0.1	0	0	0	1.0	0.0													
58	2	1	18.4	18.4	18.7	0.8	0.2	0	0	0	0.0	1.0	18.4	18.4	0.0	18.7	18.7	0.2	1.0	58.0	0	57.99	15	0	0
59	0	1	18.6	18.6	19.2	0.3	0.7	0	1	0	0.0	1.0	19.1	19.1	0.5	19.3	19.3	0.2	1.0	79.9	0	79.9161882	4	0	0
60	3	0	18.6	18.6	19.1	0.9	0.1	0	1	0	1.0	0.0													
61	0	1	18.7	18.7	19.1	0.9	0.1	0	0	0	0.0	1.0	18.7	18.7	0.0	19.1	19.1	0.4	1.0	62.9	0	62.947442	8	0	0
62	0	1	18.7	18.7	18.8	0.6	0.4	0	0	0	0.0	1.0	18.7	18.7	0.0	19.1	19.1	0.4	0.9	62.9	3.1473721	59.8000699	8	0	0
63	2	4	18.8	18.8	18.9	0.7	0.3	0	0	0	0.0	1.0	18.8	18.9	0.0	19.4	19.4	0.5	0.9	105.6	5.27891527	100.29939	13	0	0
64	0	1	18.8	18.8	19.3	0.4	0.6	0	0	0	0.0	1.0	18.8	18.8	0.0	19.1	19.1	0.3	0.9	60.1	3.0073721	57.1400699	8	0	0
65	3	0	18.9	18.9	19.6	0.3	0.7	0	0	0	0.0	1.0	18.9	18.9	0.0	19.2	19.2	0.2	1.0	58.0	0	57.99	2	0	0
66	0	6	19	19	19.5	0.4	0.6	0	0	0	0.0	1.0	19.2	19.2	0.2	19.4	19.4	0.2	1.0	58.0	0	57.99	2	0	0
67	1	5	19.1	19.1	19.3	0.6	0.4	0	1	0	0.0	1.0	19.1	19.1	0.0	19.5	19.5	0.4	1.0	137.2	0	137.165714	3	0	0
68	0	4	19.2	19.2	19.8	0.4	0.6	0	0	0	0.0	1.0	19.7	19.7	0.5	19.9	19.9	0.2	1.0	59.1	0	59.097442	2	0	0
69	5	1	19.6	19.6	20.1	0.9	0.1	0	0	0	0.0	1.0	19.6	19.6	0.0	20.1	20.1	0.5	1.0	105.6	0	105.578305	3	0	0
70	1	2	19.7	19.7	20.1	0.4	0.6	0	0	0	0.0	1.0	19.7	19.7	0.0	20.0	20.0	0.2	1.0	58.0	0	57.99	5	0	0
71	0	2	19.7	19.7	20.2	0.5	0.5	0	0	0	0.0	1.0	19.7	19.7	0.0	20.0	20.0	0.2	1.0	59.1	0	59.097442	4	0	0
72	0	1	19.8	19.8	20.1	0.4	0.6	0	0	0	0.0	1.0	19.9	19.9	0.1	20.1	20.1	0.3	1.0	59.4	0	59.447442	3	0	0
73	1	6	19.9	19.9	20.3	0.1	0.9	0	1	0	0.0	1.0	19.9	19.9	0.0	20.2	20.2	0.2	1.0	79.9	0	79.9161882	6	0	0
74	0	5	19.9	19.9	20.3	0.1	0.9	0	1	0	0.0	1.0	20.2	20.2	0.3	20.5	20.5	0.2	1.0	79.9	0	79.9161882	5	0	0
75	6	1	19.9	19.9	20.2	0.1	0.9	0	0	0	0.0	1.0	20.2	20.2	0.3	20.4	20.4	0.2	1.0	59.1	0	59.097442	6	0	0
76	3	1	20.1	20.1	20.6	0.9	0.1	0	1	0	0.0	1.0	20.5	20.5	0.4	20.9	20.9	0.4	1.0	134.6	0	134.583692	4	0	0
77	2	1	20.4	20.4	20.6	0.2	0.8	0	0	0	0.0	1.0	20.4	20.4	0.0	20.9	20.9	0.4	0.6	65.0	12.998	51.992	23	0	0
78	6	0	21	21	21.4	0.4	0.6	0	0	0	1.0	0.0													
79	2	1	16	16	16.5	0.5	0.5	0	0	0	0.0	1.0	16.0	16.0	0.0	16.3	16.3	0.2	1.0	58.0	0	57.99	9	0	0
80	0	4	16.3	16.3	16.8	0.5	0.5	0	1	0	0.0	1.0	16.3	16.3	0.0	16.6	16.6	0.2	1.0	79.9	0	79.9161882	15	0	0
81	3	1	16.3	16.3	16.8	0.5	0.5	0	1	0	0.0	1.0	16.3	16.3	0.0	16.7	16.7	0.4	1.0	134.6	0	134.583692	1	0	0
82	4	1	16.2	16.2	16.7	0.5	0.5	0	0	0	0.0	1.0	16.6	16.6	0.4	17.0	17.0	0.4	1.0	116.304884	0	116.304884	7	0	0
83	0	1	16.3	16.3	16.8	0.5	0.5	0	0	0	0.0	1.0	16.3	16.3	0.0	16.6	16.6	0.3	1.0	59.447442	0	59.447442	11	0	0
84	1	4	17.3	17.3	17.8	0.5	0.5	0	0	0	0.0	1.0	17.6	17.6	0.3	18.2	18.2	0.6	0.8	123.15769	12.315769	110.841921	12	0	0
85	3	6	17.1	17.1	17.6	0.5	0.5	0	0	0	0.0	1.0	17.3	17.3	0.2	17.9	17.9	0.6	0.8	121.101621	12.1101621	108.991459	13	0	0
86	2	1	17.3	17.3	17.8	0.5	0.5	0	0	0	0.0	1.0	17.3	17.3	0.0	17.6	17.6	0.3	1.0	58.34	0	58.34	12	0	0
87	0	1	17.9	17.9	18.4	0.5	0.5	0	0	0	0.0	1.0	17.9	17.9	0.0	18.2	18.2	0.3	0.9	60.147442	3.0073721	57.1400699	25	0	0
88	2	1	17.7	17.7	18.2	0.5	0.5	0	1	0	0	1	17.8538397	17.86383973	0.153839732	18.1064112	18.1164112	0.242571429	0.995414207	78.4157143	0	78.4157143	19	0	0
89	0	1	17.8	17.8	18.3	0.5	0.5	0	1	0	0	1	18.0861186	18.09611865	0.286118648	18.3422373	18.3522373	0.246118648	0.991818172	79.9161882	0	79.9161882	4	0	0
90	2	0	17.7	17.7	18.2	0.5	0.5	0	0	0	0	1	17.81	17.82	0.11	18.0761186	18.0861186	0.256118648	0.977110795	59.447442	0	59.447442	4	0	0
91	0	5	18.1	18.1	18.6	0.5	0.5	0	1	0	0	1	18.4822373	18.4922373	0.382237297	18.7383559	18.7483559	0.246118648	0.989394542	79.9161882	0	79.9161882	25	0	0
92	0	1	17.9	17.9	18.4	0.5	0.5	0	0	0	0	1	17.91	17.92	0.01	18.1961186	18.2061186	0.276118648	0.945117491	60.147442	3.0073721	57.1400699	25	0	0
93	1	0	18.5	18.5	19	0.5	0.5	0	0	0	0	1	18.5	18.51	0	18.7561186	18.7661186	0.246118648	0.999729876	59.097442	0	59.097442	4	0	0
94	4	1	18.3	18.3	18.8	0.5	0.5	0	0	0	0	1	18.3	18.31	0	19.0561186	19.0661186	0.746118648	0.786703599	127.080731	12.7080731	114.372658	8	0	0
95	2	3	18.7	18.7	19.2	0.5	0.5	0	0	0	0	1	18.8323512	18.84235115	0.132351152	19.0984698	19.1084698	0.256118648	0.976700373	59.447442	0	59.447442	13	0	0
96	0	4	18.6	18.6	19.1	0.5	0.5	0	1	1	0	0													
97	6	1	18.6	18.6	19.1	0.5	0.5	0	1	1	0	0													
98	0	1	18.9	18.9	19.4	0.5	0.5	0	0	1	0	0													
99	4	0	20.2	20.2	20.7	0.5	0.5	0	0	0	0	1	20.2	20.21	0	20.4561186	20.4661186	0.246118648	0.999752598	59.097442	0	59.097442	12	0	0
100	5	0	20.2	20.2	20.7	0.5	0.5	0	0	0	0	1	20.3525714	20.36257143	0.152571429	20.6186901	20.6286901	0.256118648	0.976548588	59.447442	0	59.447442	13	0	0
101	4	5	20.1	20.1	20.6	0.5	0.5	0	0	0	0	1	20.1	20.11	0	20.3625714	20.3725714	0.252571429	0.979954987	58.34	0	58.34	13	0	0
102	2	1	20.6	20.6	21.1	0.5	0.5	0	0	0	0	1	20.6	20.61	0	20.8625714	20.8725714	0.252571429	0.979961019	58.34	0	58.34	23	0	0
103	4	1	20.9	20.9	21.4	0.5	0.5	0	0	0	0	1	20.9	20.91	0	21.3382373	21.3482373	0.428237297	0.99976088	115.954884	0	115.954884	2	0	0

2

Appendix 2: Vehicle routes under dynamic demand

Routes for instance 1 under dynamic demand:

r1 = [0, 7, 6, 94, 93, 38, 125, 42, 73, 129, 160, 61, 64, 56, 148, 151, 143, 65, 152]
r2 = [0, 31, 118, 29, 30, 116, 117, 66, 153]
r3 = [0, 5, 92, 4, 91, 15, 102, 23, 28, 33, 110, 115, 120, 77, 51, 164, 138, 67, 154]
r4 = [0, 3, 90, 20, 107, 24, 111, 46, 133, 54, 141, 62, 81, 149, 168]
r5 = [0, 2, 89, 40, 127]
r6 = [0, 16, 11, 14, 103, 98, 101, 52, 139, 58, 145]
r7 = [0, 1, 88, 8, 95, 18, 105, 72, 159, 36, 123, 75, 162, 59, 146]
r8 = [0, 10, 97, 9, 13, 96, 100, 37, 124, 79, 166]
r9 = [0, 17, 12, 104, 99, 34, 121]
r10 = [0, 19, 106, 45, 132]
r11 = [0, 21, 108, 22, 32, 109, 119, 50, 74, 137, 161, 53, 140]
r12 = [0, 27, 71, 25, 114, 158, 112, 80, 167, 60, 147]
r13 = [0, 35, 122, 55, 142, 69, 156]
r14 = [0, 57, 144, 80, 167]
r15 = [0, 39, 126, 78, 165]
r16 = [0, 44, 131, 76, 163, 47, 134]

Routes for instance 2 under dynamic demand:

r1 = [0, 2, 96, 28, 122, 80, 174]
r2 = [0, 72, 166, 48, 142, 50, 144, 58, 152, 63, 157]
r3 = [0, 4, 98, 74, 23, 168, 117, 81, 175, 80, 174, 92, 186]
r4 = [0, 24, 118, 27, 121, 82, 83, 176, 177]
r5 = [0, 3, 97, 14, 108, 26, 120, 47, 141, 57, 55, 151, 149, 62, 156, 65, 159, 66, 160]
r6 = [0, 15, 6, 22, 109, 100, 116, 31, 125, 84, 178, 87, 181]
r7 = [0, 13, 107, 11, 29, 105, 123, 86, 180]
r8 = [0, 1, 95, 71, 165, 93, 187]
r9 = [0, 10, 104, 17, 20, 111, 114, 69, 163, 91, 185]
r10 = [0, 12, 106, 18, 112, 40, 134, 42, 37, 136, 131]
r11 = [0, 30, 124, 33, 39, 133, 127, 61, 64, 155, 158, 85, 179]
r12 = [0, 78, 172, 94, 188]
r13 = [0, 35, 129, 36, 130, 54, 148]
r14 = [0, 8, 7, 102, 101, 25, 9, 119, 16, 103, 110, 77, 79, 171, 173]
r15 = [0, 21, 115, 73, 167]
r16 = [0, 5, 99, 46, 140, 67, 161]
r17 = [0, 19, 113, 79, 45, 173, 139]
r18 = [0, 34, 38, 128, 132, 51, 53, 145, 147]
r19 = [0, 41, 135, 44, 138, 56, 150, 60, 154]

Routes for instance 3 under dynamic demand:
r1 = [0, 21, 30, 124, 133, 81, 184, 49, 152, 91, 194]
r2 = [0, 41, 144, 55, 158, 65, 168, 66, 169, 68, 171, 103, 206]
r3 = [0, 13, 116, 11, 114, 67, 170, 69, 72, 172, 175]
r4 = [0, 4, 107, 53, 90, 156, 193, 89, 192, 93, 196, 59, 162, 71, 174, 76, 179]
r5 = [0, 3, 106, 70, 173, 74, 177]
r6 = [0, 1, 5, 79, 8, 104, 108, 182, 111, 73, 176, 75, 178]
r7 = [0, 80, 183, 14, 82, 117, 185]
r8 = [0, 31, 134, 38, 46, 141, 149, 94, 61, 62, 64, 197, 164, 165, 167]
r9 = [0, 36, 139, 79, 182, 50, 87, 92, 52, 153, 190, 195, 51, 155, 154, 91, 194]
r10 = [0, 10, 113, 7, 110, 9, 112]
r11 = [0, 6, 83, 109, 186, 28, 35, 34, 45, 131, 138, 137, 148]
r12 = [0, 24, 127, 86, 29, 189, 84, 132, 187, 99, 202]
r13 = [0, 12, 115, 33, 85, 136, 188, 56, 159, 95, 63, 198, 166, 101, 100, 204, 203]
r14 = [0, 18, 121, 17, 22, 25, 120, 125, 128, 39, 142]
r15 = [0, 80, 183, 54, 157, 58, 161]
r16 = [0, 16, 20, 119, 15, 123, 118]
r17 = [0, 27, 130, 47, 42, 37, 150, 145, 140]
r18 = [0, 19, 122, 44, 147, 43, 146]
r19 = [0, 23, 126, 32, 135, 40, 143, 88, 191]
r20 = [0, 48, 151, 77, 102, 180, 205]

Bibliography

[1] Vehicle Routing Problem : Models and Solutions. *Journal of Quality Measurement and Analysis*, 4(1): 205–218, 2008.

[2] The Share-A-Ride Problem: People and parcels sharing taxis. *European Journal of Operational Research*, 238(1):31–40, 2014. ISSN 03772217. doi: 10.1016/j.ejor.2014.03.003.

[3] Enabling airspace integration for high-density on-demand mobility operations. *17th AIAA Aviation Technology, Integration, and Operations Conference, 2017*, (June):1–24, 2017. doi: 10.2514/6.2017-3086.

[4] Jeph Abara. Applying Integer Linear Programming to the Fleet Assignment Problem. *Interfaces*, 19(4): 20–28, 1989. ISSN 0092-2102. doi: 10.1287/inte.19.4.20.

[5] Sohaib Afifi, Duc Cuong Dang, and Aziz Moukrim. A simulated annealing algorithm for the vehicle routing problem with time windows and synchronization constraints. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7997 LNCS:259–265, 2013. ISSN 03029743. doi: 10.1007/978-3-642-44973-4_27.

[6] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences of the United States of America*, 114(3):462–467, 2017. ISSN 10916490. doi: 10.1073/pnas.1611675114.

[7] Vincent Armant and Kenneth N. Brown. Minimizing the Driving Distance in Ride Sharing Systems. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, 2014-December:568–575, 2014. ISSN 10823409. doi: 10.1109/ICTAI.2014.91.

[8] Andrea Attanasio, Jean François Cordeau, Gianpaolo Ghiani, and Gilbert Laporte. Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387, 2004. ISSN 01678191. doi: 10.1016/j.parco.2003.12.001.

[9] Nabila Azi, Michel Gendreau, and Jean Yves Potvin. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers and Operations Research*, 41(1):167–173, 2014. ISSN 03050548. doi: 10.1016/j.cor.2013.08.016. URL http://dx.doi.org/10.1016/j.cor.2013.08.016.

[10] Alessandro Bacchini and Enrico Cestino. Electric VTOL configurations comparison. *Aerospace*, 6(3), 2019. ISSN 22264310. doi: 10.3390/aerospace6030026.

[11] Mohamed Barkaoui and Michel Gendreau. An adaptive evolutionary approach for real-time vehicle routing and dispatching. *Computers and Operations Research*, 40(7):1766–1776, 2013. ISSN 03050548. doi: 10.1016/j.cor.2013.01.022. URL http://dx.doi.org/10.1016/j.cor.2013.01.022.

[12] Cynthia Barnhart, Douglas Fearing, Amedeo Odoni, and Vikrant Vaze. Demand and capacity management in air transportation. *EURO Journal on Transportation and Logistics*, 1(1-2):135–155, 2012. ISSN 2192-4376. doi: 10.1007/s13676-012-0006-9.

[13] Claudia Bongiovanni, Mor Kaspi, and Nikolas Geroliminis. The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological*, 122:436–456, 2019. ISSN 01912615. doi: 10.1016/j.trb.2019.03.004.

[14] Christabelle S. Bosson and Todd A. Lauderdale. Simulation evaluations of an autonomous urban air mobility network management and separation service. *2018 Aviation Technology, Integration, and Operations Conference*, 2018. doi: 10.2514/6.2018-3365.

[15] Zhi Long Chen and Hang Xu. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40(1):74–88, 2006. ISSN 15265447. doi: 10.1287/trsc.1050.0133.

[16] Christian H. Christiansen and Jens Lysgaard. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 35(6):773–781, 2007. ISSN 01676377. doi: 10.1016/j.orl.2006.12.009.

[17] Jean François Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006. ISSN 0030364X. doi: 10.1287/opre.1060.0283.

[18] Jean François Cordeau and Gilbert Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003. ISSN 01912615. doi: 10.1016/S0191-2615(02)00045-0.

[19] Jean François Cordeau and Gilbert Laporte. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007. ISSN 02545330. doi: 10.1007/s10479-007-0170-8.

[20] Luca Coslovich. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. 175:1605–1615, 2006. doi: 10.1016/j.ejor.2005.02.038.

[21] G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, 1959. ISSN 0025-1909. doi: 10.1287/mnsc.6.1.80.

[22] Michel Gendreau. Solving an ambulance location model by tabu search. *Location Science*, 5(2):75–88, 1997. ISSN 09668349. doi: 10.1016/S0966-8349(97)00015-6.

[23] Michel Gendreau, François Guertin, Jean Yves Potvin, and René Séguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies*, 14(3):157–174, 2006. ISSN 0968090X. doi: 10.1016/j.trc.2006.03.002.

[24] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986. ISSN 03050548. doi: 10.1016/0305-0548(86)90048-1.

[25] Bill Gurley. A deeper look at ubers dynamic pricing model. URL http://abovethecrowd.com/2014/03/11/a-deeper-look-at-ubers-dynamic-pricing-model/.

[26] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, New York, NY, USA, seventh edition, 2001.

[27] Jeff Holden and Nikhil Goel. Fast-Forwarding to a Future of On-Demand Urban Air Transportation. pages 1–98, 2016. URL https://www.uber.com/elevate.pdf.

[28] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992.

[29] Ta-Yin Hu and Chin-Ping Chang. Exact Algorithm for Dial-A-Ride Problems with Time-Dependent Travel Cost. *Journal of the Eastern Asia Society for Transportation Studies*, 10:916–933, 2013.

[30] Ramon Iglesias, Federico Rossi, Kevin Wang, David Hallac, Jure Leskovec, and Marco Pavone. Data-driven model predictive control of autonomous mobility-on-demand systems. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6019–6025, 2018. ISSN 10504729. doi: 10.1109/ICRA.2018.8460966.

[31] Parida Jewpanya, Yu-wei Chen, and Vincent F Yu. The pickup and delivery multi-depot vehicle routing problem. 2016.

[32] Hongmei Jia, Yang Li, Bo Dong, and Hongying Ya. An Improved Tabu Search Approach to Vehicle Routing Problem. *Procedia - Social and Behavioral Sciences*, 96(Cictp):1208–1217, 2013. ISSN 18770428. doi: 10.1016/j.sbspro.2013.08.138. URL http://dx.doi.org/10.1016/j.sbspro.2013.08.138.

[33] R. M. Jorgensen, J. Larsen, and K. B. Bergvinsdottir. Solving the dial-a-ride problem using genetic algorithms. *Journal of the Operational Research Society*, 58(10):1321–1331, 2007. ISSN 14769360. doi: 10.1057/palgrave.jors.2602287.

[34] Ece Kamar and Eric Horvitz. Collaboration and shared plans in the open world: Studies of ridesharing. *IJCAI International Joint Conference on Artificial Intelligence*, pages 187–194, 2009. ISSN 10450823.

[35] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598): 671–680, 1983.

[36] Imke C. Kleinbekman, Mihaela Mitici, and Peng Wei. Rolling-Horizon Electric Vertical Takeoff and Landing Arrival Scheduling for On-Demand Urban Air Mobility. *Journal of Aerospace Information Systems*, 17(3):150–159, 2020. ISSN 2327-3097. doi: 10.2514/1.i010776. URL https://doi.org/10.2514/1.I010776.

[37] Natalia Kliewer, Taïeb Mellouli, and Leena Suhl. A time-space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175(3):1616–1627, 2006. ISSN 03772217. doi: 10.1016/j.ejor.2005.02.030.

[38] Lee W. Kohlman and Michael D. Patterson. System-level urban air mobility transportation modeling and determination of energy-related constraints. *2018 Aviation Technology, Integration, and Operations Conference*, pages 1–38, 2018. doi: 10.2514/6.2018-3677.

[39] Monirehalsadat Mahmoudi and Xuesong Zhou. Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state-space-time network representations. *Transportation Research Part B: Methodological*, 89:19–42, 2016. ISSN 01912615. doi: 10.1016/j.trb.2016.03.009.

[40] Felim McGrath. The demographics of ubers us users, 2017. URL https://blog.globalwebindex.com/chart-of-the-day/uber-demographics/.

[41] Fei Miao, Shuo Han, Shan Lin, Qian Wang, John A. Stankovic, Abdeltawab Hendawi, Desheng Zhang, Tian He, and George J. Pappas. Data-Driven Robust Taxi Dispatch under Demand Uncertainties. *IEEE Transactions on Control Systems Technology*, 27(1):175–191, 2019. ISSN 1558-0865. doi: 10.1109/TCST.2017.2766042.

[42] Nasa Urban Air Mobility. Urban air mobility (uam) market study, 2018. URL https://www.nasa.gov/sites/default/files/atoms/files/uam-market-study-executive-summary-v2.pdf.

[43] Yves Molenbruch, Kris Braekers, and An Caris. Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, 259(1-2):295–325, 2017. ISSN 15729338. doi: 10.1007/s10479-017-2525-0.

[44] Mariella Moon. Dubai tests a passenger drone for its flying taxi service. URL https://www.engadget.com/2017/09/26/dubai-volocopter-passenger-drone-test/. Access date: 09.03.2020.

[45] William P. Nanry and J. Wesley Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121, 2000. ISSN 01912615. doi: 10.1016/S0191-2615(99)00016-8.

[46] Michael D. Patterson, Kevin R. Antcliff, and Lee W. Kohlman. A proposed approach to studying urban air mobility missions including an initial exploration of mission requirements. *Annual Forum Proceedings - AHS International*, 2018-May, 2018. ISSN 15522938.

[47] Marco Pavone, Stephen L. Smith, Emilio Frazzoli, and Daniela Rus. Robotic load balancing for mobility-on-demand systems. *International Journal of Robotics Research*, 31(7):839–854, 2012. ISSN 02783649. doi: 10.1177/0278364912444766.

[48] Gregory L. Plett. *Battery Management Systems, Volume II: Equivalent-Circuit Methods*. Artech House, 2016. ISBN 9781630810276.

[49] Jean-Yves Potvin and Michel Gendreau. *Handbook of Metaheuristics Associate Series Editor*. 2019. ISBN 9783319910857.

[50] Harilaos N. Psaraftis. Dynamic Programming Solution To the Single Vehicle Many-To-Many Immediate Request Dial-a-Ride Problem. *Transportation Science*, 14(2):130–154, 1980. ISSN 00411655. doi: 10.1287/trsc.14.2.130.

[51] Harilaos N. Psaraftis, Min Wen, and Christos A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):331, 2016. ISSN 0028-3045. doi: 10.1002/net.21628.

[52] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006. ISSN 15265447. doi: 10.1287/trsc.1050.0135.

[53] Laporte G Ropke S, Cordeau J.F. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 60(1):45–58, 2007. ISSN 1097-0037. doi: 10.1002/net.

[54] Bruno F. Santos. Demand models and market share. University Lecture, 2018.

[55] M.V.P. Savelsbergh and M. Sol. The general pick-up and delivery problem. (1):17 – 29, 1995. doi: http://dx.doi.org/10.1287/trsc.29.1.17.

[56] Verena Schmid. Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research*, 219(3):611–621, 2012. ISSN 03772217. doi: 10.1016/j.ejor.2011.10.043. URL http://dx.doi.org/10.1016/j.ejor.2011.10.043.

[57] Nicola Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers and Operations Research*, 27(11-12):1201–1225, 2000. ISSN 03050548. doi: 10.1016/S0305-0548(99)00146-X.

[58] Paul Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In Michael Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming — CP98*, pages 417–431, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-49481-2.

[59] Hanif D. Sherali, Ebru K. Bish, and Xiaomei Zhu. Airline fleet assignment concepts, models, and algorithms. *European Journal of Operational Research*, 172(1):1–30, 2006. ISSN 03772217. doi: 10.1016/j.ejor.2005.01.056.

[60] Syed A M Shihab and Peng Wei. By Schedule or On-Demand ? - A Hybrid Operational Concept for Urban Air Mobility Services. pages 1–13.

[61] Pawel Sitek and Jarosław Wikarek. Capacitated vehicle routing problem with pick-up and alternative delivery (CVRPPAD): model and implementation using hybrid approach. *Annals of Operations Research*, 273(1-2):257–277, 2019. ISSN 15729338. doi: 10.1007/s10479-017-2722-x. URL https://doi.org/10.1007/s10479-017-2722-x.

[62] Eiichi Taniguchi and Hiroshi Shimamoto. Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. *Transportation Research Part C: Emerging Technologies*, 12(3-4 SPEC.ISS.):235–250, 2004. ISSN 0968090X. doi: 10.1016/j.trc.2004.07.007.

[63] David P. Thipphavong, Rafael D. Apaza, Bryan E. Barmore, Vernol Battiste, Christine M. Belcastro, Barbara K. Burian, Quang V. Dao, Michael S. Feary, Susie Go, Kenneth H. Goodrich, Jeffrey R. Homola, Husni R. Idris, Parimal H. Kopardekar, Joel B. Lachter, Natasha A. Neogi, Hok K. Ng, Rosa M. Oseguera-Lohr, Michael D. Patterson, and Savita A. Verma. Urban air mobility airspace integration concepts and considerations. *2018 Aviation Technology, Integration, and Operations Conference*, 2018. doi: 10.2514/6.2018-3676.

[64] Finbarr Toesland. Five futuristic modes of transport transforming travel. URL https://www.raconteur.net/business-innovation/five-futuristic-transport-modes.

[65] Vigo D. Toth, P. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 3600 University City Science Center Philadelphia, PA, United States, January 2001. ISBN 978-0-89871-498-2.

[66] UN DESA. *World Population Prospects 2019*. Number 141. 2019. ISBN 9789211483161. URL http://www.ncbi.nlm.nih.gov/pubmed/12283219.

[67] Parker D. Vascik and R. John Hansman. Constraint identification in on demand mobility for aviation through an exploratory case study of Los Angeles. *17th AIAA Aviation Technology, Integration, and Operations Conference, 2017*, (June 2017):1–25, 2017. doi: 10.2514/6.2017-3083.

[68] Alex Wallar, Menno Van Der Zee, Javier Alonso-Mora, and Daniela Rus. Vehicle Rebalancing for Mobility-on-Demand Systems with Ride-Sharing. *IEEE International Conference on Intelligent Robots and Systems*, pages 4539–4546, 2018. ISSN 21530866. doi: 10.1109/IROS.2018.8593743.

[69] Donghyung Yook and Kevin Heaslip. Effective modeling for a distance-based fare structure with a time-expanded network. *Journal of Public Transportation*, 18(1):1–13, 2015. ISSN 1077291X. doi: 10.5038/2375-0901.18.1.2.

[70] Kai Zhang and Stuart Batterman. Air pollution and health risks due to vehicle traffic. *Science of the Total Environment*, 450-451:307–316, 2013. ISSN 00489697. doi: 10.1016/j.scitotenv.2013.01.074.