Deliverable 3.4

Final report on the formulation, evaluation, and prototyping of the advanced methods and tools

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101007153.

Project acronym	nuMIDAS
Project title	New Mobility Data and Solutions Toolkit
Project number	Horizon 2020 MG-4-8 – GA No 101007153
Work package	WP3 – Definition of advanced methods and tools
Editor(s) / main author(s)	Chrysostomos Mylonas (CERTH) Dimitris Tzanis (CERTH) Evangelos Mitsakis (CERTH) Maria Stavara (CERTH)
Contributing authors	MAPTM CERTH FACTUAL CTU POLIEDRA AMB INFO AMAT LEUVEN
Reviewer(s)	CTU
Dissemination level	Public
Contractual delivery date	31/01/2022 (M13)
Actual delivery date	30/12/2022 (M24)
Version	v1.0

Document revision history			
Version	**Date**	**Description**	**Editor(s) (Affiliation Short Name)**
v0.1	21/01/2022	Structure and first draft	Chrysostomos Mylonas (CERTH) Maria Stavara (CERTH) Evangelos Mitsakis (CERTH)
v0.2	28/01/2022	Content of Section 2, Section 3, and 4.1 added to the document	Chrysostomos Mylonas (CERTH) Dimitris Tzanis (CERTH) Maria Stavara (CERTH)
v0.3	04/02/2022	Content of Section 4.2, 4.3, and 4.6 added to the document	Maria Stavara (CERTH) Chrysostomos Mylonas (CERTH)
v0.4	11/02/2022	Content of Section 4.4 added to the document	Chrysostomos Mylonas (CERTH) Dimitris Tzanis (CERTH)
v0.5	18/02/2022	Content of Section 4.5 added to the document	Chrysostomos Mylonas (CERTH) Dimitris Tzanis (CERTH)
v0.6	27/02/2022	Content of all remaining section (incl. executive summary and concluding remarks) added to the document – First consolidated draft	Chrysostomos Mylonas (CERTH) Dimitris Tzanis (CERTH) Maria Stavara (CERTH) Evangelos Mitsakis (CERTH) Sven Maerivoet (TML)
v0.7	21/11/2022	Integrating remaining UC3 information	Chrysostomos Mylonas (CERTH) Dimitris Tzanis (CERTH) Maria Stavara (CERTH) Evangelos Mitsakis (CERTH) Sven Maerivoet (TML)
v1.0	30/12/2022	Finalisation of all use cases Final version	Chrysostomos Mylonas (CERTH) Dimitris Tzanis (CERTH) Maria Stavara (CERTH) Evangelos Mitsakis (CERTH) Sven Maerivoet (TML)

Table of contents

List of figures

List of tables

1 Executive summary

Over the past few years, the transport sector is dealing with major challenges attributed to megatrends, such as climate change, shared mobility, and user-eccentricity. Taking into consideration requirements stemming from sustainability and quality of life principles, the need for developing new methods and tools supporting the planning, management, and monitoring of mobility solutions ensuring a well-structured and well-operating mobility system seems more than needed. Hence, the scope of this document is to describe the outcomes dedicated to the formulation, validation, verification, and prototyping of the advanced methods and tools of the nuMIDAS project.

Within this deliverable the overview of each tool is provided, including data needs, data outputs, and targeted users. In addition to that, each tool, which is in a full development stage, is accompanied by a comprehensive flow diagram explaining its algorithmic framework. Moreover, the outputs of these tools are presented and evaluated from a technical and logical perspective by utilizing either sample or real-world data.

In summary, the first tool provides a solution to an optimisation problem involving the fleet size of shared mobility services. Although this is the main output of the tool, there are also several other outputs, such as demand coverage, expected profits, and walking time for several fleet size values. The second tool to be integrated into the nuMIDAS toolkit is focused on the assessment of the spatial extent of a shared mobility service across a territorial unit, i.e., its operative area, with the aim of promoting spatial accessibility and equality respecting simultaneously financial limitations on the side of service operators. Although this description points toward the provision of a solution to an optimisation problem, the second tool mainly aims to provide decision support to policy makers by providing to them a set of indicators encompassing accessibility, equality, and operational cost concerns for different configurations of an operative area. Thus, despite a fact that optimisation is not used for selecting an optimal configuration (this choice is subject to several – potentially subjective – factors), an efficient searching strategy is applied for delimiting the number of configurations reported to policy makers and the required computational time as well. The third tool focuses on the analysis of traffic-related data placing special emphasis on the contribution of traffic to emissions and, thus, the prevailing air quality conditions. The fourth tool of the toolkit aims to support the assessment of traffic-related impacts of on-street parking restriction policies, placing special emphasis of the parking pressure relocation from the area into which the policy is enforced to the adjacent ones. Subsequently, the fifth tool involves the estimation of inflows and outflows between the districts of a metropolitan area (in the form of an OD table) building upon data generated by point-to-point detection systems and vehicle registration databases. Moreover, this tool supports the temporal analysis of data generated by point-to-point detection systems considering vehicle characteristics. To this extend, it provides support to the refinement and proper enforcement of environmental policies and policy instruments (e.g., low emission zones). Finally, the sixth tool aims to support the assessment of traffic management scenarios that may consist of various traffic management measures serving the goals of a traffic management authority. The focus of this tool is not on the analysis of the full details of traffic management scenarios that may be composed by both conventional and C-ITS enabled measures but on the provision of an assessment mechanism that is based on the use of synthetic data as well as the microscopic and macroscopic modelling of traffic.

This document ends up by summarising its outputs and identifies the further steps to be taken. Therefore, this deliverable is deemed critical as it summarises everything that has been done during the development of each tool of the nuMIDAS toolkit.

2 Introduction

2.1 About nuMIDAS

The mobility ecosystem is rapidly evolving, whereby we see the rise of new stakeholders and services. Examples of these are the presence of connected and automated vehicles, a large group of organisations that rally to establish various forms of shared mobility, with the pinnacle being all of these incorporated into a large MaaS ecosystem. As these new forms of mobility offerings start to appear within cities, so do new ways in which data are being generated, collected, and stored. Analysing this (Big) data with suitable (artificial intelligence) techniques becomes more paramount, as it leads to insights in the performance of certain mobility solutions and is able to highlight (mobility) needs of citizens in a broader context, in addition to a rise in new risks and various socio-economic impacts.

Successfully integrating all these disruptive technologies and solutions with the designs of policy makers remains a challenge at current. let alone being able to analyse, monitor, and assess mobility solutions and their potential socio-economic impacts.

nuMIDAS, the New Mobility Data & Solutions Toolkit, bridges this (knowledge) gap, by providing insights into what methodological tools, databases, and models are required, and how existing ones need to be adapted or augmented with new data. To this end, it starts from insights obtained through (market) research and stakeholders, as well as quantitative modelling. A wider applicability of the project's results across the whole EU is guaranteed as all the research is validated within a selection of case studies in pilot cities, with varying characteristics, thereby giving more credibility to these results. Finally, through an iterative approach, nuMIDAS creates a tangible and readily available toolkit that can be deployed elsewhere, including a set of transferability guidelines, thus thereby contributing to the further adoption and exploitation of the project's results.

nuMIDAS, the New Mobility Data and Solutions Toolkit, started at the beginning of 2021 under the Horizon 2020 programme and it is being developed by a European Consortium, composed of 9 partners from 6 countries: Belgium, Czech Republic, Greece, Italy, The Netherlands, and Spain.

The project builds on a distributed selection of case studies in pilot cities to provide a geographic coverage of the EU. The three pilot cities are: Barcelona (Spain), Milan (Italy), and Leuven (Belgium). Thessaloniki has been also added to the pilot cities with the aim of adding to the scope of the project a use case involving traffic management and the use of cooperative technologies.

2.2 Purpose of this document

This document aims to describe the outcomes of the final iteration of the tasks dedicated to the formulation, validation, verification, and prototyping of the advanced methods and tools of the nuMIDAS toolkit. The scope and computational capabilities of these methods and tools derive from the analysis of their use cases and their resulting requirements settled into D2.2. Additional correlation also exists with D3.1 providing that the parameters which are considered critical to be incorporated into these tools are described in detail within D3.1. Moreover, in D3.1 a list of risks is included that feed the scope and functional variations of the project's toolkit. The current deliverable focuses, firstly, on the description of the features and computational structure and capabilities of the methods and tools of which the project's toolkit is comprised. Secondly, it focuses on their testing aiming to ensure their proper functioning and the provision of meaningful/rational results. The developed methods and tools will be further evaluated in the context of WP4 utilizing real-world data from pilot cities.

Furthermore, it should be noticed that this document constitutes the final deliverable of the advanced methods and tools formulation, validation, verification, and prototyping. Two more deliverables will successively follow. A textual and schematic description/representation as well as the results from the application of the methods and tools responding to the requirements related to a) the pre-planning of shared mobility services (UC1), the analysis of operative areas of mobility services (UC2), c) the analysis of air quality (UC3), d.) the assessment of parking restrictions policy impacts (UC4), e) the estimation of inflows and outflows in a metropolitan areas (UC5), and f) the assessment of traffic management scenarios (UC6) are thoroughly discussed and analysed within this deliverable.

2.3 Structure of this document

Apart from the introduction and the short overview of the nuMIDAS project included in the second chapter, this deliverable consists of three additional chapters the content of which is described below.

Chapter 3 sets out the methodological framework upon which the development and functional prototyping of the project's methods and tools is based. This chapter also sets out the adopted methodological framework concerning the testing and evaluation of the methods and tools as well as the assessment of the quality of the code enclosed in each functional prototype.

Chapter 4 constitutes the core chapter of the current deliverable. It provides an extensive description of each tool in a different chapter. Apart from a brief overview of each tool, including its purpose, scope, and operational structure, such a description also focuses on both the involved users of the tool and the prerequisite data inputs. Afterwards, the outputs of each tool are listed and analysed in brief. Next, the computational flow of each tool is analysed in detail through schematic representations, verbal descriptions, and/or the presentation of its mathematical background and formulation. The results of the code quality control assessment are provided as well. Finally, the outputs of each tool are demonstrated with the use of sample input. Such input is differentiated in the context of test scenarios with the aim of serving the purpose of the evaluation process of each tool.

It should be noted that specific reference is made to new features embedded in the tools that were under full development and described in detail in the context of D3.1. This is done by interlaying a dedicated sub-section (titled “Additional features”) between the ones revolving around the computational flow and code control. Hence, both the quality control subsection and the presentation/discussion of each tool’s outputs (that follows) consider and demonstrate any additional new features embedded.

The structure analysed above concerns the tools, which have been already prototyped and as such they are in the first completed phase of their development. The correlation analysis mentioned in the previous section is provided in Annex I. The next version of the deliverable will include a full description for all tools complying to the structure adopted for the description of the already developed tools. Chapter 5 sums up, discusses the main outcomes of the advanced methods and tools formulation, validation, verification, and prototyping, and sets out future research directions.

2.4 Acronyms

ANPR	Automatic Number Plate Recognition
C-ITS	Cooperative Intelligent Transport Systems
CO	Carbon Monoxide
CO_2	Carbon Dioxide
EC	European Commission
FD	Fundamental Diagram (of traffic flow)
GA	Grant Agreement
GIS	Geographic Information System
HC	Hydrocarbon
ITS	Intelligent Transport Systems
LEZ	Low-Emission Zone
LUR	Land-Use Regression
MFD	Macroscopic Fundamental Diagram
NOx	Nitrogen Oxide
nuMIDAS	New Mobility Data and Solutions Toolkit
PM	Particulate Matter
SO_2	Sulphur Oxide
WP	Work package
UC	Use case
UVAR	Urban Vehicle Access Restrictions

3 Adopted approach

3.1 Formulation and prototyping approach

The current chapter sets out the methodological approach followed with regards to the formulation and prototyping of the advanced methods and tools of the nuMIDAS toolkit. Based on this methodology, the problem to be solved by each tool acquires a delimited and formulated structure through appropriate parameters and mathematical relationships.

The methods and tools of the toolkit consider as a prerequisite the requirements derived from the analysis of the toolkit uses cases (D2.2) and, in turn, the needs of the pilot cities[1]. Furthermore, the parameters identified as critical in D3.1 are considered during the methods and tools formulation. Along these lines, the conceptual framework of the problem to be solved by each of the toolkit's element is defined, paving the ground for its mathematical formulation.

Furthermore, considering potential risks that could complicate the development process of each tool and hamper its usefulness and usability, the adopted formulation and prototyping approach considers the list of risks identified in D3.1, along with appropriate techniques to mitigate these risks. For instance, multiple versions of each tool are conceptualised with each of them requiring varying level of input as a means of accounting for risks related to data availability. Once a tool version is identified as stable and able to tackle various risks, this version is adopted.

Subsequently, having defined the conceptual framework of each problem to be solved, the next step includes its mathematical formulation. This is achieved by using spreadsheets and sample values, supporting its delimitation to a typical example. Afterwards, both the conceptual framework and the solution spreadsheet are translated into programming scripts. The first runs of these scripts are performed by providing the same input values with those utilised in the solution spreadsheets. Afterwards, the range of provided input is increased, while new capabilities are added to better address the requirements resulting from use case analysis as well as to increase the validity of the tool's outputs. Moreover, in all the iterative versions of programming scripts, new challenges derived from the needs of pilot cities are taken into consideration. Since the computational algorithm included in each tool is addressed as having reached an acceptable level, the process continues with the execution of comprehensive test scenarios and the assessment of derived outputs.

By modifying the existing tools, until the desired outcome meets the project's expectations, nuMIDAS consortium takes the project's output one step further. This involves the enhancement of the practical utility of the computational algorithms based on the accumulated experience from user testing endeavours. In this respect, adjustments and incorporation of new features becomes increasingly smooth, efficient, and evidence based.

[1] This is based on the premise that the toolkit's use cases have been designed following a pragmatic approach. Such an approach involved the conceptualisation of each use case content considering the needs of the pilot cities of nuMIDAS and other cities as well with the aim of enhancing their transferability.

3.2 Code quality control process

Code quality control is typically achieved through the calculation of specific metrics based on which the quality of the code is classified as “low,” “moderate,” “high,” or any other level (e.g., “extremely low” or “very high”) (Bellairs, 2019). For instance, a code that follows a well-documented structured that is easily readable/understandable and highly testable is considered as of high quality (Plösch, et. al., 2010). On the other hand, a code that follows an unstable structure is considered as of lower quality. Keeping code quality high is essential and a significant prerequisite for preventing the extraction of non-valid outputs. The most efficient approach to assure the quality of the code constitutes its regular testing during the various stages of the development process in line the so-called life cycle of code quality control. The quality control life cycle is an ongoing and iterative process involving the planning, monitoring, assessment, comparison, correction, and improvement of a code block that serves a specific purpose.

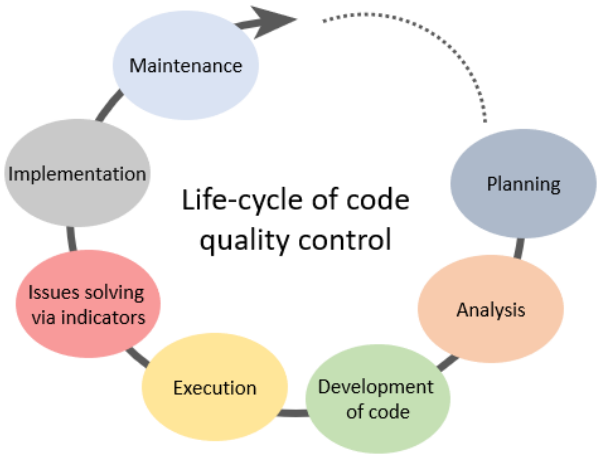


Figure 1: Life cycle of code quality control

In the planning stage, well-structured thoughts should be established to achieve a deep understanding of the purpose that a code block serves. Next, in the analysis phase, the definition of a code block’s requirements takes place. The design and the development of the code follows. In this phase, all requirements stemming from the previous stages are functionally translated and prototyped. After the completion of the development, the next step includes the execution of the code to evaluate its operability. A further step constitutes the identification and resolution of bugs through the calculation of appropriate metrics providing relevant information to the developer. Once the code is error-free the implementation phase follows, including semantics enhancement and object-oriented approaches. The final stage, which is an ever-lasting one is the maintenance of the code. The processes included in this stage stem from the need of continuously assessing and incorporating evolving needs arising from the from the long-term usage of a code block. In case that new needs have arisen, the life cycle starts again from the first phase.

Code quality control in the context of nuMIDAS toolkit is achieved via the use of dedicated libraries, such the Radon library designed for Python. This library computes various metrics, including raw metrics, metrics related to cyclomatic complexity, halstead metrics, and maintainability metrics. Raw metrics are useful for calculating the number of logical lines of code (LLOC) and the number of source lines of code (SLOC) (Hirayama, et. al., 1990). Cyclomatic complexity is used to indicate the complexity of a code. It is a quantitative metric aiming to assess the number of linearly independent paths within the source code. Halstead metrics, on the other hand, aim to measure the program module's complexity. Halstead metrics are a) Code Volume (V), b) Code Level (L), c) Code Difficulty, d) Code Effort (E), e) Estimated Code Length, f) Code vocabulary, g) Code Time, h) Language Level, and i) Intelligence Content. Lastly, maintainability metrics represents the relative ease of maintaining the code. A high value indicates better maintainability.

3.3 Evaluation process

With the aim of enhancing the toolkit's processing capabilities and the validity of its results, it is deemed needed to include in the workflow of the toolkit development an evaluation process. According to the adopted approach described in Section 3.1, before the development of the programming script, the conceptual framework of a problem to be solved is translated into mathematical expressions. Such a step facilitates the better understanding of the problem to be solved but do not purely serve the purpose of evaluating in a broader context the validity of each tool. This can be achieved through logical examples with a wide range of input values. By that means, a range of test scenarios are executed targeting to examine whether the outcomes are realistic, or the code needs further improvement. For instance, bearing in mind the scope of the first use case, a logical experiment is to increase the value of the demand parameter. In such a case, it is expected from the tool to also increase the value of the optimal fleet size. In a similar fashion, in case of increasing the area parameter, it is expected from the tool to increase the average walking time of the users. This process, which is also called validation testing, is important to ensure the rationality of the results.

Apart from the validation process, there is also the verification process which evaluates whether the tool's specifications fulfil the requirements and the design standards or not. In both evaluation processes, further discussions among project stakeholders are necessary to ensure that the entirety of envisaged functionalities are incorporated. In short, the evaluation process is one of the main contributors to the continuity of a tool.

4 Formulation, evaluation, and prototyping

As already noted in Section 2.2, the current deliverable aims to describe the outcomes of the formulation, evaluation, and prototyping of the six methods and tools of which the project's toolkit is comprised. This description is the focus of the current chapter. Furthermore, in the context of this deliverable six out of the six methods and tools are reported as being under full development, while the remaining one will be incorporated into the final version of the same deliverable. For all the methods and tools a comprehensive process flow diagram is provided. The scope of the methods and tools of the project's toolkit has been determined during its use case analysis being the focus of D2.2. Table 1 provides a summary of this scope.

Table 1: Brief description of the nuMIDAS toolkit's use cases.

ID	Title	Status	Brief description
UC1	Preplanning of shared mobility services	Full development	Supports the definition of the optimal fleet size of shared mobility services (e.g., shared bikes, scooters) taking as input socioeconomic, mobility, financial, and service provision-related parameters and constraints.
UC2	Operative areas analysis	Full development	Supports the assessment of the spatial extent of shared mobility services considering a) the fact that several sub-areas of a larger – metropolitan – area exhibit higher or lower mobility demand, b) the need to minimise the economic losses of service providers and ensure an acceptable level of service for end users, and c) the principles of inclusive and equitable transport systems.
UC3	Air quality analysis and forecasting	Full development	Supports the analysis of air quality focusing on vehicle-induced emissions, considering the correlation of multi-source data (including air quality-, traffic-, and weather-related data) and, thus, contributing to the evidence-based formulation and enforcement of access control and other emission-related policies and policy instruments.
UC4	Planning for parking	Full development	Supports the assessment of the impact of reducing on-street parking space, thus supporting the formulation and enforcement of relevant policies and policy instruments. Such impacts will focus on the parking pressure relocated to adjacent areas and generalised cost increases expensed by road users.

UC5	Inflows and outflows in a metropolitan area	Full development	Supports the estimation of inflows and outflows between the districts of a metropolitan area based on data generated by point-to-point detection systems (i.e., Automated Number Plate Recognition systems) as well as of census data (i.e., vehicle registration). The ultimate purpose is to support the refinement and proper enforcement of environmental policies and policy instruments (e.g., low emission zones). Besides inflows and outflows, the analysis of raw data (corresponding to point-to-point detection) related with census data are on the spotlight.
UC6	Assessment of traffic management scenarios	Full development	Focuses on the assessment of traffic management measures being building blocks of dynamic traffic management scenarios/strategies by exploiting various types of synthetic data generated from traffic simulators providing microscopic and macroscopic insights into traffic flow conditions and other aspects of interest (e.g., emissions).

4.1 Pre-planning of shared mobility services

4.1.1 Overview

The first tool of the nuMIDAS toolkit is aimed at improving the process of issuing tenders for shared mobility services, including various types of services, such bike-sharing, car-sharing, (kick-)scooter-sharing services, by supporting the determination of an optimal value for their fleet size given a certain level of demand and characteristics of the area of interest. The main output of this tool is considered as an important intermediate step towards the definition and enforcement of proper fleet management policies considering the long-established practice of service operators to rebalance their fleet to meet successfully existing demand (Shaheen and Cohen, 2016; Nikitas, 2019). In this respect, if the fleet size is a priori inadequate the effectiveness of such policies will most probably prove ineffective, hampering both the financial viability and the level of provided services. In contrast, an excessive value for the fleet size will, on the one hand, provide an increased level of service but, on the other hand, significantly question the financial viability of provided services. Furthermore, an excessive value for the fleet size may lead to an excessive and superfluous use of public space. As it appears, there are two perspectives from which the optimal value for the fleet size should be identified, reflecting both the level of service and the financial viability of provided services. In the context of this tool, these perspectives are understood and termed as the “perspective of end users” and the “perspective of service operators.” What is more, providing that policy makers should in principle achieve a balance between these perspectives, the tool enables its users to find an optimal compromise.

The above analysis indicates that the first tool of the nuMIDAS toolkit provides a solution to an optimisation problem. This problem involves the identification of a value for the fleet size of a mobility service that jointly maximises demand coverage and the profitability of service operators considering both its stochastic and multi-periodic dimensions. The multi-periodic dimension concerns the number of vehicle trips provided as an input to the calculations, which is addressed in an hourly basis. The stochastic dimension concerns the specific time within an hour at which each individual traveller is considered to be in need of a vehicle, which is approximated through probability distributions and/or empirical data. The stochastic dimension also concerns the estimation of the spatial distribution of demand as well as the spatial distribution of either vehicles or stations depending on the type of service to be analysed (i.e., station-based, or free-floating). The tools reports to its users, apart from the value of the optimal fleet size, several indicators related to demand coverage, expected profits of service operators, walking time, and waiting time.

4.1.2 Targeted users

The stakeholders involved in the ecosystem of the first tool are the following:

- Mobility service operators (including micromobility service operators, bike-sharing service operators, and car-sharing service operators)
- Departments of local governments tasked with issuing tenders for service (policy makers)
- Transport planners supporting policy makers
- Travellers (end users)

However, among the above stakeholders the ones that belong to the targeted users of the tool are transport planners and policy makers. The formers are understood as the ones providing input to the tool and the latter as the ones receiving the outputs of the tool.

4.1.3 Data inputs

The inputs provided to the tool can be divided into values imported from a file or database and user defined. The former includes:

- Historical demand data (trips/day hour)

As it is described in Section 4.1.5, the above input is used by the tool to calculate demand factors corresponding to each daily interval (day hour). Given that a) the purpose of the first tool is to support the process of issuing tenders for shared mobility services through their optimal fleet sizing and b) the tool requires multi-periodic demand input, it is not expected from its user to know in advance the demand profile of the analysed mobility service. To this end, the tool prompts the user to upload a file including multi-periodic trip data stemming either a) from the operation of the same mobility service within an area equivalent (in socioeconomic terms) with the area of interest or b) from the operation of a comparable shared mobility service within the area of interest. Alternatively, the tool can communicate with a knowledge base (database) providing historical trip data from the operation of the same shared mobility service in other geographical areas. Furthermore, in line with the outcomes of D3.1 concerning the risk of data availability, this type of input is optional. If the user selects to not provide historical trip data (of either form, the tool makes use of prespecified values. The process for determining these values is described in Section 4.1.5.

On the other hand, the user defined input includes the following:

- Expected daily demand (trips/day)
- Selection of the type of service to be analysed (station-based or free-floating)
- Selection of the type of mode to be analysed (bike, scooter, kick-scooter, or car sharing)
- Size of the area of interest (in km^2)
- Operating cost per vehicle per minute (in Euros)
- Expected revenues per minute of rent (in Euros)
- Average users walking speed (km/h)
- Mean trip duration (in minutes)
- Weighting factors assigned to service operator's and end-user's perspective
- Minimum and maximum value of the fleet size

As mentioned in Section 4.1.1, the types of mobility services that can be analysed by the tool are discerned into station-based and free-floating services. To this end, the user is prompted to select the value of a pre-specified parameter affecting the computational flow of the tool. Moreover, with the aim of supporting the estimation of the demand profile to be fed to calculations, the user is prompted to select the type of analysed mode. The remaining inputs included in the above list are provided in numerical form and are used for quantifying directly the variables included in the algorithmic framework of the tool.

4.1.4 Data outputs

The outputs of the tool in its current version are the following:

- Optimal fleet size
- Optimal fleet size from the operator's perspective
- Optimal fleet size from the end user's perspective
- Demand coverage corresponding to the optimal solution
- Profits corresponding to the optimal solution
- Walking time corresponding to the optimal solution
- Waiting time corresponding to the optimal solution (if applicable)
- Demand coverage for several fleet size values
- Profits for several fleet size values
- Walking time for several fleet size values
- Waiting time for several fleet size values (if applicable)

As it becomes readily observable by the above list, the optimal fleet size constitutes the major output. However, the tool provides additional outputs with the aim of informing policy makers concerning a) the variation of the decision variables for several fleet size values (i.e., the minimum and maximum values of the fleet size provided as an input by the user) and b) the values of relevant KPIs (e.g., walking and waiting time) corresponding both to the optimal solution and several fleet size values. It should be noted that the outputs included in the above list are calculated through the quantification of other variables that may be viewed as intermediate outputs.

These outputs are quantified either iteratively of after the completion of a computational loop and they include the following:

- Served demand
- Expected revenues
- Operating costs

Served demand correspond to the net amount of demand that is served assuming the operation of a certain fleet size within the area of interest. Expected revenues are a by-product of served demand, while operating costs are a by-product of the fleet size. Expected revenues and operating costs are used for the quantification of the profits of service operators.

4.1.5 Computational flow

As already mentioned in Section 4.1.1, the first tool of the nuMIDAS toolkit provides a solution to an optimisation problem involving the fleet sizing of shared mobility services. In the context of these services, end users arrive at rental stations to rent a vehicle or book a vehicle via an app at some point in time, they use it for some amount of time, and they either return it in a nominated place (e.g., another or the same station) or drop it off near their destination. In line with the suggestions of Sayarshad et al. (2012) the fleet sizing problem should be formulated considering both the multi-periodic and stochastic dimension of the demand for such services. The multi-periodic dimension implies that the rate of end users arriving at a rental station or requesting to book a vehicle from an app is not constant during a day. In contrast, there are time intervals within a day during which the level of demand is higher (peak period) and time intervals during which the level of demand is lower (off-peak period). The stochastic dimension implies, among others, on the exact time at which users arrive in a rental station or request to book a vehicle is subject to uncertainty. These two dimensions are respected by the first tool of the project's toolkit. Its overall computational flow is schematically represented in Figure 1.

The first step involves the estimation of the daily demand profile. This is achieved through demand factors calculated based on acquired input or implied by the tool itself. In the first case, the acquired input includes historical data from the operation of the same or comparable service within the area of interest or an equivalent area in socioeconomic terms. The demand factor corresponding to each daily interval *i* (day hour) is quantified through the following formula:

$$DF_i = \frac{\sum_j T_{i,j}}{\sum_{i,j} T_{i,j}} \mid i \in [0,24) \; and \; j \in [1, N_{days}]$$

where $T_{i,j}$ denotes the number of trips recorded during hour *i* of day *j* and N_{days} the number of days for which information is available.

In the second case, the demand factors are prespecified through an off-line approach building upon data provided by the pilot city of Milan. Provided data include the number of trips recorded during a period of 324 days for bike, scooter, kick-scooter, and car-sharing mobility services. The shape of the demand profiles and the values of the demand factors per mode are depicted in Figures 3-6. It is noteworthy that there are two peak periods within a day, except for the case of kick-scooter services. Furthermore, in all cases, except for bike sharing, the intensity of travel demand is higher during afternoon hours.

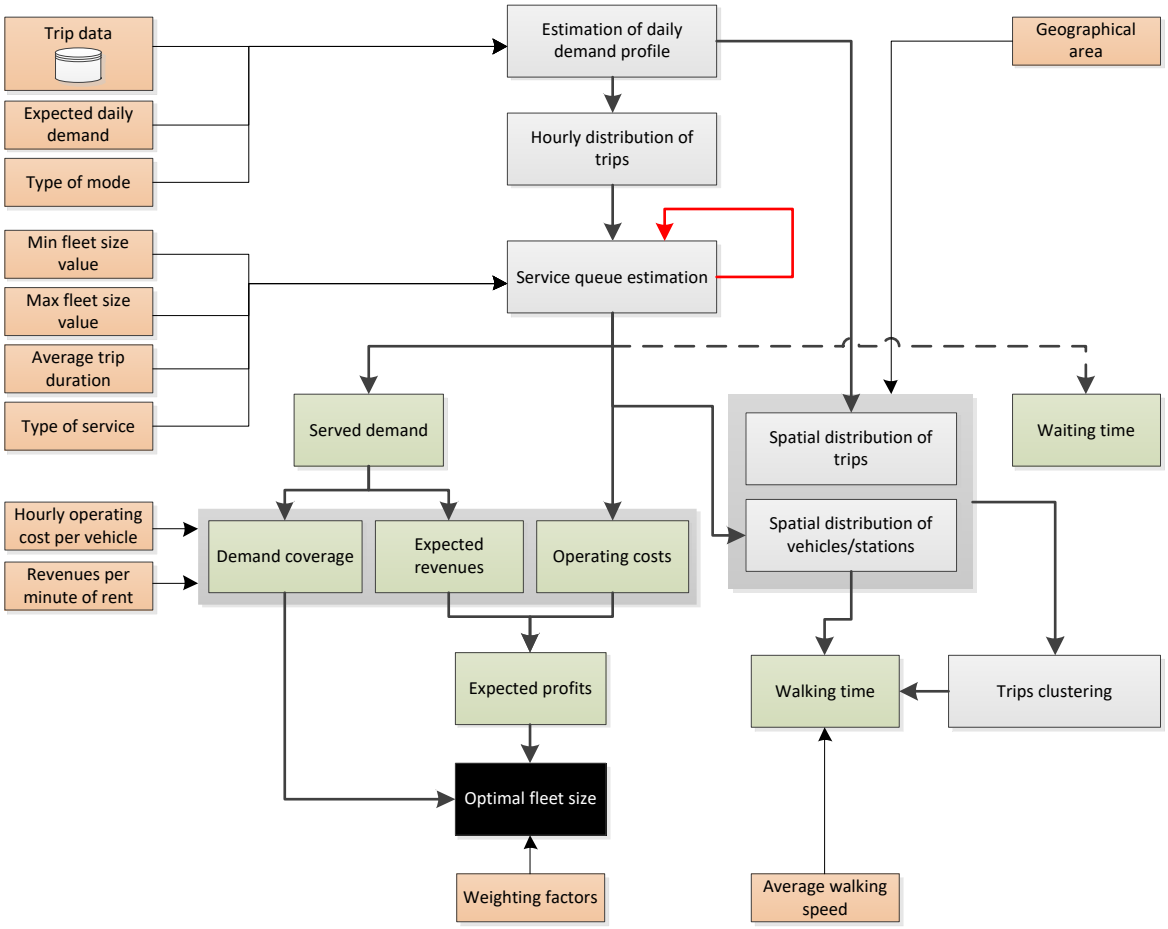


Figure 2: Computational flow of the first tool of nuMIDAS toolkit

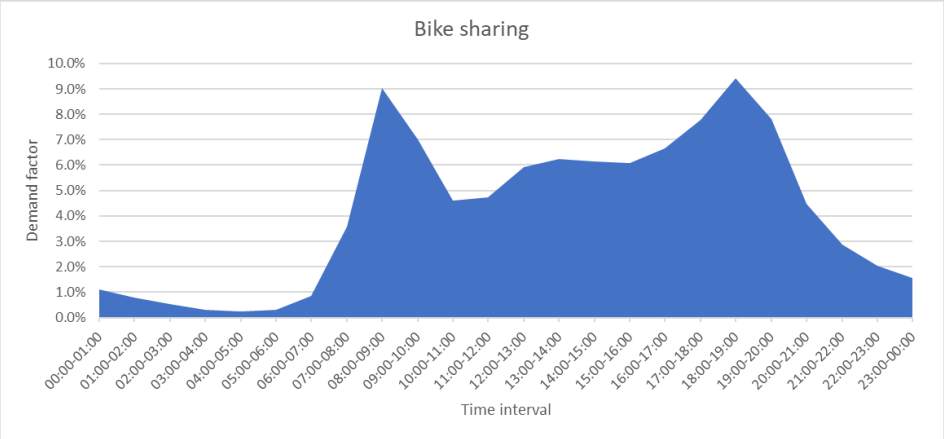


Figure 3: Demand profile of bike sharing services obtained from the pilot city of Milan

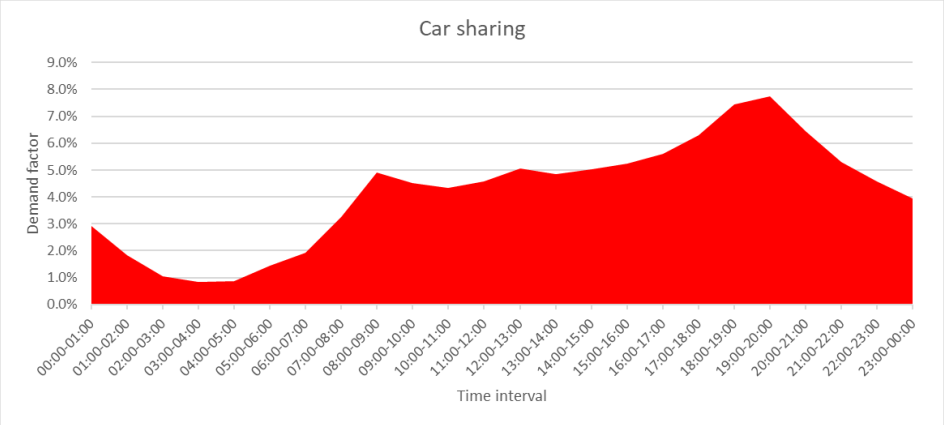


Figure 4: Demand profile of car sharing services obtained from the pilot city of Milan

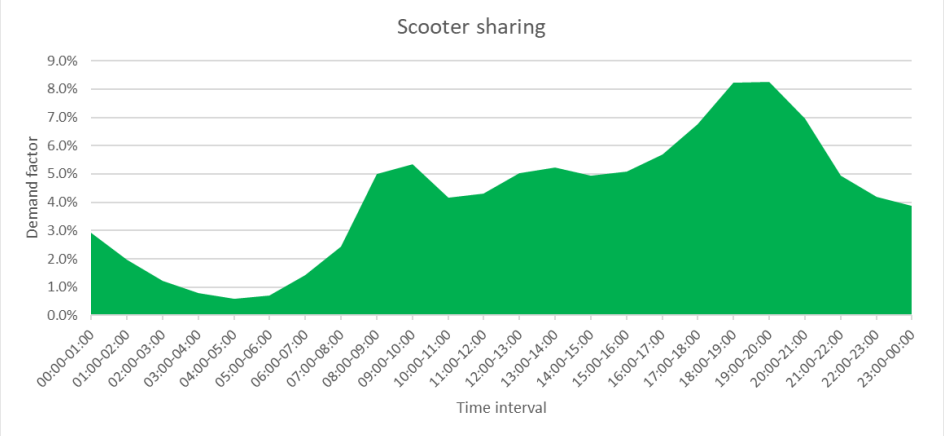


Figure 5: Demand profile of scooter sharing services obtained from the pilot city of Milan

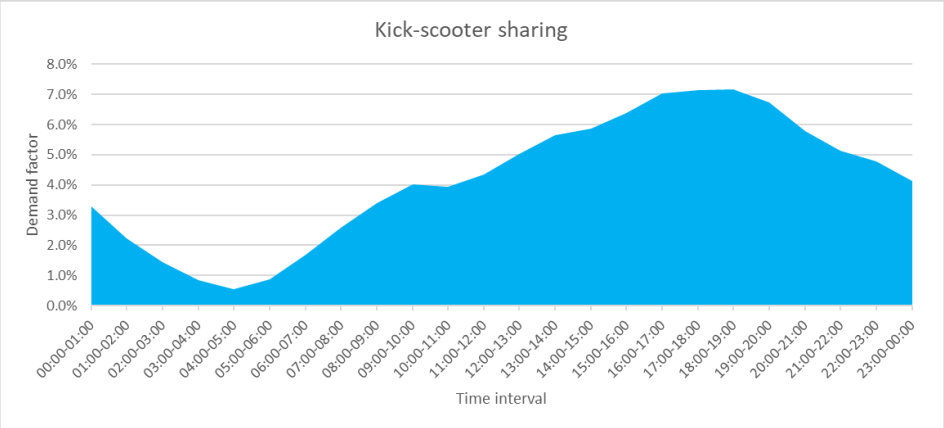


Figure 6: Demand profile of kick-scooter sharing services obtained from the pilot city of Milan

Having defined the demand factors, the demand profile is calculated through the following formula:

$$D_i = DF_i \times Expected\ daily\ demand, i \in [0,24)$$

where D_i denotes the level of demand (trips/hour) corresponding to each day hour *i*.

The next step of the analysis involves the distribution of D_i during each day hour *i*. With the aim of taking into consideration the uncertainty characterizing the arrival rate of end users in stations or the rate with which end users demand to book a vehicle via an app, a uniform probability distribution is used. The choice of this type of probability distribution is based on the analysis of historical data from taxi services in the metropolitan area of Thessaloniki, covering a total period of 3 months. In Figure 7, which depicts the outcomes of this analysis, it is shown that the demand for taxi trips is uniformly distributed in each day hour with minor exceptions. Therefore, the use of such a type of probability distribution appears to be rational.

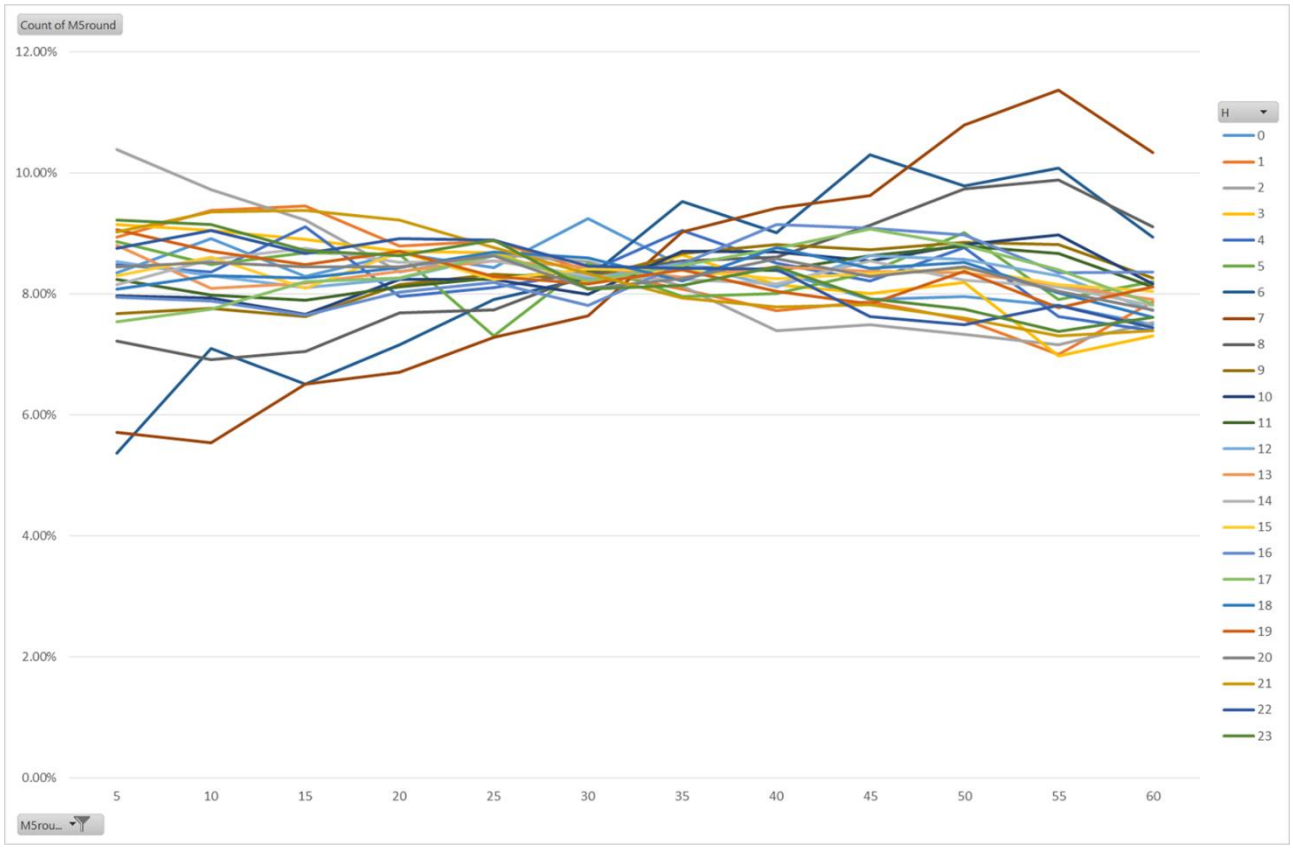


Figure 7: Percentage of taxi trips records during each 5-min interval of each day hour for a period of 3 months (Thessaloniki metropolitan area)

Having distributed the hourly demand, the next step involves the quantification of the number of served demand for each value of the fleet size falling into the range stated by the user (i.e., minimum, and maximum fleet size). This quantification is achieved by adopting a queue theory approach. According to this approach, a demand unit (end user) can be served only in the premise that a vehicle is available. Occupied vehicles become re-available after a period of time equal to the average trip duration. Furthermore, it is assumed that end users who are not served until the upper bound of each daily interval disappear from the queue. By that means, the effect of the provided level of service to the level of demand is taken into consideration (i.e., end users may choose an alternative transport mode if they keep waiting for long). End users are served through the First-In-First-Out (FIFO) principle if the analysed mobility service is station-based. In contrast, end users are randomly served (irrespective of their arrival time) if the analysed mobility service is free-floating. The adopted approach is schematically represented in Figure 8, assuming that the operation of a free-floating shared mobility service consisting of three vehicles. In this figure, three discrete hourly intervals are depicted. In the first hourly interval, three end users are assumed to need a vehicle. Given that all vehicles are available, the entirety of end users are served. In the second hourly interval, five end users are assumed to need a vehicle. By that time, the first three end users have completed their trip and, therefore, three vehicles are available for use. Consequently, three out of five end users are served, while the remaining two end users are considered as waiting in a queue. In third hourly interval, one of the three served users of the previous interval is assumed to have completed his/her trip, while an additional user appears in the queue. Provided that t_4 constitutes the upper bound of the daily interval *i,* one additional end user is served and the remaining two end users in the queue are addressed as not served at all.

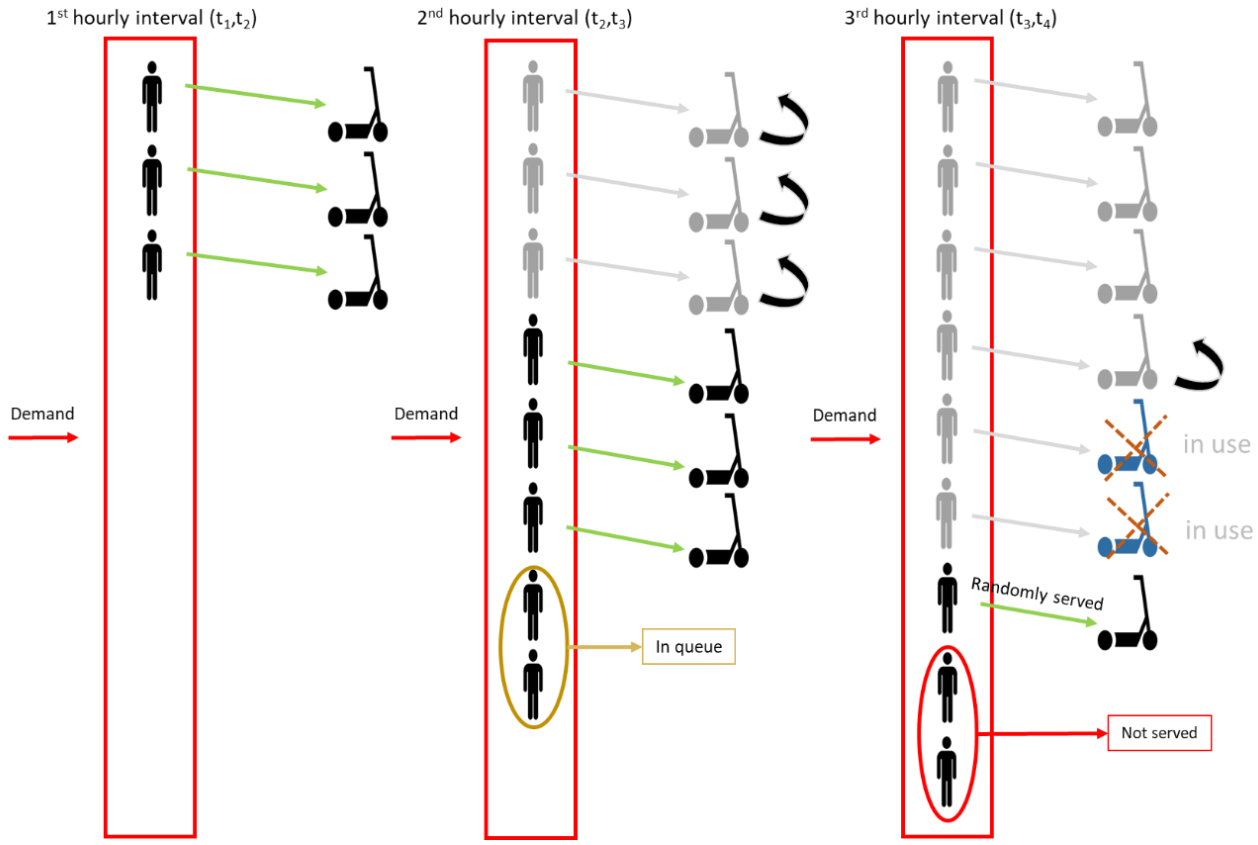


Figure 8: Demonstration of adopted approach for estimating the amount of served demand

From a technical perspective, the amount of served demand is calculated by assessing the value of the trip end time. In particular, when a demand unit appears in time t_i falling into period [t_1, t_4] it is assigned with a trip start time equal to t_1. This demand unit is also assigned with a trip end time equal to infinity. Depending on the availability of vehicles, each demand unit is also assigned with an actual trip start time equal to t_i plus the time needed for a vehicle to become available. If a vehicle is already available when a demand unit appears in time, then the trip's start time and the actual trip start time is the same. The value of the trip end time of served demand units is replaced with a value being equal to the actual start time plus the average trip duration. By that means, the amount of served demand is equal to the number of demand units the trip end time of which is finite. Moreover, the actual trip start time of non-served demand units is replaced with a value being equal to t_4. This technique facilitates the calculation of the waiting time as follows:

$$Waiting\ time_z = \frac{\sum_i \sum_j \frac{Actual\ trip\ start\ time_{i,j} - Trip\ start\ time_{i,j}}{D_i}}{24} \mid i \in [0,24), j \in [1, Di]\ and$$

$$z \in [Min\ fleet\ size, Max\ fleet\ size]$$

Waiting time is only applicable for station-based services on the premise that free-floating vehicles, once available, are typically booked remotely via an app and therefore their users do not need to wait. For this reason, the line connecting service queue estimation and waiting time is dashed in Figure 1. The only generalised cost parameter associated with free-floating services is walking time. Walking time is approximated through the spatial distribution of the maximum value of the hourly demand (hereafter

referred to as maximum hourly demand) as well as the spatial distribution of either vehicles or stations within a geographical area of square shape and size equal with that declared by the user. The number of stations, in station-based services, is estimated via the following rules:

- If the analysed area is less than 5 km^2, then it is assumed that one station hosts up to 10 vehicles
- If the analysed area is greater than 5 km^2 and less than 15 km^2, then it is assumed that one station hosts up to 15 vehicles
- If the analysed area is greater than 15 km^2, then it is assumed that one station hosts up to 20 vehicles

Such a spatial distribution is achieved with the use of a uniform probability distribution for both demand and vehicles or stations. For the case of stations, a constraint is used aiming to ensure that they are placed to a satisfactory extent apart from each other. This constrained is formulated as follows:

$$Distance\ between\ stations\ \geq \frac{\sqrt{A}}{n/2}$$

where *A* denotes the size of the geographical area and the *n* the number of stations.

Having distributed the maximum hourly demand and vehicles or stations in space, an average[2] walking distance is calculated by clustering the demand into an equal number of clusters with the number of stations (if the analysed service is station-based) or with the looped number of fleet size (if the analysed service is free-floating). This is achieved by utilizing Lloyd's k-means hierarchical clustering algorithm in a simplified fashion[3] aiming to ensure that the centroid of each demand cluster coincides with the uniformly (or constrained uniformly) generated position of stations/vehicles. Figure 9 provides a schematic demonstration of the clustering of 372 trips into 6 clusters being equal with the number of stations assumed to be operated within an area extended over 9 km^2. The values of both axis included in Figure 9's graph are in kilometres, i.e., each dot represents the position of each trip or station within the assumed area.

[2] The average value of each and subsequently of all clusters is computed by the tool.

[3] The initial position of the centroids is set to be equal with the position of stations/vehicle, while the maximum number of iterations is set to be equal with 1.

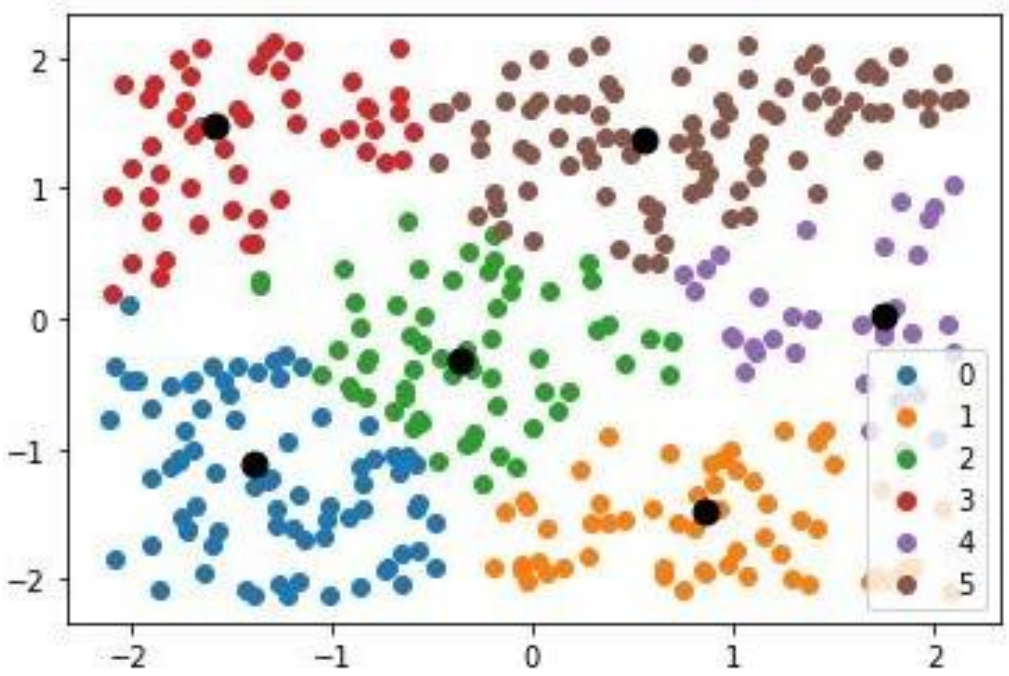


Figure 9: Adopted approach for estimating walking distance

Having estimated the average walking distance of users, walking time is calculated trough the following formula:

$$Walking\ time = \frac{Average\ walking\ distance\ (in\ km)}{Average\ walking\ speed\ (in\ km/h)}$$

The last step of the computational flow involves the calculation of the optimal fleet size. The decision variables used by the tool include the demand coverage and the profitability corresponding to various values of the fleet size. The former is assumed to reflect the perspective of travellers (or end users), while the latter is assumed to reflect the perspective of service operators. From a mathematical point of view, it should be borne in mind that shape of the demand coverage curve closely resembles the shape of a square root function's curve, while the profitability curve is concave. Given that end users always wish to enjoy a greater level of service, the demand coverage curve is monotonically increasing. This does not hold true for the shape of the profitability curve, which is increasing for a lower range of fleet size and decreasing for a higher range of fleet size (Figure 10a). This is attributed to the fact that while service operators require a considerable fleet size to serve demand and thus increase their profits, there is a certain value of fleet size beyond which the profits are decreasing given that the utilisation rate of vehicles gets steadily lower. Furthermore, for considerable high values of fleet size, the profits may become even negative given that the cost of operating a large size of fleet exceeds the revenue margin of service operators.

Moreover, the maximum value of demand coverage as shown in Figure 10a may occur for a value of fleet size leading to negative profits. In this respect, the maximisation of demand coverage concludes to a negative externality for service operators. This situation purely implies the need to incorporate in the tool's algorithm the principles of the general theory of the second best. According to this theory, which was originally postulated by Lipsey and Lancaster (1956) with the aim of analysing how the removal of a market distortion may lead to the introduction of a new market distortion, there is a second-best policy that maximises social welfare from a utilitarian point of view. This can be achieved as suggested by Bennear and

Stavins (2007) through the introduction of an appropriate constraint. In the context of the problem solved by the first tool of the nuMIDAS toolkit, this constraint involves the exclusion from the analysis of the values of the fleet size that leads to negative profits. In this respect, as shown in Figure 10b it is accepted that there is a solution that maximises the welfare of service operators and a second-best solution maximizing to the extent possible the welfare of travellers (end users).

The optimal solution derives from the weighted combination of the fleet size values corresponding to the optimal solution from the operator's perspective and the second-best solution from the perspective of travellers. The overall formulation of the problem is as follows:

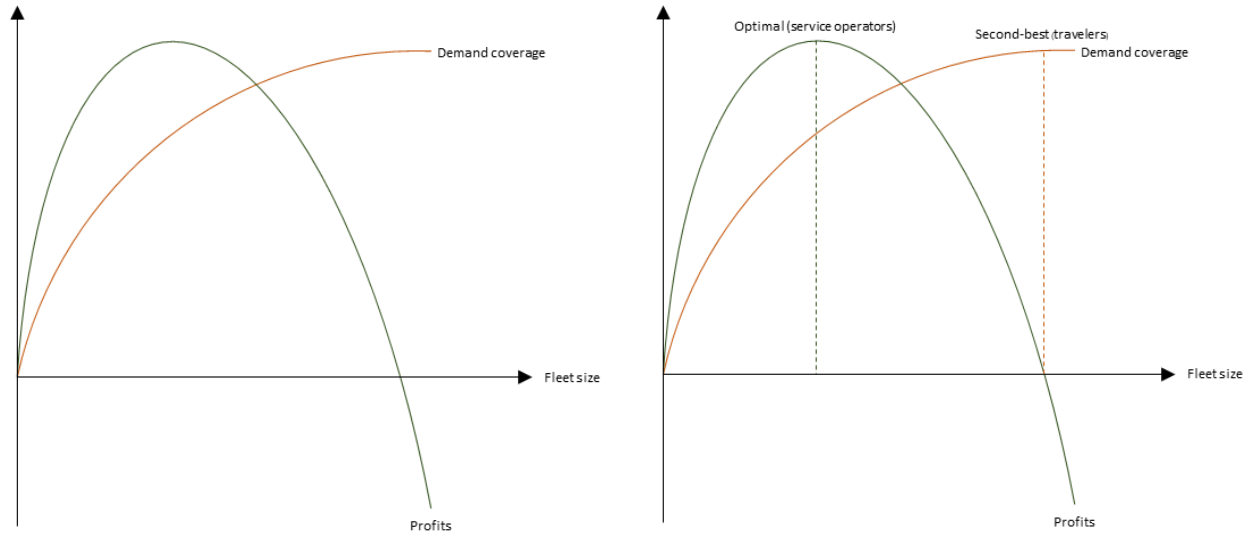
$$\text{maximise } (w_{traveler} * demand\ coverage + w_{service\ operators} * profits)$$

subject to:

$$w_{traveler} + w_{service\ operators} = 1$$

$$profits > 0$$

$$\min fleet\ size\ value \leq fleet\ size \leq \max fleet\ size\ value$$



(a) (b)

Figure 10: Shape of demand coverage and profit curve (a) and the adopted approach involving second-best theory (b)

In the above objective function, demand coverage corresponding to fleet size value *z* is calculated as the ratio of the total served demand to total demand. Similarly, profits corresponding to fleet size value *z* is calculated as the difference between the expected revenues of and costs incurred by service operators.

$$demand\ coverage_z = \frac{total\ served\ demand_z}{total\ demand}$$

$$profits_z = revenues_z - operating\ costs_z$$

$$revenues_z = total\ served\ demand_z * expected\ revenues\ per\ minute\ of\ rent * mean\ trip\ duration$$

$$operating\ costs_z = z * operating\ cost\ per\ vehicle\ per\ minute * 1440$$

4.1.6 Additional features

The additional features that have been embedded in the first tool of the nuMIDAS toolkit involve three main aspects: a) the possibility on behalf of the users of the tool to modify demand profiles, b) demand elasticity, and c) the possibility on behalf of the users of the tool to allow for negative profits. More details are provided below.

The first aspect involves the incorporation of three additional procedures in the computational flow of the first tool that allow users to a) use a demand profile that stems from available mode dependent demand factors, b) to *increase* the demand factors corresponding to the peak hours, or c) to *decrease* (smooth) the demand factors corresponding to the peak hours. The description of the available mode dependent demand factors is provided in Section 4.1.5. On the other hand, the increase of the demand factors corresponding to the peak hours is executed through the following successive steps:

- Sort the entirety of demand factors in descending order
- Select the demand factors that sum up to the 40% of the total demand (i.e., peak factors)
- Multiply by a value equal to 1.5 the selected factors
- Calculate the sum of the increased factors (ΣpDF')
- Calculate the difference between the sum of the increased factors (ΣpDF') and the sum of their prior values (ΣpDF) (ΔpDF = ΣpDF' - ΣpDF)
- Calculate the sum of the unselected (non-peak) factors (ΣnpDF)
- Calculate the weight of each unselected (non-peak) factor on ΣnpDF ($W_{npDF,i}$)
- Calculate the product $W_{npDF,i}$ x ΔpDF for each unselected (non-peak) factor ($PnpDF_i$)
- Set each unselected (non-peak) factor equal to $npDF_i$ - $PnpDF_i$

A similar process is followed in the case that a user wishes to decrease the peak demand factors. The sole difference is that the selected (peak) factors are multiplied by a value equal to 0.8. Figure 11 demonstrates the outputs of the above procedure, including both cases.

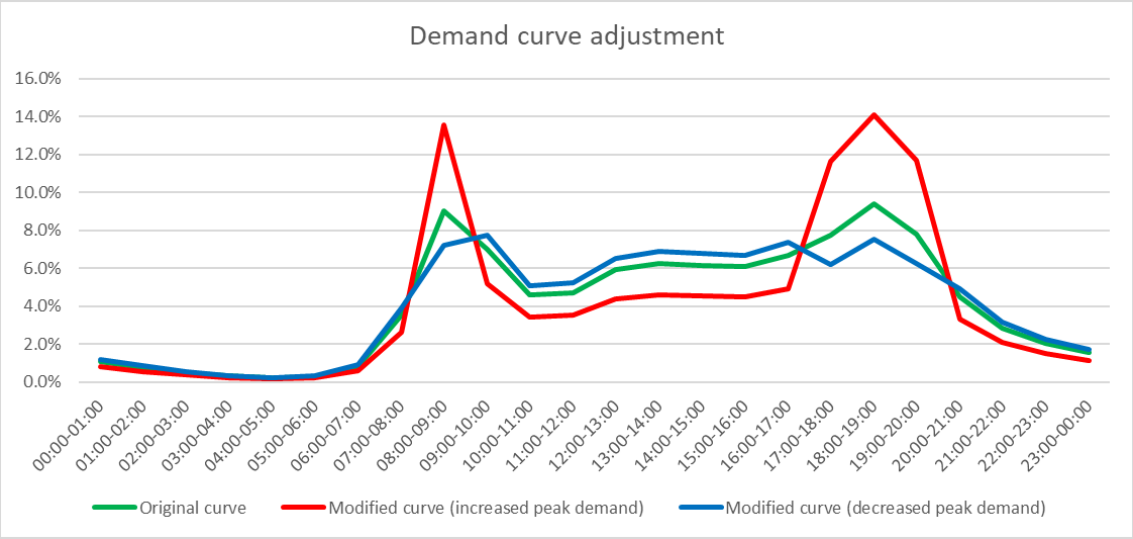


Figure 11: Demand curve adjustment

The second aspect involves, in line with the relevant literature (Narayan et al. 2021), the possibility on behalf of the users of the tool to account for the effect of demand elasticity. In social and economics sciences (Menz et al., 2018), it is generally accepted that a change in the unit price of an offered service or a change in the performance of an offered service (i.e., level of service) affects the demand for that service. In that sense, in the context of the first tool of the nuMIDAS toolkit an increased demand coverage (expressed through an increased fleet size) and, thus, decreased walking and waiting time is assumed to affect the modal split and, in turn, lead to an increased demand. Contrariwise, a decreased fleet size is assumed to lead to a decreased demand. The incorporation of demand elasticity is achieved through the pre-estimation of the optimal fleet size of the analysed service (following a simplified procedure) and subsequently the adjustment of the expected total number of trips to be served. The procedure for pre-estimating the optimal fleet size of the analysed service does not consider the stochasticity in the exact time at which end users arrive at a rental station or request to book a service. Therefore, the simulation of the amount of served demand relying on the adopted queue-theoretic approach (see Figure 8) is not engaged in this pre-estimation. Thus, it is assumed that end users arrive or request to book a service at the same time (i.e., at the start of each hour interval). This procedure enables a fast estimation of the optimal fleet size that is denoted in the remainder of this section as FCrV (Fleet Critical Value). Subsequently, in each iteration of the analytical estimation corresponding to each fleet size value, the expected daily demand (EDD) corresponding to each accounted – current – fleet value (FCurV) is adjusted as follows.

$$EDD' = \begin{cases} EDD + \dfrac{|\mathrm{FCrV} - \mathrm{FCurV}|}{|max\ fleet\ size - min\ fleet\ size|} \times f, \mathrm{FCurV} > \mathrm{FCrV} \\ EDD - \dfrac{|\mathrm{FCrV} - \mathrm{FCurV}|}{|max\ fleet\ size - min\ fleet\ size|} \times f, \mathrm{FCurV} < \mathrm{FCrV} \end{cases}$$

where f denotes a demand scaling factor. Higher values of this factor enable a wider adjustment of the expected daily demand, while lower values enable a narrower adjustment. During the development process of the first tool and after an experimentation stage this value is set to $f = 15$.

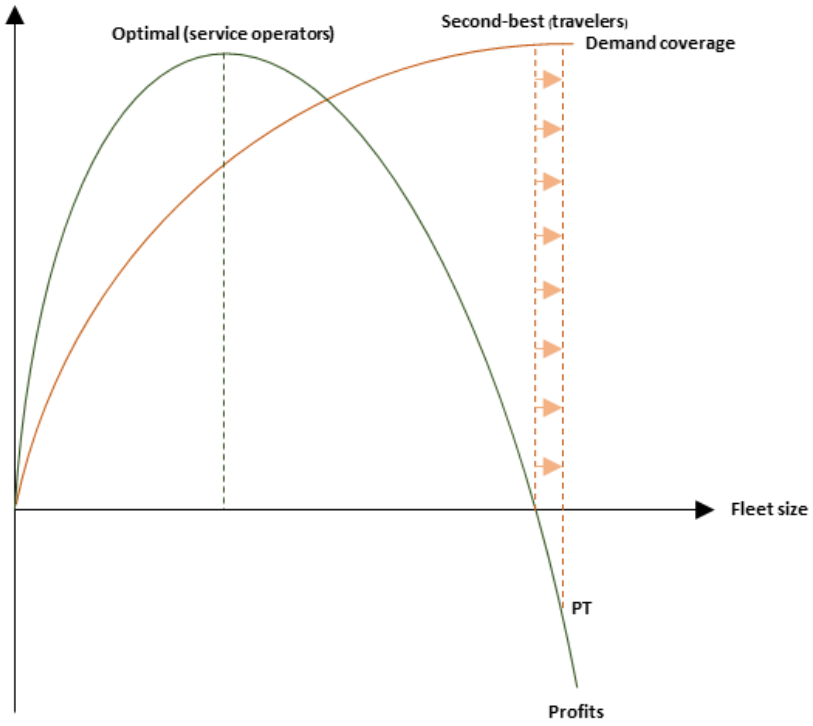


Figure 12: Graphical representation of the use of negative profits within the tool

The final aspect, as already stated, involves the possibility on behalf of the users of the tool to allow for negative profits. In this respect, the constraint *profits > 0* mentioned in Section 4.1.5 is relaxed. In particular, the user of the tool can provide as input a negative profit threshold value (*PT*) so that *profits > PT*. The impact of the relaxation of this constraint to the assessment of the optimal fleet size is schematically represented in Figure 12.

As it can be observed in Figure 12, the perspective of travellers complying to the theory of second-best is affected. Specifically, the introduction of negative profits leads to an increased optimal fleet size value from the perspective of travellers, while the optimal fleet size value from the perspective of service operators remains unaffected. Similarly, depending on the assigned weights to both perspectives (i.e., service operators versus end users) the optimal fleet size will be to a greater or lesser extent larger. Such an outcome appears to more closely resemble reality given that in several European cities, many service operators are willing to sacrifice an amount of their profits to be able to better pursuit competition, understand the market, or even develop side products that are based on the increased use of their fleet by the end users.

The updated computational flow of the first tool is summarised in Figure 13. New processes are yellow coloured.

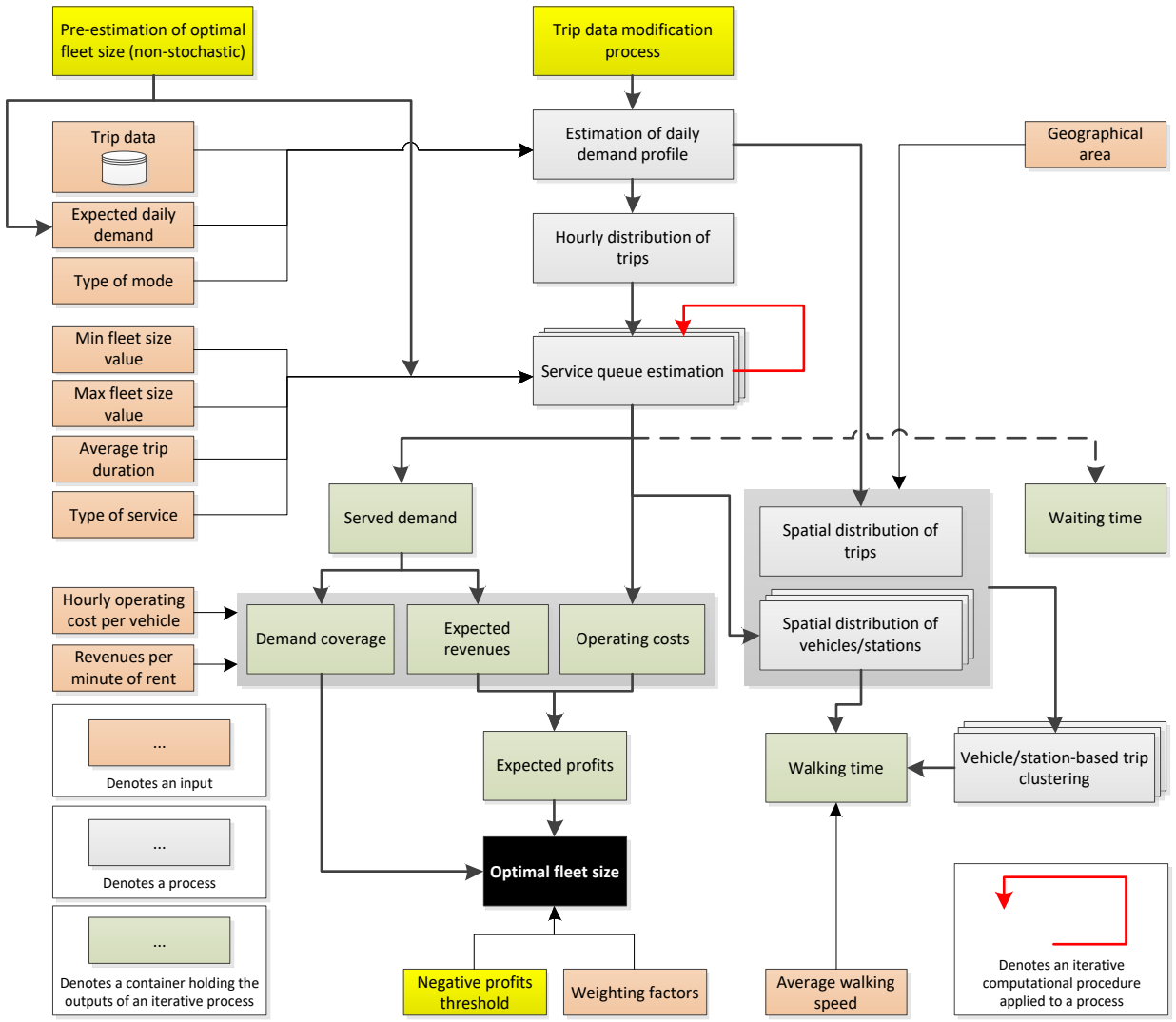


Figure 13: Updated computational flow of the first tool of nuMIDAS toolkit

4.1.7 Code and quality control

The computational flow described in conjunction with the additional features that have been added in the previous sections are functionally prototyped using Anaconda Individual Edition 2020.02 relying upon Python Release 3.7.0. The dependencies of the code are as follows:

- Sys library
- Json library
- Math library
- Random library
- Datetime library
- Numpy library
- Pandas library
- Multiprocessing library
- Mean method of Statistics library
- Cdist method of Scipy Spatial Distance library
- Sqlalchemy method of Create_engine library

The first tool's code is comprised of the following functions:

- main: is the function in which all the necessary processes are conducted.
- main_new: is the function which calls the "main" function. The execution of the "main" function sets to a thread and the current function monitors the execution process of the "main" function to not exceed the time-limit of the AWS Lambda function (15 minutes).
- user_input: is the function by which the algorithm retrieves all the necessary input data from the database (PostgreSQL) for the selected scenario.
- db_connection: is the function through which the results of the simulations as well as from the analysis are stored in the database (PostgreSQL).
- demand_percentage_calculation_0: is the function through which the demand distributed within the hours of the day by following the historic distribution of demand within the hours of the day.
- demand_percentage_calculation_1: is the function through which the demand distributed within the hours of the day by following an increased demand profile of peaks.
- demand_percentage_calculation_2: is the function through which the demand distributed within the hours of the day by following a decreased demand profile of peaks.
- elastic_demand_parameters: is the function through which the calculation of the demand is achieved by taking into account the elasticity of the mode type and service type.
- generate_random_starttime: is the function through which a random number between 0 and 60 minutes is assigned to each trip of each hour of the day.
- generate_tripduration: is the function through which a random number between "mean_trip_duration" - 3 and "mean_trip_duration" + 3 is assigned to each trip of each hour of the day.
- generate_tripduration_spec: is the function through which a random number between "mean_trip_duration" - 3 and "mean_trip_duration" + 3 is assigned to each trip of each hour of the day.

- generate_random_starttime_spec: is the function through which a random number between 0 and 60 minutes is assigned to each trip of each hour of the day.
- user_input_2_1: is the function which calculates the average walking time when the service is declared as station based by the user. It takes as input the size of the area, the optimal fleet size, and the average walking speed.
- user_input_2_2: is the function which calculates the average walking time when the service declared as free-floating by the user. It takes as input the size of the area, the optimal fleet size, and the average walking speed.
- convert: is the function which takes as input a value in seconds and converts it to hours:minutes:seconds format.
- kmeans_impl: is the function in which the kmeans algorithm is implemented. It takes as input the demand, the number of the optimal fleet size, the coordinates of the centroids which are the stations or the position of the vehicles comprising the fleet, as well as the number iterations to be executed. The output of this function is the classification of the demand to the nearest centroid.
- random_generator_constrained: is the function which generates randomly the position of shared mobility service stations utilizing as constrained the minimum distance between them.
- random_generator_nonconstrained: is the function which generates randomly the position of shared mobility service stations utilizing as constrained the minimum distance between them.
- calc_proc_sim: is a core function which takes as input all user defined parameters required for the execution of the algorithm (i.e., demand factors, operating cost per vehicle per minute, expected revenues per minute of rent, minimum fleet size, maximum fleet size, average walking speed, mean trip duration, weights assigned to the perspectives of service operators and end users, type of service, and expected daily demand). It returns as output the optimal fleet size from the perspective of service operators and end users, as well as the values of decision variables and KPIs of interest (i.e., waiting and walking time) corresponding to a range of fleet size defined by the maximum and minimum values declared by the user.

The results of the code quality control, by utilizing the methodology described in Section 3.2, are included in Table 2.

Table 2: Code quality control results - 1st tool (UC1)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	Main function	CC index	C (17)	moderate - slightly complex block
	main_new function	CC index	A (2)	low - simple block
	user_input function	CC index	A (1)	low - simple block

	db_connection function	CC index	A (1)	low - simple block
	demand_percentage_calculation_0 function	CC index	B (7)	low - well-structured and stable block
	demand_percentage_calculation_1 function	CC index	C (16)	moderate - slightly complex block
	demand_percentage_calculation_2 function	CC index	C (16)	moderate - slightly complex block
	elastic_demand_parameters function	CC index	B (9)	low - well-structured and stable block
	generate_random_starttime function	CC index	A (4)	low - simple block
	generate_tripduration function	CC index	A (4)	low - simple block
	generate_tripduration_spec function	CC index	A (3)	low - simple block
	generate_random_starttime_spec function	CC index	A (3)	low - simple block
	user_input_2_1 function	CC index	B (10)	low - well-structured and stable block
	user_input_2_2 function	CC index	B (6)	low - well-structured and stable block
	convert function	CC index	A (1)	low - simple block
	kmeans_impl function	CC index	A (5)	low - simple block
	random_generator_constrained function	CC index	B (6)	low - well-structured and stable block
	random_generator_nonconstrained function	CC index	A (4)	low - simple block

	calc_proc_sim function	CC index	E (32)	high - complex block
Maintainability	Entire UC1 .py file	Maintainability index	B (13.96)	Medium maintainability
Raw metrics	Entire UC1 .py file	LOC	1225	total # lines of code
	Entire UC1 .py file	LLOC	777	total # logical lines of code[4]
	Entire UC1 .py file	SLOC	759	total # source lines of code[5]
	Entire UC1 .py file	comments	109	total # comment lines
	Entire UC1 .py file	multi	4	total # lines representing multi-line strings
	Entire UC1 .py file	blank	356	total # blank lines
Hal metrics	Entire UC1 .py file	h_1	16	total # distinct operators[6]
	Entire UC1 .py file	h_2	337	total # distinct operands[7]
	Entire UC1 .py file	N_1	250	total # operators
	Entire UC1 .py file	N_2	495	total # operands
	Entire UC1 .py file	Vocabulary	353	$n = n_1 + n_2$
	Entire UC1 .py file	Length	745	$N = N_1 + N_2$
	Entire UC1 .py file	calculated_length	2893.66	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Entire UC1 .py file	Volume	6305.33	$V = N \log_2(n)$
	Entire UC1 .py file	Difficulty	11.75	$D = (n_1/2)x (N_2/n_2)$

[4] Logical lines of code may be defined as lines of code including executable expressions.
[5] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.
[6] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[7] The values that an operator operates.

	Entire UC1 .py file	Effort	74092.25	E = D x V
	Entire UC1 .py file	Time	4116.24	T = E/18 seconds
	Entire UC1 .py file	Bugs	2.10	B = V/3000

The average cyclomatic complexity of the UC1 code file is ranked as B (7.125). Hence, it can be considered as a well-structured and stable piece of code.

4.1.8 Demonstration and testing

This section aims, firstly, to demonstrate the results of the application of the first tool and, secondly, to assess the validity of its results based on various test case scenarios. The first purpose is served by presenting the outcomes of running the first tool's Python script providing the inputs included in Table 3 (base test scenario).

Table 3: Inputs corresponding to the base test scenario of the first tool

Input	Value
Expected daily demand (trips/day)	5000
Type of service	Station-based
Type of mode	Bike
Size of the area of interest (in km^2)	5
Operating cost per vehicle per minute (in €)	0.20
Expected revenues per minute of rent (in €)	0.45
Average users walking speed (km/h)	3.6
Mean trip duration (in minutes)	18
Weighting factor (service operators)	0.5
Weighting factor (travellers)	0.5
Minimum fleet size	1
Maximum fleet size	500

The numerical outputs of the tool by providing the input included in Table 3 are as follows:

- Optimal fleet size: **91**
- Optimal fleet size (service operators): **50**
- Optimal fleet size (travellers): **132**
- Demand coverage corresponding to the optimal solution: **86,11%**
- Expected profits corresponding to the optimal solution: **€7.582 (daily)**
- Average walking time corresponding to the optimal solution: **0:08:24**
- Average waiting time corresponding to the optimal solution: **0:00:01**

In addition, the graphical outputs of the first tool indicating the variation of the above outcomes for several fleet size values are depicted in Figure 14[8].

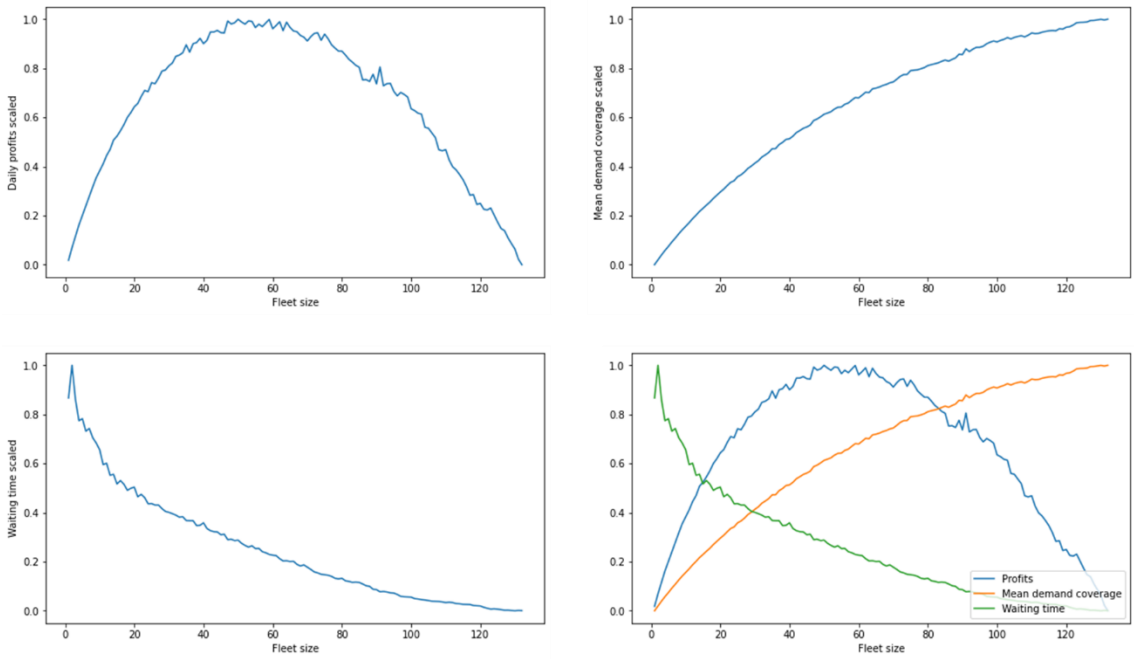


Figure 14: Graphical outputs corresponding to the base test scenario of the first tool

In the first test case scenario it is assumed that the user provides a greater input value to the expected daily demand (Table 4). The expected response of the tool constitutes an increased optimal fleet size value compared to the base test scenario, given that a higher number of end users shall be served.

Table 4: Inputs corresponding to the 1st test case scenario of the first tool

Input	Value
Expected daily demand (trips/day)	**7500**
Type of service	Station-based

[8] Despite the fact that the user is assumed to have declared that the maximum fleet is equal to 500, values greater than 152 has been excluded because they are addressed as leading to negative profits.

Type of mode	Bike
Size of the area of interest (in km^2)	5
Operating cost per vehicle per minute (in €)	0.20
Expected revenues per minute of rent (in €)	0.45
Average users walking speed (km/h)	3.6
Mean trip duration (in minutes)	18
Weighting factor (service operators)	0.5
Weighting factor (travellers)	0.5
Minimum fleet size	1
Maximum fleet size	500

The numerical outputs of the tool by providing the input included in Table 3 are as follows:

- Optimal fleet size: **142**
- Optimal fleet size (service operators): **86**
- Optimal fleet size (travellers): **199**
- Demand coverage corresponding to the optimal solution: **86,4%**
- Expected profits corresponding to the optimal solution: **€10.195 (daily)**
- Average walking time corresponding to the optimal solution: **0:07:00**
- Average waiting time corresponding to the optimal solution: **0:00:01**

In the second test case scenario it is assumed that the user provides a lower input value to the mean trip duration (Table 5). The expected response of the tool constitutes a decreased optimal fleet size value compared to the base test scenario, given that the increased service rate of vehicles composing the fleet of assessed shared mobility service.

Table 5: Inputs corresponding to the 2nd test case scenario of the first tool

Input	Value
Expected daily demand (trips/day)	5000
Type of service	Station-based
Type of mode	Bike
Size of the area of interest (in km^2)	5
Operating cost per vehicle per minute (in €)	0.20

Expected revenues per minute of rent (in €)	0.45
Average users walking speed (km/h)	3.6
Mean trip duration (in minutes)	**9**
Weighting factor (service operators)	0.5
Weighting factor (travellers)	0.5
Minimum fleet size	1
Maximum fleet size	500

The numerical outputs of the tool by providing the input included in Table 4 are as follows:

- Optimal fleet size: **40**
- Optimal fleet size (service operators): **22**
- Optimal fleet size (travellers): **59**
- Demand coverage corresponding to the optimal solution: **74,31%**
- Expected profits corresponding to the optimal solution: **€2.613 (daily)**
- Average walking time corresponding to the optimal solution: **0:12:48**
- Average waiting time corresponding to the optimal solution: **0:00:02**

In the third test case scenario it is assumed that the user provides a lower input value to the operating cost per vehicle per minute (Table 6). The expected response of the tool constitutes an increased optimal fleet size value compared to the base test scenario, given that the profit margin of service operators will be larger. This larger profit margin affects the optimal fleet size from the perspective of service operators but also the optimal fleet size from the perspective of end users. The latter is attributed to the use of the "second best theory."

Table 6: Inputs corresponding to the 3rd test case scenario of the first tool

Input	Value
Expected daily demand (trips/day)	5000
Type of service	Station-based
Type of mode	Bike
Size of the area of interest (in km^2)	5
Operating cost per vehicle per minute (in €)	**0.10**
Expected revenues per minute of rent (in €)	0.45
Average users walking speed (km/h)	3.6

Mean trip duration (in minutes)	18
Weighting factor (service operators)	0.5
Weighting factor (travellers)	0.5
Minimum fleet size	1
Maximum fleet size	500

The numerical outputs of the tool by providing the input included in Table 5 are as follows:

- Optimal fleet size: **131**
- Optimal fleet size (service operators): **106**
- Optimal fleet size (travellers): **156**
- Demand coverage corresponding to the optimal solution: **96,06%**
- Expected profits corresponding to the optimal solution: **€19.054 (daily)**
- Average walking time corresponding to the optimal solution: **0:06:24**
- Average waiting time corresponding to the optimal solution: **0:00:00**

In the fourth test case scenario it is assumed that the user provides a lower input value to the expected revenues per minute per rent (Table 7). The expected response of the tool constitutes a decreased optimal fleet size value, given that the profit margin of service operators will be evidently lower. This lower profit margin affects the optimal fleet size from the perspective of service operators but also the optimal fleet size from the perspective of end users. The latter is attributed to the use of the "second best theory."

Table 7: Inputs corresponding to the 4th test case scenario of the first tool

Input	Value
Expected daily demand (trips/day)	5000
Type of service	Station-based
Type of mode	Bike
Size of the area of interest (in km^2)	5
Operating cost per vehicle per minute (in €)	0.20
Expected revenues per minute of rent (in €)	**0.30**
Average users walking speed (km/h)	3.6
Mean trip duration (in minutes)	18
Weighting factor (service operators)	0.5
Weighting factor (travellers)	0.5

Minimum fleet size	1
Expected daily demand (trips/day)	5000

The numerical outputs of the tool by providing the input included in Table 6 are as follows:

- Optimal fleet size: **46**
- Optimal fleet size (service operators): **65**
- Optimal fleet size (travellers): **28**
- Demand coverage corresponding to the optimal solution: **56,38%**
- Expected profits corresponding to the optimal solution: **€1.569 (daily)**
- Average walking time corresponding to the optimal solution: **0:13:12**
- Average waiting time corresponding to the optimal solution: **0:00:04**

The above results suggest that the tool developed in response to the requirements set out by the first use case produce rational results.

4.2 Operative areas analysis

4.2.1 Overview

The scope of the second tool to be integrated in the nuMIDAS toolkit is to provide support to policy makers of a metropolitan area towards the determination of the spatial extent of a shared mobility service. This necessity stems from the fact that metropolitan areas within and beyond Europe are typically sprawled across large geographical areas, including both densely and less densely populated sub-areas. In fact, some of these sub-areas eventually end up being overserved and others being underserved in terms of offered shared mobility services. This is in line with the conceptual framework behind the first tool based on which service operators should decide on which areas to serve with the aims of minimizing their operational expenditures and guaranteeing an acceptable profitability[9], while travellers wish to enjoy an increased level of service and spatial accessibility. Spatial accessibility is especially the case for the travellers located in remote (or even isolated) areas.

In this respect, it is ideal for policy makers to take the role of a regulating authority that will identify an optimal trade-off between the perspectives of service operators and travellers and conclude to the regulation of an operative area for a shared mobility service that will a) provide an adequate level of service for the end users (travellers), b) comply to the principles of spatial equity and accessibility, and c) conclude to an adequate profitability for service operators, thus safeguarding its financial viability in the long-run.

The tool is built by exploiting the functionalities offered by the first tool, especially the possibility to approximate an optimal (or near optimal) value for the operable fleet size of a shared mobility within an area (or sub-area) of interest, given its spatial and socioeconomic characteristics. Specifically, the tool receives as input the origin-destination (OD) table of a metropolitan area of interest, the population of each zone, the distance between each zone pair, and some input data reflecting the current spatial extent and

[9] It is rationally assumed here that service operators are typically interested in serving smaller geographical areas of high population density.

the (modal) share of a shared mobility service. Subsequently, it includes a computational flow that adjusts the demand for the analysed mobility service considering the extent to which it promotes long-distance or short-distance trips, generates various combinations reflecting various configurations of an operative area, updates the total assumed fleet size (using a simplified version of the first tool[10]), and calculates for each combination a set of Key Performance Indicators (KPIs).

The content of the KPI set encompasses the outputs of the first tool (operating costs and revenues), the amount of population covered, but is further inspired by the accessibility index that has been proposed by Hansen in 1959 (Peña Carrera, 2002). According to this index, the accessibility of a zone *i* is computed through the following formula:

$$A_i = \sum_{j \neq i} \frac{B_j}{d_{ij}^a}$$

where B_j denotes the opportunities at zone *j* for a given purpose, d_{ij} the generalised distance from *i* to *j*, and *a* an appropriate constant. The opportunities at zone *j* can be approximated as the population of zone j, the product of populations of zones i and j, or the value of the (i,j) entry of a given OD table.

The scenario generation process follows an intelligent logic that minimises the search space and ensures the connectivity of the zones to be included in each configuration. The search space minimisation[11] is grounded on the distinction of the sub-areas/zones of a metropolitan area into:

- Initial: zones that are currently/already served
- Mandatory: zones that shall be considered for being served
- Optional: zones that may be considered for being served
- Excluded: zones that will not be considered in any configuration

Moreover, the search space minimisation logic relies on the assumption that the various configurations of an operative area follow an inner-outer expansion pattern. In that sense, the algorithm executes a stepwise search by starting from a central sub-area (i.e., typically the initially/currently served one) and moving out towards the peripheral ones. During this expansion scheme, several configurations are generated by grouping the already included areas and successively adding new zones based on a) their proximity and b) the extent to which are marked (by the user) as mandatory or identified as "attractive" by the algorithm itself[12].

[10] This simplified version of the first tool respects, in line with the version used for pre-estimation purposes (see Section 4.1.6) the multi-periodic nature of the problem but not the stochasticity in the exact time at which end users arrive at a rental station or request to book a service. The only difference between the version used here and the version used for pre-estimation purposes is that the range corresponding to the minimum and maximum fleet size and the expected daily demand are not provided as input by the user but are approximated by the tool itself.

[11] In combinatorial mathematics, search space minimisation is the process of delimiting the number of scenarios that should be evaluated for solving a problem. Translating this analogy to the scope of the second tool, it is impossible to assess all possible configurations of an operative area. The tool should be capable of assessing only the meaningful ones, otherwise a "combinatorial explosion" should be expected.

[12] Attractive zones are identified by clustering the entirety of zones not included in any configuration based on their population by using k-means algorithm (k=2).

The connectivity of the zones to be included in each configuration is ensured by connecting[13] mandatory or optional zones that have been identified as "attractive" to the group of zones corresponding to the previous iteration by selecting a path comprised of a set of zones that maximise system accessibility. This is achieved by using Dijkstra's shortest path algorithm applied to a generalised graph[14] and setting system inaccessibility as a cost function.

The above-described logic is addressed as closely resembling the rationale of a regulatory authority that wishes to assess various configurations of an operative area that exhibit spatial compactness and follow an expansion logic conforming to equitable transport systems.

4.2.2 Targeted users

The stakeholders involved in the ecosystem of the second tool are the following:

- Mobility service operators (including micromobility service operators, bike-sharing service operators, and car-sharing service operators)
- Departments of local government (e.g., municipality) tasked with issuing tenders for service (policy makers)
- Transport planners supporting policy makers
- Travellers (end users)

However, among the above stakeholders the ones that belong to the targeted users of the tool are transport planners and policy makers. The formers are understood as the ones providing input to the tool and the latter as the ones receiving the outputs of the tool (supported by the tool).

4.2.3 Data inputs

The inputs provided to the tool can be divided into values imported from a file or database and user defined. The former includes:

- Geofenced zones/districts (shapefile)
- Adjacency matrix (generalised graph)
- Origin-Destination table
- Origin-Destination distance between districts/zones (interzonal impedance)
- Historical demand data (trips/day hour/mode)
- Statistical information about mobility (modal split factors, average, minimum, and maximum distances travelled per mode, average trip duration per mode)
- Initial set of served districts (or defined by the user)
- Population of each district

[13] In case that they are not already connected.

[14] Defined by abstracting zone centroids as vertices and the straight lines connecting neighbouring zone pairs as arcs. Arcs are bidirectional and weighted by the geometric distance corresponding to each straight-line connection.

On the other hand, the user defined input includes the following:

- Selection of the mode to be analysed (bike, scooter, kick-scooter, or car sharing)
- Operational cost per minute
- Operational revenue per minute
- Weight – service operator perspective
- Weight – end user perspective
- Indication of:
    - Initial served zones/districts (optional - unless extracted from a database)
    - Mandatory zones/districts
    - Excluded zones/districts

4.2.4 Data outputs

The outputs of the tool include:

- Container of generated configurations of an operative area
- A set of KPI values for each configuration
    - Estimated (total) service provision cost
    - Estimated (total) profits
    - Population served
    - Number of served zones/districts
    - System accessibility

4.2.5 Computational flow

As already mentioned in Section 4.2.1, the second tool of nuMIDAS toolkit serves as a decision support tool assisting the assessment of the spatial extent (operative area) of shared mobility services, including bike-, car-, scooter-, and kick scooter-sharing services. The current section aims to provide further details on the computational flow of this tool.

The first step involves the estimation of OD demand corresponding to a shared mobility service. The demand corresponding to each OD pair and the mobility mode/service selected by the user is initially identified as follows:

$$demand_{od,mode} = split_{ave,mode} * demand_{od}$$

where $demand_{od}$ denotes the total demand between each OD pair, $split_{ave,mode}$ the modal split factor corresponding to the select mobility mode/service, and $demand_{od,mode}$ the demand between each OD pair corresponding to the selected mobility mode/service. In an effort to increase the accuracy of the estimated value, this step also includes a process for adjusting the estimated value based on whether the selected mobility mode/service favours short- versus long-distance trips. The modes/services addressed as favouring short-distance trips include bike-, scooter-, and kick-scooter, while car-sharing is addressed as favouring long-distance trips. The adjusted value is denoted as $demand^{*}_{od,mode}$, while the adjustment process follows the logic presented in the following block.

If selected mode $=$ bike $-$ sharing OR selected mode $=$ scooter $-$ sharing OR selected mode $=$ kick $-$ scooter sharing:

If $\text{dist}_{od} > 0$ AND $\text{dist}_{od} < \text{dist}_{ave,mode}$:

$$\text{demand}^{*}_{od,mode} = \text{demand}_{od,mode}\left(\frac{\text{dist}_{ave,mode}}{\text{dist}_{od}}\right)$$

Else if $\text{dist}_{od} = \text{dist}_{ave,mode}$:

$$\text{demand}^{*}_{od,mode} = \text{demand}_{od,mode}$$

Else if $\text{dist}_{od} > \text{dist}_{ave,mode}$ AND $\text{dist}_{od} < \text{dist}_{max,mode}$:

$$\text{demand}^{*}_{od,mode} = \text{demand}_{od,mode}\left(\frac{\text{dist}_{od}}{\text{dist}_{max,mode}}\right)$$

Else:

$$\text{demand}^{*}_{od,mode} = 0$$

Else if selected mode $=$ car $-$ sharing:

If $\text{dist}_{od} > \text{dist}_{min,mode}$ AND $\text{dist}_{od} < \text{dist}_{ave,mode}$:

$$\text{demand}^{*}_{od,mode} = \text{demand}_{od,mode}\left(\frac{\text{dist}_{od}}{\text{dist}_{ave,mode}}\right)$$

Else if $\text{dist}_{od} = \text{dist}_{ave,mode}$:

$$\text{demand}^{*}_{od,mode} = \text{demand}_{od,mode}$$

Else if $\text{dist}_{od} > \text{dist}_{ave,mode}$:

$$\text{demand}^{*}_{od,mode} = \text{demand}_{od,mode}\left(\frac{\text{dist}_{ave,mode}}{\text{dist}_{od}}\right)$$

Else:

$$\text{demand}^{*}_{od,mode} = 0$$

Once the demand between each OD pair is adjusted, the next step involves the identification of the set of attractive zones/districts. The elements of this set include the zones/districts indicated as mandatory by the user as well as a fraction of the zones/districts indicated as optional by the user. This fraction is identified by classifying optional zones/districts based on the total demand produced by each of them. This is achieved by using Lloyd's k-means hierarchical clustering algorithm, specifically by setting the number of output clusters equal to 2 (k=2). By that means, the zones/districts included in the cluster of zones/districts producing or attracting a high amount of travel demand are addressed as attractive and, thus, included in the set, while the zones/districts included in the cluster of zones/districts producing or attracting a lower amount of travel demand are addressed as non-attractive and, thus, excluded from the set.

The third step revolves around the generation of the various configurations of an operative area. This is achieved by aggregating the initially served zones/districts into a single origin zone. Subsequently, the entirety of attractive zones/districts are sorted in ascending order based on their straight-line geometric distance from the source zone[15]. Following the acquired order, a set of combinations is generated with each of them including the source zone and additional instances of the source zone in which the *closest* attractive zone is *successively appended*. This set of combinations or scenarios represent various configurations of an operative area exhibiting an inner-outer expansion/enlargement pattern. The number of generated combinations is equal to the number of the attractive zones + 1[16]. It is noted that newly added/appended districts should not be isolated. For this reason, we find a path for connecting each

[15] This is achieved by using the provided interzonal impedance and calculating the distance between each zone included in the set of attractive zones and the closest zone included in the source zone group (i.e., minimum distance).

[16] By that means, we make a first step towards the reduction of the search space of the second tool of the nuMIDAS toolkit.

previous instance of a source zone with the next attractive zone/district. This path consists of consecutive intermediate zones that should *not* be marked by the user as excluded[17].

With the aim of speeding up the search of that path, a constrained network is defined between the source zone and the next attractive zone/district. The adopted principle is to remove from the search space zones/districts the distance of which to the source zone exceeds a certain threshold[18]. This is achieved by defining two buffer zones and delimiting the search space to their union (Figure 15). The radius of both circular zones is set equal to the minimum distance between the source zone (S) and the zone/district (D) to be added/appended to the group.

Subsequently, the exact identification of the zones/districts through which the connection will be made relies on the computation of the shortest path between S and D (as denoted in Figure 14). For such a computation, we make use of the Dijkstra's shortest path algorithm by setting inaccessibility as a cost function:

$$inaccessibility_z = \frac{1}{\frac{demand_{z,mode}}{d_{sz}}}$$

where d_{sz} denotes the distance between the source zone (S) and each intermediary district (Z) included in the constrained network and $demand_{z,mode}$ the total demand produced by each intermediary district (Z) included in the constrained network. Therefore, the chosen path (sequence of neighbouring zones/districts) between source and destination zones is the one that minimises total inaccessibility.

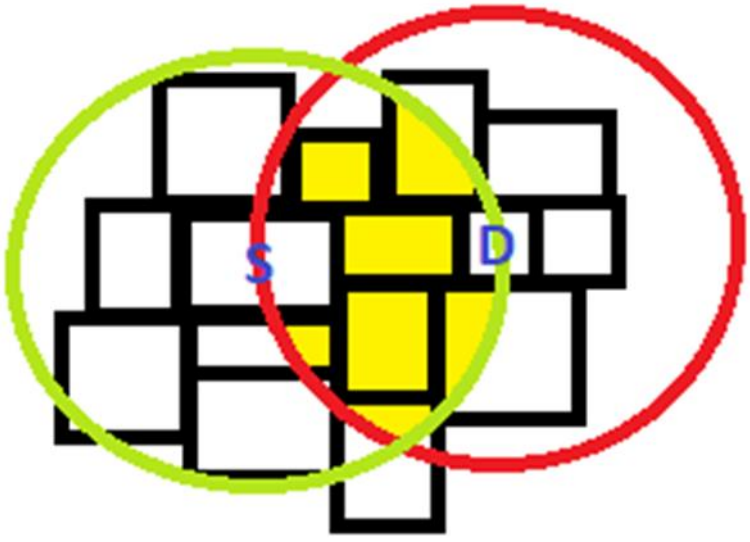


Figure 15: Search space zones/districts

[17] In the extreme case that the connection of the previous instance of a source zone group with the next attractive zone is feasible only via zones marked as excluded the newly added/appended attractive zone remains isolated.

[18] By that means, we make the second step towards the reduction of the search space of the second tool of the nuMIDAS toolkit.

The final step involves the quantification of several indicators corresponding to each generated configuration. As mentioned in Section 4.2.4, such indicators include: (a) the estimated (total) service provision cost, (b) the estimated (total) profits, (c) the percentage of served population, (d) the number of served zones/districts, and (e) system accessibility.

System accessibility is approximated on the basis of Hansen's definition for accessibility:

$$System\ accessibility(z) = \sum_{i \neq j \in A} \frac{demand^{*}_{od,mode,ij}}{d_{z,ij}}, z \in C$$

where *A* denotes the set of zones/districts of a whole metropolitan area, *C* the set of generated configurations of an operative area, $demand^{*}_{od,mode,ij}$ the *(i,j)* entry of the adjusted OD table corresponding to the selected mode, and $d_{z,ij}$ the distance between *i* and *j* assuming configuration *z*. Inspired by the literature on the analysis of robust transportation networks, when either zone/district *i* or *j* is not served under configuration *z*, $d_{z,ij}$ is set equal to infinity. Through this technique, the part of the system accessibility corresponding to an unserved pair of zones/districts gets equal to zero.

The estimation of the (total) service provision cost and (total) profits requires the solution of the optimal fleet size problem. In particular, one problem is solved for each zone/district included in at least one configuration of the operative area. Therefore, given that a large metropolitan area may require several solutions, such solutions are achieved by following a more simplified/computationally efficient approach than that presented in Section 4.1. As already mentioned in Section 4.2.1, the multiperiodic nature of the problem is maintained, while the stochasticity with which end-users are assumed to request to book a service is relaxed. The difference of the solution framework used in the context of the second tool and the simplified framework used for pre-estimating fleet size (see Section 4.1.6) is that the range corresponding to the minimum and maximum fleet size and the expected daily demand are not provided as input by the user but are approximated by the tool itself. In particular, the expected daily demand of each zone/district is approximated as the total number of trips produced by this zone/district taking as input the adjusted OD table corresponding to the selected mobility service/mode. Subsequently, the total demand of each zone/district *i* corresponding to time interval/hour *t* ($demand^{*}_{i,mode,t}$) is calculated by multiplying the expected daily demand by the 24h demand factor obtained from the analysis of the historical demand data provided as input. On the other hand, the minimum and maximum fleet size values are determined as follows:

$$min\_fleet\_size_i = min\ (demand^{*}_{i,mode,t})$$

$$max\_fleet\_size_i = max\ (demand^{*}_{i,mode,t})$$

The assignment of the minimum and maximum total demand of each zone/district *i* corresponding to time interval/hour *t* to the minimum and maximum fleet size values, respectively, appears to constitute a rational assumption, given that a fleet size equal to the minimum total demand provides a presumably low level of service, while a fleet size equal to the maximum total demand provides a presumably high (potentially extravagant) level of service. Therefore, an optimal value ($fleet\_size^{*}_i$) should be in between these extreme values. The following block demonstrates the followed process for solving from that point on the problem of the optimal fleet size within a zone/district *i*.

For each fleet size value within the range $[min\_fleet\_size_i, max\_fleet\_size_i]$:
  For each day hour t = {0..h}:
    If fleet_size > $\text{demand}^*_{\text{i,mode,t}}$:
      $\text{trips\_covered}_{\text{fleet\_size,t,i}} = \text{demand}^*_{\text{i,mode,t}}$
      $\text{demand\_coverage}_{\text{fleet\_size,t,i}} = \mathbf{1}$
    Else:
      $\text{trips\_covered}_{\text{fleet\_size,t,i}} = \text{fleet\_size} \times \mathbf{60} / (\text{duration\_trip}_{\text{mean}})$
      $\text{demand\_coverage}_{\text{fleet\_size,t,i}} = (\frac{\text{fleet\_size} \times \mathbf{60}}{\text{duration\_trip}_{\text{mean}}})/(\text{demand}^*_{\text{i,mode,t}})$
For each fleet size value within the range $[min\_fleet\_size_i, max\_fleet\_size_i]$:
  $\text{trips\_covered}_{\text{fleet\_size,i}} = \sum_t \text{trips\_covered}_{\text{fleet}_{\text{size}},\text{t,i}}$
  $\text{demand\_coverage}_{\text{fleet\_size,mean,i}} = \frac{\sum_t \text{demand\_coverage}_{\text{fleet\_size,t,i}}}{\text{h}}$
  $\text{revenues\_operator}_{\text{fleet\_size,i}} = \text{trips\_covered}_{\text{fleet\_size,i}} \times \text{op\_rev\_unit} \times \text{duration\_trip}_{\text{mean}}$
  $\text{cost\_operator}_{\text{fleet\_size,i}} = \text{fleet\_size} \times \text{op\_cost}_{unit} \times 24 \times 60$
  $\text{profits}_{\text{fleet\_size,i}} = \text{revenues\_operator}_{\text{fleet\_size,i}} - \text{cost\_operator}_{\text{fleet\_size,i}}$
$\text{profits}_{\text{max,i}} = \text{argmax}(\text{profit}_{\text{fleet\_size,i}})$
$\text{demand\_coverage}_{\text{max,i}} = \text{argmax}(\text{demand\_coverage}_{\text{fleet\_size,mean,i}})$
$\text{fleet\_size}^*_{\text{i}} = \text{int}(\text{weight}_{\text{operator}} * \text{profit}_{\text{max,i}} + \text{weight}_{\text{traveller}} \times \text{demand\_coverage}_{\text{max,i}})$
$\text{estimated\_service\_provision\_cost}_{\text{i}} = \text{fleet\_size}^*_{\text{i}} \times \text{op\_cost}_{unit} \times 24 \times 60$
$\text{estimated\_revenues}_{\text{i}} = \text{trips\_covered}_{\text{fleet\_size}^*_{\text{i}}} \times \text{op\_rev\_unit} \times \text{duration\_trip}_{\text{mean}}$
$\text{estimated\_profits}_{\text{i}} = \text{estimated\_service\_provision\_cost}_{\text{i}} - \text{estimated\_revenues}_{\text{i}}$

Finally, once the set of potential zones/districts is determined, the calculation of the percentage of served population is trivial:

$$served_{population(z)}\% = \frac{served_{population(z)}}{total_{population}}, z \in C$$

where *C* denotes the set of generated configurations of an operative area, $served_{population(z)}$ the amount of population served under configuration *z*, and $total_{population}$ the total population of the analysed metropolitan area. Figure 16 provides an overview of the computational flow behind the second tool of nuMIDAS toolkit.

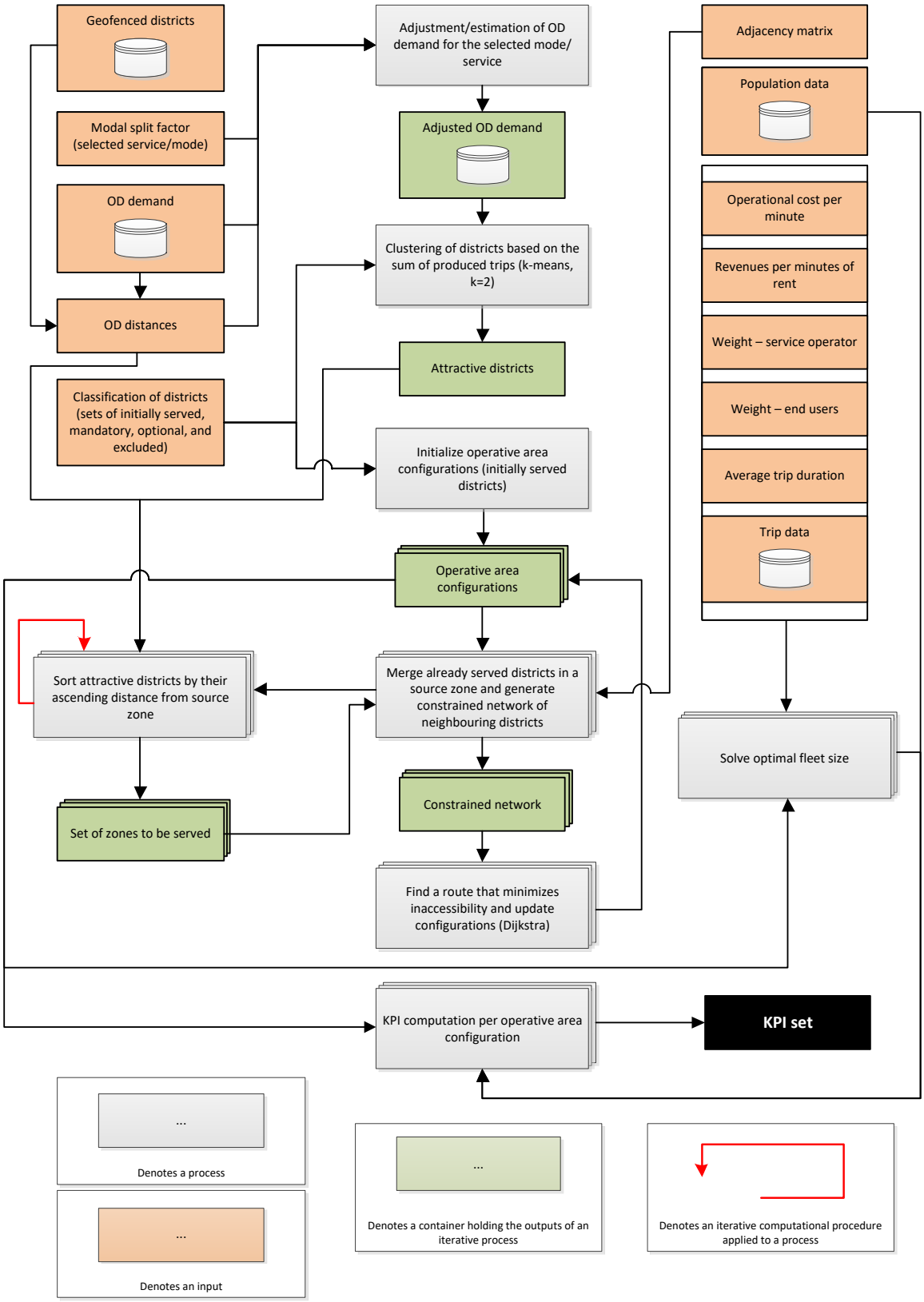


Figure 16: Computational flow of the second tool of nuMIDAS toolkit

4.2.6 Additional features

No additional features have been incorporated in the second tool of the nuMIDAS toolkit as all the requirements settled into D2.2 have taken into account through the first version of the tool.

4.2.7 Code and quality control

The computational flow described in the previous section is functionally prototyped using Anaconda Individual Edition 2020.02 relying upon Python Release 3.9.12. The dependencies of the code are as follows:

- Json library
- Math library
- Numpy library
- Pandas library
- Multiprocessing
- Datetime library
- Threading library
- Queue library
- Kmeans1d library
- Sqlalchemy method of Create_engine library

The second tool's code is comprised of the following functions:

- main: is the function in which all the necessary processes are conducted.
- main_new: is the function which calls the "main" function. The execution of the "main" function sets to a thread and the current function monitors the execution process of the "main" function to not exceed the time-limit of the AWS Lambda function (15 minutes).
- db_connection: is the function through which the results of the simulations as well as from the analysis are stored in the database (PostgreSQL).
- input_parameters: is the function by which the algorithm retrieves all the necessary input data from the database (PostgreSQL) for the selected scenario.
- load_edit_od_data: is the function through which the algorithm retrieves the demand data for all the districts and transport modes.
- shapefile_distance_calculation: is the function by which the algorithm retrieves the shapefile of the district border dans calculates the distance among the centroids of the districts (skim matrix).
- load_adj_matrix: is the function by which the algorithm retrieves the adjacent matrix of each of the districts (the adjacent matrix provides information about the neighbour districts that each district has).
- mode_selection_changes: is the function through which the algorithm retrieves the modal split information for each of the four transport modes (bike sharing, car sharing, scooter, and kickscooter) and also calculates the demand for the chosen transport mode for each district based on the distance among the districts.
- identify_currently_served_zones: is the function through which the algorithm retrieves the initial (currently) served districts, only in case the user does not manually / insert.

- identify_potential_zones: is the function by which the algorithm identifies the districts that are worth to be included into the assessment among the districts that are unserved. The algorithm splits into groups of two the unserved districts by considering the estimated demand of each district for the chosen transport mode.
- define_net: is the function through which the algorithm generates the (initial) network of neighbouring districts where links (from or to excluded districts) are not included in the network, according to the adjacent matrix obtained (through another function).
- calc_dist: is the function through which the algorithm calculates the distance between each district and the source zone, given the skim matrix.
- calc_demand: is the function by which the algorithms calculates the sum of the demand among each district and the source zone.
- sort_zones: is the function through which the algorithm sorts the districts to be included in the assessment by their distance (ascending) from source zone or by their demands (descending) and then initialises the order of the destination districts that will be included in the evaluation.
- define_constrained_net: is the function of the algorithm through which a constrained network of neighbouring districts generated where links (from or to districts unlikely to be included between the source zone and the destination district being evaluated are removed from the initial network generated at function named “define_net.”
- calc_inaccessibility: is the function by which the algorithm calculates the inaccessibility index/ cost for each (intermediary) district between the source zone and the destination district being evaluated. The algorithm calculates the inaccessibility as the ratio between the sum of the demand originating in the district and the distances to other zones.
- dijkstra: is the function by which the algorithm uses the Dijkstra routing algorithm to find the route between a source zone and a destination district which minimises the total inaccessibility index.
- initialiser: is the function of the algorithm through which the initialisation of the multiprocessing and multithreading process accomplishes.
- Worker: is the function by which the algorithm calculates the accessibility of a subset of a combination.
- accessibility_calculator: is the function by which the algorithm calculates the accessibility of each combination, i. e. scenario where a new set of districts are assessment to be included in the expansion of the operative areas.
- calc_accessibility: is the function by which the algorithm calculates the accessibility of an origin-destination pair.
- cost_profit_calculator: is the function by which the algorithm calculates both the cost as well as the profits of each service. At first, the algorithm calculates the demand profile for each district and for each hour of the day based on the hourly demand factors and then calculates the fleet size, the revenues of the system, the cost of the system the trips covered and finally the profits.
- kpis_calculator: is the function of the algorithm through which the needed KPIs are calculated.

The results of the code quality control, by utilizing the methodology described in Section 3.2, are included in Table 8.

Table 8: Code quality control results – 2nd tool (UC2)

	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	main function	CC index	B (7)	low - well-structured and stable block
	main_new function	CC index	A (2)	low - simple block
	db_connection function	CC index	A (1)	low - simple block
	input_parameters function	CC index	A (1)	low - simple block
	load_edit_od_data function	CC index	A (1)	low - simple block
	shapefile_distance_calculation function	CC index	A (1)	low - simple block
	load_adj_matrix function	CC index	A (5)	low - simple block
	mode_selection_changes function	CC index	C (18)	moderate - slightly complex block
	identify_currently_served_zones function	CC index	A (5)	low - simple block
	identify_potential_zones function	CC index	A (4)	low - simple block
	define_net function	CC index	B (8)	low - well-structured and stable block
	calc_dist function	CC index	A (3)	low - simple block
	calc_demand function	CC index	A (3)	low - simple block
	sort_zones function	CC index	A (4)	low - simple block

	Evaluation element	Metric	Rank (Score)	Interpretation
	define_constrained_net function	CC index	A (3)	low - simple block
	calc_inaccessibility function	CC index	A (5)	low - simple block
	Dijkstra function	CC index	C (16)	moderate - slightly complex block
	Initialiser function	CC index	A (1)	low - simple block
	worker function	CC index	A (4)	low - simple block
	accessibility_calculator function	CC index	C (15)	moderate - slightly complex block
	calc_accessibility function	CC index	A (3)	low - simple block
	cost_profit_calculator function	CC index	C (12)	moderate - slightly complex block
	kpis_calculator function	CC index	A (4)	low - simple block
Maintainability	Entire UC2 .py file	Maintainability index	A (26.96)	Very high maintainability
Raw metrics	Entire UC2 .py file	LOC	908	total # lines of code
	Entire UC2 .py file	LLOC	544	total # logical lines of code[19]

[19] Logical lines of code may be defined as lines of code including executable expressions.

	Evaluation element	Metric	Rank (Score)	Interpretation
	Entire UC2 .py file	SLOC	541	total # source lines of code[20]
	Entire UC2 .py file	comments	143	total # comment lines
	Entire UC2 .py file	multi	5	total # lines representing multi-line strings
	Entire UC2 .py file	blank	219	total # blank lines
Hal metrics	Entire UC2 .py file	h_1	18	total # distinct operators[21]
	Entire UC2 .py file	h_2	217	total # distinct operands[22]
	Entire UC2 .py file	N_1	149	total # operators
	Entire UC2 .py file	N_2	295	total # operands
	Entire UC2 .py file	Vocabulary	235	$n = n_1 + n_2$
	Entire UC2 .py file	Length	444	$N = N_1 + N_2$
	Entire UC2 .py file	calculated_length	1759.32	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Entire UC2 .py file	Volume	3497.17	$V = N \log_2(n)$
	Entire UC2 .py file	Difficulty	12.24	$D = (n_1/2)x (N_2/n_2)$

[20] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.
[21] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[22] The values that an operator operates.

	Evaluation element	Metric	Rank (Score)	Interpretation
	Entire UC2 .py file	Effort	42787.99	E = D x V
	Entire UC2 .py file	Time	2377.11	T = E/18 seconds
	Entire UC2 .py file	Bugs	1.17	B = V/3000

The average cyclomatic complexity of the UC2 code file is ranked as B (5.42). Hence, it can be considered as a well-structured and stable piece of code.

4.2.8 Demonstration and testing

This section aims, firstly, to demonstrate the results of the application of the second tool and, secondly, to assess the validity of its results based on various test case scenarios. The first purpose is served by presenting the outcomes of running the second tool's Python script providing the inputs included in Table 9 (base test scenario).

Table 9: Inputs corresponding to the base test scenario of the second tool

Input	Value
Type of mode	Bike
Average trip duration (in minutes)	14
Operating cost per vehicle per minute (in €)	0.2
Expected revenues per minute of rent (in €)	0.5
Weighting factor (service operators)	0.5
Weighting factor (travellers)	0.5

The initial set of zones, which are set as currently served, the included ones and the excluded ones are depicted in the following figure (Figure 17).

The numerical outputs of the tool by providing the input included in Table 9 are as follows:

- Cost in €: **€ 4.310**
- Profit in €: **€ 150.384**
- Accessibility: **26,68**
- Population coverage: **93,21%**

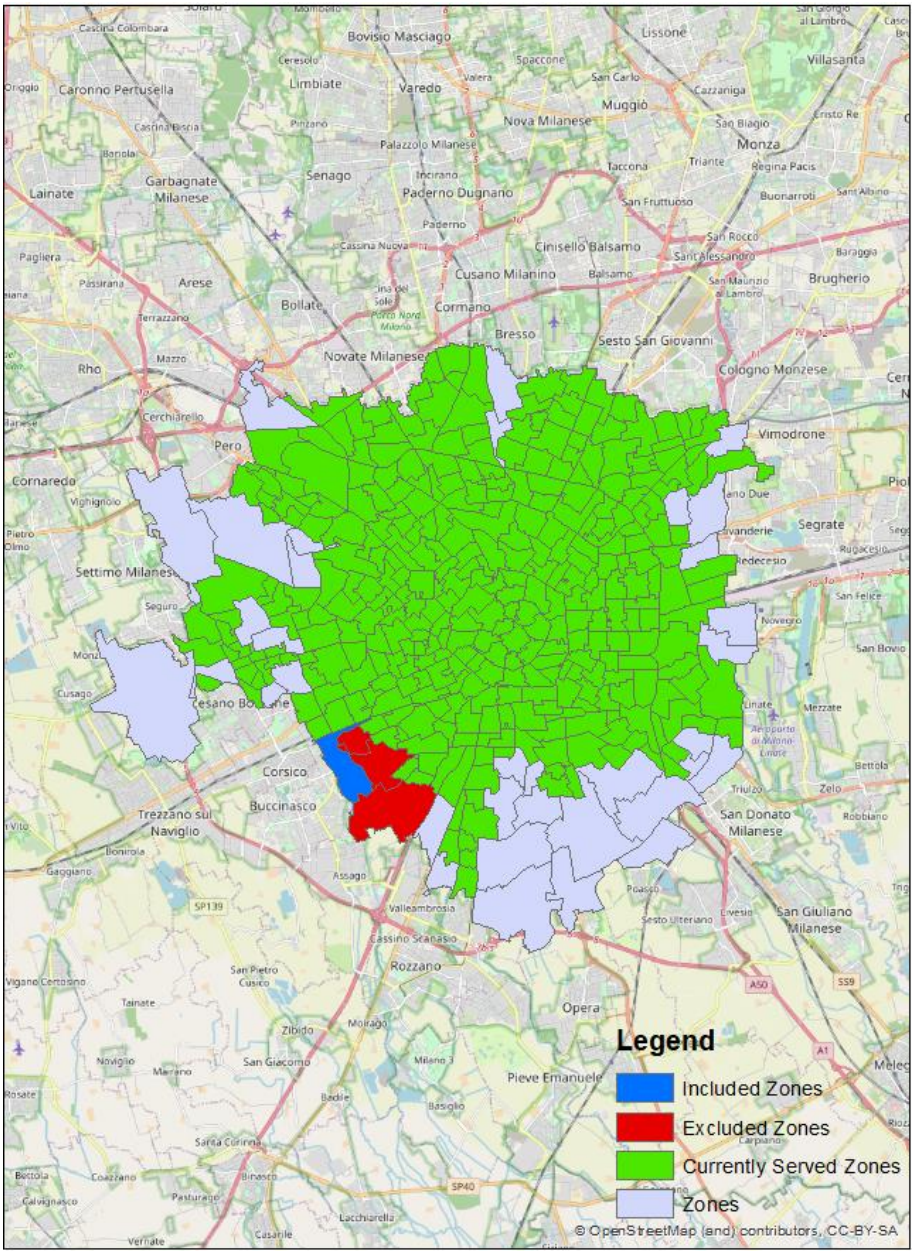


Figure 17: Base test scenario edited zones

In the first test case scenario it is assumed that the user provides a greater input value to average trip duration (Table 10). The expected response of the tool constitutes an increased profit value compared to the base test scenario, given that the users will use each bike for a greater value of time.

Table 10: Inputs corresponding to the 1st test case scenario of the second tool

Input	Value
Type of mode	Bike
Average trip duration (in minutes)	**18**
Operating cost per vehicle per minute (in €)	0.2

Expected revenues per minute of rent (in €)	0.5
Weighting factor (service operators)	0.5
Weighting factor (travellers)	0.5

The numerical outputs of the tool by providing the input included in Table 10 are as follows:

- Cost in €: **€ 4.310**
- Profit in €: **€ 155.772**
- Accessibility: **26,68**
- Population coverage: **93,21%**

In the second test case scenario it is assumed that the user provides a lower input value to expected revenues per minute of rent (Table 11). The expected response of the tool constitutes a decreased cost and profit value compared to the base test scenario, given that the expected revenues per minute of rent are lower.

Table 11: Inputs corresponding to the 2nd test case scenario of the second tool

Input	Value
Type of mode	Bike
Average trip duration (in minutes)	14
Operating cost per vehicle per minute (in €)	0.2
Expected revenues per minute of rent (in €)	**0.3**
Weighting factor (service operators)	0.5
Weighting factor (travellers)	0.5

The numerical outputs of the tool by providing the input included in Table 10 are as follows:

- Cost in €: **€ 1.877**
- Profit in €: **€ 30.960**
- Accessibility: **26,68**
- Population coverage: **93,21%**

The above results suggest that the tool developed in response to the requirements set out by the second use case produce rational results.

4.3 Air quality analysis and forecasting

4.3.1 Overview

The rapid rate of growth of vehicles rises the need of understanding the environmental impacts that caused by the massive usage of private vehicles within urban areas. In most cases, this is reflected by the road congestion and more specifically by the excessive vehicles' starts and stops induced mainly at signalised intersections. On the other hand, promoted concepts, including sustainability, liveability, and quality of life, indicate that there is a clear need to reduce vehicle emissions within urban centres, thus alleviating adverse environmental impacts of traffic (Nešić et al., 2015). To address this environmental issue with regards to air quality, another tool that we planned to be integrated into the nuMIDAS toolkit is responsible for supporting the execution of relevant data analyses based on multi-source data. This is expected to support policy makers towards better planning and assessing enforced policy instruments, such as Low-Emission Zones (LEZ) and Urban Vehicle Access Restrictions (UVAR). The original purpose of the tool was to analyse and correlate various data sources providing information about traffic intensity, weather conditions, air quality, and events. The ultimate purpose would then be to forecast the effect of vehicle traffic and weather on air quality in a short- to medium-term basis (i.e. time horizons covering at maximum the next 10 days).

A critical consideration to be made is that the involved parameters are to certain extent interrelated. For instance, weather conditions may affect both traffic intensity and air quality, while air quality are affected by traffic intensity and weather conditions. This translates to the need for developing two models. The first one will provide an improved forecasting of traffic intensity based on traffic-related historical data, (planned) event-related data, and meteorological forecasts for the upcoming days. The second one will provide a forecast of air quality based on traffic-related data and meteorological forecasts. In this respect, traffic-related information to be provided as an input to the second model will be the output of the first model. For both models to be developed supervised Machine Learning algorithms are considered for utilisation (e.g., linear regression, logistic regression, artificial neural networks, deep neural networks).

Two important aspects that underpin the development of this use case, were (i) the link between weather and emissions, and (ii) the link between traffic and recorded immission (i.e. the measured dispersed emissions).

4.3.1.1 Preliminary correlation analysis (weather and emissions)

Weather conditions have no causal relation to pollution but are important mediators. The influence on pollutant measurements involves multiple aspects:

- **Wind speed and wind directions** affect the transport of pollutants and strengthen the correlation between the pollutant source(s) and the air quality monitoring stations if the direction matches the geographic relationship between the source and the measurement. If the air is calm and pollutants cannot spread, the concentration of these pollutants will accumulate near the pollutant source. Still air is a predictor of high air pollution close around intensive and dense road networks.
- **Sunlight** (solar radiation) play a vital role in the chemical reactions that occur in the atmosphere to form photochemical smog and ground-level ozone from other pollutants, especially during heat waves.
- **Temperature** in the winter and the summer has a dual influence:
    - About 80 percent of the mass of the overall atmosphere pollution disperse in the troposphere, the lowest layer of the atmosphere (up to 18 km). The air temperature of the atmosphere decreases with increasing altitude. The usual warm ground-level temperature and colder higher-level air layer cause convection with rising air near the ground lifting pollution in the sky. This happens more often in the summer.
    - During the winter, thermal inversion can take place with warmer air at higher altitudes in the troposphere. The layer of warm air acts as a lid keeping cold air at the surface. This creates a thermal inversion trapping cool air and pollution close to the ground. Thermal inversions of air layers are more common above cities surrounded by mountains.
- **Rain** washes particulate matter out of the air and break down gaseous pollutants. The measured pollution offset measured at air quality monitoring stations will decrease and pollution might infiltrate the soil.

The emissions of pollutants from one vehicle is generally low but the numbers of vehicles is concentrated around metropolitan cities increasing the pollution many moving emission sources to dangerous levels in the surroundings of heavy traffic roads. About 35 % of carbon monoxide (CO), 30 % of hydrocarbons (HC) and 25 % percent of nitrogen oxides (NOx) produced into the atmosphere is from the transportation sector[23]. High concentration of emissions impacts health and nature's environment.

Long-term exposure of these substances can cause respiratory and cardiovascular diseases, and cancer. CO is one of the main toxic air pollutants. It is the product from the incomplete combustion of fossil fuels. It combines with haemoglobin and reduces the oxygen carrying capacity of the human blood. Carbon dioxide (CO_2) and sulphur oxide (SO_2) can affect the lung function. Motor vehicle exhausts are known to emit NO_x, hydrocarbons causing cancer, and suspended particulate matter (PM). NO_X causes breathing problems but affects the ecosystems seriously, harming animals and plants through polluted water and land. The NO and NO2 substance is a key component for the formation of photochemical smog around cosmopolitan cities. PM has different effects depending on the size. The fine particles (PM2.5) pose the greatest problem as it can penetrate deep into the lungs and the bloodstream. Coarse particles (PM10) are of less concern, but can irritate a person’s eyes, nose, and throat.

[23] Sayani D & Niraj M (2020) Automobile pollution control using catalysis. Resources, Environment and Sustainability. 2. 100006. 10.1016/j.resenv.2020.100006.

Major cities around the world have set up various automatic air quality monitoring stations that detect the levels of PM such as PM2.5 and PM10 in specific areas spread throughout the city. Air quality forecasting methods have gained in importance because of their societal value to predict poor air quality and warn citizens. As a result, there is a significant need for measuring and estimating the contribution of road traffic emissions to outdoor air pollution in large cities. Methods until now can be classified into statistical methods and deep learning methods. An overview of these techniques can be found in Table 12. Multiple factors are considered such as motorised traffic, seasonality, industrialisation, as well as meteorological and socio-economic trends.

There are different research problems that are tackled in the literature of pollution modelling:

- Based on parameters coming from traffic and other emission sources in addition to parameters coming from meteorological, seasonal, road, terrain and geographic elements, the air quality is estimated as a correlate of one or more of these source parameters and mediator variables.
- Based on localised point measurements of air quality at a limited number of stations, air quality is spatially interpolated (e.g. using kriging techniques) over a larger geographical region of the size of a suburban area or a whole city.
- Based on parameters coming from traffic and other emission sources in addition to parameters coming from meteorological, seasonal, road, terrain and geographic elements, the air quality is predicted for the forthcoming minute, hour, day, or week.
- Based on localised point measurements of air quality at a limited number of stations, air quality is predicted for the forthcoming minute, hour, day, or week.

Although all these problems are different from each other they share many features. Predictions are regularly based on machine learning techniques while estimations are more based on regression methods. When it comes to estimation, lagging variables are often used to take for instance the delay into account that the dispersion process takes place or the wind flow moving the particles at a particular velocity. These models differ in the spatial-temporal resolution ranging from coarse to fine-grained scales in space and time covering a region as large as a (sub)urban areas to a whole nation.

Table 12: Research papers, used methods, and description (source see footnote[24]).

Article	***Method***	***Description***
Mahajan S, Chen LJ, Tsai TC (2017) An empirical study of PM2.5 forecasting using neural network. https://doi.org/10.1109/UIC-ATC.2017.8397443	Neural network auto regression (NNAR)	Hourly forecast of PM2.5 was performed and its prediction was compared with ARIMA and Holt–Winters models

[24] Nath, P., Saha, P., Middya, A.I. et al. (2021)Long-term time-series pollution forecast using statistical and deep learning methods. Neural Comput & Applic 33, 12551–12570. https://doi.org/10.1007/s00521-021-05901-2
Espinosa, R., Palma, J., Jiménez, F., Kamińska, J., Sciavicco, G. and Lucena-Sánchez, E (2021). A time series forecasting based multi-criteria methodology for air quality prediction, *Applied Soft Computing, vol. 113,* Article ID 107850, 2021.

Xiang X (2019) Forecasting air pollution PM2.5 in Beijing using weather data and multiple kernel learning. J Forecast. https://doi.org/10.1002/for.2599	Multiple kernel learning (MKL) framework	MKL was proposed to forecast the near future PM2.5 values and was compared to single kernel-based support vector regression (SVR) model
Xie J (2017) Deep neural network for PM2.5 pollution forecasting based on manifold learning. In: 2017 international conference on sensing, diagnostics, prognostics, and control (SDPC), pp 236–240	Deep neural network	The proposed model was based on manifold learning along with a deep belief network (DBN) developed to learn the features of the input candidates for local PM2.5 forecast
Luo C, Yang H, Huang L, Mahajan S, Chen L (2018) A fast PM2.5 forecast approach based on time-series data analysis, regression, and regularisation. In: 2018 conference on technologies and applications of artificial intelligence (TAAI), pp 78–81	Adaptive iterative forecast (AIF) Model	The proposed AIF model could predict the value of PM2.5 for the next few hours (by linear programming, normalisation, and time series) based on the trend of historical data
Feng X, Li Q, Zhu Y, Hou J, Jin L, Wang J (2015) Artificial neural networks forecasting of PM2.5 pollution using air mass trajectory based geographic model and wavelet transformation. Atmos Environ 107:118–128. https://doi.org/10.1016/j.atmosenv.2015.02.030.	Hybrid artificial neural network (ANN)	A hybrid model combining air mass trajectory analysis and wavelet transformation was proposed to improve the forecast's accuracy
Haiming Z, Xiaoxiao S (2013) Study on prediction of atmospheric PM2.5 based on RBF neural network. In: 2013 4th international conference on digital manufacturing automation, pp 1287–1289	RBF neural network	Along with PM2.5, other influence factors were chosen to predict its concentration and then compared with the classic BP network model
Yan L, Wu Y, Yan L, Zhou M (2018) Encoder–decoder model for forecast of PM2.5 concentration per hour. In: 2018 1st international cognitive cities conference (IC3), pp 45–50	Encoder–decoder model	Three prediction models: BP, stack GRU and encoder–decoder were constructed to predict the PM2.5 concentration of every hour of the next day

Cortina-Januchs MG, Quintanilla-Dominguez J, Vega-Corona A, Andina D (2015) Development of a model for forecasting of PM10 concentrations in Salamanca, Mexico. Atmos Pollut Res 6(4):626–634. https://doi.org/10.5094/APR.2015.071. http://www.sciencedirect.com/science/article/pii/S1309104215301951	Multilayer perceptron neural network and clustering algorithm	In addition to multilayer neural network, clustering algorithm was used to find relationships between PM10 and meteorological variables for increasing accuracy of forecasting
Al-kasassbeh M, Sheta A, Faris H, Turabieh H (2013) Prediction of PM10 and tsp air pollution parameters using artificial neural network autoregressive, external input models: a case study in salt, jordan. Middle-East J Sci Res 14:999–1009. https://doi.org/10.5829/idosi.mejsr.2013.14.7.2171	Nonparametric artificial neural network (ANN)	For prediction of PM10, other meteorological parameters were also considered and an artificial neural network based auto regressive with external input (ANNARX) model was proposed to provide high calibre modelling
Lam LH, Mok KM (2007) Prediction of ambient pm10 concentration with artificial neural network. In: Computational methods in engineering and science. Springer, Berlin, Heidelberg, p 276	ANN applied three-layer feed-forward network (TLFN)	Along with six input parameters for each seasonal model, highest absolute values of correlation coefficients were selected to form the model input pattern to feed into the ANN for 24 hour predictions
Zoest, V.V., Osei, F.B., Hoek, G., & Stein, A. (2020). Spatio-temporal regression kriging for modelling urban NO2 concentrations. International Journal of Geographical Information Science, 34, 851 - 865.	A spatio-temporal regression kriging approach	This method interpolates air quality measurements at particular points in the city.
Shi J.P., Harrison R.M. (1997) Regression modelling of hourly no and no concentrations in urban air in London Atmos. Environ., 31 (24), pp. 4081-4094	multidimensional regression models	Based on hourly measurements of NOx NO2 and O3 and meteorological data, an ordinary least squares (OLS) model and a first-order autocorrelation (AR) model were developed to analyse the regression and prediction of NOx and NO2 concentrations in the city
Lucena-Sánchez E., Jiménez F., Sciavicco G., Kaminska J. (2020) Simple versus composed temporal lag regression with feature selection, with an application to air quality modelling. 2020 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS 2020, Bari, Italy, May 27-29, 2020, IEEE, pp. 1-8	Temporal lag regression	Generalisation of the simple temporal lag regression model, and introduce a composed temporal regression model that entails a transformation of the data to improve the effectiveness of classical learning algorithms.

Patra S. (2017). Time series forecasting of air pollutant concentration levels using machine learning Adv. Comput. Sci. Inf. Technol., 4 (5), pp. 280-284	multi-layer perceptron (MLP), support vector regression (SVR) and autoregressive integrated moving average (ARIMA) model	The best results are obtained with MLP with an architecture (4-8-1) for CO and with an architecture (10-2-1) for NO2.
Brunello, A., Kamińska, J., Marzano, E., Montanari, A., Sciavicco, G., Turek, T. (2019). Assessing the Role of Temporal Information in Modelling Short-Term Air Pollution Effects Based on Traffic and Meteorological Conditions: A Case Study in Wrocław. In: , et al. New Trends in Databases and Information Systems. ADBIS 2019. Communications in Computer and Information Science, vol 1064. Springer, Cham. https://doi.org/10.1007/978-3-030-30278-8_45	decision tree model, called J48SS	managing heterogeneous datasets consisting of static (i.e., categorical, and numerical) attributes, as well as sequential and time series data, such as related to meteorological conditions and traffic flow in the city. The study shows the importance to use lagged variables.
I. Kok, M. Simsek, S. Ozdemir (2017). A deep learning model for air quality prediction in smart cities, in: IEEE International Conference on Big Data, pp. 1983–1990.	deep learning model based on Long Short Term Memory	Network structure consists of an input layer, a hidden layer with 24 LSTM units, and an output layer. A considerable share of data is needed to tune the network.
Fan J., Li Q., Hou J., Feng X., Karimian H., Lin S. (2017). A spatiotemporal prediction framework for air pollution based on deep RNN, ISPRS Annal. Photogramm. Remote Sens. Spatial Inf. Sci., 44W2 (2017), pp. 15-22.	deep recurrent neural networks (DRNN)	a spatio-temporal framework incorporating deep recurrent neural networks (DRNN) along with interpolation algorithms to deal with missing values in time series data. Data includes air quality of neighbouring stations, local air quality properties, local meteorological properties and time and spatial properties

When air quality in the city is represented by a categoric index to warn citizen for unhealthy polluted air or to advise people to keep the windows close, a low spatial-temporal resolution can be adequate. The prediction can be concluded from coarse variables such as overall traffic intensity, or even only the day of the week as a precursor of traffic intensity in combination with an effortless measurable parameter like temperature. The relation among those variables is modelled using data from a range of readings recorded by the air quality stations. From the previous appendix, it is clear that the correlations between overall traffic intensity and air quality measured at one station is modest. More complicated treatments incorporating the wind speed and delays might uncover the relation between traffic and air quality in Barcelona. When air quality in the city is represented by some sort of heat map, a much more complex exercise needs to be made keeping pace of smaller spatial and temporal resolutions. For more fine-grained treatment, a lot of variability in air pollution can be distinguished. The air pollution at a busy street may be many times higher than that a little further away. Concentrations of emission particles are highly variable in urban areas. When using air quality measurements at stations as independent variables, it is best to do so with a fine-grained network of air quality sensors.

A more fine-grained solution is achieved when the air quality is estimated using a two-stage model: an emission model immediately followed by a dispersion model to model the emission releases and the spread of the particles in different meteorological conditions. The most important input parameters of the dispersion model are meteorological parameters. A dispersion model can model the range of fine air pollutant particles under the influence of weather at a time resolution of one to several hours. An example of a multiple stage model is that of Forehead en Huynh (2018)[25]. They complement their emission model with a dispersion model (Figure 18). most important input factors for the emission models are values obtained from the traffic simulator, aggregating traffic passing at the spatial locations of interest in addition to vehicle types and model years that composes the fleet. Other important parameters can be the seasonal period, how much time has passed since a cold engine start of the passing vehicles. the chemistry of traffic emissions is highly variable over short distances and intervals of time, due to fleet composition, congestion, and the shape of street canyons. Finally, meteorological parameters do have influence on multiple levels such as the combustion activity, the evaporation of fuel from fuel tanks, and the discharge of particles from tyre-tear and tyres-road friction. They note that the data from traffic emissions models is chemically detailed with resolution of tens of metres and 1 Hz, which is good for the time resolution of the predicted emissions (if the dispersion is low resulting from windless condition) having near to real-time and spatially resolved emissions estimation in chunks of tens of meters. The dispersion model is also highly dependent on meteorological parameters. The composition of the mixture of gases and particles changes with time after release in the air. There are a number of possible chemical reactions such as aerosols, condensation, and coagulation of gases. The transformations can be affected by local conditions such as the concentration of pollutants, temperature, turbulence (particularly wind), sunlight and humidity. Forehead en Huynh use open source code available in GitHub and work in three steps: SUMO to simulate traffic patterns, MOVES to simulate emissions, and FLUIDITY to simulate dispersion. The meteorological component is most contained in the dispersion model.

[25] Forehead H., Huynh, N. (2018). Review of modelling air pollution from traffic at street-level - The state of the science, Environmental Pollution, Volume 241, P. 775-786,

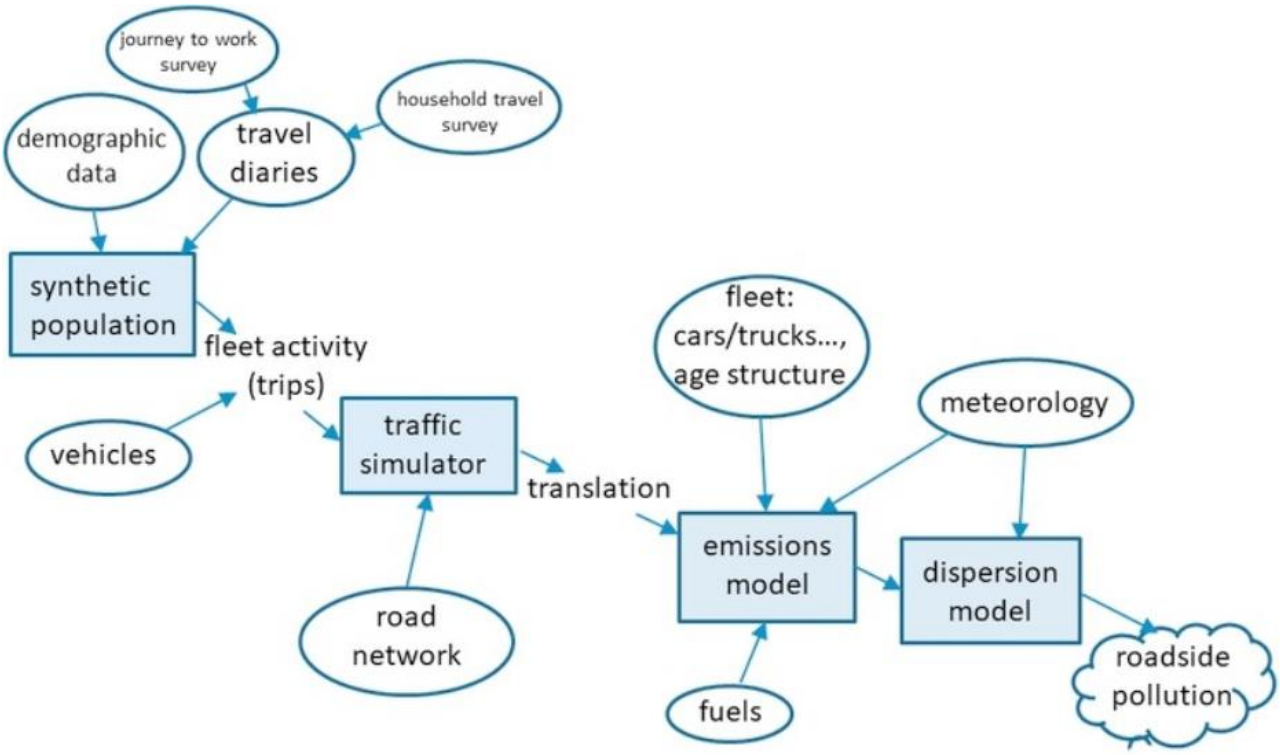


Figure 18: Graph reproduced from Forehead H., Huynh, N. (2018). Review of modelling air pollution from traffic at street-level - The state of the science, Environmental Pollution, Volume 241, P. 775-786, https://doi.org/10.1016/j.envpol.2018.06.019.

Dispersion modelling can evolve to a complex matter. Dispersion of emissions near a source can be modelled by Gaussian models. Two categories of models find their use in research: plume and puff models. Plume models assume steady-state conditions (e.g., wind flow below 1 m/s) while puff models simulate instantaneous releases in a changing environment. The following elements can be integrated in the emission-dispersion model:

- Traffic parameters and fleet composition (traffic intensity 50–250 meter from sampling site and congestion);
- The terrain data (hills, street canyons, buildings, forest);
- Meteorological data and season (wind speed the most important one);
- the transformations of emissions once released.

Fortunately, many applications are available today to accept the challenge of pollution modelling. Many of these tools are open source.

Table 13: Software package for modelling pollution in the air from traffic emission.

Tool/Application/ Software	Use	Description
CALPUFF	Non-steady-state puff dispersion model	It capture the effects of terrain and meteorology and various transformations of emissions over time in the model
CALINE3	Steady-state Gaussian dispersion model	Mainly highway pollution modelling including traffic density in the form of traffic hot-spots and queuing, and meteorological data input.
CAL3QHCR	Dispersion model	CO and PM based on meteorological data and traffic
CALRoads	Dispersion modelling	predicting air pollution concentrations of carbon monoxide (CO), nitrogen dioxide (NO2), particulate matter (PM), and other inert gases near roadways. It combines CALINE4, CAL3QHC, and CAL3QHCR air dispersion models into one integrated graphical interface. predicting from idle or moving motor vehicles
AEROMOD	Dispersion modelling	uses CAL3QHCR as a meteorological data pre-processor and AERMAP as a terrain pre-processor
R-LINE	Dispersion modelling	Freely available model frequently used in academics
ADMS-Roads	Dispersion modelling	Advanced model frequently used in academics
PHOENICS	Dispersion modelling based on computational fluid dynamics	CFD Code for solving 3-D Navier–Stokes
FLUIDITY	Dispersion modelling based on computational fluid dynamics	Open source simulator with fine-grained resolution and representation of turbulence
MOVES	Emission model	Estimates emissions for mobile sources for air pollutants, greenhouse gases, and air toxics.
SUMO	Traffic simulator with emission modelling	multi-modal traffic simulation package including emission calculation

A final approach that deserves some attention for its simplicity and applicability to the Barcelona case is the land-use regression (LUR) model [26]. Land-use regression uses the monitored levels of the pollutant of interest as the dependent variable and spatially distributed variables such as traffic, topography, and other geographic variables as the independent variables in a multivariate regression model. the precision of the LUR model is dependent upon the location of these sites to optimally characterise the sources and pattern of exposure. A raster graphic image of the area is generated and intersected with area-level population data to formulate the exposure distribution.

The LUR model is a simple multiregression model. The air quality measuring stations are the samples of the dependent variable distributed geographically. GIS parameters such as the length of all roads, or the number of houses, or the number of dwellings in circular buffers ranging from 100 to 1000 m around the station site. In addition, the emission from industry within windrose buffers can be included. The method and application is well explained in (Bertazzon et al., 2015)[27]. Next to these spatial varying parameters, time-varying parameters can be considered such as emissions from vehicles on an hourly basis. When multiple measuring stations are used as dependent variable, then the influence of geospatial parameters can be filtered or split away from the influences from time-varying parameters such as the emission from vehicles. When the regression is done on a time basis of consecutive hours, The traffic parameters with zero lag, one-hour lag, two-hour lag can all be initially included. Correlation analysis can help to determine which lag-depending variables can be remained in the model. When the correlation among predictors is higher than 0.6 (Pearson's r), than one predictor should be remained. The predictor with the highest correlation to the dependent variables should remain. A more ambitious approach is to apply a spatio-temporal autoregressive model (for the method, see [28]) to deal with the spatial (land-use) and temporal (hourly) correlations in the data.

[26] Ryan PH, LeMasters GK. (2007). A review of land-use regression models for characterizing intraurban air pollution exposure. Inhal Toxicol.;19 Suppl 1(Suppl 1):127-33. doi: 10.1080/08958370701495998.

[27] Bertazzon S., Markey J., Eccles K. and Kaplan G.G. (2015). Accounting for spatial effects in land use regression for urban air pollution modeling. Spatial and Spatio-temporal Epidemiology 14–15 (2015) 9–21

[28] Pace, R. & Barry, Ronald & Clapp, John & Rodriquez, Mauricio. (1998). Spatio-Temporal Autoregressive Models of Neighborhood Effects. The Journal of Real Estate Finance and Economics. 17. 15-33. 10.1023/A:1007799028599.

4.3.1.2 Preliminary correlation analysis (traffic and emissions)

In order to evaluate whether or not possible correlations between camera and emission data exist, we investigated certain locations. To do this, we first looked at which emission measurement stations recorded at least two pollutants and are in close proximity to cameras for which we have actual detections.

Air quality data for Barcelona is available via an open data stream. In Figure 19 we show the locations of all emission measurement stations, with the circles denoting the local availability of CO, NO, NO_2, and PM10 pollutants being measured. For reference, we superimposed the locations of the available cameras as yellow dots. As can be seen, the correlation between locations of cameras and emission measurement stations is very poor, i.e. not a lot of them seem to be in close proximity to each other.

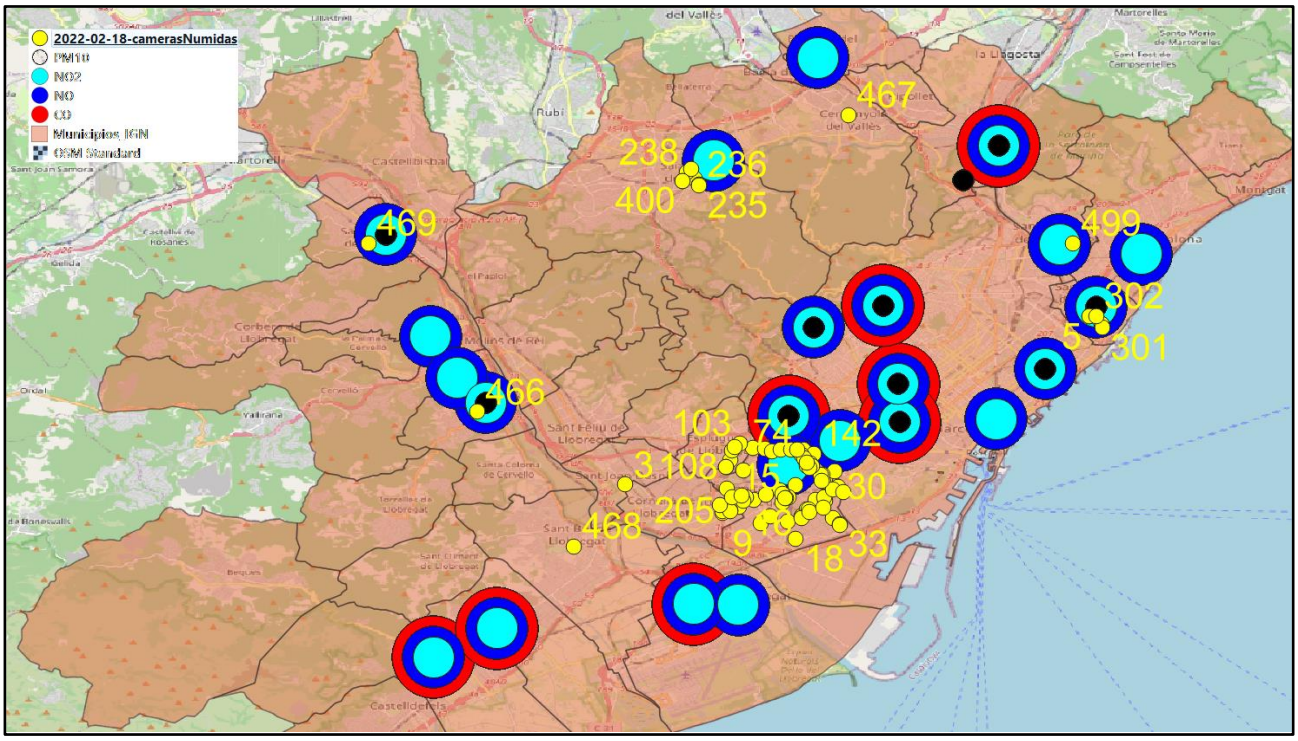


Figure 19: Locations of available emission measurement stations, showing the different pollutants being measured (red: CO, blue: NO, cyan: NO2, and black: PM10).

There are about five locations for which there is a good enough geospatial match between cameras and emission measurement stations. One of them, station #8125002 even measures all four pollutants. We conducted further tests using this station, which has three cameras in close proximity, i.e. #236, #237, and #238.

Looking at the number of vehicles detected by each of these cameras, we can see in how typical daily profiles are present within the data (the data spans approximately two months). There are also large correlations between pairs of cameras. This enhances the credibility of the data to faithfully capture the typical trend of traffic.

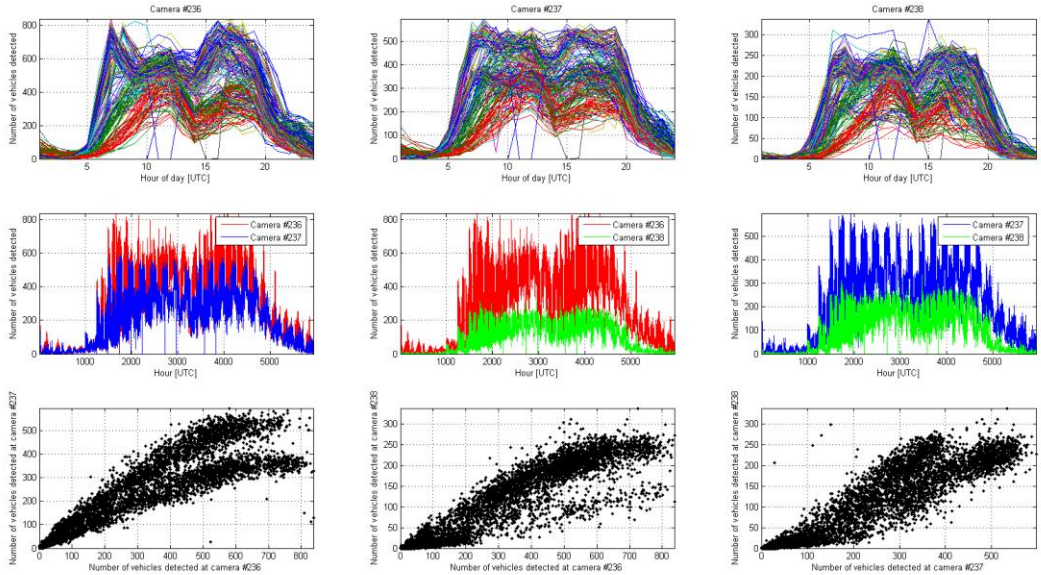


Figure 20: Daily patterns for each of the three cameras (top row) and correlations between each pair of cameras (middle and bottom rows).

These trends are also made more clearly visible, where we can see the differences between weekdays (i.e. Monday through Friday), Saturdays, and Sundays.

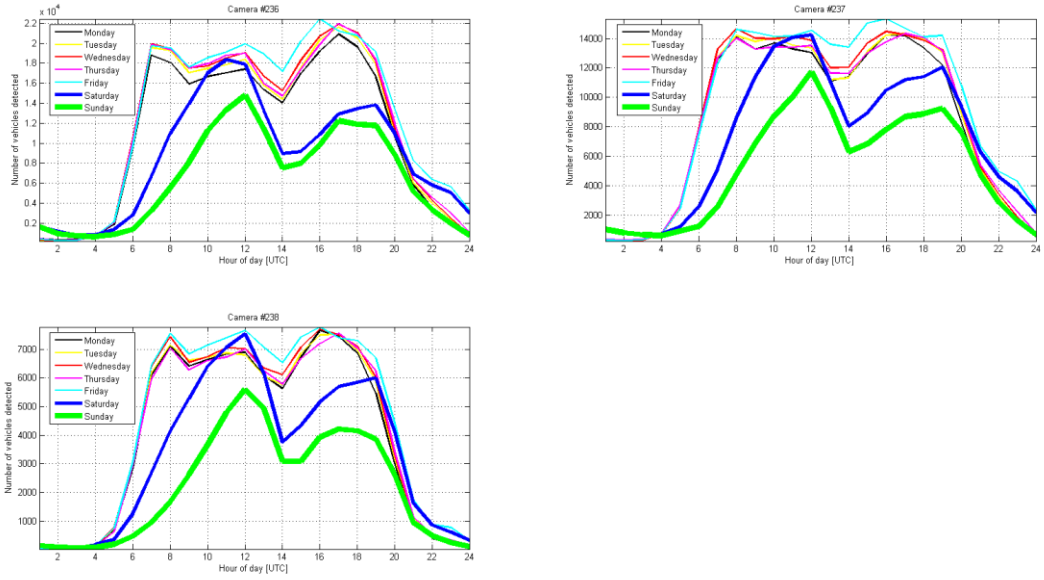


Figure 21: Differences in trends between weekdays (i.e. Monday through Friday), Saturdays, and Sundays.

The boxplots of the measured emissions at station #8125002 for CO, NO, NO2, and PM10 show how the data for the CO emissions is rather flat (mainly because the data contains only a limited set of values). Data for NO and NO~~2~~ is rather similar, showing a larger morning peak and a lesser evening peak. The data for PM10 also exhibits two peaks, albeit slightly less pronounced. Note that the evening peaks occur after 20h, implying that there may be another phenomenon at work besides traffic.

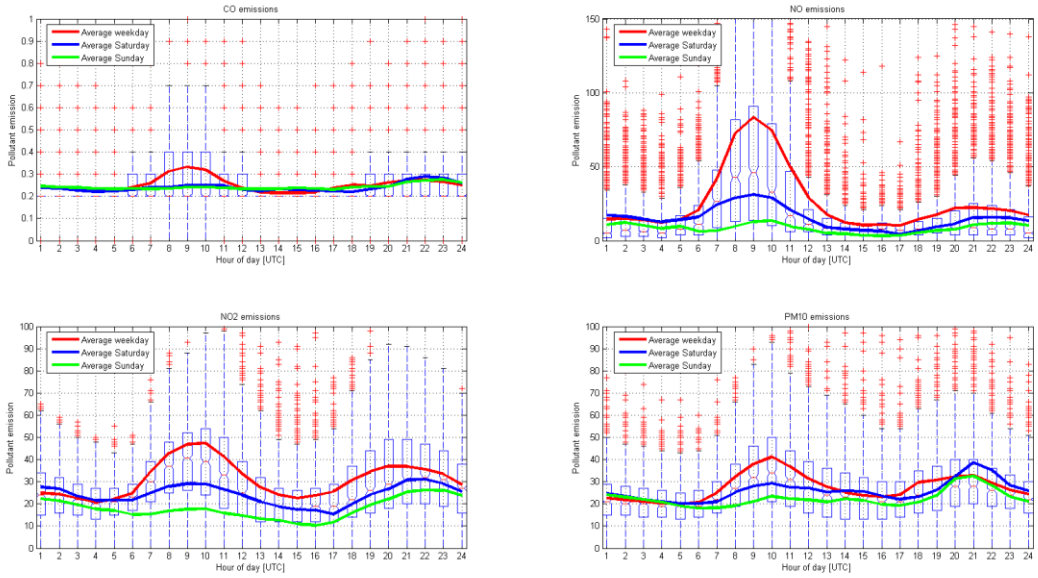


Figure 22: Boxplots of the measured emissions at station #8125002 for CO, NO, NO2, and PM10.

Finally, the correlations between the detections at each of the three cameras and the various measured pollutants show no discernible differences between individual camera measurements (as also evidenced from the correlation plots). As there is a large scatter of data at the hour level, we also averaged them to daily values (i.e. resulting in an average hour for each day), as well as the total amount of emissions and detections recorded during that day. There is some correlation present for the hour averages, albeit quite low.

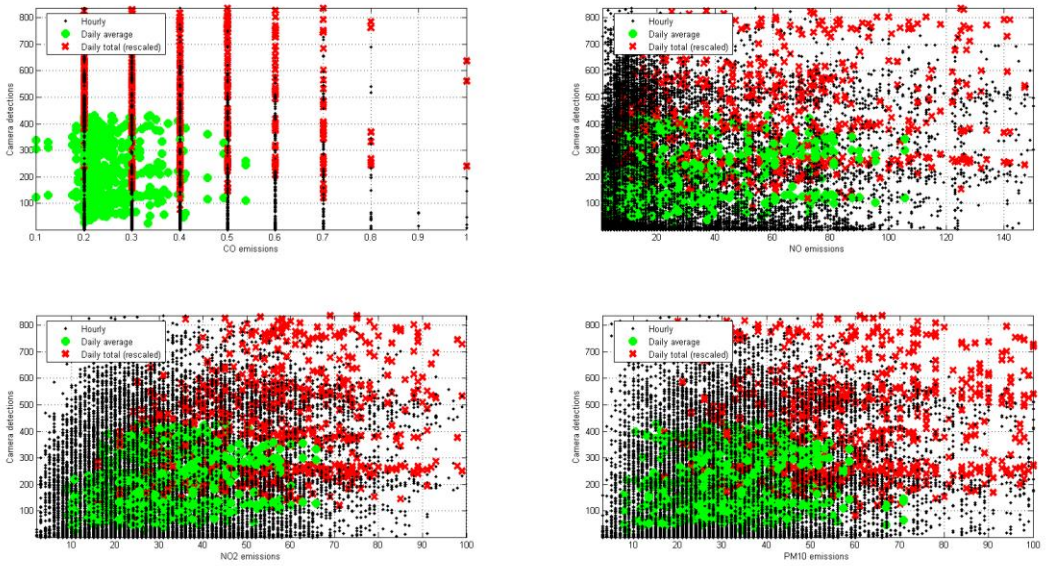


Figure 23: Correlations between the detections at each of the three cameras and the various measured pollutants.

4.3.1.3 Conclusions

Given the research done in the previous section, we make the following observations:

- Only using the number of vehicles may prove to result in a bad correlation with emissions once congestion sets in, hence we also need to take their (average) speed into account.
- All available literature mentions that proximity (until 250 m) is important when one wants to draw a correlation between traffic on the one hand and recorded emissions on the other hand. Emissions rise quickly up through the convection of hot air. Only in winter, when there is a temperature inversion (as the cold air sits lower), the emissions keep hanging over the ground and the original source may have a further-reaching effect. An emission detection station in the neighbourhood of an ANPR camera that records a lot of traffic may result in a higher correlation.
- Of all meteorological aspects, wind speed is the most important parameter (in addition to the season that can be taken into account). Below 1 m/s there is practically no wind and emissions spread out in a Gaussian spatial distribution. At higher wind speeds turbulence occurs, requiring the need of more complex models.

This leads us to the following conclusions for developing, setting up, and deploying the use case given the constraints for creating a prototype:

- We will not create a full-blown model for the immissions, but will rather focus on the emissions of the vehicles.
- For the latter we will create a straightforward model that uses the vehicle counts (as recorded by the ANPR cameras), the emission factors, and speed profiles.

Finally, regarding the emission factors, the go-to reference is Copert[29], the industry standard. Currently, Copert version V has a ready-to-go vehicle fleet, activity, emissions, and energy consumption data for the EU 27 member states. Such emission factors are per vehicle type speed dependent, with an example shown in Figure 24. The figure also shows the average emission curve typically used, and how it is obtained from lots of individual measurements that by their own distribution highlight how widely varying these are.

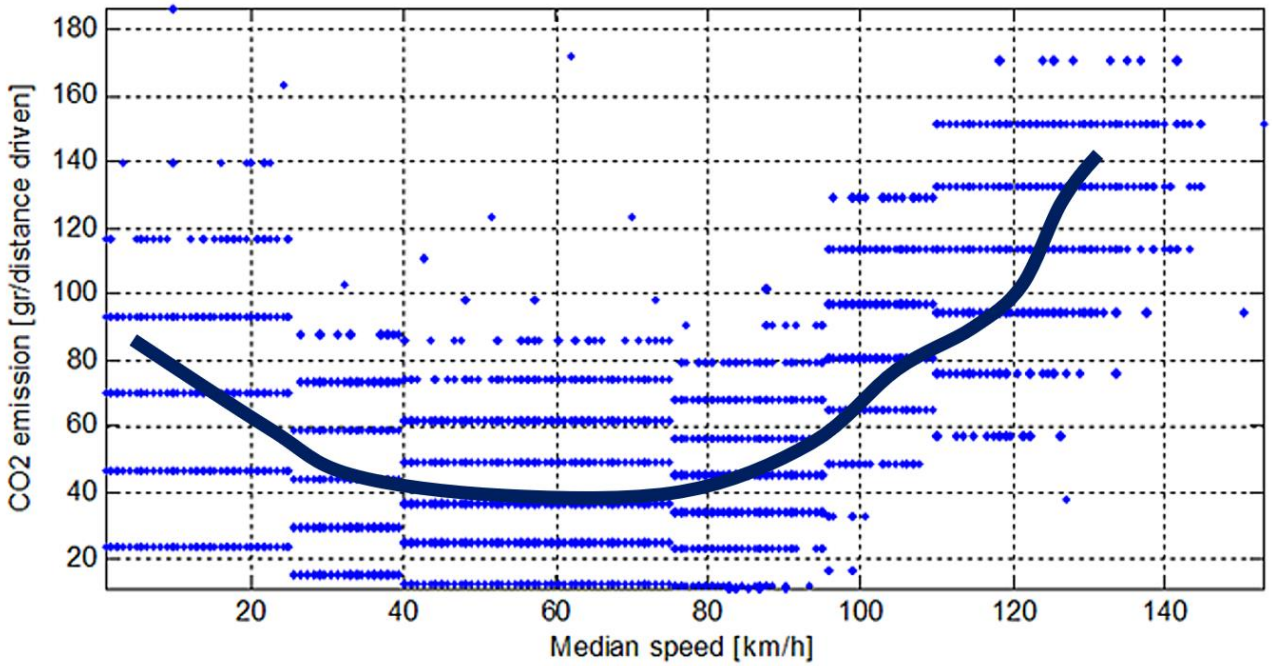


Figure 24: Example of Copert speed-dependent emissions for CO2.

However, in this project we do not have the Copert V speed-dependent emissions readily available, and in addition they are costly to obtain and as such deemed out of scope for the project. Therefore we will rely on more static emission factors that are valid for a mixture of speeds.

[29] https://www.emisia.com/utilities/copert/

4.3.2 Targeted users

The stakeholders involved in the ecosystem of the third tool are the following:

- Data providers (incl. traffic management centres)
- Departments of local government (e.g., municipalities) responsible for the enforcement of traffic restriction policies (policy makers) like (dynamically) adopting LEZ-regulations
- Transport planners supporting policy makers
- Third-party service providers (e.g., weather bureaus)

4.3.3 Data inputs

The inputs to be provided to the tool either by the user or through a database include the following:

- Traffic volumes from ANPR cameras at certain points in the road network
  Emission factors for pollutants CO2, NOx, and PM2.5 (dependent on the type of vehicle)

4.3.4 Data outputs

The outputs of the tool include:

- Traffic volumes at each ANPR camera location
- CO2, NOx, and PM2.5 emissions estimated from traffic at each ANPR camera location

4.3.5 Computational flow

Based on the data for a single camera, we consider the **number of passenger cars** recorded during a 5-minute interval *t*.

Originally, we could consider the speed *v* associated with that 5-min interval (i.e. introducing an extra dependency on the time of day), and the different fuel types *f* of the observed vehicles. For each pollutant *p* (CO2, NOx, or PM2.5) we can then calculate the emission as:

Emission (*p*) = sum over all fuel types *f* { traffic_counts(*t,f*) * emission_factor (*f,v,p*) }

And as we are in urban areas, the speed depends more on the type of road and time of the day than the type of vehicle. In addition, as mentioned earlier, we can not work with speed-dependent emission factors, therefore we resort to a more static approach using an average speed. This then leads to the following simplification whereby the average speed is omitted:

Emission (*p*) = sum over all fuel types *f* { traffic_counts(*t,f*) * emission_factor (*f,p*) }

A final further simplification can be made considering the fleet mix observed in Barcelona, whereby we assume it is incorporated in the emission factors themselves, leading to:

Emission (*p*) = traffic_counts(*t*) * emission_factor (*p*)

As such, we will calculate the emissions based on the observed types of vehicles, and not their fuel types; for the latter we assume that their distribution is encoded in the emission factors themselves.

Note that aggregations over multiple 5-min intervals is straightforward and can be done as summations or averages.

For the average emission factors (expressed in g/km), we rely on calculations done with COPERT V5.41, thereby considering an average speed of circulation for each of the municipalities according to Table 14.

Table 14: Overview of the average speed in each municipality (source: "Anàlisi de les dades de les càmeres de la ZBE – 2022: Tota la base de dades de les càmeres", AMB INFO, 05/09/2022, p. 89).

Location	Average speed	Explanation
Barcelona	28	BARCELONA: average speed to calculate the average emission factor in g/km. At an average speed of circulation of 28 kmh, this speed leaves the average circulation between the city and Rondes: city (75% veh-km * 24 kmh) + Rounds (25% veh-km * 40 kmh). The '*unidentified*' vehicles are assigned the average Emission Factor of the other vehicles
Cornellà de Llobregat	71	CORNELLA: from the ZBE work, it comes out of weighting 20.17% veh-km in URBAN at 25.86kmh + 13.87% interurban at 44.4 kmh + 65.95% motorway at 90.23 kmh
Esplugues de Llobregat	63	ESPLUGUES: from the ZBE work, it comes out of weighting 27.92% veh-km in URBAN at 29.9kmh + 7% interurban at 43.36 kmh + 65% motorway at 78.7 kmh
l'Hospitalet de Llobregat	50	HOSPITAL: from the ZBE work, it comes out of weighting 43.14% veh-km in URBAN at 27.41kmh + 14.57% interurban at 43.68 kmh + 42.3% motorway at 76.27 kmh
Sant Adrià de Besòs	58	SANT ADRIA: from the ZBE work, it comes out of weighting 24.32% veh-km in URBAN at 28.75kmh + 13.82% interurban at 43 kmh + 61.85% motorway at 73.18 kmh
Others	45	REST OF MUNICIPALITIES, average speed for the rest of the municipalities that are not on the list

The emission factors themselves are summarised in Table 15.

Table 15: Overview of the main emission factors per vehicle type (source: "Anàlisi de les dades de les càmeres de la ZBE – 2022: Tota la base de dades de les càmeres", AMB INFO, 05/09/2022, p. 89).

Vehicle type	Counts (abs.)	Counts (rel.)	NOx	NO2	PM10	PM2.5	BC	CO2
Bus [AMB]	526062	0.40%	2.4263	0.2992	0.1121	0.0690	0.0199	684.52
Bus [TMB]	940554	0.70%	2.5495	0.2884	0.1165	0.0727	0.0204	812.87
Average bus	733308	0.55%	**2.4879**	0.2938	0.1143	**0.0709**	0.0202	**748.70**
Autocar	933623	0.70%	3.1665	0.3694	0.1271	0.0828	0.0266	1002.03
Articulated truck	517894	0.40%	**2.1708**	0.2477	0.1116	**0.0700**	0.0200	**510.43**
Rigid truck	2680454	2%	**1.3690**	0.1657	0.0990	**0.0578**	0.0133	**344.40**
Mobile	139982	0.10%	0.0697	0.0028	0.0164	0.0102	0.0012	39.92
Van	13113652	9.60%	**0.8418**	0.3051	0.0469	**0.0286**	0.0078	**244.38**
Microcar	122783	0.10%	0.3362	0.1317	0.0581	0.0543	0.0424	84.79
Motorcycle	15219431	11.10%	**0.0561**	0.0022	0.0176	**0.0116**	0.0018	**88.22**
Unidentified	11129453	8.10%	0.3816	0.1120	0.0365	0.0225	0.0057	195.05
TAXI (van)	40964	0%	0.8862	0.3021	0.0411	0.0227	0.0025	251.65
TAXI (passenger car)	11631763	8.50%	0.2400	0.0724	0.0303	0.0169	0.0016	158.71
Passenger car	79600567	58.30%	**0.2738**	0.0982	0.0343	**0.0212**	0.0058	**188.12**

Note that these emission factors do not incorporate the increase due to remote sensing devices from other studies, nor real measurement campaigns.

Input data consists of the following data fields:

- **date_time_stamp**
  - Format: 2022-01-29 22:00:00+00:00
- **from_camera**
  - Format: integer
- **vehicle_type_info**
  - A list containing the numbers for:
    - Articulated truck
    - Bus
    - Disabled car
    - Moped/quad
    - Motorcycle
    - Not Applicable
    - Passenger car
    - Road works vehicle
    - Tractor
    - Truck
    - Van

    ➔ **From these values we only use Articulated truck, bus, motorcycle, passenger car, truck, van**

Their respective volumes are each time multiplied with the associated emissions factors, and then summed together. For the emission factors, we use the following associations with the camera fields:

- Articulated truck → Articulated truck
- Bus → Average bus
- Motorcycle → Motorcycle
- Passenger car → Passenger car
- Truck → Rigid truck
- Van → Van

Vehicle type	NOx	PM2.5	CO2
Average bus	2.4879	0.0709	748.70
Articulated truck	2.1708	0.0700	510.43
Rigid truck	1.3690	0.0578	344.40
Van	0.8418	0.0286	244.38
Motorcycle	0.0561	0.0116	88.22
Passenger car	0.2738	0.0212	188.12

4.3.6 Additional features

Based on the technical elaboration in Section 4.3.1, we foresee some possible additional features that can be incorporated to make the model more applicable.

In first instance, using real Copert V functions would augment the correctness of the emission estimations. This however entails the use of speed-dependent information in the network, but will automatically give time-of-day dependence. In order to accomplish this, the following inputs need to be available:

- Emission factors for pollutants CO2, NOx, and PM2.5 (dependent on the type of vehicle and its speed)
- Average speed information corresponding to links in the road network that are associated with the earlier mentioned ANPR cameras

Furthermore, the emission factors are now based on class (i.e. vehicle type) averages. A further refinement can be done by separating out the individual vehicle types. A first approach could be to use:

- Fleet composition

This would allow us to further refine the fleet mix used to calculate the average emission factor. Further details can be obtained by another step, in which each detected vehicle type is used together with a specific speed-dependent emission factor.

At the level of further calculations and comparisons with the real-word recorded emissions by the emission detector stations, we could analyse their correlations by also incorporating:

- Historical records of traffic conditions and/or volumes
- Historical records of weather conditions
- Historical records of events
- Historical air quality-related data
- Meteorological forecasts
- Planned events in the area of interest

4.3.7 Code and quality control

The computational flow described in the previous section is functionally prototyped using Anaconda Individual Edition 2020.02 relying upon Python Release 3.9.12. The dependencies of the code are as follows:

- Json library
- Pandas library
- Multiprocessing library
- Datetime library
- Sqlalchemy library

The third tool's code is comprised of the following functions:

- main: is the function in which all the necessary processes are conducted.
- main_new: is the function which calls the "main" function. The execution of the "main" function sets to a thread and the current function monitors the execution process of the "main" function to not exceed the time-limit of the AWS Lambda function (15 minutes).
- db_connection: is the function through which the results of the simulations as well as from the analysis are stored in the database (PostgreSQL).
- input_parameters: is the function by which the algorithm retrieves all the necessary input data from the database (PostgreSQL) for the selected scenario.
- allsundays: is the function by which the algorithm finds all the dates of the Sundays for a given year.
- find_dates_list: is the function through which the algorithm calculates all the dates between the start date and the end date (based on the user input).
- data_extraction_from_sql: is the function by which the algorithm loops through the selected cameras and dates and retrieves all the information from the respective tables in PostgreSQL.
- aata_aggregation: is the function through which the algorithm loops through the retrieved data and aggregates these data in an hour interval.
- graph_index_calculation: is the function of the algorithm which calculates all the time intervals between the start time and the end time as have been selected by the user.
- table_formated_data: is the function which formats the calculated – aggregated data in to the final format that is need it in order to be stored at the database (PostgreSQL) and presented at the front-end.

The results of the code quality control, by utilizing the methodology described in Section 3.2, are included in Table 16.

Table 16: Code quality control results – 3rd tool (UC3)

	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	main function	CC index	C (19)	moderate - slightly

	Evaluation element	Metric	Rank (Score)	Interpretation
				complex block
	main_new function	CC index	A (2)	low - simple block
	db_connection function	CC index	A (1)	low - simple block
	input_parameters function	CC index	A (3)	low - simple block
	allsundays function	CC index	A (2)	low - simple block
	find_dates_list function	CC index	A (2)	low - simple block
	data_extraction_from_sql function	CC index	A (4)	low - simple block
	data_aggregation function	CC index	B (7)	low - well structured and stable block
	graph_index_calculation function	CC index	A (2)	low - simple block
	table_formated_data function	CC index	B (6)	low - well structured and stable block
Maintainability	Entire UC3 .py file	Maintainability index	A (51.80)	Very high maintainability
Raw metrics	Entire UC3 .py file	LOC	414	total # lines of code
	Entire UC3 .py file	LLOC	235	total # logical lines of code[30]

[30] Logical lines of code may be defined as lines of code including executable expressions.

	Evaluation element	Metric	Rank (Score)	Interpretation
	Entire UC3 .py file	SLOC	235	total # source lines of code[31]
	Entire UC3 .py file	comments	87	total # comment lines
	Entire UC3 .py file	multi	4	total # lines representing multi-line strings
	Entire UC3 .py file	blank	90	total # blank lines
Hal metrics	Entire UC3 .py file	h_1	8	total # distinct operators[32]
	Entire UC3 .py file	h_2	48	total # distinct operands[33]
	Entire UC3 .py file	N_1	33	total # operators
	Entire UC3 .py file	N_2	66	total # operands
	Entire UC3 .py file	Vocabulary	56	$n = n_1 + n_2$
	Entire UC3 .py file	Length	99	$N = N_1 + N_2$
	Entire UC3 .py file	calculated_length	292.08	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Entire UC3 .py file	Volume	574.93	$V = N \log_2(n)$
	Entire UC3 .py file	Difficulty	5.50	$D = (n_1/2)x (N_2/n_2)$
	Entire UC3 .py file	Effort	3162.1	$E = D \times V$

[31] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.
[32] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[33] The values that an operator operates.

	Evaluation element	Metric	Rank (Score)	Interpretation
			1	
	Entire UC3 .py file	Time	175.67	T = E/18 seconds
	Entire UC3 .py file	Bugs	0.19	B = V/3000

The average cyclomatic complexity of the UC3 code file is ranked as A (4.80). Hence, it can be considered as a well-structured and stable piece of code.

4.3.8 Demonstration and testing

Let us take for example camera #236, and consider the recorded traffic volumes between 01/07/2021 and 31/07/2021, as shown in Table 17.

Table 17: Traffic volumes recorded by camera #236, green/white/red denoting low/medium/high traffic counts.

year	month	day	camera_id	h01	h02	h03	h04	h05	h06	h07	h08	h09	h10	h11	h12	h13	h14	h15	h16	h17	h18	h19	h20	h21	h22	h23	h24
2021	7	1	236	13	5	4	25	87	337	570	644	599	593	624	655	443	444	528	582	703	676	580	367	194	158	127	30
2021	7	2	236	11	13	8	30	87	308	576	583	588	564	594	646	511	484	591	728	573	498	477	364	190	172	132	68
2021	7	3	236	49	25	22	13	32	86	201	344	418	447	495	464	307	204	222	292	329	343	403	320	167	140	135	79
2021	7	4	236	50	39	21	14	20	37	100	140	223	290	326	352	286	156	178	263	334	325	373	323	174	92	43	19
2021	7	5	236	10	8	11	26	61	321	601	625	455	516	565	623	451	401	516	573	596	630	525	337	171	137	51	29
2021	7	6	236	7	9	6	31	66	308	629	624	511	553	527	635	485	433	563	610	667	591	585	261	126	121	118	32
2021	7	7	236	10	7	10	19	79	309	605	632	498	538	548	642	478	440	486	620	637	554	531	329	204	128	78	32
2021	7	8	236	7	10	4	20	70	290	627	654	522	539	571	585	447	440	576	601	658	641	560	358	201	144	117	52
2021	7	9	236	14	17	14	21	75	282	547	647	442	525	606	647	564	519	568	611	563	524	486	345	178	162	126	88
2021	7	10	236	40	31	13	16	33	94	194	295	343	417	456	395	310	208	202	255	300	353	352	314	180	182	129	92
2021	7	11	236	42	25	14	11	18	37	83	134	225	253	308	329	236	172	173	279	341	358	374	304	154	103	106	23
2021	7	12	236	9	9	18	22	67	312	542	592	503	509	506	588	418	398	554	582	628	592	491	345	163	102	60	22
2021	7	13	236	6	6	8	27	94	335	605	631	521	544	564	589	413	398	511	587	594	599	518	380	177	139	77	34
2021	7	14	236	13	7	12	25	78	307	590	617	473	503	533	623	460	399	528	614	616	633	565	373	182	130	87	29
2021	7	15	236	9	8	9	21	72	309	593	636	508	502	545	644	426	406	532	586	638	613	546	385	182	158	126	36
2021	7	16	236	8	6	12	27	74	301	560	592	530	573	583	602	490	421	533	593	557	566	481	369	169	151	153	38
2021	7	17	236	6	7	3	10	39	80	218	311	415	431	437	454	295	168	183	228	279	355	371	288	195	156	128	40
2021	7	18	236	6	2	5	9	18	34	92	145	240	292	313	343	249	121	163	223	298	346	349	269	178	91	68	16
2021	7	19	236	6	3	7	23	62	292	524	554	489	511	512	576	392	420	483	566	585	551	473	302	162	116	57	17
2021	7	20	236	2	3	11	27	61	292	558	639	528	547	531	566	420	358	492	592	580	609	535	368	165	142	92	14
2021	7	21	236	3	3	8	11	68	299	541	572	496	536	487	576	415	437	527	564	566	594	506	332	174	138	91	17
2021	7	22	236	4	4	8	27	74	290	556	612	534	599	555	597	437	438	519	595	622	617	617	406	234	188	121	30
2021	7	23	236	4	7	2	21	84	277	530	576	524	554	536	588	488	449	561	616	588	541	488	340	211	156	165	49
2021	7	24	236	12	5	5	11	21	88	183	317	369	414	427	379	268	182	161	224	304	354	360	269	177	157	175	48
2021	7	25	236	12	6	4	7	17	41	86	133	210	248	305	353	243	142	187	235	290	313	365	281	166	113	83	10
2021	7	26	236	6	5	6	20	79	287	443	550	485	517	529	514	448	390	508	489	561	495	468	286	154	132	68	16
2021	7	27	236	3	8	7	15	72	268	494	516	455	517	530	567	412	326	460	515	587	621	556	323	193	164	118	25
2021	7	28	236	4	5	9	20	74	281	461	515	444	519	502	527	418	372	425	521	604	594	525	356	183	147	146	25
2021	7	29	236	5	2	6	15	62	280	476	563	520	502	511	557	418	402	461	548	557	584	517	380	201	158	156	33
2021	7	30	236	14	2	4	19	75	277	477	489	503	532	523	593	502	417	521	553	527	528	508	342	197	145	173	47
2021	7	31	236	3	12	7	8	34	94	193	309	328	413	482	381	307	184	215	273	290	369	320	226	198	116	102	27

This is shown graphically in Figure 25.

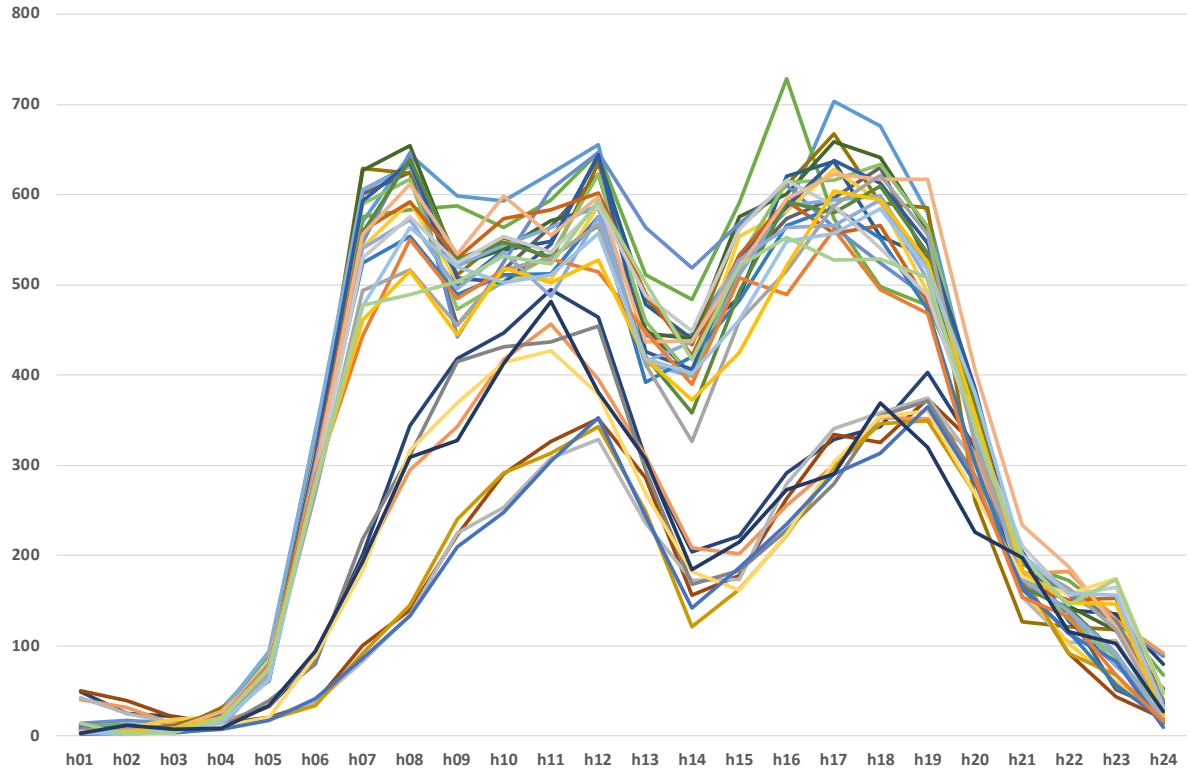


Figure 25: Traffic volumes for each of the 31 days in July 2021, shown for camera #236 per hour of the day.

Here we can see the presence of two rather sharp morning peaks and a broader evening peak during the week days, with only a later, smaller morning peak present during Saturdays and an even later one during Sundays.

Based on the emission factors from Section 4.3.5, we can now calculate the emissions. Figure 26 shows the result as a run plot of all the days in the discussed sample, whereby we each time give the sum (blue curve) and average (red curve) of all the emissions between 6h in the morning and 22h (including) in the evening.

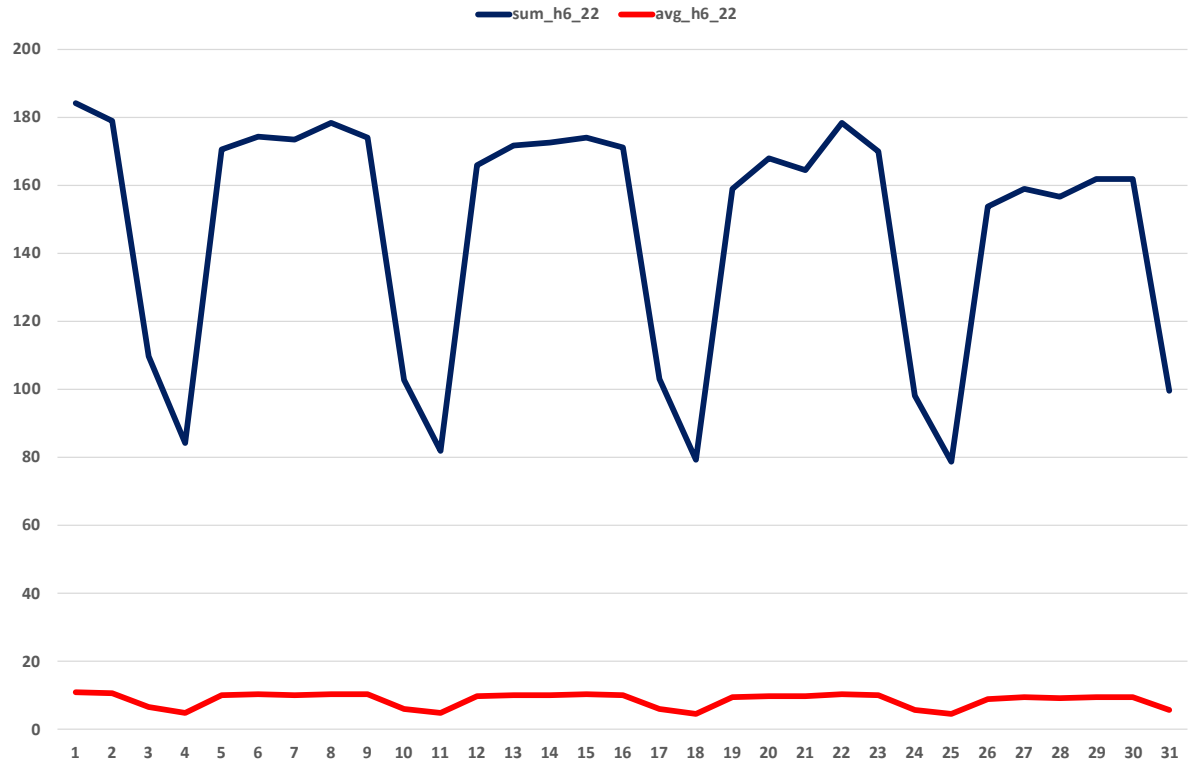


Figure 26: Sum and average emissions calculated between 6h and 22h (including) for camera #236 in July 2021.

By applying the methodology from the computational flow to an example, we get the following results.

Starting from the counts observed by, e.g., camera # 302:

date_time_stamp	from_camera	vehicle_type_info
2022-01-29 22:00:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 0, "Not Applicable": 0, "Pa
2022-01-29 22:05:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 0, "Not Applicable": 0, "Pa
2022-01-29 22:10:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 2, "Not Applicable": 0, "Pa
2022-01-29 22:15:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 2, "Not Applicable": 0, "Pa
2022-01-29 22:20:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 0, "Not Applicable": 0, "Pa
2022-01-29 22:25:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 1, "Not Applicable": 0, "Pa
2022-01-29 22:30:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 1, "Not Applicable": 0, "Pa
2022-01-29 22:35:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 2, "Not Applicable": 0, "Pa
2022-01-29 22:40:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 0, "Not Applicable": 0, "Pa
2022-01-29 22:45:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 2, "Not Applicable": 0, "Pa
2022-01-29 22:50:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 2, "Not Applicable": 0, "Pa
2022-01-29 22:55:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 0, "Not Applicable": 0, "Pa
2022-01-29 23:00:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 4, "Not Applicable": 0, "Pa
2022-01-29 23:05:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 2, "Not Applicable": 0, "Pa
2022-01-29 23:10:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 0, "Not Applicable": 0, "Pa
2022-01-29 23:15:00+00:00	302	{"Articulated truck": 0, "Bus": 0, "Disabled car": 0, "Moped/quad": 0, "Motorcycle": 0, "Not Applicable": 0, "Pa

Filtering out the relevant fields, leads to the following information:

date_time_stamp	from_camera	Articulated truck	Bus	Motorcycle	Passenger car	Truck	Van
2022-01-29 22:00:00+00:00	302	0	0	0	5	0	0
2022-01-29 22:05:00+00:00	302	0	0	0	9	0	0
2022-01-29 22:10:00+00:00	302	0	0	2	5	0	0
2022-01-29 22:15:00+00:00	302	0	0	2	4	0	0
2022-01-29 22:20:00+00:00	302	0	0	0	6	0	0
2022-01-29 22:25:00+00:00	302	0	0	1	5	0	1
2022-01-29 22:30:00+00:00	302	0	0	1	7	0	0
2022-01-29 22:35:00+00:00	302	0	0	2	5	1	1
2022-01-29 22:40:00+00:00	302	0	0	0	3	0	0
2022-01-29 22:45:00+00:00	302	0	0	2	4	1	0
2022-01-29 22:50:00+00:00	302	0	0	2	8	0	0
2022-01-29 22:55:00+00:00	302	0	0	0	6	0	1
2022-01-29 23:00:00+00:00	302	0	0	4	4	0	0
2022-01-29 23:05:00+00:00	302	0	0	2	5	1	0
2022-01-29 23:10:00+00:00	302	0	0	0	1	0	0
2022-01-29 23:15:00+00:00	302	0	0	0	5	0	0

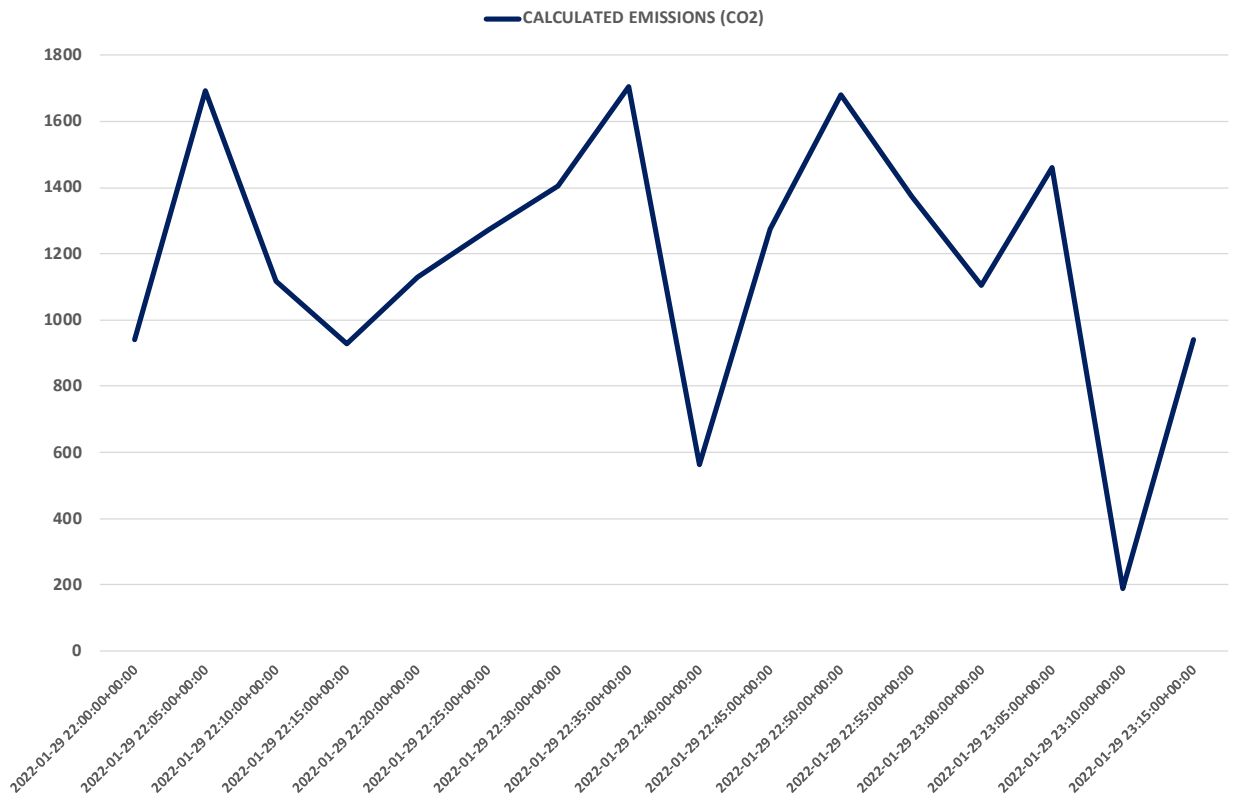
Multiplying each number by the relevant emission factors, gives the following outputs for CO2, NOx, and PM2.5:

CALCULATED EMISSIONS (CO2)								
date_time_stamp	from_camera	Articulated truck	Bus	Motorcycle	Passenger car	Truck	Van	Total
2022-01-29 22:00:00+00:00	302	0.0	0.0	0.0	940.6	0.0	0.0	**941**
2022-01-29 22:05:00+00:00	302	0.0	0.0	0.0	1693.1	0.0	0.0	**1693**
2022-01-29 22:10:00+00:00	302	0.0	0.0	176.4	940.6	0.0	0.0	**1117**
2022-01-29 22:15:00+00:00	302	0.0	0.0	176.4	752.5	0.0	0.0	**929**
2022-01-29 22:20:00+00:00	302	0.0	0.0	0.0	1128.7	0.0	0.0	**1129**
2022-01-29 22:25:00+00:00	302	0.0	0.0	88.2	940.6	0.0	244.4	**1273**
2022-01-29 22:30:00+00:00	302	0.0	0.0	88.2	1316.8	0.0	0.0	**1405**
2022-01-29 22:35:00+00:00	302	0.0	0.0	176.4	940.6	344.4	244.4	**1706**
2022-01-29 22:40:00+00:00	302	0.0	0.0	0.0	564.4	0.0	0.0	**564**
2022-01-29 22:45:00+00:00	302	0.0	0.0	176.4	752.5	344.4	0.0	**1273**
2022-01-29 22:50:00+00:00	302	0.0	0.0	176.4	1505.0	0.0	0.0	**1681**
2022-01-29 22:55:00+00:00	302	0.0	0.0	0.0	1128.7	0.0	244.4	**1373**
2022-01-29 23:00:00+00:00	302	0.0	0.0	352.9	752.5	0.0	0.0	**1105**
2022-01-29 23:05:00+00:00	302	0.0	0.0	176.4	940.6	344.4	0.0	**1461**
2022-01-29 23:10:00+00:00	302	0.0	0.0	0.0	188.1	0.0	0.0	**188**
2022-01-29 23:15:00+00:00	302	0.0	0.0	0.0	940.6	0.0	0.0	**941**

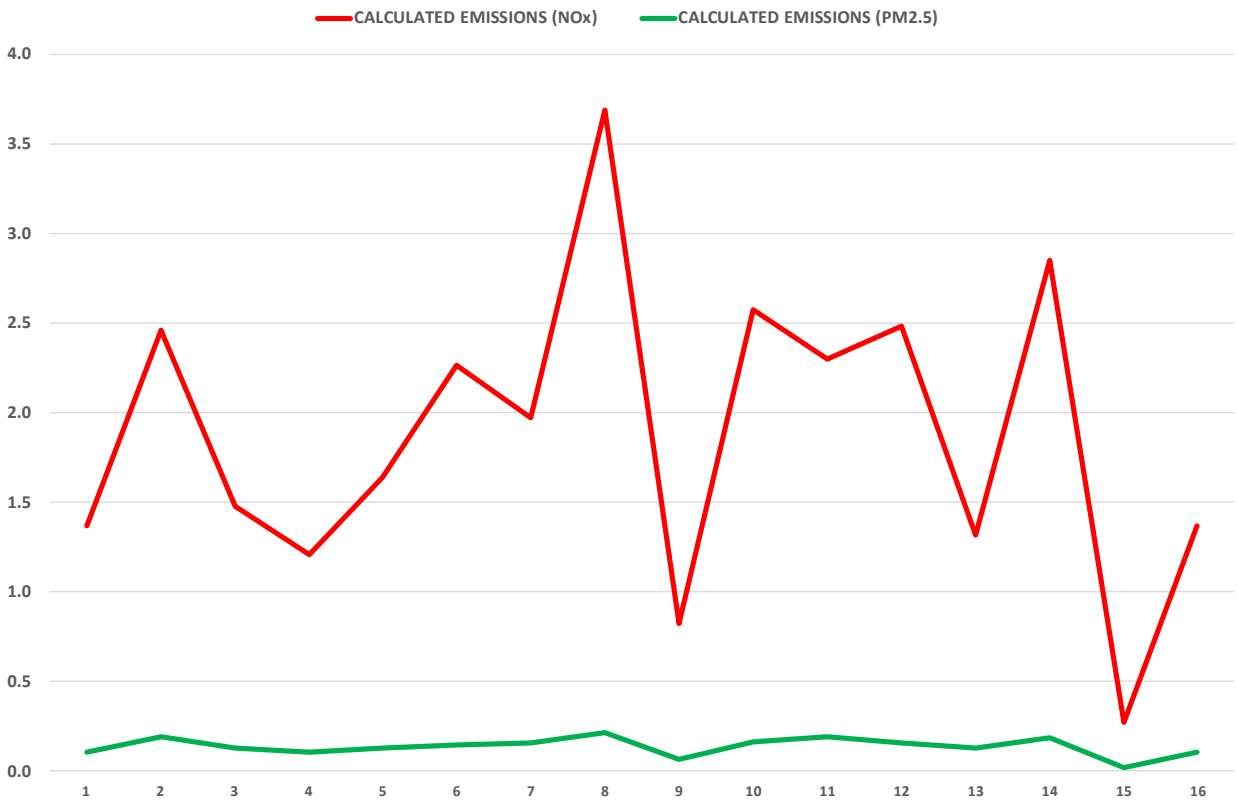
CALCULATED EMISSIONS (NOx)								
date_time_stamp	from_camera	Articulated truck	Bus	Motorcycle	Passenger car	Truck	Van	Total
2022-01-29 22:00:00+00:00	302	0.0	0.0	0.0	1.4	0.0	0.0	**1.4**
2022-01-29 22:05:00+00:00	302	0.0	0.0	0.0	2.5	0.0	0.0	**2.5**
2022-01-29 22:10:00+00:00	302	0.0	0.0	0.1	1.4	0.0	0.0	**1.5**
2022-01-29 22:15:00+00:00	302	0.0	0.0	0.1	1.1	0.0	0.0	**1.2**
2022-01-29 22:20:00+00:00	302	0.0	0.0	0.0	1.6	0.0	0.0	**1.6**
2022-01-29 22:25:00+00:00	302	0.0	0.0	0.1	1.4	0.0	0.8	**2.3**
2022-01-29 22:30:00+00:00	302	0.0	0.0	0.1	1.9	0.0	0.0	**2.0**
2022-01-29 22:35:00+00:00	302	0.0	0.0	0.1	1.4	1.4	0.8	**3.7**
2022-01-29 22:40:00+00:00	302	0.0	0.0	0.0	0.8	0.0	0.0	**0.8**
2022-01-29 22:45:00+00:00	302	0.0	0.0	0.1	1.1	1.4	0.0	**2.6**
2022-01-29 22:50:00+00:00	302	0.0	0.0	0.1	2.2	0.0	0.0	**2.3**
2022-01-29 22:55:00+00:00	302	0.0	0.0	0.0	1.6	0.0	0.8	**2.5**
2022-01-29 23:00:00+00:00	302	0.0	0.0	0.2	1.1	0.0	0.0	**1.3**
2022-01-29 23:05:00+00:00	302	0.0	0.0	0.1	1.4	1.4	0.0	**2.9**
2022-01-29 23:10:00+00:00	302	0.0	0.0	0.0	0.3	0.0	0.0	**0.3**
2022-01-29 23:15:00+00:00	302	0.0	0.0	0.0	1.4	0.0	0.0	**1.4**

CALCULATED EMISSIONS (PM2.5)								
date_time_stamp	from_camera	Articulated truck	Bus	Motorcycle	Passenger car	Truck	Van	Total
2022-01-29 22:00:00+00:00	302	0.00	0.00	0.00	0.11	0.00	0.00	**0.11**
2022-01-29 22:05:00+00:00	302	0.00	0.00	0.00	0.19	0.00	0.00	**0.19**
2022-01-29 22:10:00+00:00	302	0.00	0.00	0.02	0.11	0.00	0.00	**0.13**
2022-01-29 22:15:00+00:00	302	0.00	0.00	0.02	0.08	0.00	0.00	**0.11**
2022-01-29 22:20:00+00:00	302	0.00	0.00	0.00	0.13	0.00	0.00	**0.13**
2022-01-29 22:25:00+00:00	302	0.00	0.00	0.01	0.11	0.00	0.03	**0.15**
2022-01-29 22:30:00+00:00	302	0.00	0.00	0.01	0.15	0.00	0.00	**0.16**
2022-01-29 22:35:00+00:00	302	0.00	0.00	0.02	0.11	0.06	0.03	**0.22**
2022-01-29 22:40:00+00:00	302	0.00	0.00	0.00	0.06	0.00	0.00	**0.06**
2022-01-29 22:45:00+00:00	302	0.00	0.00	0.02	0.08	0.06	0.00	**0.17**
2022-01-29 22:50:00+00:00	302	0.00	0.00	0.02	0.17	0.00	0.00	**0.19**
2022-01-29 22:55:00+00:00	302	0.00	0.00	0.00	0.13	0.00	0.03	**0.16**
2022-01-29 23:00:00+00:00	302	0.00	0.00	0.05	0.08	0.00	0.00	**0.13**
2022-01-29 23:05:00+00:00	302	0.00	0.00	0.02	0.11	0.06	0.00	**0.19**
2022-01-29 23:10:00+00:00	302	0.00	0.00	0.00	0.02	0.00	0.00	**0.02**
2022-01-29 23:15:00+00:00	302	0.00	0.00	0.00	0.11	0.00	0.00	**0.11**

Or visually for CO_2:



And for NOx and PM2.5:



4.4 Planning for parking

4.4.1 Overview

Due to the increase of the number of private vehicles within urban areas there is a shortage in the supply of parking facilities (Guo, et al., 2016). Such a situation is further exacerbated by vehicles circulating in search of a parking lot. In this respect and considering that parking is an important traffic generator, an effective measure for alleviating the above issues and reducing private vehicle attractiveness within metropolitan areas is the reduction of on-street parking space.

The scope of the current tool of the nuMIDAS toolkit is to support the impact assessment of on-street parking restriction policies. In the current version, these impacts include the parking pressure relocated from the area in which a parking restriction policy has been enforced to adjacent areas and increases in parking searching time. The operational logic of this tool relies on the discretisation of a road network using a grid comprised of cells of appropriate size and the use of parameters, such as the demand for parking and parking capacity (number of parking places) corresponding to each cell. By that means, the tool will enable the simulation of enforcing a parking restriction policy in one or more grid cells and the assessment of its impacts on the remaining cells.

4.4.2 Targeted users

The stakeholders involved in the ecosystem of the current tool are the following:

- Departments of a local governments responsible for on-street parking management (policy makers)
- Off-street parking managers (private service providers)
- Transport planners supporting policy makers
- Traffic police
- Drivers

However, among the above stakeholders the ones that belong to the targeted users of the current tool are transport planners and policy makers. The formers are understood as the ones providing input to the tool and the latter as the ones receiving the outputs of the tool.

4.4.3 Data inputs

The inputs provided to the tool can be divided into values imported from a file or database and user defined. The former includes:

- Parking capacity
- Parking demand
- Spatial/geometric information

As indicated in Section 4.2.1, the above information should correspond to each cell of the grid network representing the whole road network of the area of interest. Spatial/geometric information constitutes a generalised input given that in some cases a grid network may be available, while in other cases a grid network can be generated by the tool by utilizing appropriate geo-processing capabilities of open-source solutions (e.g., pyQGIS). A crucial parameter that should be provided in any case constitutes the dimension of the grid network, including unique identifiers for each cell.

On the other hand, the user defined input includes the following:

- Declaration of the cell(s) into which a restriction policy is enforced
- Percentage of parking capacity reduced
- Policy effect expansion level
- Average spacing between on-street parking places
- Typical length of a parking place
- Average speed of road users while searching for parking

As it is described in detail in Section 4.2.5, the user of the tool can assess the effect of a parking restriction policy enforced in one or more cells of the grid network. Moreover, the user can assess the effect of parking restriction policy, involving either full or partial restriction of parking with an area represented by a cell. Finally, it should be noted that the user can set the expansion level of the policy effect so as to account for varying road user behaviours and responses to restriction policies.

4.4.4 Data outputs

The outputs of the tool in its current version are the following:

- Updated parking capacity
- Updated parking demand
- Initial searching time
- Updated searching time
- Searching time difference

4.4.5 Computational flow

As already mentioned in Section 4.2.1, the scope of the current tool is to assess the impacts of parking restriction policies to adjacent areas. At the current version of the tool, focus is given on the parking pressure relocated to adjacent areas as well as the increase in external costs, such as the average searching time. Moreover, as also mentioned, the operational logic of the current tool relies on the discretisation of a road network using a grid comprised of cells of appropriate size and relevant parameters. To this extent, the geographical area into which a restriction policy is enforced is indicated through the cells of the road network's grid. Moreover, the impacts of a restriction policy are reported at the grid level. The overall computational flow of the fourth tool is depicted in Figure 27.

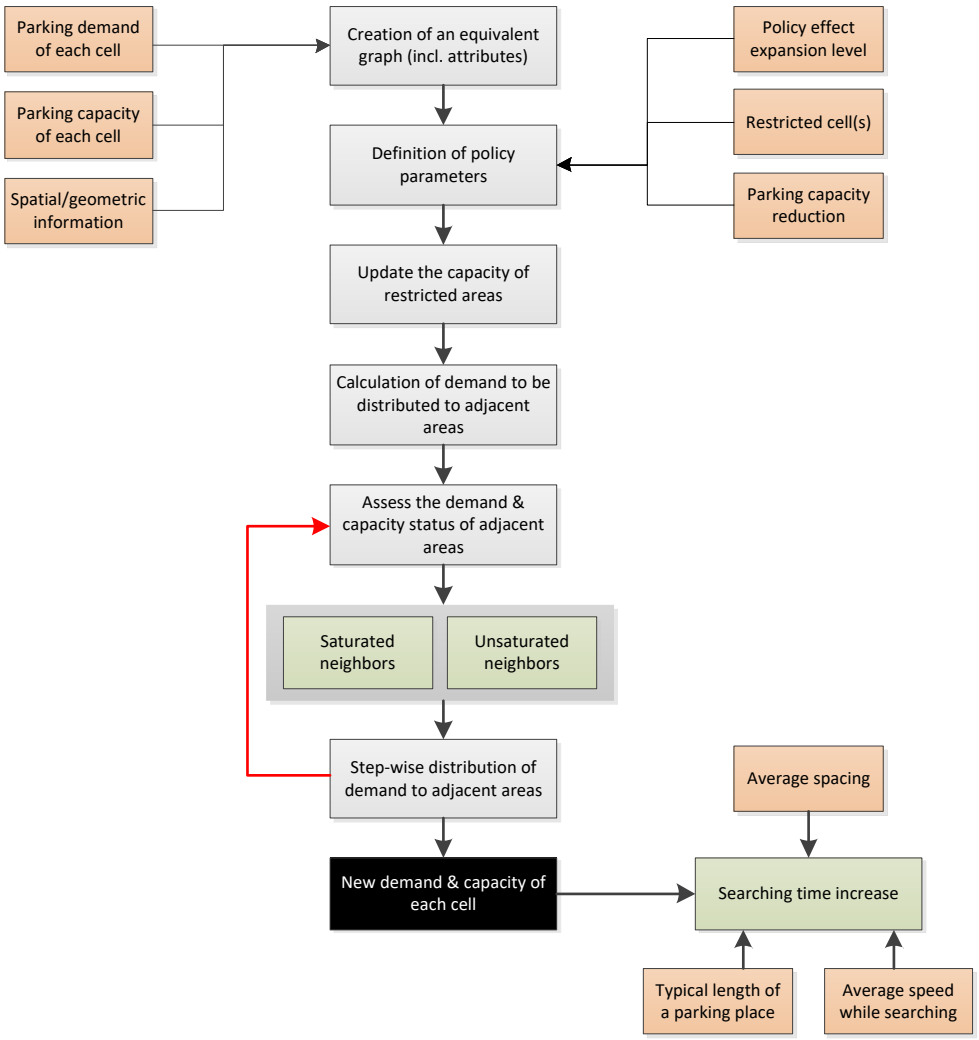


Figure 27: Computational flow of the fourth tool of nuMIDAS toolkit

The first step of the computational flow involves the creation of an undirected graph, including the same number of vertices with the number of cells of the road network's grid, thus serving as an analogue of the road network's grid. This graph is undirected considering that its purpose is solely to facilitate the indication of the connectivity among cells (or vertices). Such a graph has the shape of a grid graph in which additional vertices are added to directly connect its vertices diagonally (Figure 28). By that means, it becomes readily manageable to a define an adjacency matrix A_{ij} and, thus, identify the neighbours of each cell (or vertex). Adjacency matrix has the following structure:

$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Values equal to 0 indicate that vertex *i* is not connected with vertex *j*, while the opposite holds true for values equal to 1.

Having defined the adjacency matrix, the next step involves the assignment of the cell's attributes (i.e., parking demand and capacity) to the associated vertices of the equivalent graph. Afterwards, the user shall provide an input indicating the cell or group of cells into which a parking restriction policy is about to be enforced. Moreover, the user shall provide an input regarding the extent to which parking capacity will be reduced in this (or these) cell(s).

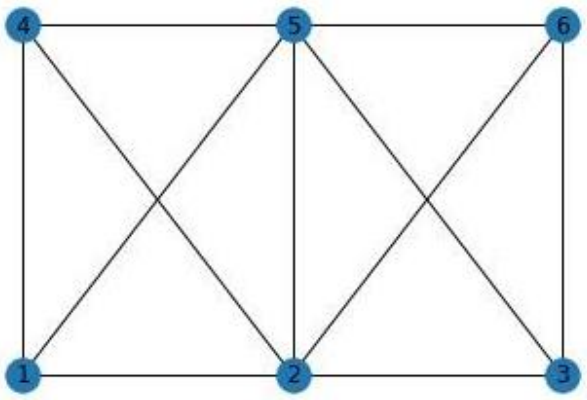


Figure 28: Structure of equivalent graph

The demand to be distributed into the adjacent areas (PD^*) is calculated by the tool via the following rules:

- If $PD_0/PC_0 \geq 1$, then $PD^* = PD_0 - PC_r$
- If $PD_0/PC_0 \leq 1$ and $PC_0 - PC_r > PC_0 - PD_0$, then $PD^* = PD_0 - PC_r$
- If $PD_0/PC_0 \leq 1$ and $PC_0 - PC_r \leq PC_0 - PD_0$, then $PD^* = 0$

where PD_0 denotes the initial demand corresponding to each cell, PC_0 denotes the initial parking capacity of each cell, and PC_r denotes the updated value of parking capacity after the enforcement of the restriction policy.

A significant computational step of the tool constitutes the identification of saturated and non-saturated neighbours in terms of parking demand to parking capacity ratio. On this assessment the stepwise distribution of PD^* relies. The steps taken for this purpose are summarised in the pseudocode included in the following block.

Step 0: Identification of saturated and non-saturated neighbours
Condition 1: Non-saturated neighbour(s) exist(s)
    Step 1.1: Calculation of the quantity d_i^* to be distributed: $d_i^* = \frac{PD^*}{non-saturated\ neighbors}$
    Condition 1.1: Non-saturated neighbours have the required capacity to attract d^*
        Step 1.1.1: Uniform distribution of d_i^*
        Step 1.1.2: END
    Condition 1.2: Non-saturated neighbours can partially attract d_i^*
        Step 1.2.1: Capacity-restrained distribution of d_i^*
        Step 1.2.2: Calculation of the residual quantity
        Step 1.2.3: Re-identification of saturated and non-saturated neighbours
        Condition 1.2.1: Non-saturated neighbours have the required capacity to attract the residual quantity
            Step 1.2.1.1: Uniform distribution of the residual quantity
            Step 1.2.1.2: END

Condition 1.2.2: Non-saturated neighbours can partially attract the residual quantity
Step 1.2.2.1: Capacity-restrained distribution of the residual quantity
Step 1.2.2.2: Calculation of the *new* residual quantity
…..
Condition 1.2.3: All neighbours are saturated
Step 1.2.3.1: Weighted distribution of the residual quantity based on demand to capacity ratio
Step 1.2.3.2: END
Condition 2: All neighbours are saturated
Step 2.1: Weighted distribution of the PD^* based on demand to capacity ratio

The logic of equal and capacity-restrained distribution of parking demand to unsaturated neighbours relies on the premise that road-users prefer to search for a parking space in areas that are to the extent possible close to their destination (thus minimizing their egress time) and provide more vacant space (thus minimizing their searching time). In addition, the algorithm included in the above block applies as is when the user indicates that the policy effect expansion level is equal to 1. If the user indicates a greater value, the tool repeats the algorithm until the neighbours of the nth level are reached. The condition based on which the tool continues this exploratory process is that all neighbours are saturated. Provided that even one neighbour is assessed as non-saturated, this translates that PD^* have been already attracted by the neighbours of the (n-j)th level (Figure 29).

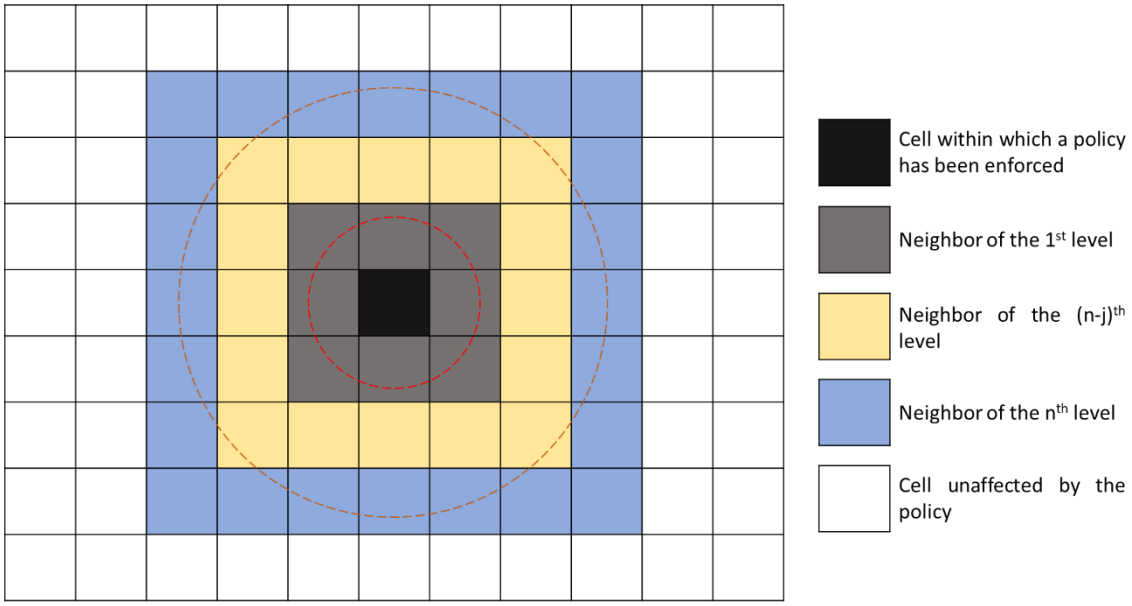


Figure 29: Neighbours notation

The last step of the algorithm involves the assessment of the searching time increase. The estimation of searching time in each cell is based on the suggestions made by Yan et al. (2019) and Inci et al. (2017). Specifically, the following formulation is adopted:

$$Searching\ time = \begin{cases} (1-V) \times C \times \dfrac{l+l'}{v_{parking}}, & when\ \dfrac{D}{C} \leq 1 \\ (1-V)^{(1+\frac{D}{C})} \times C \times \dfrac{l+l'}{v_{parking}}, & when\ \dfrac{D}{C} > 1 \end{cases}$$

where V denotes the average vacancy rate within the area of interest, C denotes the parking capacity within the area of interest, l denotes average length of a typical parking place, l' the average spacing between parking places (considering prohibited areas and intersections), and $v_{parking}$ the average travel speed when searching for time.

The exponent included in the above equation expresses that under parking saturation conditions, more than one road user should be in search of a vacant space. In such a case, the probability for each individual road user to find a vacant space is drastically reduced.

4.4.6 Additional features

No additional features have been incorporated in the fourth tool of the nuMIDAS toolkit as all the requirements settled into D2.2 have taken into account through the first version of the tool.

4.4.7 Code and quality control

The computational flow described in the previous section is functionally prototyped using Anaconda Individual Edition 2020.02 relying upon Python Release 3.9.12. The dependencies of the code are as follows:

- Networkx library
- Numpy library
- Pandas library
- Datetime library
- Time library
- Math library
- Multiprocessing library
- Sqlalchemy method of Create_engine library

The fourth tool's code is comprised of the following classes and functions per class:

- Retrieve_Input_Parameters: is the class which includes the code for retrieving all the required input from the front-end.
  - input_parameters: is the function in by which the algorithm retrieves all the necessary input data from the database (PostgreSQL) for the selected scenario.
- Graph_Attributes: is the class including the code for assigning the appropriate attributes to each node of the equivalent graph.
  - add_diagonial_edges_new: is the function through which the diagonal connections of the graph are created.
  - add_attributes_to_graph: is the function through which the demand and the capacity of each cell is assigned to the nodes of the graph.
  - add_new_attributes_to_graph: is the function through which the updated demand and capacity of each cell is assigned to the nodes of the graph (after the enforcement of a parking restriction policy).
  - remove_node_connections: is the function through which the unnecessary connections between the nodes are removed from the graph.

- add_demand_to_active_nodes: is the function through which the algorithm redistributes the excess demand to nodes which have demand lower than capacity.
- add_demand_to_active_nodes_common: is the function through which the algorithm redistributes the excess demand to nodes which have demand lower than capacity and these nodes are also common for more than two nodes.
- found_nodes_with_specific_demand: is the function through which the algorithm finds nodes of the graph which have specific demand values.

- Neighbour_Attributes: is the class including the code for identifying both the saturated and unsaturated neighbours of a graph's node.
  - neighbours_found: is the function which identifies the neighbours that a node of the graph has.
  - active_neighbours_found: is the function which determines the unsaturated neighbours a node of the graph has, based on the difference between the initial demand and capacity values.
  - active_neighbours_found_new: is the function which determines the unsaturated neighbours a node of the graph has, based on the difference between the demand and capacity values after the implementation of the parking restriction policy.
  - active_neighbours_found_new_mulitple: is the function which determines the unsaturated neighbours of multiple nodes of the graph have, based on the difference between their demand and capacity values after the implementation of the parking restriction policy.
- Graph_Calculations: is the class through which the distribution of the excess demand is applied to the neighbours of the nodes of the graph based on the number of the areas the parking restriction policy is applied and the policy effect expansion level
  - calculations: is the function which executes either the one_area_selected function or the two_areas_selected function based on the number of cells/ nodes the parking restriction policy has been applied.
  - one_area_selected: is the function which executes the one_closed_area_calculations function and updates the demand and capacity values after the restriction of the parking restriction policy on each policy effect expansion level.
  - one_closed_area_calculations: is the functions which distributes the demand to the neighbours of a node or n number of nodes when the nodes are both in the neighbours list of each one.
  - one_area_selected_common: is the functions which distributes the demand to the neighbours of a node.
  - more_areas_common_neighboor: is the function which distributes the excess demand to the neighbours of two or more selected nodes.
- Graph_Statistics_Results: is the class through which the algorithm calculates the necessary statistics
  - statistics_results: is the function which calculates all the statistics outputs of the algorithm. Specifically, for each node for which there is a change either in demand or in capacity or both of them, the algorithm calculates the new available parking spots, the new average demand, and the new searching time.
- main: is the function in which all the necessary processes are conducted.

- main_new: is the function which calls the main function and sets this function to a thread in order to monitor the execution of the tool in line with the time limit of the AWS Lambda function (15 minutes).

The results of the code quality control per class, by utilizing the methodology described in Section 3.2, are included in Tables 18, 19, 20, 21, and 22.

Table 18: Code quality control results regarding the first class of the 4th tool (Retrieve_Input_Parameters class)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	input_parameters function	CC index	A (1)	low - simple block
	Retrieve_Input_Parameters class	CC index	A (2)	low - simple block
Maintainability	Retrieve_Input_Parameters class	Maintainability index	A (85.96)	Very high maintainability
Raw metrics	Retrieve_Input_Parameters class	LOC	22	total # lines of code
	Retrieve_Input_Parameters class	LLOC	12	total # logical lines of code[34]
	Retrieve_Input_Parameters class	SLOC	12	total # source lines of code[35]
	Retrieve_Input_Parameters class	comments	2	total # comment lines
	Retrieve_Input_Parameters class	multi	0	total # lines representing multi-line strings
	Retrieve_Input_Parameters class	blank	8	total # blank lines
Hal metrics	Retrieve_Input_Parameters class	h_1	2	total # distinct operators[36]
	Retrieve_Input_Parameters class	h_2	10	total # distinct

[34] Logical lines of code may be defined as lines of code including executable expressions.
[35] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.
[36] May include arithmetic, comparison, logical, bitwise, assignment, special operators.

				operands[37]
	Retrieve_Input_Parameters class	N_1	5	total # operators
	Retrieve_Input_Parameters class	N_2	10	total # operands
	Retrieve_Input_Parameters class	Vocabulary	12	$n = n_1 + n_2$
	Retrieve_Input_Parameters class	Length	15	$N = N_1 + N_2$
	Retrieve_Input_Parameters class	calculated_length	35.22	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Retrieve_Input_Parameters class	Volume	53.77	$V = N \log_2(n)$
	Retrieve_Input_Parameters class	Difficulty	1.0	$D = (n_1/2)x (N_2/n_2)$
	Retrieve_Input_Parameters class	Effort	53.77	E = D x V
	Retrieve_Input_Parameters class	Time	2.99	T = E/18 seconds
	Retrieve_Input_Parameters class	Bugs	0.02	B = V/3000

Table 19: Code quality control results regarding the second class of the 4th tool (Graph_Attributes class)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	add_diagonal_edges_new function	CC index	A (4)	low - simple block
	add_attributes_to_graph function	CC index	A (3)	low - simple block
	add_new_attributes_to_graph function	CC index	A (4)	low – simple block
	remove_node_connections function	CC index	C (12)	moderate - slightly complex block
	add_demand_to_active_nodes function	CC index	B (7)	low - well-structured and stable block

[37] The values that an operator operates.

	add_demand_to_active_nodes_common function	CC index	B (9)	low - well-structured and stable block
	found_nodes_with_specific_demand function	CC index	A (3)	low – simple block
	Graph_Attributes class	CC index	B (7)	low - well-structured and stable block
Maintainability	Graph_Attributes class	Maintainability index	A (48.27)	Very high maintainability
Raw metrics	Graph_Attributes class	LOC	147	total # lines of code
	Graph_Attributes class	LLOC	102	total # logical lines of code[38]
	Graph_Attributes class	SLOC	103	total # source lines of code[39]
	Graph_Attributes class	comments	14	total # comment lines
	Graph_Attributes class	multi	0	total # lines representing multi-line strings
	Graph_Attributes class	blank	30	total # blank lines
Hal metrics	Graph_Attributes class	h_1	13	total # distinct operators[40]
	Graph_Attributes class	h_2	102	total # distinct operands[41]
	Graph_Attributes class	N_1	73	total # operators
	Graph_Attributes class	N_2	145	total # operands

[38] Logical lines of code may be defined as lines of code including executable expressions.
[39] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.
[40] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[41] The values that an operator operates.

	Graph_Attributes class	Vocabulary	115	$n = n_1 + n_2$
	Graph_Attributes class	Length	218	$N = N_1 + N_2$
	Graph_Attributes class	calculated_length	728.69	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Graph_Attributes class	Volume	1492.32	$V = N \log_2(n)$
	Graph_Attributes class	Difficulty	9.24	$D = (n_1/2)x (N_2/n_2)$
	Graph_Attributes class	Effort	13789.3	E = D x V
	Graph_Attributes class	Time	766.07	T = E/18 seconds
	Graph_Attributes class	Bugs	0.50	B = V/3000

Table 20: Code quality control results regarding the third class of the 4th tool (Neighbour_Attributes Class)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	neighbours_found function	CC index	A (3)	low - simple block
	active_neighbours_found function	CC index	A (4)	moderate - slightly complex block
	active_neighbours_found function	CC index	A (4)	low - simple block
	active_neighbours_found_new function	CC index	A (4)	low - simple block
	active_neighbours_found_new_mulitple function	CC index	A (4)	low - simple block
	Neighbour_Attributes class	CC index	A (5)	low - simple block
Maintainability	Neighbours class	Maintainability index	A (74.25)	Very high maintainability
Raw metrics	Neighbours class	LOC	52	total # lines of code

	Neighbours class	LLOC	38	total # logical lines of code[42]
	Neighbours class	SLOC	38	total # source lines of code[43]
	Neighbours class	comments	8	total # comment lines
	Neighbours class	multi	0	total # lines representing multi-line strings
	Neighbours class	blank	6	total # blank lines
Hal metrics	Neighbours class	n_1	3	total # distinct operators[44]
	Neighbours class	n_2	12	total # distinct operands[45]
	Neighbours class	N_1	6	total # operators
	Neighbours class	N_2	12	total # operands
	Neighbours class	Vocabulary	15	$n = n_1 + n_2$
	Neighbours class	Length	18	$N = N_1 + N_2$
	Neighbours class	calculated_length	47.77	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Neighbours class	Volume	70.32	$V = N \log_2(n)$
	Neighbours class	Difficulty	1.5	$D = (n_1/2)x (N_2/n_2)$
	Neighbours class	Effort	105.49	E = D x V
	Neighbours class	Time	5.86	T = E/18 seconds

[42] Logical lines of code may be defined as lines of code including executable expressions.
[43] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.
[44] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[45] The values that an operator operates.

	Neighbours class	Bugs	0.02	B = V/3000

Table 21: Code quality control results regarding the fourth class of the 4th tool (Graph_Calculations Class)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	calculations function	CC index	F (141)	very high - error-prone, unstable block
	one_area_selected function	CC index	C (12)	moderate - slightly complex block
	one_closed_area_calculations function	CC index	F (45)	very high - error-prone, unstable block
	more_areas_common_neighboor function	CC index	F (59)	very high - error-prone, unstable block
	Graph_Calculations class	CC index	F (65)	very high - error-prone, unstable block
Maintainability	Graph_Calculations Class	Maintainability index	C (0.00)	Very high maintainability
Raw metrics	Graph_Calculations Class	LOC	982	total # lines of code
	Graph_Calculations Class	LLOC	645	total # logical lines of code[46]
	Graph_Calculations Class	SLOC	639	total # source lines of code[47]
	Graph_Calculations Class	comments	96	total # comment lines
	Graph_Calculations Class	multi	0	total # lines representing multi-line strings

[46] Logical lines of code may be defined as lines of code including executable expressions.
[47] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.

	Graph_Calculations Class	blank	247	total # blank lines
Hal metrics	Graph_Calculations Class	h_1	16	total # distinct operators[48]
	Graph_Calculations Class	h_2	293	total # distinct operands[49]
	Graph_Calculations Class	N_1	308	total # operators
	Graph_Calculations Class	N_2	622	total # operands
	Graph_Calculations Class	Vocabulary	309	$n = n_1 + n_2$
	Graph_Calculations Class	Length	930	$N = N_1 + N_2$
	Graph_Calculations Class	calculated_length	2465.06	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Graph_Calculations Class	Volume	7692.46	$V = N \log_2(n)$
	Graph_Calculations Class	Difficulty	16.98	$D = (n_1/2)x (N_2/n_2)$
	Graph_Calculations Class	Effort	130640.56	$E = D \times V$
	Graph_Calculations Class	Time	7257.81	T = E/18 seconds
	Graph_Calculations Class	Bugs	2.56	B = V/3000

Table 22: Code quality control results regarding the fifth class of the 4th tool (Graph_Statistics_Results Class)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	statistics_results function	CC index	A (5)	low - simple block
	Graph_Statistics_Results class	CC index	B (6)	low - well-structured and stable block

[48] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[49] The values that an operator operates.

Maintainability	Graph_Statistics_Results Class	Maintainability index	A (56.42)	Very high maintainability
Raw metrics	Graph_Statistics_Results Class	LOC	109	total # lines of code
	Graph_Statistics_Results Class	LLOC	57	total # logical lines of code[50]
	Graph_Statistics_Results Class	SLOC	59	total # source lines of code[51]
	Graph_Statistics_Results Class	comments	4	total # comment lines
	Graph_Statistics_Results Class	multi	0	total # lines representing multi-line strings
	Graph_Statistics_Results Class	blank	48	total # blank lines
Hal metrics	Graph_Statistics_Results Class	h_1	7	total # distinct operators[52]
	Graph_Statistics_Results Class	h_2	55	total # distinct operands[53]
	Graph_Statistics_Results Class	N_1	34	total # operators
	Graph_Statistics_Results Class	N_2	68	total # operands
	Graph_Statistics_Results Class	Vocabulary	62	$n = n_1 + n_2$
	Graph_Statistics_Results Class	Length	102	$N = N_1 + N_2$
	Graph_Statistics_Results Class	calculated_length	337.63	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Graph_Statistics_Results Class	Volume	607.33	$V = N \log_2(n)$
	Graph_Statistics_Results Class	Difficulty	4.33	$D = (n_1/2)x$

[50] Logical lines of code may be defined as lines of code including executable expressions.
[51] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.
[52] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[53] The values that an operator operates.

				(N_2/n_2)
	Graph_Statistics_Results Class	Effort	2628.07	E = D x V
	Graph_Statistics_Results Class	Time	146.00	T = E/18 seconds
	Graph_Statistics_Results Class	Bugs	0.20	B = V/3000

4.4.8 Demonstration and testing

This section aims, firstly, to demonstrate the results of the application of the fourth tool and, secondly, to assess the validity of its results based on various test case scenarios. The first purpose is served by presenting the outcomes of running the current tool's Python script providing sample data corresponding to an abstract road/grid network (Tables 23-25). Table 23 lists the inputs related to the dimension of the area of interest, the assumed road users' speed when searching for a parking place, the assumed geometrical characteristics of parking places, as well as the policy effect expansion level. Table 24, on the other hand, includes both the demand- and capacity-side inputs of the above mentioned (4*4) area. Finally, Table 25 highlights the areas in which a parking restriction policy is enforced (Areas 3 and 4). It is assumed that there will be a 11.11% reduction of the available parking places in Area 3 and a 16.67% reduction of the available parking places in Area 4. The updated demand and capacity values of these areas (before the execution of the tool) are also highlighted. It is noted that the demand values in both areas exceeds the corresponding capacity values. As a result, the excess demand will be considered for distribution into the neighbours of these cells.

Table 23: Basic inputs corresponding to the base test scenario of the fourth tool

Input	Value
Area Dimension (x * y)	4 * 4
Searching Speed (km/h)	11
Average Length of a Parking Space (m)	5
Average Spacing Between Parking Places (m)	6.5
Policy effect expansion level	1

Table 24: Capacity- and demand-side inputs corresponding to the base test scenario of the fourth tool

Area (Capacity - Demand)			
13 Capacity:110 Demand: 95	14 Capacity:120 Demand: 118	15 Capacity:90 Demand: 94	16 Capacity:110 Demand: 111

9 Capacity:90 Demand: 95	10 Capacity:110 Demand: 110	11 Capacity:80 Demand: 79	12 Capacity:90 Demand: 92
5 Capacity:110 Demand: 108	6 Capacity:90 Demand: 99	7 Capacity:120 Demand: 112	8 Capacity:100 Demand: 110
1 Capacity:70 Demand: 75	2 Capacity:80 Demand: 79	3 Capacity:90 Demand: 90	4 Capacity:120 Demand: 123

Table 25: Areas into which a restriction policy is to be enforced (base test scenario of the fourth tool)

Area (Capacity - Demand)			
13 Capacity:110 Demand: 95	14 Capacity:120 Demand: 118	15 Capacity:90 Demand: 94	16 Capacity:110 Demand: 111
9 Capacity:90 Demand: 95	10 Capacity:110 Demand: 110	11 Capacity:80 Demand: 79	12 Capacity:90 Demand: 92
5 Capacity:110 Demand: 108	6 Capacity:90 Demand: 99	7 Capacity:120 Demand: 112	8 Capacity:100 Demand: 110
1 Capacity:70 Demand: 75	2 Capacity:80 Demand: 79	**3** **Capacity:81** **Demand: 90**	**4** **Capacity:100** **Demand: 123**

The outputs of the tool are provided in Tables 26-28. Table 26 provides the average searching time induced by all road users before the enforcement of the restriction policy. Table 27 presents the outputs of the tool in terms of distributing the excess demand into the neighbour areas of Areas 3-4. It is noted that the demand to capacity ration in the neighbour areas is greater than one. This happens because the policy expansion level was set to be equal to 1. In case that the user had provided a greater value to policy expansion level parameter the tool would have distributed the excess demand in the neighbour areas to neighbours of the n^{th} level. Finally, Table 28 presents the updated average searching time induced by all road users after the enforcement of the restriction policy.

Table 26: Searching time per cell before the enforcement of the restriction policy (base test scenario of the fourth tool)

Area (Searching Time)			
13 Searching Time: 0:05:57	14 Searching Time: 0:07:24	15 Searching Time: 0:06:10	16 Searching Time: 0:07:01
9 Searching Time: 0:06:18	10 Searching Time: 0:06:54	11 Searching Time: 0:04:57	12 Searching Time: 0:05:54
5 Searching Time: 0:06:46	6 Searching Time: 0:06:53	7 Searching Time: 0:07:01	8 Searching Time: 0:07:39
1 Searching Time: 0:05:03	2 Searching Time: 0:04:57	3 Searching Time: 0:05:38	4 Searching Time: 0:07:54

Table 27: Distribution of the excess demand to the neighbour areas (base test scenario of the fourth tool)

Area (Capacity - Demand)			
13 Capacity:110 Demand: 95	14 Capacity:120 Demand: 118	15 Capacity:90 Demand: 94	16 Capacity:110 Demand: 111
9 Capacity:90 Demand: 95	10 Capacity:110 Demand: 110	11 Capacity:80 Demand: 79	12 Capacity:90 Demand: 92
5 Capacity:110 Demand: 108	**6** **Capacity:90** **Demand: 101**	**7** **Capacity:120** **Demand: 120**	**8** **Capacity:100** **Demand: 110**
1 Capacity:70 Demand: 75	**2** **Capacity:80** **Demand: 101**	**3** **Capacity:81** **Demand: 81**	**4** **Capacity:100** **Demand: 100**

Table 28: Searching time per cell after the enforcement of the restriction policy (base test scenario of the fourth tool)

Area (Searching Time)			
13 Searching Time: 0:05:57	14 Searching Time: 0:07:24	15 Searching Time: 0:06:10	16 Searching Time: 0:07:01
9 Searching Time: 0:06:18	10 Searching Time: 0:06:54	11 Searching Time: 0:04:57	12 Searching Time: 0:05:54
5 Searching Time: 0:06:46	**6** **Searching Time: 0:07:12**	**7** **Searching Time: 0:07:31**	**8** **Searching Time: 0:07:39**
1 Searching Time: 0:05:03	**2** **Searching Time: 0:08:30**	**3** **Searching Time: 0:05:04**	**4** **Searching Time: 0:06:16**

Two types of test scenarios are executed in the context of the current tool. The first one is based on the same input data with base scenario with the main difference being the policy effect expansion level. In the first test scenario the policy expansion level is set to 2. Table 29 presents the outputs of the tool in terms of distributing the excess demand into the neighbour areas of Areas 3-4, this time including 2nd level neighbours (highlighted with light blue colour). Similarly, Table 30 presents the updated average searching time induced by all road users after the enforcement of the restriction policy assumed to expand to the 2nd level neighbours.

Table 29: Distribution of the excess demand to the neighbour areas (1st test scenario of the fourth tool)

Area (Capacity - Demand)			
13 Capacity:110 Demand: 95	14 Capacity:120 Demand: 118	15 Capacity:90 Demand: 94	16 Capacity:110 Demand: 111
9 **Capacity:90** **Demand: 95**	**10** **Capacity:110** **Demand: 110**	**11** **Capacity:80** **Demand: 95**	**12** **Capacity:90** **Demand: 95**
5 **Capacity:110** **Demand: 110**	**6** **Capacity:90** **Demand: 90**	**7** **Capacity:120** **Demand: 120**	**8** **Capacity:100** **Demand: 100**
1 **Capacity:70** **Demand: 96**	**2** **Capacity:80** **Demand: 80**	**3** **Capacity:81** **Demand: 81**	**4** **Capacity:100** **Demand: 100**

Table 30: Searching time per cell after the enforcement of the restriction policy (1st test scenario of the fourth tool)

Area (Searching Time)			
13 Searching Time: 0:05:57	14 Searching Time: 0:07:24	15 Searching Time: 0:06:10	16 Searching Time: 0:07:01
9 **Searching Time: 0:06:18**	**10** **Searching Time: 0:06:54**	**11** **Searching Time: 0:07:10**	**12** **Searching Time: 0:06:11**
5 **Searching Time: 0:06:46**	**6** **Searching Time: 0:06:09**	**7** **Searching Time: 0:08:23**	**8** **Searching Time: 0:06:39**
1 **Searching Time: 0:09:07**	**2** **Searching Time: 0:04:55**	**3** **Searching Time: 0:05:04**	**4** **Searching Time: 0:06:16**

The second test scenario uses real-world data from the pilot city of Leuven. These data include parking measurements during the morning rush morning hour of a typical working day as well as parking places across the entire city. This piece of information is transmitted to the attributes of a grid network covering the entire city through spatial join operators of QGIS platform. This grid was initially comprised was initially comprised of 54 rows and 37 columns, while the cell size is set to (250m*250m). After the execution of the tool, only cells intersecting with the statistical sectors of the city are maintained.

Figure 30 depicts the parking demand to parking capacity ratio corresponding to the rush morning hour of a typical workday in Leuven. In Figure 16, these values are updated assuming that a parking restriction policy has been enforced in cell (indicated with a blue circle) located in the City Centre of Leuven. Similarly, in Figure 31 these values are updated assuming that a parking restriction policy has been enforced in three cells (indicated with a blue oval) located in Leuven East[54].

[54] The assumed policies involves the full restriction of parking places.

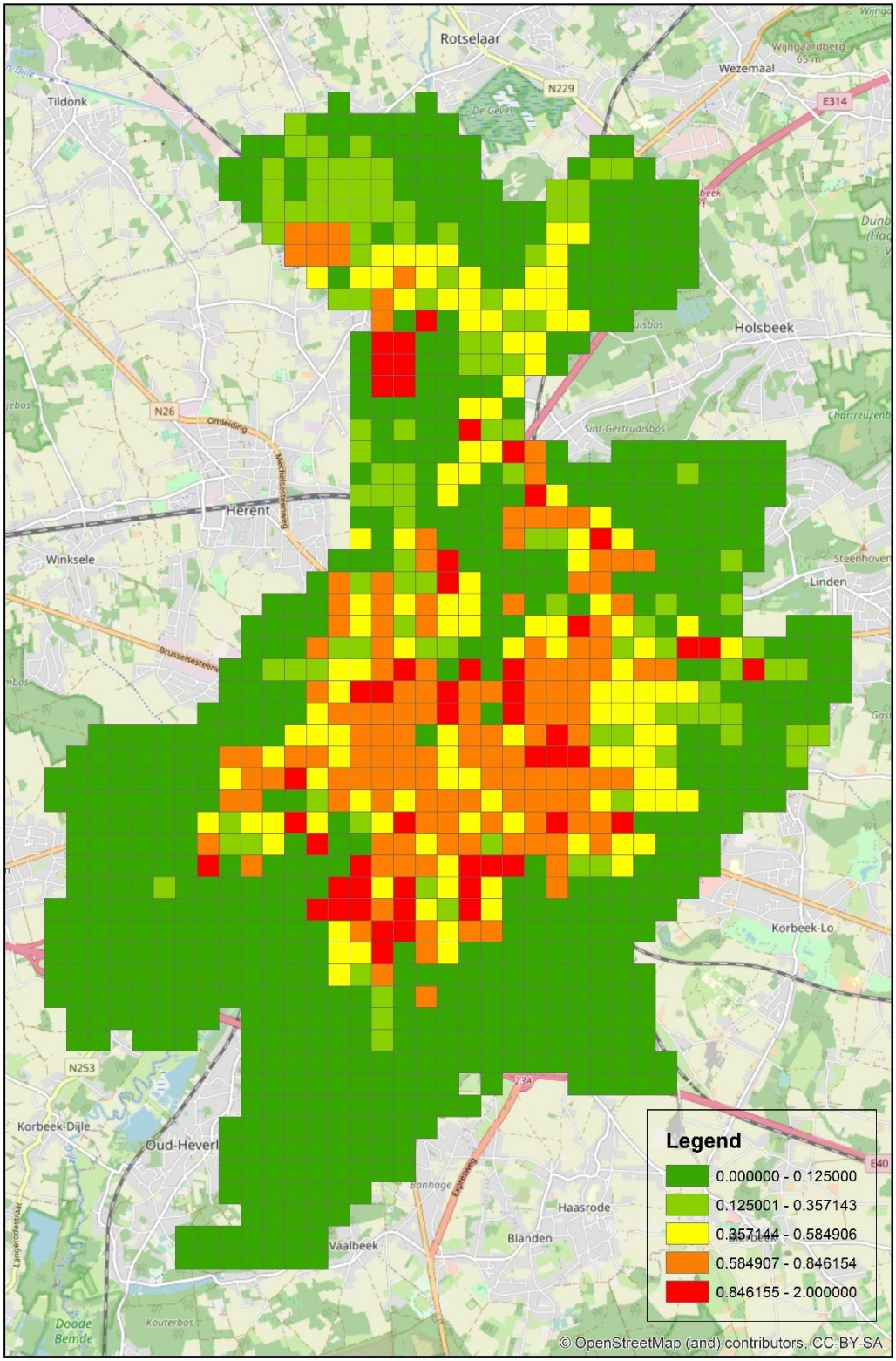


Figure 30: Parking demand to capacity ratio during the morning rush hour in Leuven

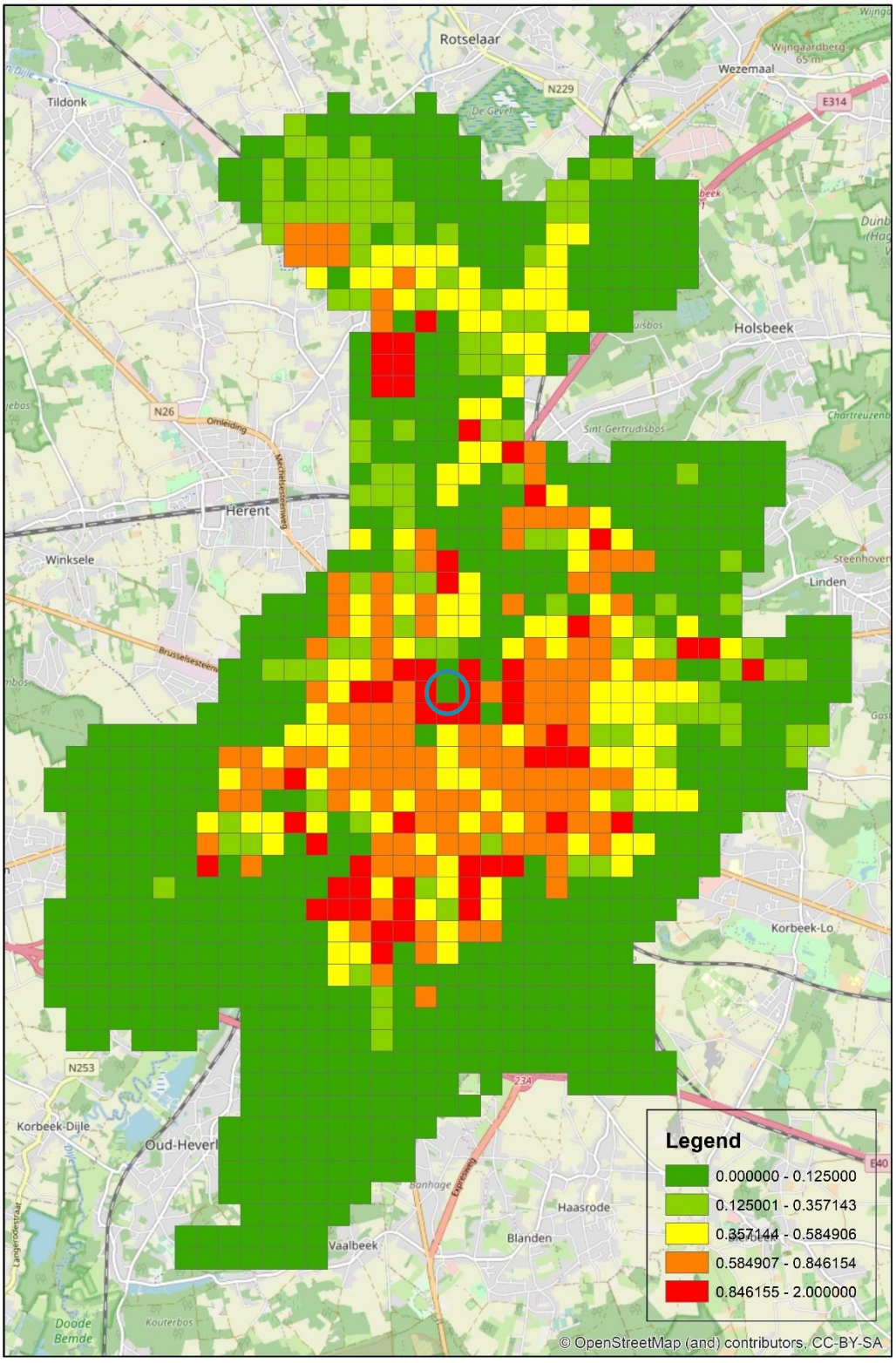


Figure 31: Updated parking demand to capacity ratio after the enforcement of a restriction policy in the city centre of Leuven (2nd test scenario of the fourth tool)

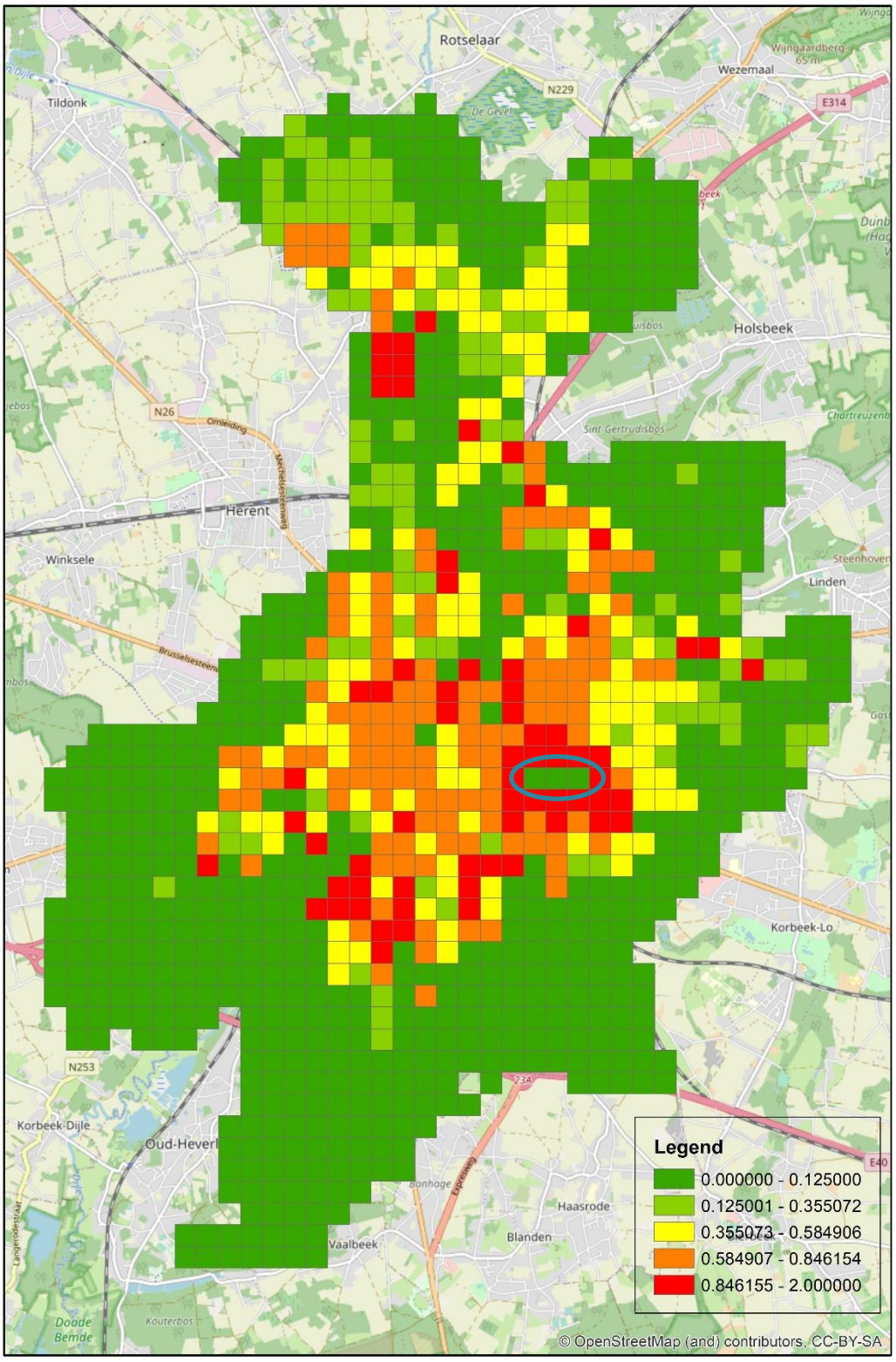


Figure 32: Updated parking demand to capacity ratio after the enforcement of a restriction policy in Leuven East (2nd test scenario of the fourth tool)

4.5 Inflows and outflows in a metropolitan area

4.5.1 Overview

The estimation of the origins and destinations of vehicle trips is of particular use for large metropolitan areas given that they inform transport planning authorities on the extent to which prevailing traffic conditions are a product of internal, inbound, outbound, or through-going vehicle trips. The scope of the fifth tool to be integrated into the nuMIDAS toolkit is the estimation of in- and out-flows of a specific zone of a metropolitan area (e.g., low emission zone), i.e., from its boundaries to the remaining districts. These estimations will be based on data generated by Automated Number Plate Recognition (ANPR) systems coupled with census data (vehicle registration). The main objective of the tool will be to improve mobility planning processes and acquire a better knowledge about users and mobility patterns. A valuable aid towards this direction would be the definition of an Origin-Destination (OD) Metropolitan matrix using as data input the detections of the available ANPR systems. By that means, a city will be able to better understand the effectiveness of policy instruments of Low Emission Zones (and any necessary adjustments) but also to plan public transport services, including, among others, park and ride facilities and services.

4.5.2 Targeted users

The stakeholders involved in the ecosystem of the fifth tool are the following:

- Data providers (including Traffic Management Centres)
- Department of a municipality responsible for the enforcement of mobility policies or for transport planning (policy makers)
- Transport planners supporting policy makers

The entirety of the above actors constitutes the targeted users of the fifth tool. It is assumed that data providers provide input to the tool, while transport planner receive the outputs of the tool (i.e., OD matrices and other data analytics results) to support policy makers to take evidence-based decisions concerning the operation of public transport services and the prioritisation of relevant investments (e.g., development of park & ride facilities).

4.5.3 Data inputs

The inputs provided to the tool can be divided into values imported from a file or database and user defined. The former includes:

- ANPR system detectors' location and any associated descriptive information
- ANPR system detections (including timestamps and detectors identifiers)
- External data (e.g., census data providing information about vehicle registration)

As it is described in Section 4.5.5, the above inputs are used by the tool to calculate the number of detections per each camera pair in a five-minute interval time. For each camera pair, the total number of vehicles is calculated as well as the distribution of these vehicles into three different categories. The three categories based on which the distribution was made are the following: a.) vehicle type, b.) fuel type, and c.) environmental badge. Given that a) the purpose of the fifth tool is to support transport planners and policy makers by providing information about the in- and out-flows of a specific camera pair or a specific

zone of a metropolitan area, and b) the tool requires the ANPR system detectors' location, the detections the later made, and external data in order to distribute the detected vehicles to different categories.

On the other hand, the user defined input includes the following:

- Selection of one or more cameras
- Definition of the start and end date for displaying the results
- Definition of the days of the weeks for displaying the results
- Definition of the start time and end time of the day for displaying the results
- Definition of the aggregation period for the calculated data

4.5.4 Data outputs

The outputs of the tool in its current version are the following:

- OD table (between cameras)
- Graphs over the selected aggregation period for each of the three categories (vehicle type, fuel type and environmental badge)
- Map showing the location of the selected pair of cameras

4.5.5 Computational flow

The computational flow of the fifth tool is depicted in Figure 33. The first step involves the removal of all single detections considering that the desired outcome of the tool is the creation of OD tables. However, it should be noted that even single detections may prove useful if they are combined with external data (including information about the registration location of each vehicle) when the purpose is to identify inflows and outflows within specific zones. Such a purpose is not directly the focus of the current version of the tool and the current version of deliverable thereof, but it will be further analysed in the next one.

The second step involves the distinction of all sequential detection pairs. A parallel step to the second one constitutes the calculation of the free-flow travel times between these pairs. This has been achieved through queries to the OSRM service[55]. The estimated travel times are then converted in a tabular form, i.e., in the form of a skim matrix. Considering that the purpose of this matrix is to be used for understanding whether two sequential detections of the same vehicle correspond to the same trip, its values are scaled up to account for congestion effects. A simplistic, yet empirically evaluated, approach towards the estimation of congestion effects has been provided by Ji et al. (2014). According to this approach a threshold equal to the one-third of the free flow speed of each link is used to distinguish congested and uncongested links within a road network. To this end, the values included in the skim matrix are multiple by a fixed value equal to three.

Having distinguished the detection pairs, the third step involves the calculation of the observed travel times by utilizing the timestamp of each detection. Subsequently, the next step involves the comparison of the observed and estimated travel times with the aim of identifying what is termed in Figure 18 as time feasible trips. As time feasible trips are understood the trips that include paths along detections, the period of which is within the limits defined by the skim matrix. After the identification of the time feasible trips, the

[55] http://project-osrm.org/

next step involves the distinction of the trip start and end location. The entirety of all trips is then aggregated and included in an OD table.

The last part of the computational flow is based on the availability of unique identifiers for each detected vehicle or device included in a detected vehicle. Through these identifiers it is possible to use data included in an external database to create a 3D OD table that will include the number of trips between each detection location or zone and, in its third dimension, vehicle-related information (e.g., number of vehicles by type of vehicle, propulsion technology, and registration location). Finally, having available such a 3D OD table it is possible to provide to the user of the tool data analytics in support of mobility policies and prioritisation of relevant investments.

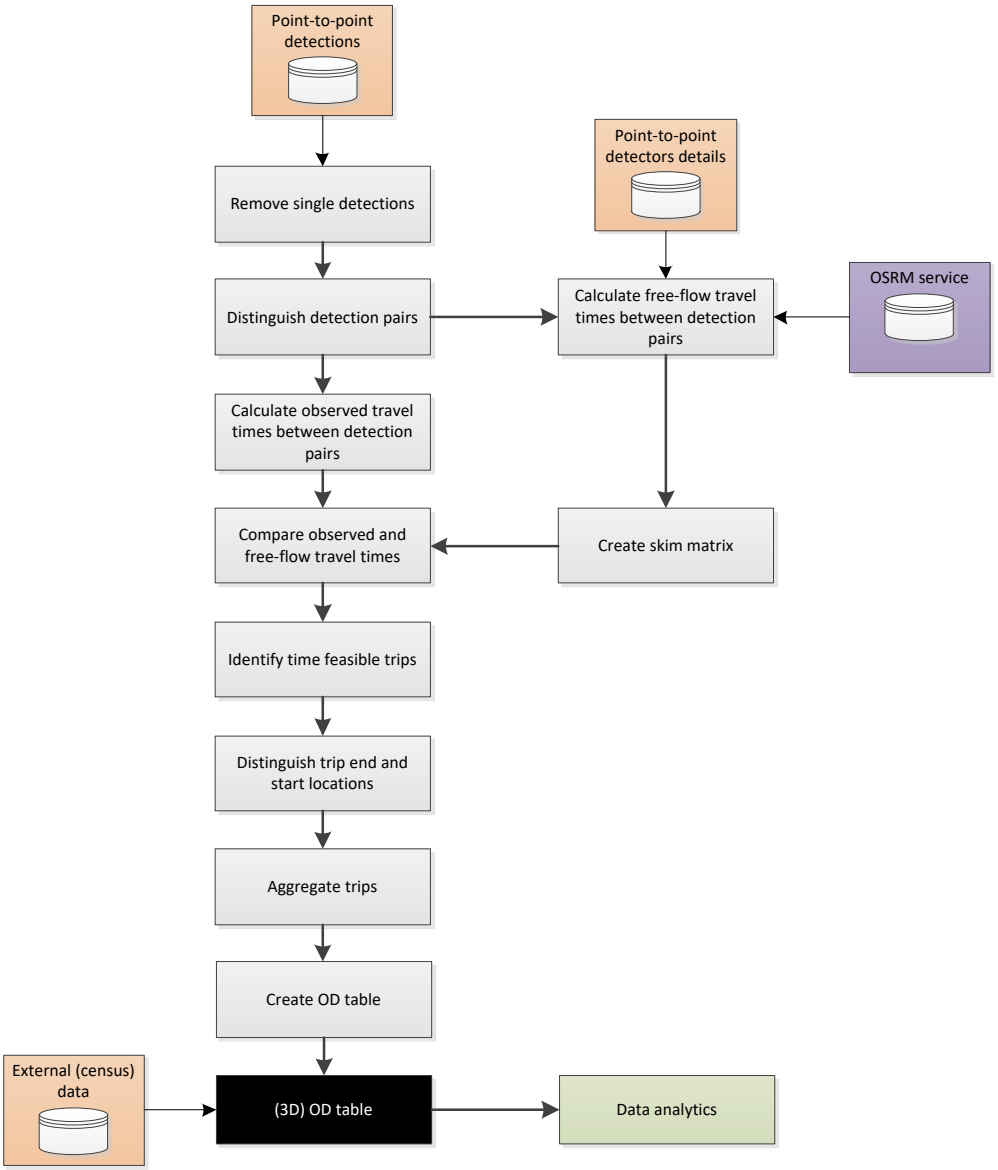


Figure 33: Computational flow of the fifth tool of nuMIDAS toolkit

4.5.6 Additional features

The additional features that have been embedded in the fifth tool of the nuMIDAS toolkit involve two main aspects: a) the creation of an off-line process of calculations, and b) the creation of an on-line process of calculations. More details are provided below.

The development of the off-line process of calculations involves the calculation of valid trips based on detection times among all the available cameras. The main reason for developing the above-mentioned off-line process was to reduce the needed computational time for identifying from all the cameras and their detections the valid trips. The available data were received in multiple files (in csv format) per each week from 28th of March 2021 up to 5th of February 2022. The total number of available csv files were 45 and each file contained about 1 million of vehicle detections. The needed time in order to retrieve all the valid trips for all the available weeks exceeds the time-limit of 15 minutes that the tool has. Based on the pre-process calculations the computed data were stored to respective tables at the database, specifically in PostgreSQL, and then the algorithm developed for the on-line process retrieves the needed data based on the user input.

Moreover, through the on-line process of the developed algorithm, the needed statistics such as the distribution of detected vehicles between two cameras based on vehicle type (e.g., passenger car, truck, bus etc.), based on fuel type (e.g., gasoline, diesel, electric etc.) and based on environmental badge is achieved.

4.5.7 Code and quality control

Individual Edition 2020.02 relying upon Python Release 3.9.12. The dependencies of the code are as follows:

- Json library
- Numpy library
- Pandas library
- Multiprocessing
- Datetime library
- Sqlalchemy library

The fifth tool's code is comprised of the following functions:

- main: is the function which calls the remaining ones and retrieve their outputs to be reported to the users of the tool.
- main_new: is the function which calls the "main" function. The execution of the "main" function sets to a thread and the current function monitors the execution process of the "main" function to not exceed the time-limit of the AWS Lambda function (15 minutes).
- db_connection: is the function through which the results of the simulations as well as from the analysis are stored in the database (PostgreSQL).
- input_parameters: is the function in which the algorithm retrieves all the necessary input data from the database (PostgreSQL) for the selected scenario.
- allsundays: is the function which calculates for a specific year the dates of all Sundays.

- camera_region_combinations: is the function which calculates all the possible combinations between the cameras that have been selected from the user.
- find_dates_list: is the function which finds all the dates between the start data and end data that the user has selected and the filters them by keeping only the selected days of week.
- get_day_of_week: is the function which converts the number of each day to the corresponded day of the week.
- data_extraction_from_sql: is the function which extracts for the selected days of week and dates and for the defined time interval of the day the data from the database (PostgreSQL).
- data_aggregation: is the function which aggregates the data from all the days in to the selected aggregation period of time.
- graph_index_calculation: is the function which calculates all the time intervals between the start time and the end time as have been selected by the user.
- table_formated_data: is the function which formats the calculated – aggregated data in to the final format that is need it in order to be stored at the database (PostgreSQL).

The results of the code quality control, by utilizing the methodology described in Section 3.2, are included in Table 31.

Table 31: Code quality control results – 5th tool (UC5)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	main function	CC index	A (4)	low - simple block
	main_new function	CC index	A (2)	low - simple block
	db_connection function	CC index	A (1)	low - simple block
	input_parameters function	CC index	A (1)	low - simple block
	allsundays function	CC index	A (2)	low - simple block
	camera_region_combinations function	CC index	A (4)	low - simple block
	find_dates_list function	CC index	A (3)	low - simple block
	get_day_of_week function	CC index	B (8)	low - well-structured and stable block

Category	Evaluation element	Metric	Rank (Score)	Interpretation
	data_extraction_from_sql function	CC index	B (8)	low - well-structured and stable block
	data_aggregation function	CC index	B (6)	low - well-structured and stable block
	graph_index_calculation function	CC index	A (5)	low - simple block
	table_formated_data function	CC index	D (28)	more than moderate - more complex block
Maintainability	Entire UC5 .py file	Maintainability index	A (42.82)	Very high maintainability
Raw metrics	Entire UC5 .py file	LOC	560	total # lines of code
	Entire UC5 .py file	LLOC	303	total # logical lines of code[56]
	Entire UC5 .py file	SLOC	319	total # source lines of code[57]
	Entire UC5 .py file	comments	93	total # comment lines
	Entire UC5 .py file	multi	4	total # lines representing multi-line strings

[56] Logical lines of code may be defined as lines of code including executable expressions.
[57] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.

Category	Evaluation element	Metric	Rank (Score)	Interpretation
	Entire UC5 .py file	blank	146	total # blank lines
Hal metrics	Entire UC5 .py file	h_1	12	total # distinct operators[58]
	Entire UC5 .py file	h_2	83	total # distinct operands[59]
	Entire UC5 .py file	N_1	60	total # operators
	Entire UC5 .py file	N_2	116	total # operands
	Entire UC5 .py file	Vocabulary	95	$n = n_1 + n_2$
	Entire UC5 .py file	Length	176	$N = N_1 + N_2$
	Entire UC5 .py file	calculated_length	572.15	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Entire UC5 .py file	Volume	1156.30	$V = N \log_2(n)$
	Entire UC5 .py file	Difficulty	8.39	$D = (n_1/2)x (N_2/n_2)$
	Entire UC5 .py file	Effort	9696.16	E = D x V
	Entire UC5 .py file	Time	538.68	T = E/18 seconds
	Entire UC5 .py file	Bugs	0.39	B = V/3000

The average cyclomatic complexity of the UC5 code file is ranked as B (6.0). Hence, it can be considered as a well-structured and stable piece of code.

[58] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[59] The values that an operator operates.

4.5.8 Demonstration and testing

Demonstration and testing in the context of the current tool is achieved through data stemming from the pilot city of Barcelona and specifically from the ANPR system which is already installed and operated. This section aims to demonstrate the results of the application of the fifth tool and to assess the validity of its results based on various test case scenarios.

The first test scenario and its input parameters for assessing the validity of the fifth tool are included in Table 32.

Table 32: Inputs corresponding to the 1st test scenario of the fifth tool

Input	Value
Start date	4/26/2021
End date	5/26/2021
Day(s) of week	Monday - Tuesday - Wednesday - Thursday - Friday
Time of day (from)	6:00 AM
Time of day (to)	19:00 PM
Aggregation period	15 minutes
Selected cameras	11 - 12 - 13 - 14 - 202

The output table that corresponds to the total number of vehicles for each camera pair combination is shown at the following Table 33.

Table 33: OD table of the 1st test scenario of the fifth tool

Origin/ Destination	11	12	13	14	202
202	34	49	9	3	3
12	9	78	66	3	29
11	66	28	34	0	18
13	12	110	11	1	9
14	3	62	25	0	0

The following figure (Figure 34) depicts the graph over 15 minutes aggregation period regarding the distribution of detected vehicles based on the fuel type for the camera pair 13 (origin/ from camera) – 12 (destination/ to camera).

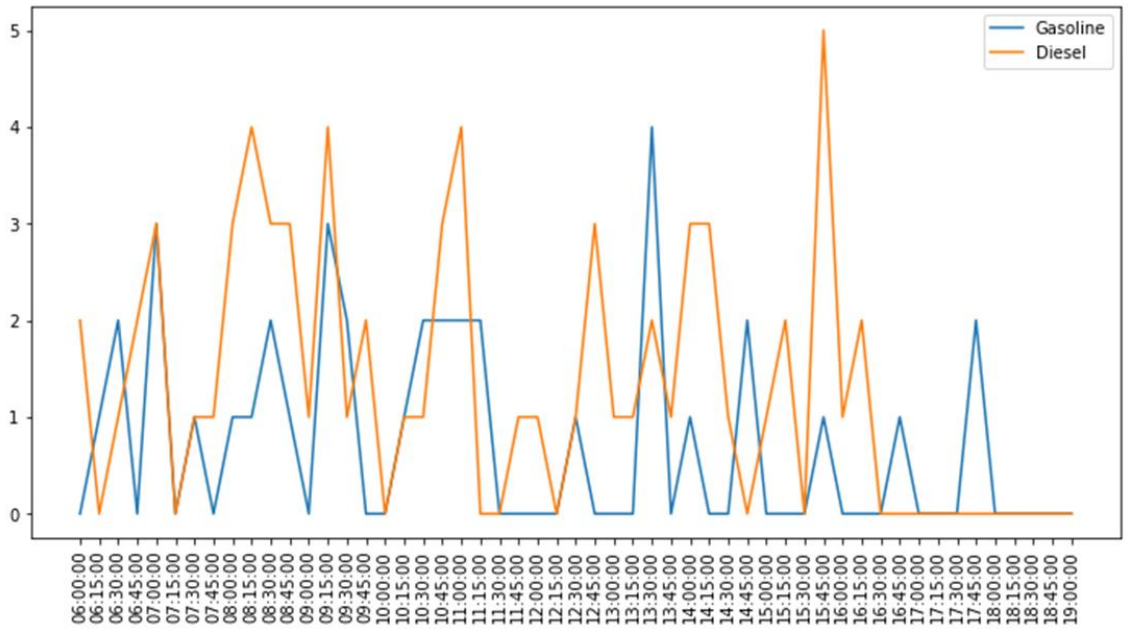


Figure 34: Distribution of detected vehicle for camera pair 13-12 based on fuel type per 15-minute intervals

The second test scenario and its input parameters for assessing the validity of the fifth tool are included in Table 34.

Table 34: Inputs corresponding to the 2nd test scenario of the fifth tool

Input	Value
Start date	4/26/2021
End date	5/26/2021
Day(s) of week	Saturday - Sunday
Time of day (from)	9:00 AM
Time of day (to)	1:00 PM
Aggregation period	1 hour
Selected cameras	11 - 12 - 13 - 14 - 143 - 202 - 203

The output table that corresponds to the total number of vehicles for each camera pair combination is shown at the following Table 35.

Table 35: OD table of the 2nd test scenario of the fifth tool

Origin/ Destination	12	13	202	203	11	143	14
202	5	1	1	1	0	0	0
203	4	0	3	0	34	20	0
11	2	3	2	0	7	31	0
12	0	3	1	0	1	2	1
13	4	1	1	0	0	0	0
14	2	0	0	0	0	0	0
143	8	8	1	0	4	4	0

The following figure (Figure 35) depicts the graph over 60 minutes (1 hour) aggregation period regarding the distribution of detected vehicles based on the fuel type for the camera pair 203 (origin/ from camera) – 11 (destination/ to camera).

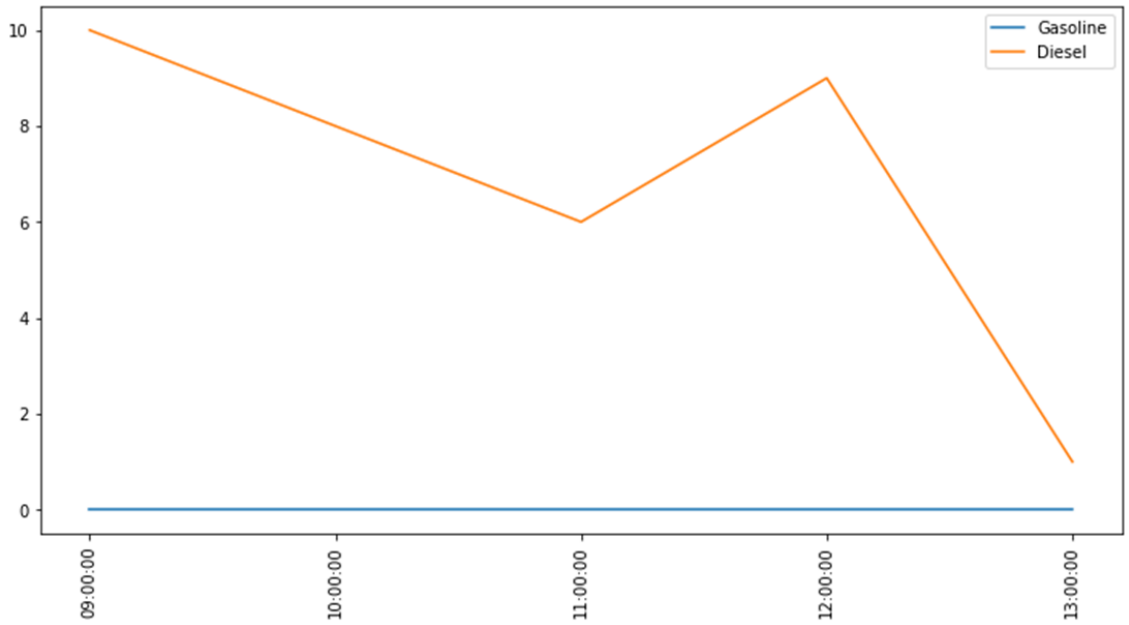


Figure 35: Distribution of detected vehicle for camera pair 203-11 based on fuel type per 1-hour intervals

The third test scenario and its input parameters for assessing the validity of the fifth tool are included in Table 36.

Table 36: Inputs corresponding to the 3rd test scenario of the fifth tool

Input	Value
Start date	4/26/2021
End date	5/26/2021
Day(s) of week	Monday - Sunday
Time of day (from)	8:00 AM
Time of day (to)	10:00 AM
Aggregation period	5 minutes
Selected cameras	11 - 12 - 13 - 14 - 143 - 202 - 203

The output table that corresponds to the total number of vehicles for each camera pair combination is shown at the following Table 37.

Table 37: OD table of the 3rd test scenario of the fifth tool

Origin/ Destination	11	12	13	202	203	205	206	14
202	10	13	2	1	2	5	19	0
11	17	3	9	4	0	3	2	0
13	3	34	0	1	0	0	0	0
12	1	13	10	8	0	1	0	0
14	0	1	1	0	0	0	0	0
203	115	8	1	0	1	0	0	0
206	3	6	1	0	0	3	0	0
205	70	149	28	3	39	1	5	1

The following figure (Figure 36) depicts the graph over 5 minutes aggregation period regarding the distribution of detected vehicles based on the fuel type for the camera pair 205 (origin/ from camera) – 12 (destination/ to camera).

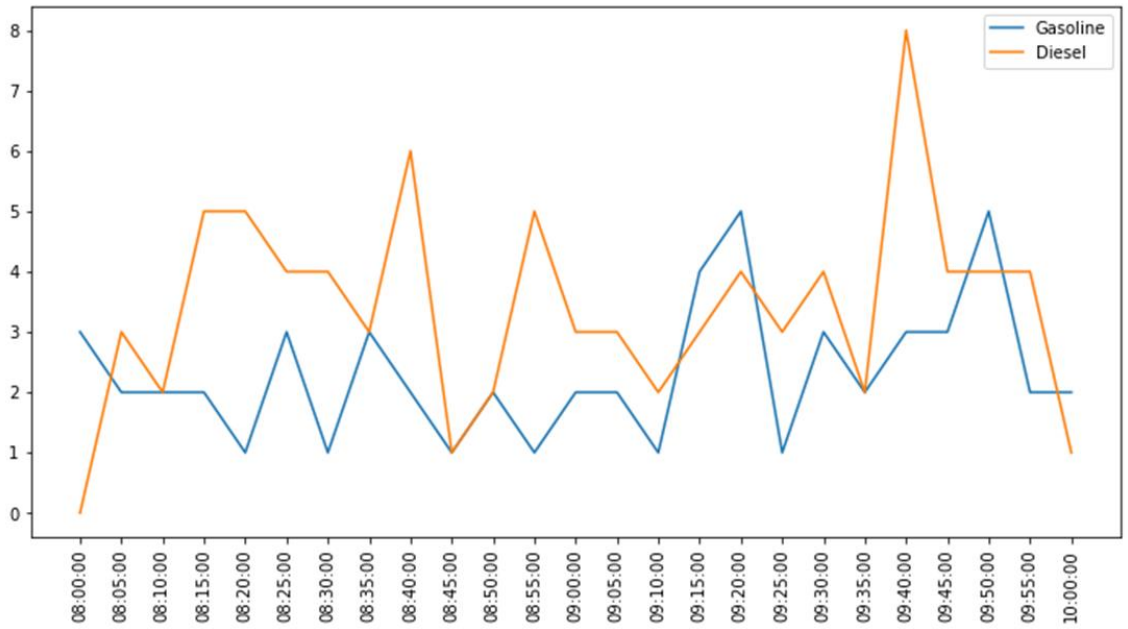


Figure 36: Distribution of detected vehicle for camera pair 205-12 based on fuel type per 5-minute intervals

The above results suggest that the tool developed in response to the requirements set out by the fifth use case produce rational results.

4.6 Assessment of traffic management scenarios

4.6.1 Overview

Last decades, the rising traffic congestion is a common situation in all large cities induced by the large number of vehicles circulating in the roads (Roelofsen et. al., 2012). Hence, vehicle emissions are provoked in a high rate and therefore the ambient air quality is even more degraded. Apart from the adverse effect on the environment, there are also other traffic-related impacts, such as the increase of travel delays and vehicular queueing. The most direct approach to reduce traffic congestion is the implementation of traffic management measures dedicated or bundled in the context of traffic management scenarios/strategies that are enabled by intelligent transport system applications, targeting the improvement of road conditions. In that sense, the development and application of traffic management measures typically relies on the knowledge/visibility (ideally in a real-time context) of a transport network, paying increased attention to prevailing traffic conditions. The data that are mainly digested for this purpose by traffic managers and planners[60] can be classified from various perspectives, mainly including their (PIARC, 2019) (a) collection source (b) provision timeline, (c) thematic area, and (d) coverage.

From a "collection source" perspective digested data can be discerned into (a) stemming from infrastructure-based detectors (e.g., inductive loops, magnetic sensors, video image processors), (b) in-vehicle or vehicle-based (e.g., Floating Car Data, dash cams, UAVs), (c) user-based (e.g., derived from installed applications), and (d) other (e.g., meteorological sensors, air pollution sensors, social networks, geospatial repositories). The "provision timeline" practically divides data into four fundamental types, namely static, real-time, dynamic, and historical. Static data reflect data types that do not change or change in very long intervals. Typical examples, in the traffic management domain, constitute road network attributes (e.g., number of lanes, direction of road segments, speed limits), traffic restrictions/regulations, road signage information, and traffic signal control information. Real-time data reflect data types that are constantly changing and concentrated into dedicated information systems. They are practically large-sized and enclose information of significant variability that is difficult to analyze without involving real-time processing techniques. Assuming that such techniques are involved real-time data can provide useful information about the *status* of a road network in a real-time basis (e.g., events, road closures, etc.). Dynamic data, on the other hand, reflect real-time that are *aggregated* at appropriate time intervals (depending on the use case), providing useful information about the *state* of a road network (e.g., travel times, delays, level of congestion, etc.). Historical data, finally, reflect filtered, aggregated, and stored dynamic data that are useful for planning purposes or checking the validity of deployed traffic management measures. On the other hand, digested data based on "thematic area" perspective are distinguished in several application areas. This is also the case when it comes to the ITS Directive and its supplementary Delegated Regulations defining four main thematic areas, namely the exchange of data for (a) safe and secure truck parking areas along motorways, (b) the provision of safety-critical traffic information, (c) the provision of real-time traffic information, and (d) the provision of multimodal travel information services. As it is can be deduced, the previously discussed perspective reveals two different goals for traffic management, with the first one corresponding to the increase of road safety and the second corresponding to the increase of traffic efficiency. Finally, digested data based on the "coverage" perspective can be

[60] Understood as transport planning and traffic flow analysis experts that are engaged in the formulation of traffic management scenarios.

divided into various categories ranging from local coverage to urban-wide, region-wide, national, and cross-border coverage.

Moreover, there are several approaches of varying sophistication for implementing traffic management (Klein, 2018). The first and less complex approach is the *static* management of traffic according to which specific traffic management plans are drafted for each day type and time of day. Such an approach may provide satisfactory results only on the premise of limited variability of critical demand- and supply-side parameters. The second approach is the *responsive* management of traffic according to which specific strategies or measures are applied as a means of addressing observed traffic conditions. The third and most complex approach is the *initiative-taking* management of traffic based on which the applied traffic management strategies or measures are called to respond to predicted demand- and supply-side changes or even delay and eliminate breakdowns. Both the last two approaches can be classified as dynamic traffic management approaches with their main difference being that in the former applied strategies or measures may have been evaluated in several circumstances, while in the latter the strategies and measures are by nature more experimental.

Traffic management measures can be conceptualised and developed following a narrow approach, such as the resolution of a specific (observed) situation with any available means, or following a more structured approach, such as the "Regelaanpak," i.e., the Dutch dynamic traffic management scheme (Kotsi & Mitsakis, 2020). According to this scheme, traffic management measures should stick to a traffic control strategy perspective that aims to mitigate congestion and secure policy objectives by optimizing traffic flow. A traffic control strategy practically consists of several sub-strategies ranging from less severe to radical that are typically applied in a successive manner:

- Inform traffic
- Enlarge outflow
- Reduce inflow
- Reroute traffic

The means of applying these sub-strategies are mainly based on I2V (infrastructure-to-vehicle) communication technologies, involving several Day 1 and Day 1.5 C-ITS services (e.g., warnings, in-vehicle signage, smart routing, green light optimal speed advisory) or even conventional traffic management measures addressed from a dynamic perspective (e.g., dynamic traffic signal control). By that means, it is possible for traffic management authorities to fulfil their policy objectives by activating or deactivating a set services and measures in a coordinated manner. A critical need for achieving such a goal is the availability of an appropriate set of trigger and target variables and (threshold) values. Trigger variables and values support the definition of what constitutes a traffic-related problem and lead to the activation of a set of treatment services and measures. Target variables and values, on the other hand, support the definition of what constitutes a satisfactory state and lead to the deactivation of the previously mentioned set. In short, a critical need for making this traffic management scheme happens constitutes the quantitative assessment of any set of traffic management measures and services.

The above analysis indicates that the chain of developing a traffic management scenario or control strategy includes various step ranging from the collection of required data to its assessment (or even execution of predictions if proactive management is the case). The assessment of a traffic management scenario or control strategy shall be implemented by following a dual-stage approach. The first stage corresponds to its

ex-ante assessment (before its application) to evaluate its suitability to treat a specific (observed) unwanted situation. The second stage corresponds to its ex-post assessment (after its application) to practically evaluate its success rate. There are two crucial questions that should be answered at this point. The first involves to what measures and KPIs such an assessment should be based, while the second involves data availability.

Regarding the second question, data sources that can be used in the assessment of traffic management scenarios are discussed in the beginning of the current section. In case of limited data availability, a typical approach followed by traffic planners constitutes the generation of synthetic data through traffic simulation experiments. There are several commercial or open-source tools that can be used for this purpose, such as AIMSUN, VISSIM, SUMO, and MATSIM. A brief overview and comparison of these tools can be found in the studies of Saidallah et al. (2016), Boxill and Yu (2000), Pell et al. (2013), and Alghamdi et al. (2022), while a more detailed discussion of their features, functionalities, strengths, and weaknesses can be found in the book published by Barceló (2010). An undoubted need in such cases is the proper calibration of traffic simulation models utilizing any available data and traffic observations as well as their proper configuration (e.g., proper configuration of simulation duration, parametrisation of underlying car-following and lane-changing models). A wonderful set of guidelines for such a purpose is provided by Antoniou et al. (2014).

With respect to the first question, the measures and KPIs that can be used in the assessment of traffic management scenarios may encompass several concepts that govern the functionality/ level of service of a transportation network and especially a road network, such as mobility, reliability, operational efficiency, and environmental performance. As already mentioned in Deliverable 3.1 of nuMIDAS (Mylonas et al., 2021), the CONDUITS project has elaborated a performance measurement framework to evaluate the success rate of Intelligent Transportation Systems (ITS) that may addressed as a means of applying in an operational manner a traffic management scenario. Example measures included in this framework constitute average travel time, delays, number of stops, value of fuel savings, etc. A more pragmatic approach for selecting a set of performance indicators to be used constitutes the outputs of a traffic simulation tool. The Simulation of Urban MObility (SUMO) tool discerns the outputs of a simulation experiment into several categories, including (DLR, 2022):

- Outputs of simulated detectors
- Edge-specific or lane-specific outputs
- Vehicle-based outputs (aggregated or disaggregated)
- Junction-specific outputs
- Network-wide outputs (summary statistics)
- Outputs related to the operation of traffic lights

Specific examples of measures falling into the first three categories are presented in Table 38.

Table 38: Types of outputs and their respective measures

Type of output file	Measure
Detector outputs	Flow rate
	Speed
	Occupancy
Edge data (emissions)	CO emissions (aggregated per edge)
	CO_2 emissions (aggregated per edge)
	Travel time (aggregated per edge)
Edge data (mean data)	Travel time losses (aggregated per edge)
Queue output	Queueing time (aggregated per lane)
	Queueing time (aggregated per lane)
Trip info output	Trip duration (per vehicle trip)
	Waiting time (per vehicle trip)
	Travel time losses (per vehicle trip)
	Number of times the speed of a vehicle was lower than 0,1 m/s (per vehicle trip)
	Total time a vehicle was taking a planned stop (per vehicle trip)

The insightfulness of the above measures into the performance of a traffic management scenario can be furthered enhanced through their proper post-processing. A straightforward approach for such a purpose constitutes their (further) aggregation at a desired level and the application of descriptive or inferential statistics (e.g., frequency distributions, central tendency measures, variability or dispersion measures, hypothesis testing). Figure 37 presents a histogram depicting the distribution/frequency of queueing lengths (value ranges) for two different traffic management scenarios/configurations of an urban arterial.

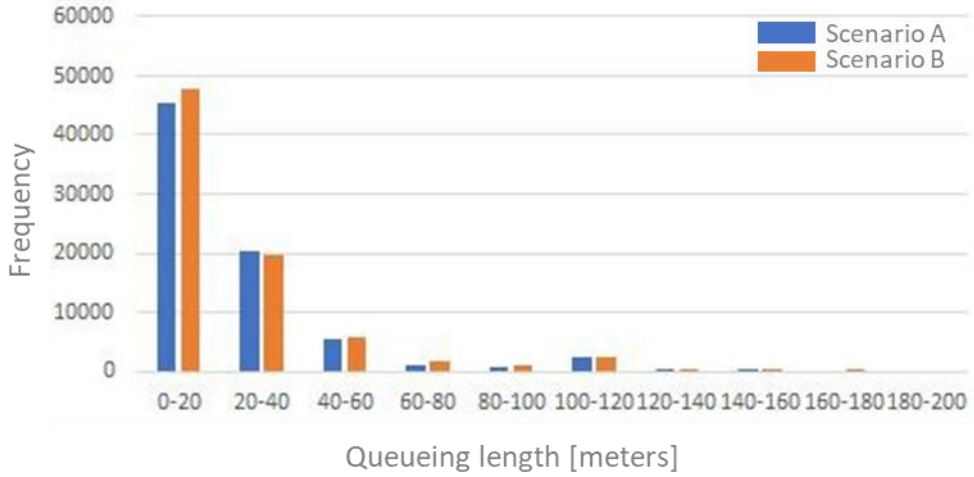


Figure 37: Histogram of queueing lengths

As it can be observed in Figure 28, Scenario A exhibits a lower frequency of queues spanning over 0 to 20 meters and 40 to 100 meters, while Scenario B exhibits a lower frequency of queues spanning over 20 to 40 meters. Moreover, the above-mentioned scenarios exhibit an almost equal frequency of queues spanning over 100 to 120 meters. Therefore, it is up to the planner's judgement to select which traffic management scenario is preferable considering the relative size of the bins of this histogram, the peculiarities of each urban arterial, and other performance measures as well in the context of a multiparametric/multi-criteria analysis. No matter which approach is adopted for selecting a suitable traffic management scenario, the provision of relevant statistics provides significant assistance to traffic managers and planners.

Another approach for judging the performance of a traffic network from a macroscopic – network-wide – perspective constitutes the so-called macroscopic fundamental diagram of traffic (MFD). The theoretical foundation of MFDs relies on the fact that traffic is characterised by three key variables: flow rate (vehicles/time), speed, and density or occupancy (number of vehicles per unit length or percentage of time that a point location is occupied). The relationships between these fundamental variables have been initially analysed in an empirical manner along continuous traffic streams (motorway sections) by Greenshields et al. (1935) and Greenberg (1959) and have been termed as Fundamental Diagrams (see Figure 29).

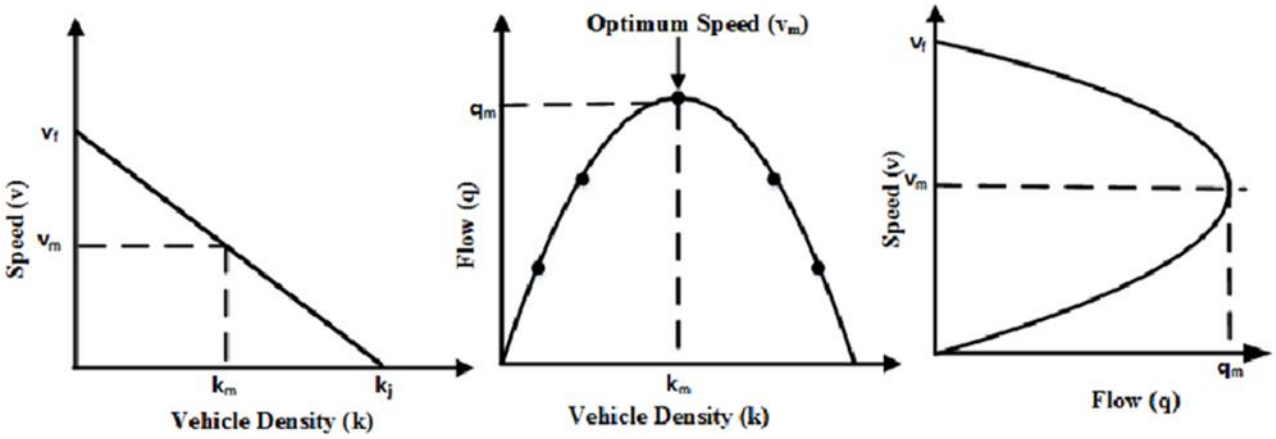


Figure 38: Fundamental diagrams

Later analyses (Thomson, 1967; Godfrey, 1969; Zahavi, 1972; Williams et al., 1987; Mahmassani et al., 1987) have highlighted that similar relationships exist in the context of interrupted traffic conditions (e.g., in case of traffic signal-controlled intersections). In addition, when average flow, speed, and density are analysed at the network level, the derived relationships are termed as *Macroscopic* Fundamental Diagrams. Geroliminis and Daganzo (2007) and Daganzo and Geroliminis (2008), building upon recordings of loop detectors placed at the Yokohama city of Japan supplemented by data derived from GPS devices installed into a taxi fleet, have made some breakthrough observations. In short, they have proved that a) uniformly congested areas (partitions) of a road network present a well-defined MFD, i.e., a relationship of predictable shape between production and accumulation of vehicles, b) the completion rate of vehicle trips is a function of vehicle production, c) MFDs exist regardless of travel/traffic demand (and thus they are a property of road infrastructure and not travel/traffic demand), and d) they can be used for defining and optimizing perimeter traffic control schemes. Moreover, and much importantly, they have showed that MFDs are reproducible within uniformly congested areas (partitions) no matter whether an FD exist along individual arterials.

Considering that a) in the context of (Macroscopic) Fundamental Diagrams the flow rate (q_m) corresponding to jam density (k_m) indicates the *capacity* of an arterial or the *average capacity* of a network partition and and b) the observations made by Geroliminis and Daganzo (2007) and Daganzo and Geroliminis (2008), we have a powerful and demand-agnostic tool/KPI for assessing a traffic control scheme or traffic management scenario at a macroscopic scale bounded by the borders of a network partition exhibiting homogeneous traffic conditions. In short, we expected that different traffic management scenarios will conclude to a better or worse arterial or average network capacity. The only assumption that should be made at that point is that a different control scheme or traffic management scenario does not considerably affect the homogeneity of traffic conditions. Such an assumption appears rational considering that a) the behavior of road users is not rapidly adapted to an applied control scheme or traffic management scenario and b) a different functional network configuration introduced by a new traffic control scheme or management scenario is not typically expected to rapidly affect other substantial factors regulating congestion generation and propagation (e.g., land uses).

The focus of the sixth tool of nuMIDAS toolkit aims to prove the concept of a framework for assessing traffic management scenarios making use of data generated by traffic simulators. In so doing, it provides outputs based on (Macroscopic) Fundamental Diagrams and other KPIs used in the context of assessing ITS

(incl. C-ITS) services and applications. Our main purpose in that context is not on defining comprehensive traffic management scenarios, but on providing a mechanism for assessing a set of simpler measures that can later be further parametrised and used for assessing more complex/comprehensive ones. In addition, in the following sections we present the high-level process for configuring a traffic simulation experiment using SUMO, although the toolkit itself in its current version is limited on digesting generated data. Finally, given that MFD-based outputs are obtainable by both digesting synthetic and real-world data, traffic simulation can be completely disengaged from our framework (if needed).

4.6.2 Targeted users

The stakeholders involved in the ecosystem of the sixth tool are the following:

- Data providers (incl. ITS service providers)
- Traffic managers and Traffic Management Centres
- Local government departments responsible for enforcing traffic restriction/control policies (policy makers)
- Transport planners (in that case traffic planners) supporting policy makers

The entirety of the above-mentioned stakeholders is targeted by the sixth tool. Data providers, although not in the focus of the current version, may find a suitable mechanism for increasing the added value of the data provided them. Traffic managers will be provided with the opportunity of further assessing, validating, or even adjusting traffic management scenarios. Traffic planners will have a valuable tool in their hands for implementing their main duty, i.e., definition of effective traffic management scenarios, while policy makers will have the opportunity to be informed in a comprehensive manner (including visualisations) of a traffic management to be enforced.

4.6.3 Data inputs

The inputs to be provided to the tool through a database include the following:

- Geometry of the road network to be analysed
- Identifiers, names, and other attributes of included arterials/corridors and edges
- Parameters of the baseline configuration of the network to be analysed (in the current version limited to speed limit and applied traffic signal control program)
- Detector output data:
    - Flow measurements
    - Occupancy measurements
    - Speed measurements
    - Corresponding arterial/corridor identifiers
    - Corresponding demand scaling factor
    - Corresponding simulation timestep
    - Corresponding speed limit
    - Corresponding traffic signal control program
- Edge output data:
    - CO emissions (aggregated per edge)
    - CO_2 emissions (aggregated per edge)
    - Time losses (aggregated per edge)

    - Queueing time (aggregated per edge)
    - Queueing length (aggregated per edge)
    - Corresponding edge identifiers
    - Corresponding corridor/arterial identifiers
    - Corresponding demand profile
    - Corresponding speed limit
    - Corresponding traffic signal control program
- Trip info output data:
    - Trip duration
    - Waiting time (per trip)
    - Time losses (per trip)
    - Number of stops (per trip)
    - Corresponding edge identifiers
    - Corresponding corridor/arterial identifiers
    - Corresponding demand profile
    - Corresponding speed limit
    - Corresponding traffic signal control program

4.6.4 Data outputs

The outputs of the tool (indicatively) include:

- Flow-density correlation along an arterial/corridor
- Flow-speed correlation along an arterial/corridor
- Speed-density correlation along an arterial/corridor
- Flow-density correlation along a partition or group of arterials/corridors
- Flow-speed correlation along a partition or group of arterials/corridors
- Speed-density correlation along a partition or group of arterials/corridors
- Histograms of edge and trip-based outputs
- Descriptive statistics (mean value and standard deviation) of edge and trip-based outputs
- Map visualisation of edge-based outputs

4.6.5 Computational flow

The computational flow for generating synthetic data through SUMO, hereafter referred to as off-line process, is discerned into two different pipelines. The first relates to the generation of synthetic data for the MFD-based analysis. The second relates to the generation of synthetic data for the analysis of edge- or trip-based outputs. Figure 39 provides a summary of the process followed for executing the required traffic simulation experiments and generating the required synthetic data.

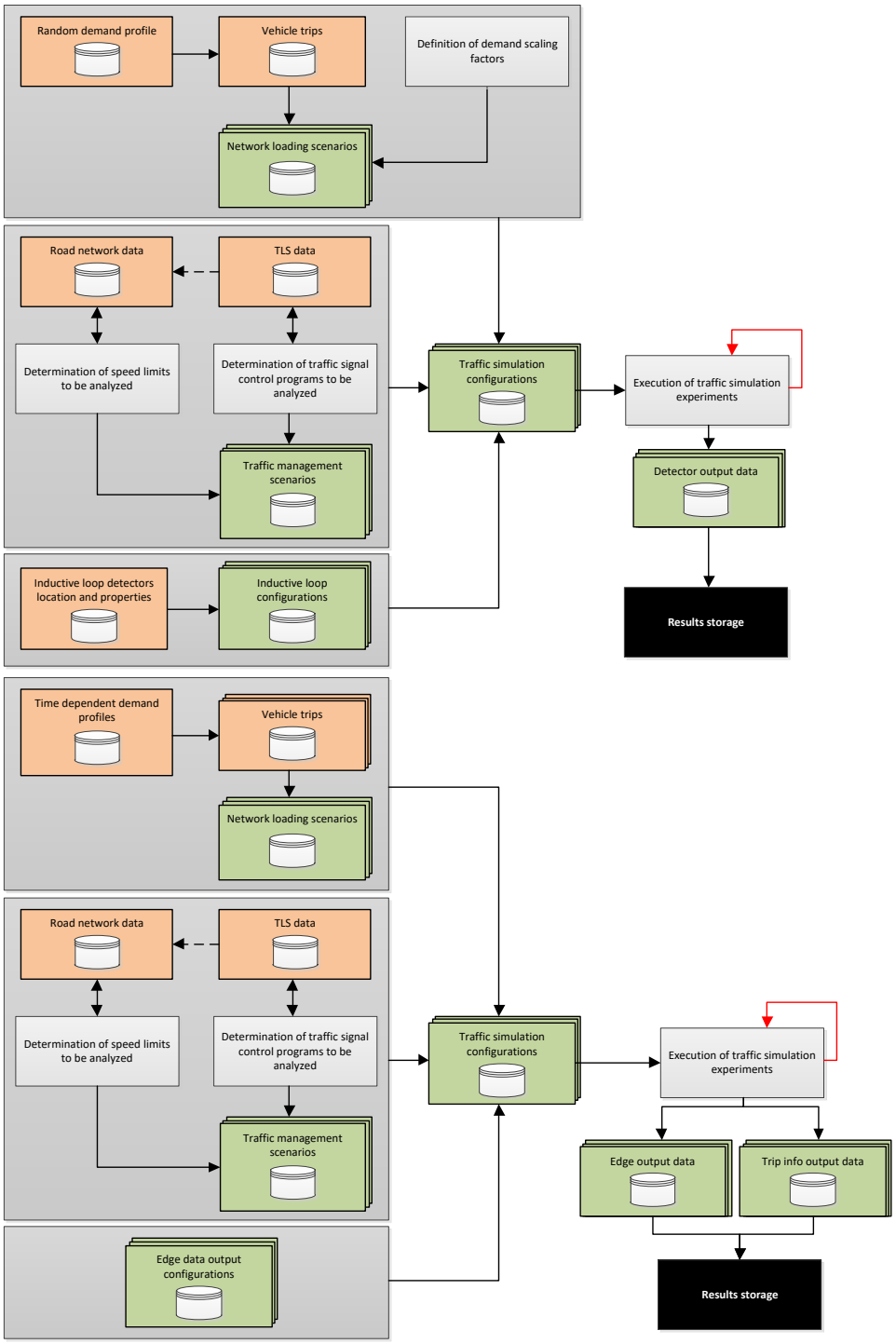


Figure 39: Computational flow of the sixth tool of nuMIDAS toolkit

The followed process within the first pipeline involves the random selection of a route file (that may correspond for instance to a specific demand profile, such as weekday morning peak). Subsequently, a series of traffic simulation experiment is executed by scaling up and down the total number of trips from 10% to 200%. This technique implies the usage of synthetic travel/traffic demand data that are stepwise reduced and increased to ensure that all possible traffic conditions are adequately captured (the whole

range from uncongested to congested state)[61]. The only type of outputs that is requested in the configuration of these experiments constitutes detector outputs. The configuration file of the detectors installed along the simulated corridors is setup to provide measurements every 60 seconds. Moreover, the series of executed experiments involves one experiment for each speed limit and available traffic signal control program the combination of which compose the applied traffic management measure(s). Finally, the outputs of each simulation experiment are stored in a PostgreSQL database.

The followed process within the second pipeline involves the execution of a series of traffic simulation experiments for each available demand profile (i.e., workday morning peak, workday off-peak, workday morning peak, weekend peak, and weekend off-peak) as well as for each speed limit and available traffic signal control program. This is done since specific edge- or trip-based outputs, say time losses, heavily depend on the simulated demand profile. Consistently, a traffic management measure may prove more effective during peak hours and less effective during off-peak hours. The types of outputs that are requested in the configuration of the executed experiments constitute edge output data and trip info output data. Similarly with the first pipeline, the final step involves the storage of the derived outputs in a PostgreSQL database.

The computational flow for extracting Fundamental or Macroscopic Fundamental Diagrams (depending on whether a single or a group of corridors is selected by the user) as well as edge- or trip-based statistics is quite straightforward.

As regards the MFD-based analysis, the computational flow includes the retrieval of the detector outputs corresponding to the baseline and user-defined scenario, their spatial aggregation, and their correlation in the form of a scatter plot. The user can classify the dots in the derived plot based on either the corresponding demand scaling factor or the corresponding simulation time step. A classification based on the demand scaling factor provides insight into whether a specific scatter group is associated with uncongested (free flow), bound flow, or congested traffic states. On the other hand, a classification based on the simulation time step provides the possibility to the users to assess whether the simulated corridor or group of corridors suffers from traffic hysteresis.

Finally, as regards the edge- or trip-based statistics, the computational flow includes the retrieval of edge- or trip-based outputs, the calculation of descriptive statistics (arithmetic mean and standard deviation), as well as the construction of respective histograms, such as the one included in Figure 37. Each bin in the derived histograms includes two bars. The first corresponds to the baseline scenario, while the second to the user-defined scenario.

4.6.6 Additional features

No additional features have been incorporated in the fourth tool of the nuMIDAS toolkit as all the requirements settled into D2.2 have taken into account through the first version of the tool.

[61] We recall at this point that MFD are reproducible regardless of travel/traffic demand and that they constitute a property of road infrastructure. However, the simulation of every possible network/arterial state is required to acquire a complete curve.

4.6.7 Code and quality control

The computational flow described in the previous section is functionally prototyped using Anaconda Individual Edition 2022 relying upon Python Release 3.9.12. The dependencies of the code are as follows:

- Sqlalchemy library
- Shutil library
- Numpy library
- Pandas library
- Matplotlib.pyplot library
- SUMO library

The sixth tool's code for calculation of all the needed data (code for pre-calculation) through the simulation process is comprised of the following functions:

- main: is the function which calls the remaining ones and retrieve their outputs to be reported to the database of the tool.
- db_connection: is the function through which the results of the simulations as well as from the analysis are stored in the database (PostgreSQL).
- simulation_process: is the function which takes as input all the corridors for which calibrated networks are available and runs all the simulations through SUMO software for all traffic management scenarios.
- calculate_format_output_data: is the function which takes as input all the output files from the simulation of each corridor and calculates both the Fundamental Diagram parameters as well as the Macroscopic Fundamental Diagram parameters and stores the calculated data in a table format in the PostgreSQL database.
- statistica_calculation_process: is the function which takes as input all the output files from the simulation of each corridor and calculates the statistics of each corridor such as: CO, CO_2, travel time, time loss, queueing time and queueing length, waiting time, trip duration, and stores the calculated data in a table format in the PostgreSQL database.
- exact_statistics_calculation_process_edge_emissions: is the function which takes as input the output files of the corridors related to edge emissions data and calculates the respective statistics.
- exact_statistics_calculation_process_edge_mean_data: is the function which takes as input the output files of the corridors related to edge mean data and calculates the respective statistics.
- exact_statistics_calculation_process_queue: is the function which takes as input the output files of the corridors related to queue data and calculates the respective statistics.
- exact_statistics_calculation_process_tripinfo: is the function which takes as input the output files of the corridors related to trip data and calculates the respective statistics.

The results of the code quality control, by utilizing the methodology described in Section 3.2, are included in Table 39.

Table 39: Code quality control results – 6th tool (UC6)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	Main function	CC index	A (1)	low - simple block
	db_connection function	CC index	A (1)	low - simple block
	simulation_process function	CC index	E (38)	high - complex block
	calculate_format_output_data function	CC index	D (22)	more than moderate - more complex block
	statistica_calculation_process function	CC index	C (18)	moderate - slightly complex block
	exact_statistics_calculation_process_edge_emissions function	CC index	B (10)	low - well-structured and stable block
	exact_statistics_calculation_process_edge_mean_data function	CC index	A (4)	low - simple block
	exact_statistics_calculation_process_queue function	CC index	C (12)	moderate - slightly complex block
	exact_statistics_calculation_process_tripinfo function	CC index	B (10)	low - well-structured and stable block
Maintainability	Entire UC6 .py file	Maintainability index	A (30.17)	Very high maintainability
Raw metrics	Entire UC6 .py file	LOC	781	total # lines of code

Category	Evaluation element	Metric	Rank (Score)	Interpretation
	Entire UC6 .py file	LLOC	447	total # logical lines of code[62]
	Entire UC6 .py file	SLOC	444	total # source lines of code[63]
	Entire UC6 .py file	comments	149	total # comment lines
	Entire UC6 .py file	multi	0	total # lines representing multi-line strings
	Entire UC6 .py file	blank	188	total # blank lines
Hal metrics	Entire UC6 .py file	h_1	12	total # distinct operators[64]
	Entire UC6 .py file	h_2	179	total # distinct operands[65]
	Entire UC6 .py file	N_1	151	total # operators
	Entire UC6 .py file	N_2	301	total # operands
	Entire UC6 .py file	Vocabulary	191	$n = n_1 + n_2$
	Entire UC6 .py file	Length	452	$N = N_1 + N_2$
	Entire UC6 .py file	calculated_length	1382.62	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Entire UC6 .py file	Volume	3424.9	$V = N \log_2(n)$

[62] Logical lines of code may be defined as lines of code including executable expressions.
[63] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.
[64] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[65] The values that an operator operates.

Category	Evaluation element	Metric	Rank (Score)	Interpretation
			9	
	Entire UC6 .py file	Difficulty	10.09	$D = (n_1/2)x (N_2/n_2)$
	Entire UC6 .py file	Effort	34556.12	E = D x V
	Entire UC6 .py file	Time	1919.78	T = E/18 seconds
	Entire UC6 .py file	Bugs	1.14	B = V/3000

The average cyclomatic complexity of the UC6 pre-calculation code file is ranked as C (12.89). Hence, it can be considered as a moderated-structured and stable piece of code with slight complex blocks.

The developed code regarding the on-line process of the sixth tool, as well as the results of the code quality control, by utilizing the methodology described in Section 3.2, will be presented in the final version of this deliverable (Deliverable 3.4).

The sixth tool's code for implementing the on-line process in order to provide the simulation results to the user of the tool is comprised of the following functions:

- main: is the function which calls the remaining ones and retrieve their outputs to be reported to the users of the tool.
- main_new: is the function which calls the "main" function. The execution of the "main" function sets to a thread and the current function monitors the execution process of the "main" function to not exceed the time-limit of the AWS Lambda function (15 minutes).
- db_connection: is the function through which the results of the simulations as well as from the analysis are stored in the database (PostgreSQL).
- input_parameters: is the function in which the algorithm retrieves all the necessary input data from the database (PostgreSQL) for the selected scenario.
- Id_to_name_connection: is the function through which the algorithm retrieves the names of the selected corridor or group of corridors from the appropriate tables in the database by utilizing the respective id as key.
- retrieve_data_mfd: is the function of the algorithm which takes as input the names of the selected corridor or group of corridors as well as the selected traffic light program and speed limit and retrieves the respective data that have been derived from the simulations in case the user has set a Macroscopic Fundamental Diagram analysis.
- retrieve_data_edge: is the function of the algorithm which takes as input the names of the selected corridor or group of corridors as well as the selected traffic light program and speed limit and retrieves the respective data that have been derived from the simulations in case the user has set an edge-based analysis.

- formated_data_mfd: is the function of the algorithm which transforms the retrieved data about the Macroscopic Fundamental Diagram analysis to the needed format and stores them to the relevant table in the database so the result to be shown in the front-end environment.
- formated_data_edges: is the function of the algorithm which transforms the retrieved data about the edge-based analysis to the needed format and stores them to the relevant table in the database so the result to be shown in the front-end environment.

The results of the code quality control, by utilizing the methodology described in Section 3.2, are included in Table 39.

Table 40: Code quality control results – 6th tool (UC6 – on-line process)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	main function	CC index	C (14)	moderate - slightly complex block
	main_new function	CC index	A (2)	low - simple block
	db_connection function	CC index	A (1)	low - simple block
	input_parameters function	CC index	A (3)	low - simple block
	id_to_name_connection function	CC index	B (7)	low - well structured and stable block
	retrieve_data_mfd function	CC index	A (3)	low - simple block
	retrieve_data_edge function	CC index	A (3)	low - simple block
	formated_data_mfd function	CC index	C (19)	moderate - slightly complex block
	formated_data_edges function	CC index	C (19)	moderate - slightly complex block

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Maintainability	Entire UC6 .py file	Maintainability index	B (18.05)	Medium maintainability
Raw metrics	Entire UC6 .py file	LOC	682	total # lines of code
	Entire UC6 .py file	LLOC	485	total # logical lines of code[66]
	Entire UC6 .py file	SLOC	478	total # source lines of code[67]
	Entire UC6 .py file	comments	56	total # comment lines
	Entire UC6 .py file	multi	4	total # lines representing multi-line strings
	Entire UC6 .py file	blank	144	total # blank lines
Hal metrics	Entire UC6 .py file	h_1	11	total # distinct operators[68]
	Entire UC6 .py file	h_2	173	total # distinct operands[69]
	Entire UC6 .py file	N_1	183	total # operators
	Entire UC6 .py file	N_2	348	total # operands
	Entire UC6 .py file	Vocabulary	184	$n = n_1 + n_2$

[66] Logical lines of code may be defined as lines of code including executable expressions.
[67] Source lines of code may differ from logical ones given that two executable expressions may be included in each line.
[68] May include arithmetic, comparison, logical, bitwise, assignment, special operators.
[69] The values that an operator operates.

Category	Evaluation element	Metric	Rank (Score)	Interpretation
	Entire UC6 .py file	Length	531	$N = N_1 + N_2$
	Entire UC6 .py file	calculated_length	1324.24	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Entire UC6 .py file	Volume	3995.01	$V = N \log_2(n)$
	Entire UC6 .py file	Difficulty	11.06	$D = (n_1/2)x(N_2/n_2)$
	Entire UC6 .py file	Effort	44199.14	E = D x V
	Entire UC6 .py file	Time	2455.51	T = E/18 seconds
	Entire UC6 .py file	Bugs	1.33	B = V/3000

The average cyclomatic complexity of the UC6 pre-calculation code file is ranked as C (15.55). Hence, it can be considered as a moderated-structured and stable piece of code with slight complex blocks.

4.6.8 Demonstration and testing

This section aims, firstly, to demonstrate the results of the application of the sixth tool and, secondly, to assess the validity of its results based on various test case scenarios. The first purpose is served by presenting the outcomes of running the sixth tool's Python script providing the inputs included in Table 40 (1st test scenario).

Table 41: Inputs corresponding to the 1st test scenario of the sixth tool

Input	Value
Corridor	Vas. Olgas 1
Traffic Light Program	2
Speed Limit	50

The results of the Macroscopic Fundamental Diagram analysis based on the input of Table 34 produces the following Flow – Occupancy diagram (Figure 40).

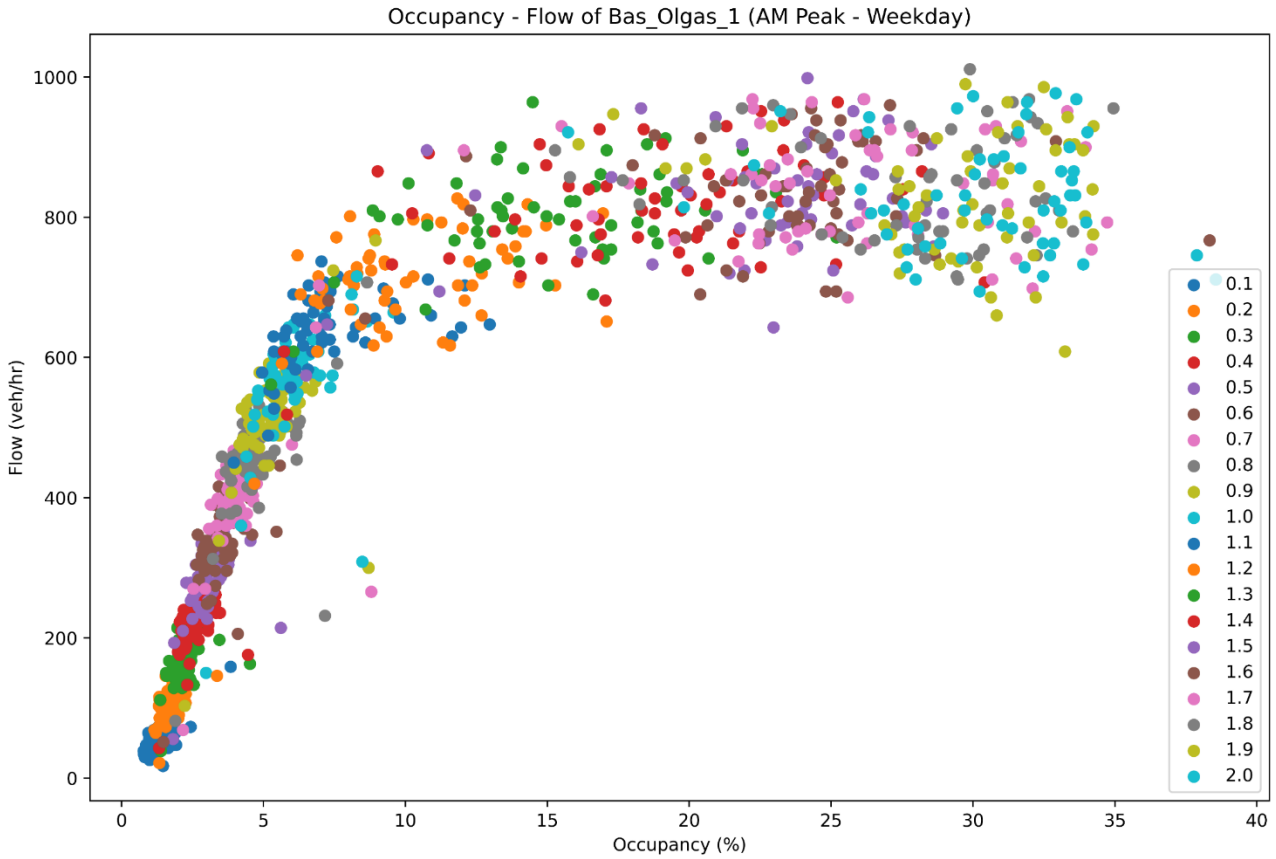


Figure 40: Macroscopic Fundamental Diagram of 1st test scenario

In the second test case scenario it is assumed that the user provides a lower speed limit value (Table 40). The expected response of the tool constitutes a decreased flow value compared to the first test scenario.

Table 42: Inputs corresponding to the 2nd test scenario of the sixth tool

Input	Value
Corridor	Vas. Olgas 1
Traffic Light Program	2
Speed Limit	**30**

The results of the Macroscopic Fundamental Diagram analysis based on the input of Table 40 produces the following Flow – Occupancy diagram (Figure 41).

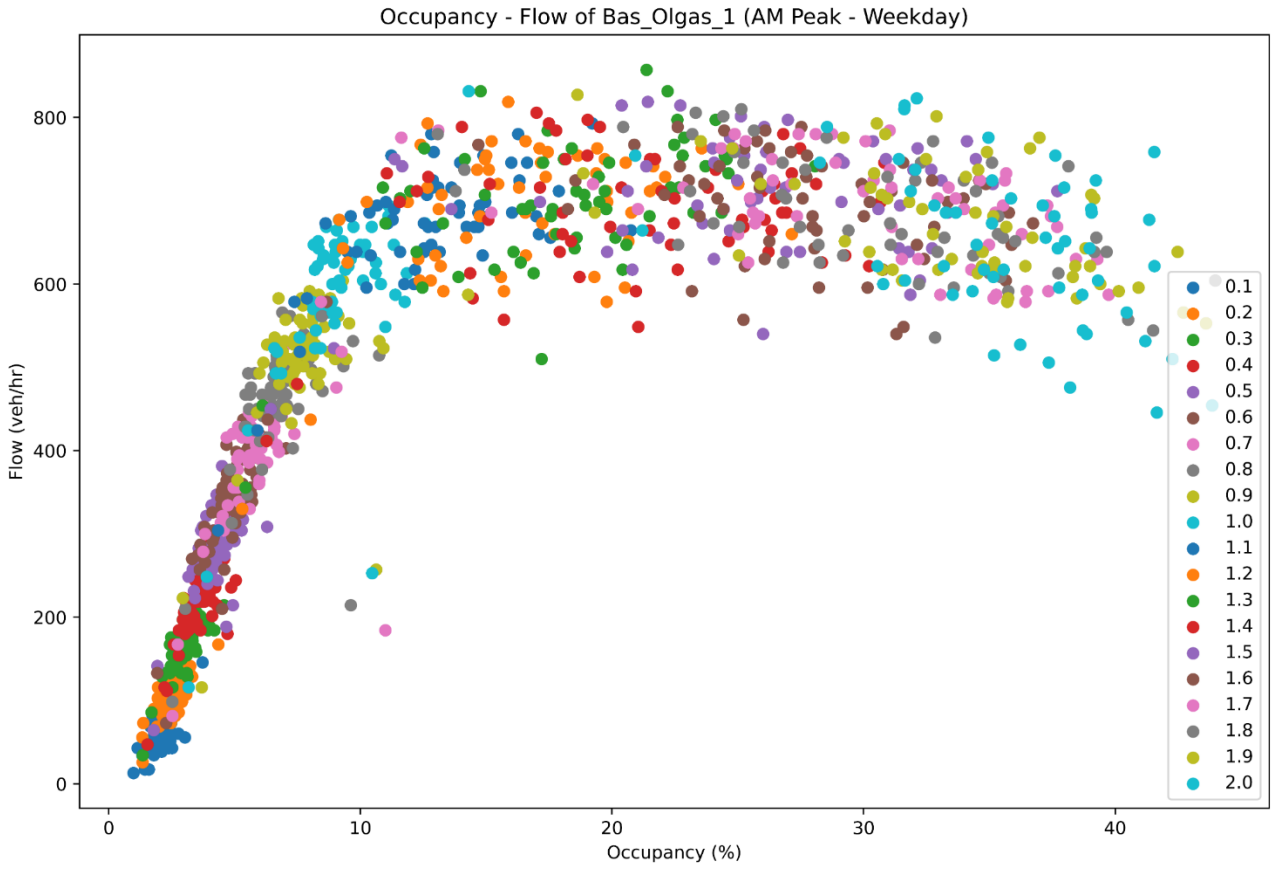


Figure 41: Macroscopic Fundamental Diagram of 2nd test scenario

In the third test case scenario it is assumed that the user selects a group of corridors with the traffic light programs 1 and with a speed limit of 50 km/h (Table 42).

Table 43: Inputs corresponding to the 3rd test scenario of the sixth tool

Input	Value
Group of Corridors	Vas. Olgas 1 - Vas. Olgas 2 - Egnatia 1 - Egnatia 2 - Georgiou Papandreou - Megalou Alexandrou 2
Traffic Light Program	1
Speed Limit	50

The results of the Macroscopic Fundamental Diagram analysis based on the input of Table 36 produces the following Flow – Occupancy diagram (Figure 42).

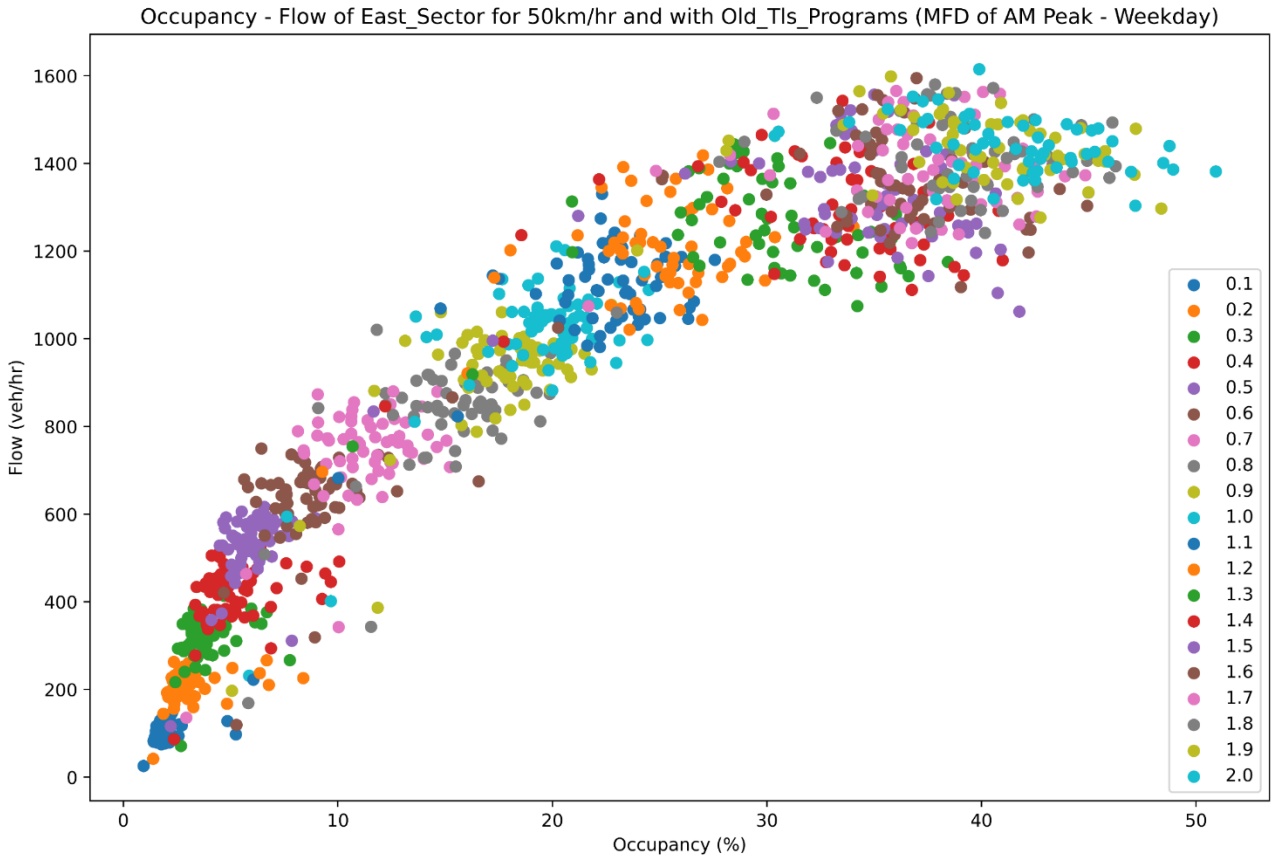


Figure 42: Macroscopic Fundamental Diagram of 3rd test scenario

In the fourth test case scenario it is assumed that the user provides different traffic light programs (Table 43). The expected response of the tool constitutes an increased flow value compared to the third test scenario.

Table 44: Inputs corresponding to the 4th test scenario of the sixth tool

Input	Value
Group of Corridors	Vas. Olgas 1 - Vas. Olgas 2 - Egnatia 1 - Egnatia 2 - Georgiou Papandreou - Megalou Alexandrou 2
Traffic Light Program	**2**
Speed Limit	50

The results of the Macroscopic Fundamental Diagram analysis based on the input of Table 43 produces the following Flow – Occupancy diagram (Figure 43).

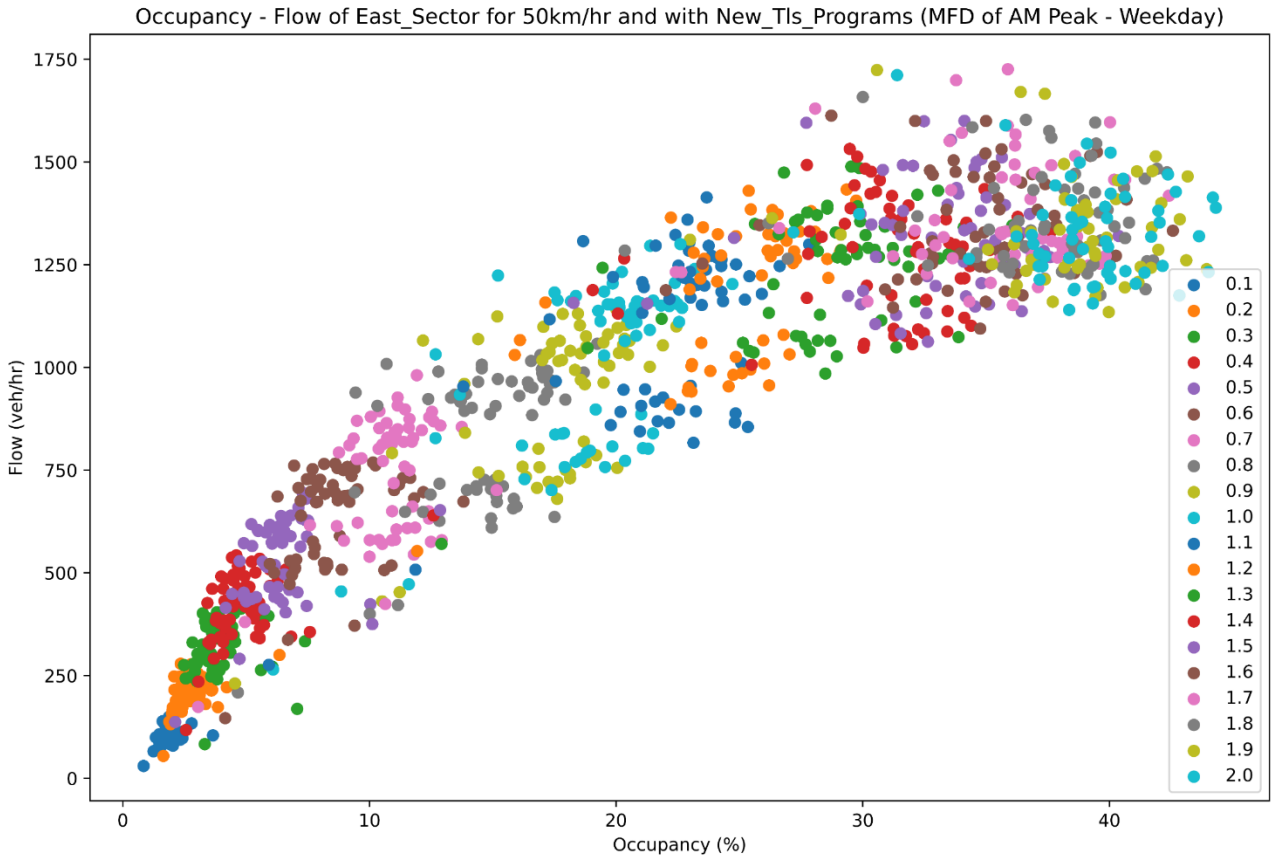


Figure 43: Macroscopic Fundamental Diagram of 4th test scenario

In the fifth test case scenario it is assumed that the user selects a statistical analysis of data. For this scenario, the user selects a corridor with the 1st traffic light program and with a speed limit of 50 km/h. Also, for the statistical analysis the parameter queueing times over time is selected to be analysed by the user. For running the sixth tool's Python script the inputs included in Table 44.

Table 45: Inputs corresponding to the 5th test scenario of the sixth tool

Input	Value
Group of Corridors	Vas. Olgas 1
Traffic Light Program	1
Speed Limit	50
Statistics Analysis	Queueing times over time

The results of the Statistics analysis based on the input of Table 44 produces the following queueing times over time (Figure 44).

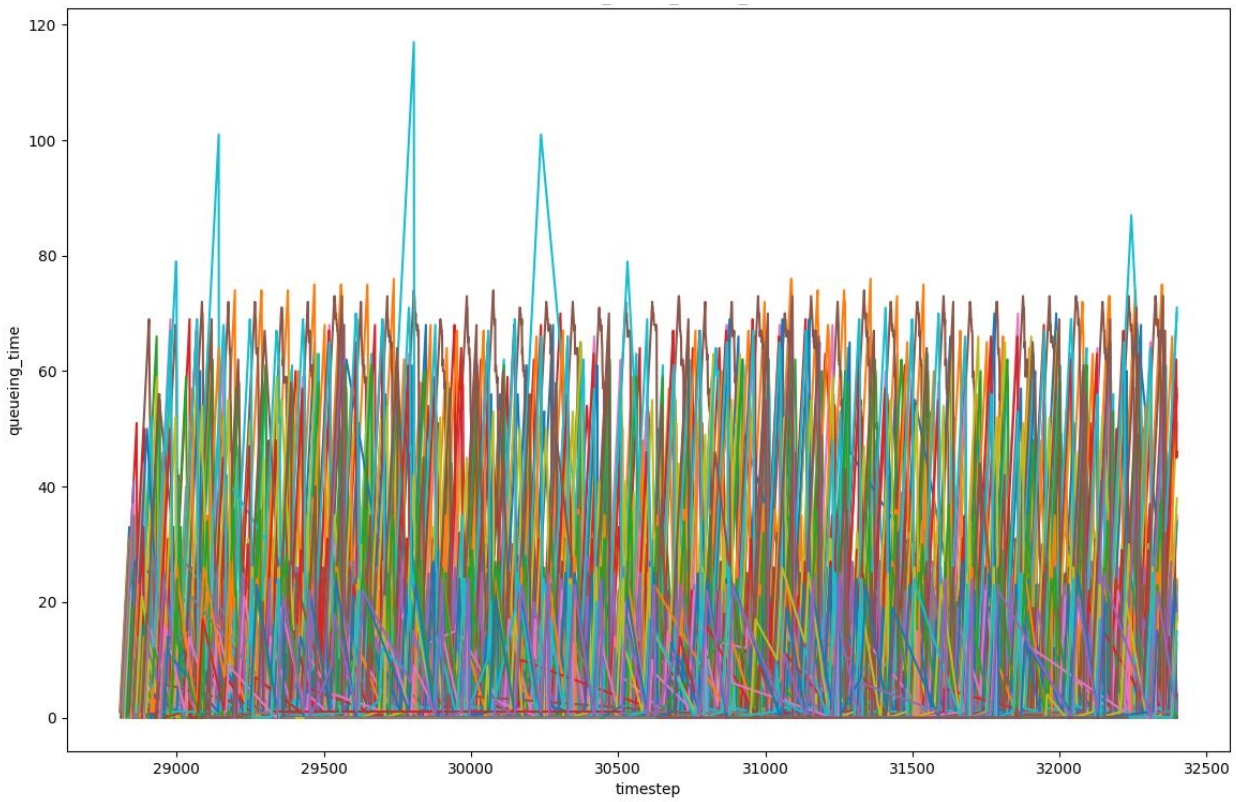


Figure 44: Statistics analysis of queueing times over time – 5th test case scenario

In the sixth test case scenario it is assumed that the user provides different traffic light programs (Table 45). The expected response of the tool constitutes a decreased queueing times values over time compared to the fifth test scenario.

Table 46: Inputs corresponding to the 6th test scenario of the sixth tool

Input	Value
Group of Corridors	Vas. Olgas 1
Traffic Light Program	**2**
Speed Limit	50
Statistics Analysis	Queueing times over time

The results of the Statistics analysis based on the input of Table 38 produces the following queueing times over time (Figure 45).

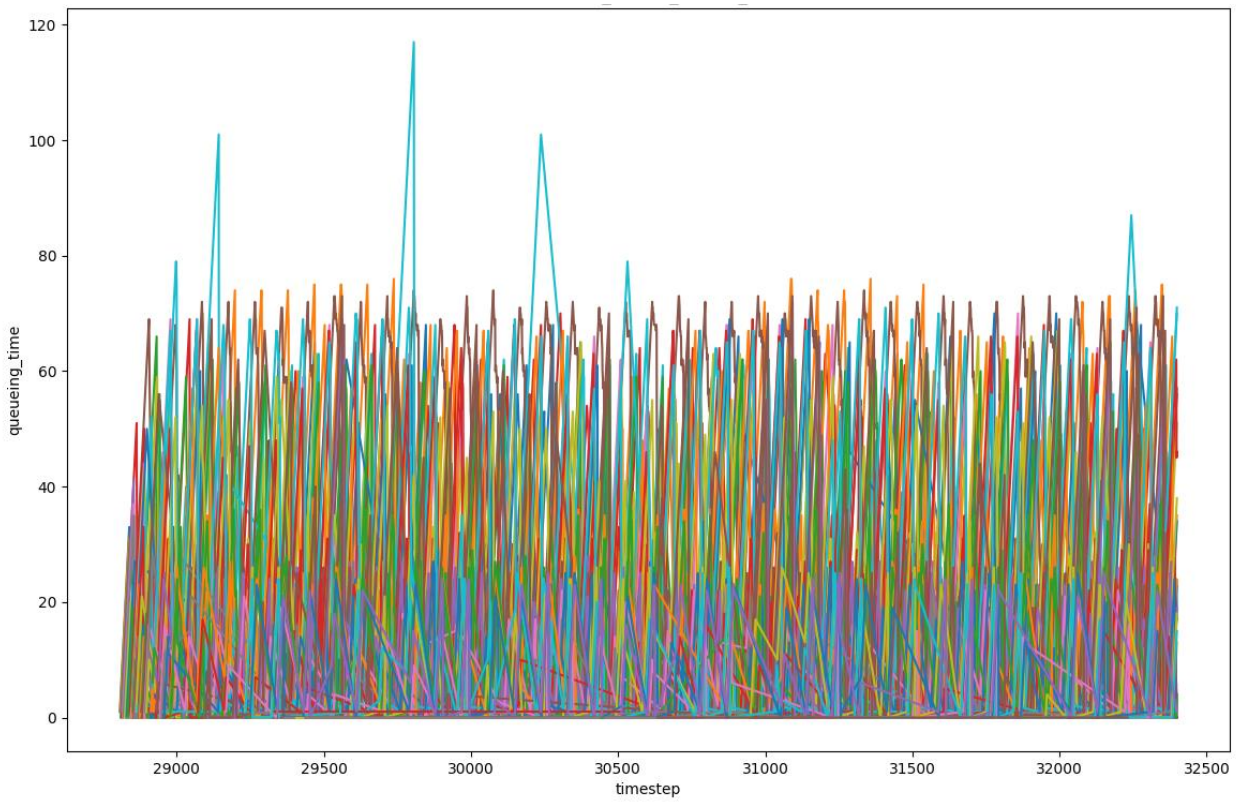


Figure 45: Statistics analysis of queueing times over time – 6th test case scenario

The above results suggest that the tool developed in response to the requirements set out by the fifth use case produce rational results.

5 Concluding remarks

This deliverable described the final outcomes of the formulation, evaluation, and prototyping of the methods and tools of which the project's toolkit is comprised. For each of the methods and tools a brief description highlighting its scope and purposes is provided. In addition, the targeted users, data inputs, and data outputs of each of them are mentioned. In the context of the relevant subsections, an illustrative diagram is delivered emphasising on the basic steps into which each tool's algorithm can be decomposed. In parallel, the methodological and mathematical framework behind these steps is described in detail. In addition, for each tool in full development, a description of its code and code quality is provided. In the context of the relevant subsections, the functions of which their code is comprised is described (including inputs, purpose, and intermediate or final outputs). In these subsections, the results of the code quality assessment are also presented by following the approach stated in the beginning of the document. The results of this quality control process indicate that the functional prototypes of the project's toolkit components are robust and stable from a technological point of view. In an effort to logically assess the outputs of these components, several test scenarios are executed. In these test scenarios the value of one or more of input parameters is differentiated and an assessment is made whether the tool responds as expected. The results of these test scenarios highlight that the methods and tools perform as expected and produce rational results.

6 References

Bellairs, R. (2019). What Is Code Quality? Overview & How to Improve Code Quality. Perforce Software. https://www.perforce.com/blog/sca/what-code-quality-overview-how-improve-code-quality

Bennear, L. S., & Stavins, R. N. (2007). Second-best theory and the use of multiple policy instruments. Environmental and Resource economics, 37(1), 111-129.

Guo, W., Zhang, Y., Xu, M., Zhang, Z., & Li, L. (2016). Parking spaces repurchase strategy design via simulation optimization. Journal of Intelligent Transportation Systems, 20(3), 255-269.

Hirayama, M., Sato, H., Yamada, A., & Tsuda, J. (1990, March). Practice of quality modeling and measurement on software life-cycle. In [1990] Proceedings. 12th International Conference on Software Engineering (pp. 98-107). IEEE.

Inci, E., Ommeren, J. V., & Kobus, M. (2017). The external cruising costs of parking. Journal of Economic Geography, 17(6), 1301-1323.

Ji, Y., Luo, J., & Geroliminis, N. (2014). Empirical observations of congestion propagation and dynamic partitioning with probe data for large-scale systems. Transportation Research Record, 2422(1), 1-11.

Klein, L. A. (2018). ITS Sensors and Architectures for Traffic Management and Connected Vehicles. Boca Baton: CRC Press

Lipsey, R. G., & Lancaster, K. (1956). The general theory of second best. The review of economic studies, 24(1), 11-32.

Nešić, A., Čavka, I., & Čokorilo, O. (2015). Shifting to more environmentally friendly modes in long-distance transport. KEEPING UP WITH TECHNOLOGIES TO MAKE HEALTHY PLACES.

Nikitas, A. (2019). How to save bike-sharing: An evidence-based survival toolkit for policy-makers and mobility providers. Sustainability, 11(11), 3206.

Roelofsen, M., Bie, J. J., & DHV, W. W. F. (2012). Dynamic modelling of traffic management scenarios using Dynasmart. Auckland, New Zealand.

Sayarshad, H., Tavassoli, S., & Zhao, F. (2012). A multi-periodic optimization formulation for bike planning and bike utilization. *Applied Mathematical Modelling*, *36*(10), 4944-4951.

Shaheen, S., Cohen, A., & Zohdy, I. (2016). Shared Mobility. Current Practices and Guiding Principles (Report no. FHWA-HOP-16-022 for the Federal Highway Administration). https://ops.fhwa.dot.gov/publications/fhwahop16022 (Accessed on 20.01.2021).

Yan, X., Levine, J., & Marans, R. (2019). The effectiveness of parking policies to reduce parking demand pressure and car use. Transport Policy, 73, 41-50.

Menz, P., Mulberry, N., Guichard, D., & Team, L. L. (2018). Calculus Early Transcendentals Differential & Multi-Variable Calculus for Social Sciences.

Narayan, J., Cats, O., van Oort, N., & Hoogendoorn, S. P. (2021). Fleet size determination for a mixed private and pooled on-demand system with elastic demand. Transportmetrica A: Transport Science, 17(4), 897-920.

Peña Carrera, L. (2002). Tracing accessibility over time: two Swiss case studies.