Deliverable 3.2

First 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) Evangelos Mitsakis (CERTH) Dimitris Tzanis (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	27/02/2022 (M14)
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)
V1.0	27/02/2022	Content of all remaining section (incl. executive summary and concluding remarks) added to the document – First consolidated draft Submitted 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 optimization 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 allocation of existing shared mobility services' supply (i.e., operable fleet) to specific sub-areas of a metropolitan area with the aim of minimizing economic losses of service providers and ensuring an acceptable level of service within each sub-area. This tool will also have the aim of promoting equity. The third tool focuses on air quality analysis based on multi-source data. The fourth tool, which is also in full development, aims to evaluate the traffic-related impacts of on-street parking restrictions within urban areas. Subsequently, the objective of the fifth tool, which is also in full development involves the estimation of inflows and outflows between the districts of a metropolitan area based on data generated by point-to-point detection systems as well as external (e.g., census) data. 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 to be integrated into the nuMIDAS toolkit concerns the evaluation of traffic management scenarios comprised of conventional and C-ITS enabled measures utilizing historical, real-time, and synthetic data.

This document ends up by summarizing its outputs and identifies the further steps to be taken. Therefore, this deliverable is deemed critical both for the development of the remaining elements of the toolkit and the refinement of the developed ones.

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 first 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 results. The developed methods and tools will be further tested in the context of WP4 utilizing real-world data from pilot cities.

Furthermore, it should be noticed that this document constitutes the first deliverable of the advanced methods and tools formulation, validation, verification, and prototyping. Two more deliverables will successively follow. The results from the methods and tools responding to the requirements related to a) the pre-planning of shared mobility services (UC1), b) the assessment of parking restriction policy impacts (UC4) and c) the estimation of inflows and outflows in a metropolitan area (UC5) are thoroughly discussed and analyzed within this deliverable. Concerning the remaining use cases, including the operative areas analysis (UC2), the air quality and vehicle emissions analysis based on multi-source data (UC3), and the assessment of traffic management scenarios (UC6), a high-level description of the currently foreseen features and computational structure and capabilities is enclosed within this deliverable. The next version of the deliverable will also include the results of their tools. Any additional needs that may arise from the analysis of the tools presented in this document will be discussed and fulfilled in the next versions as well. Finally, a schematic representation of all advanced methods and tools is also enclosed.

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 analyzed in brief. Next, the computational flow of each tool is analyzed 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.

The structure analyzed above concerns the tools, which have been already prototyped and as such they are in the first completed phase of their development. The structure of the sub-chapters concerning the remaining tools, which are not yet in a consolidated version, includes a brief overview of their purpose and scope, their targeted users, as well as their foreseen inputs and outputs. 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 discusses the outcomes of the advanced methods and tools formulation, validation, verification, and prototyping, sets out future goals, and concludes the document.

2.4 Acronyms

EC	European Commission
GA	Grant agreement
nuMIDAS	New Mobility Data and Solutions Toolkit
WP	Work package
UC	Use case
ANPR	Automatic Number Plate Recognition
C-ITS	Cooperative Intelligent Transport Systems

3 Adopted approach

3.1 Formulation and prototyping approach

As already mentioned, this 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 conceptualized 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 iterative process terminates.

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 utilized 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 is attributed both to the improvement of the practical utility of the computational algorithms as well as to the accumulation of useful experience. In this respect, future developments, including the remaining elements of the project's toolkit, are expected to be smooth and efficient.

[1] This is based on the premise that the toolkit's use case has been designed following a pragmatic approach. Such an approach involved the conceptualization of the use case content considering the needs of the cities and their generic analysis 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 high quality (Plösch, et. al., 2010). On the other hand, a code that follows an unstable structure is considered as 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. To facilitate the process and to be always on track without any deviations, it is important to keep in mind the life cycle of code quality control. The quality control life cycle is an ongoing process of planning, monitoring, assessing, comparing, correcting, and improving the code through continuous reinvention.

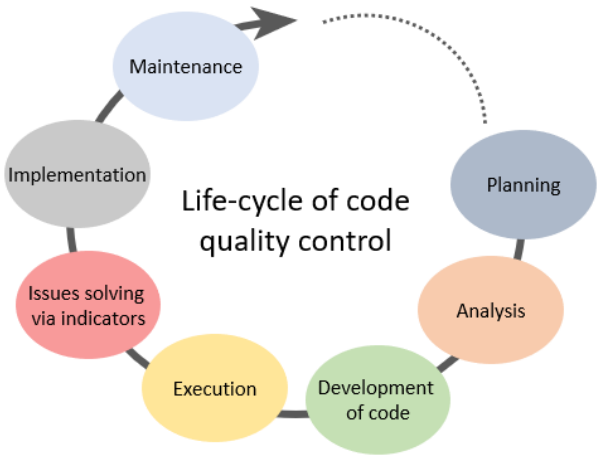


Figure 1: Life cycle of code quality control

In the planning stage, well-structured thoughts should be established to achieve a deep understanding of the code’s usage. Next, in the analysis phase, the definition of code’s requirements takes place. The designing and the development of the code follows. In this phase, all requirements stemming from the previous stages are translated and incorporated into code. After the completion of the development, the next step includes the execution of the code to test 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. The final stage, which is ever-lasting is the maintenance of the code. The processes included in this stage stem from the need of continuously assessing and incorporating new needs arising from the users of the code. 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

In an attempt to enhance 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 the optimization problem is converted into mathematical expressions. Despite the better understanding of the problem to be solved by each tool, the validity of each tool results has not yet evaluated. This can be achieved through logical examples with a wide range of input values. Along these lines, the exported outcomes can be easily anticipated and evaluated as well. By that means, a range of test scenarios are executed targeting to examine whether the outcomes are realistic, or the code needs to be improved. 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. Correspondingly, 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 tests whether the tool's specifications fulfill the requirements and the design standards or not. In both evaluation processes, further discussions among project stakeholders are necessary with regards to ensure that all current operational functionalities are covered. It is also important to embed new potential features when needed. Hence, evaluation process is partly responsible for the continuity of the 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 three out of six methods and tools are in full development, while the remaining three will be either incorporated in the next deliverable or in an updated version of the current deliverable. For this reason, a different structure is followed to describe the outcomes of the formulation, evaluation, and prototyping process. In addition, for each of the methods and tools being under full development, a comprehensive process flow diagram is provided. These diagrams will be translated into UML diagrams upon the finalization of the methods and tools. 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 brief description 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	In progress	Supports the distribution of existing shared mobility services supply to specific (high- or low-demand) sub-areas of a metropolitan area to minimize economic losses of service providers and to ensure an acceptable level of service for citizens in line with the principles of equitable transport systems.
UC3	Air quality analysis and forecasting	In progress	Supports the linkage and correlation of multi-source data including traffic-related, emissions-related, and weather-related data. The ultimate purpose of this linkage/correlation is to assess the impact of traffic on air quality, forecast air quality on a short- to medium-term basis (e.g., 10 days), and/or simulate scenarios for the development and enforcement of air quality-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 generalized 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).
UC6	Assessment of traffic management scenarios	In progress	Supports the assessment of C-ITS enabled traffic management scenarios based on simulation-based tools and data analytics, making use of multi-source data.

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 optimization problem. This problem involves the identification of a value for the fleet size of a mobility service that jointly maximizes 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 (policymakers)
- Transport planners supporting policymakers
- Travelers (end-users)

However, among the above stakeholders the ones that belong to the targeted users of the tool are transport planners and policymakers. 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 utilized 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 analyzed mobility service. To this end, the tool prompts the user to upload a file including multi-periodic trip data stemming either from the operation of the same mobility service within an area equivalent in socioeconomic terms with the area of interest or 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 analyzed (station-based or free-floating)
- Selection of the type of mode to be analyzed (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 analyzed 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

analyzed mode. The remaining inputs included in the above list are provided in numerical form and are utilized 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 utilized 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 optimization 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.

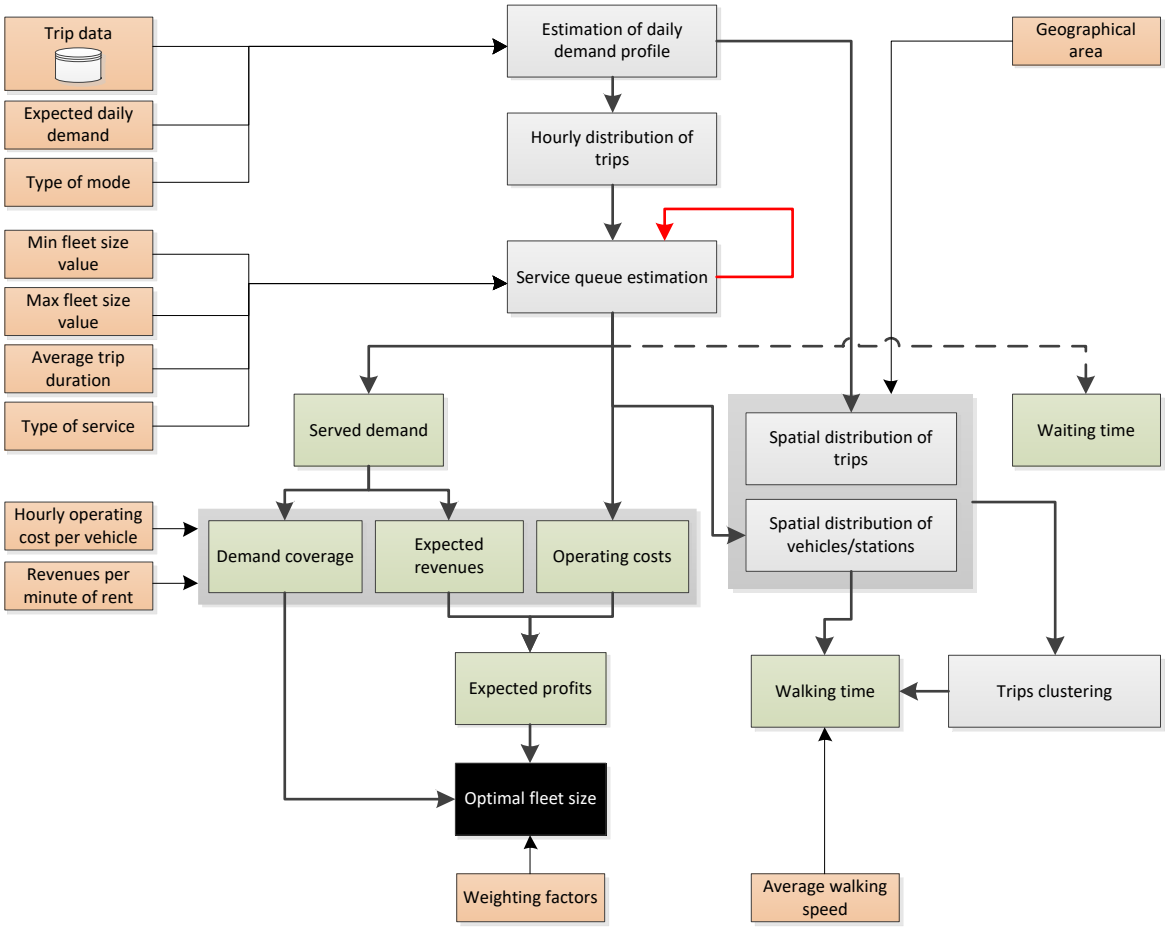


Figure 2: Computational flow of the first tool of nuMIDAS toolkit

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 2-5. 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.

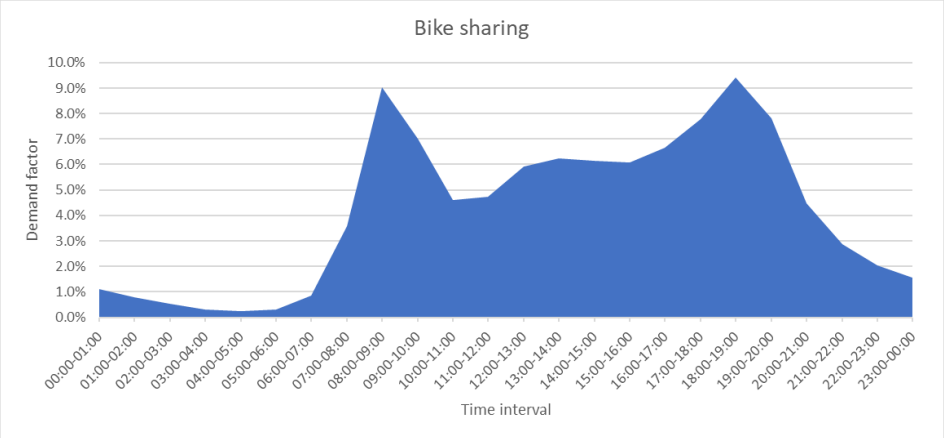


Figure 3: Demand profile of bike sharing services obtained from the pilot city of Milan

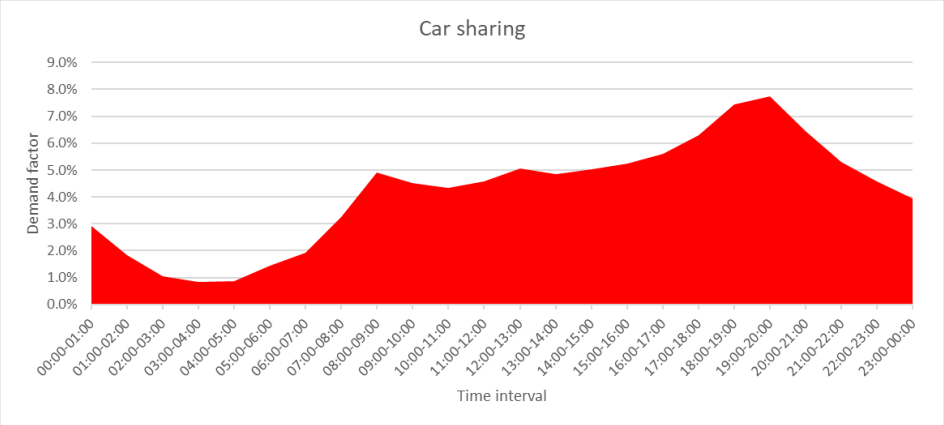


Figure 4: Demand profile of car sharing services obtained from the pilot city of Milan

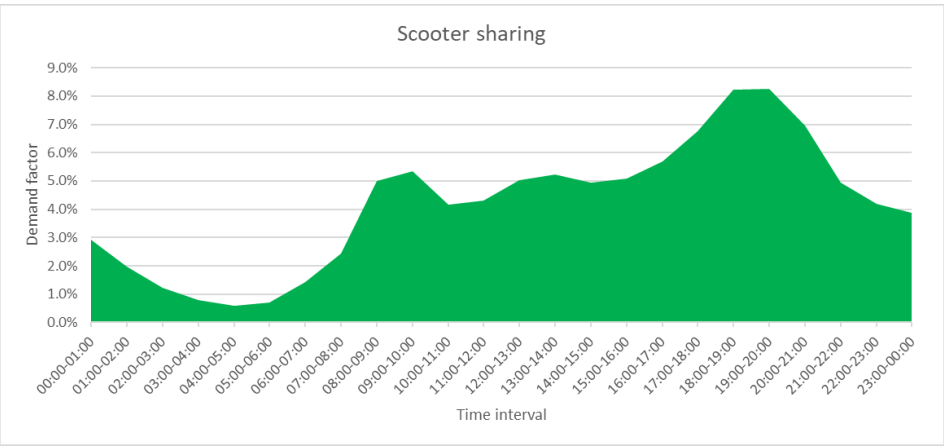


Figure 5: Demand profile of scooter sharing services obtained from the pilot city of Milan

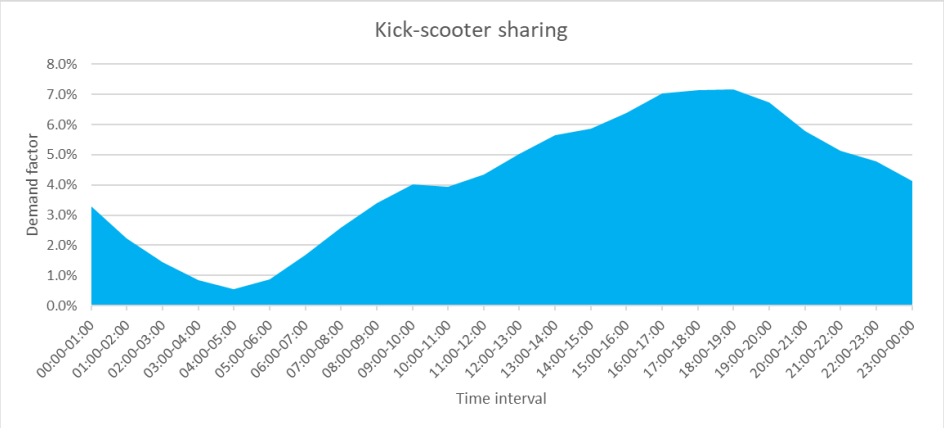


Figure 6: Demand profile of kick-scooter sharing services obtained from the pilot city of Milan

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 utilized. The choice of this type of probability distribution is based on the analysis of historical data from taxi services in the pilot city of Thessaloniki, covering a total period of 3 months. In Figure 6, 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 NOT AVAILABLE

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

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 analyzed mobility service is station-based. In contrast, end users are randomly served (irrespective of their arrival time) if the analyzed 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.

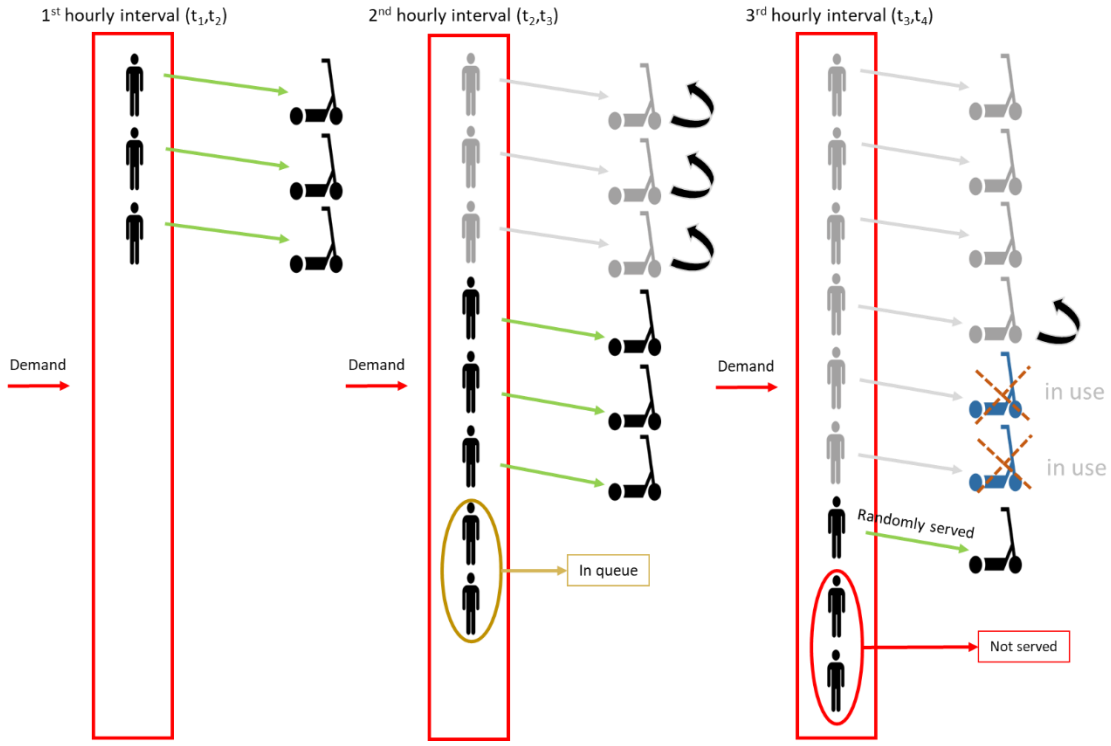


Figure 8: Demonstration of adopted approach for estimating the amount of served demand

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 start time and the actual trip start time is one and 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 generalized 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 analyzed area is less than 5 km^2, then it is assumed that one station hosts up to 10 vehicles
- If the analyzed 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 analyzed 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 utilized 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 analyzed service is station-based) or with the looped number of fleet size (if the analyzed service is free-floating). This is achieved by utilizing k-means 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 kimlometers, i.e., each dot respresents 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.

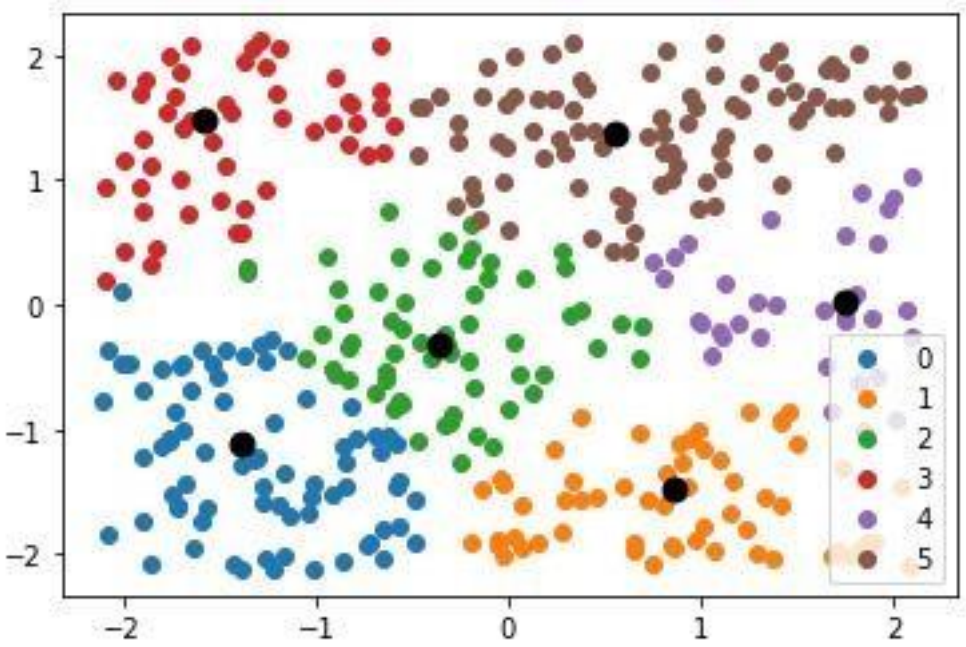


Figure 9: Adopted approach for estimating walking distance

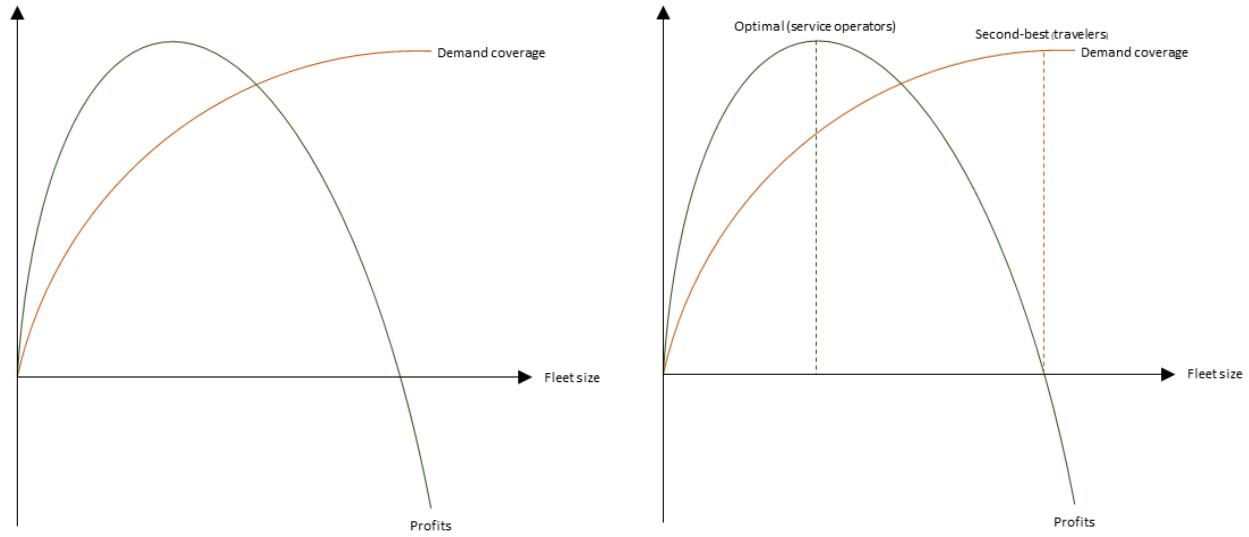
Having estimated the average walking distance of users, walking time is calculated trough the following formula:

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

The last step of the computational flow involves the calculation of the optimal fleet size. The decision variables utilized 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 travelers (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 utilization 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 maximization 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 analyzing how the removal of a market distortion may lead to the introduction of a new market distortion, there is a second-best policy that maximizes 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 maximizes the welfare of service operators and a second-best solution maximizing to the extent possible the welfare of travelers (end users).



(a) (b)

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

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 travelers. The overall formulation of the problem is as follows:

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

subject to:

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

$$profits > 0$$

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

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 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 3.7.0. The dependencies of the code are as follows:

- Sys library
- Math library
- Random library
- Psycopg2 library
- Numpy library
- Pandas library
- Mean method of Statistics library
- Matplotlib.pyplot library
- Sqlalchemy method of Create_engine library

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

- main: is the function which prompts the user to provide inputs, pass it to the hourly_sim_solution function described below and retrieves its outputs.
- 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 the vehicles comprising the fleet and the demand for shared mobility service without any constraint.
- 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.
- 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.
- hourly_sim_solution: 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)	hourly_sim_solution function	CC index	C (16)	moderate - slightly complex block
	main function	CC index	B (7)	low - well-structured and stable block
	user_input_2_1 function	CC index	B (7)	low - well-structured and stable block
	random_generator_constrained function	CC index	B (6)	low - well-structured and stable block
	kmeans_impl function	CC index	A (5)	low - simple block
	random_generator_nonconstrained function	CC index	A (4)	low - simple block
	user_input_2_2 function	CC index	A (3)	low - simple block
	convert function	CC index	A (1)	low - simple block
Maintainability	Entire UC1 .py file	Maintainability index	A (38.27)	Very high maintainability
Raw metrics	Entire UC1 .py file	LOC	610	total # lines of code
	Entire UC1 .py file	LLOC	307	total # logical lines of code[4]
	Entire UC1 .py file	SLOC	295	total # source lines of code[5]

[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.

	Entire UC1 .py file	comments	46	total # comment lines
	Entire UC1 .py file	multi	4	total # lines representing multi-line strings
	Entire UC1 .py file	blank	268	total # blank lines
Hal metrics	Entire UC1 .py file	n_1	14	total # distinct operators[6]
	Entire UC1 .py file	n_2	155	total # distinct operands[7]
	Entire UC1 .py file	N_1	111	total # operators
	Entire UC1 .py file	N_2	219	total # operands
	Entire UC1 .py file	Vocabulary	169	$n = n_1 + n_2$
	Entire UC1 .py file	Length	330	$N = N_1 + N_2$
	Entire UC1 .py file	calculated_length	1181.10	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Entire UC1 .py file	Volume	2442.29	$V = N \log_2(n)$
	Entire UC1 .py file	Difficulty	9.89	$D = (n_1/2)x (N_2/n_2)$
	Entire UC1 .py file	Effort	24155.04	E = D x V
	Entire UC1 .py file	Time	1341.95	T = E/18 seconds
	Entire UC1 .py file	Bugs	0.81	B = V/3000

The average cyclomatic complexity of the UC1 code file is ranked as B (6.125). Hence, it can be considered as a well-structured and stable piece of code.

4.1.7 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).

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

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 (travelers)	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: **108**
- Optimal fleet size (service operators): **63**
- Optimal fleet size (travelers): **152**
- Demand coverage corresponding to the optimal solution: **83,7%**
- Expected profits corresponding to the optimal solution: **€6.980 (daily)**
- Average walking time corresponding to the optimal solution: **0:06:39**
- Average waiting time corresponding to the optimal solution: **0:02:30**

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 11[8].

[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.

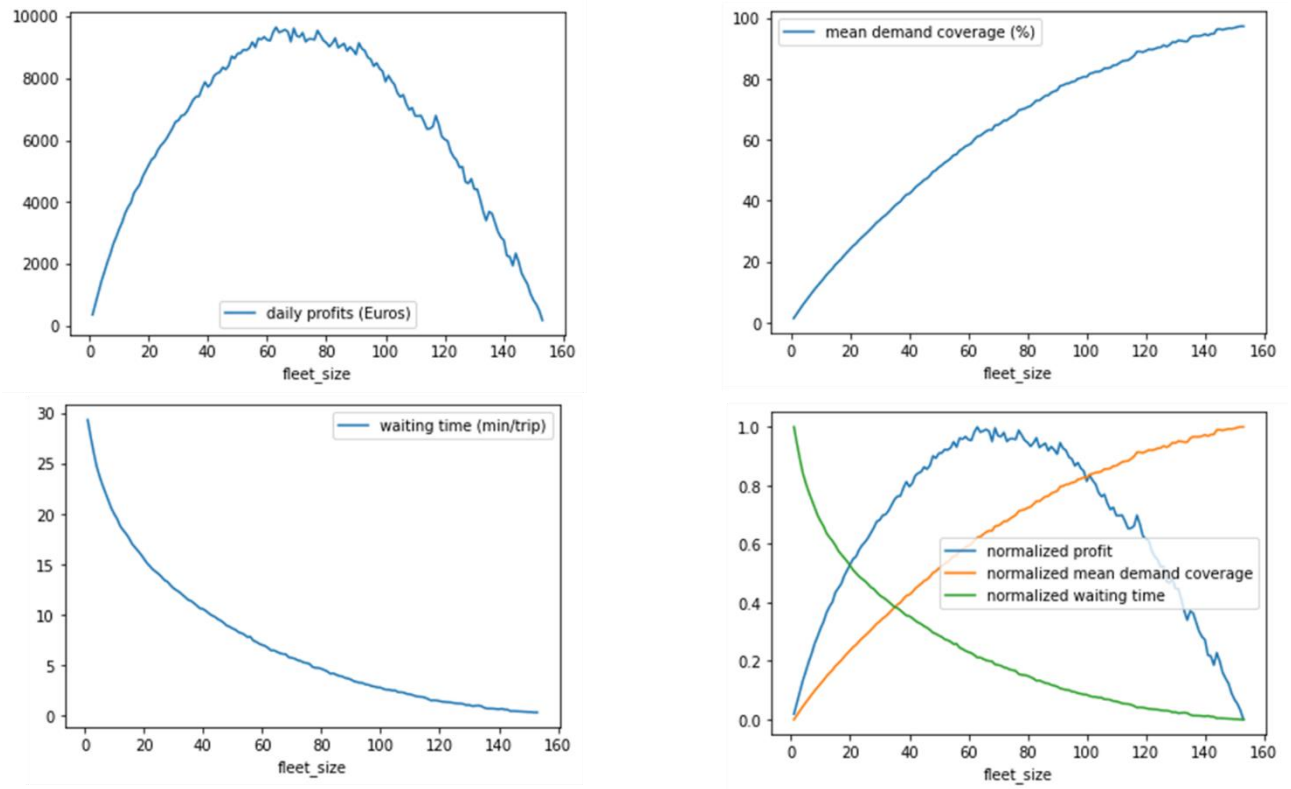


Figure 11: Graphical outputs corresponding to the base test scenario of the first tool

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
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 (travelers)	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: **174**
- Optimal fleet size (service operators): **124**
- Optimal fleet size (travelers): **224**
- Demand coverage corresponding to the optimal solution: **87,4%**
- Expected profits corresponding to the optimal solution: **€9.511 (daily)**
- Average walking time corresponding to the optimal solution: **0:04:59**
- Average waiting time corresponding to the optimal solution: **0:01:47**

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 (travelers)	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: **55**
- Optimal fleet size (service operators): **34**
- Optimal fleet size (travelers): **76**
- Demand coverage corresponding to the optimal solution: **73,6%**
- Expected profits corresponding to the optimal solution: **€2.764 (daily)**
- Average walking time corresponding to the optimal solution: **0:09:10**
- Average waiting time corresponding to the optimal solution: **0:04:29**

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 (travelers)	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: **158**
- Optimal fleet size (service operators): **125**

- Optimal fleet size (travelers): **192**
- Demand coverage corresponding to the optimal solution: **97,0%**
- Expected profits corresponding to the optimal solution: **€21.410 (daily)**
- Average walking time corresponding to the optimal solution: **0:05:09**
- Average waiting time corresponding to the optimal solution: **0:00:19**

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 (travelers)	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: **58**
- Optimal fleet size (service operators): **38**
- Optimal fleet size (travelers): **78**
- Demand coverage corresponding to the optimal solution: **57,0%**
- Expected profits corresponding to the optimal solution: **€1.542 (daily)**
- Average walking time corresponding to the optimal solution: **0:09:20**
- Average waiting time corresponding to the optimal solution: **0:07:14**

The above results suggest that the tool developed in response to the requirements set out by the first use case produce rational results. Further stress testing will be executed in the context of WP4 by utilizing real-world data from the pilot city of Milan. Based on the derived results the parameters of the tool will be further refined/calibrated.

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 allocation of the operable fleet of multiple mobility service operators into specific service areas (operative areas). This necessity stems from the fact that metropolitan areas in Europe are typically sprawled across large geographic areas, including both densely and less densely populated sub-areas. As such, some of these sub-areas eventually end up being overserved and others being underserved in terms of offered mobility shared services. Thus, this tool targets to support policy makers to distribute service operators including their operable fleet into specific sub-areas of a metropolitan area. Policy makers should follow a sound approach ensuring that this division obey in the following: a) the level of provided services will be adequate for the end-users, b) the full area will be divided into service zones considering constraints related to equity (i.e. service operators should be treated equally and/or citizens should have equal access to shared mobility services), and c) the profit margin of service operators will be adequate to avoid financial losses, thus safeguarding the viability of operated services. A critical assumption to be made is that service operators are typically interested in serving smaller geographical areas of high population density with the aim of minimizing operating costs and maximizing their revenues (i.e., maximizing their profits). In this respect, policy makers should make a trade-off between assigning to each service operator an attractive service area and a service area of lower attractiveness that should be served to ensure a minimum level of service.

The tool will receive as input the value of the fleet size that should optimally be operated in each sub-area from the tool associated with the UC1. By that means, minimum and maximum operable fleet will be identified for each zone enabling the determination of the range of the decision variable, which in this case will be the amount of the capacity of each service operator allocated in each sub-area. Subsequently, the tool will be capable of approximating the profitability of service operators during a day for a given allocation of their capacity into specific sub-areas. The suggested value of the operable fleet per service operator and service will seek to optimize the level of service and the profitability of provided services in each area in a manner similar with the tools associated with UC1. Furthermore, and most importantly, the tool will seek a solution that will also minimize the difference in the level of service among the defined sub-areas and the difference in profitability among service operators, thus promoting equity. Taking into consideration that the land uses in European metropolitan are organized in such a manner so that a city center exists in each city (and typically city centers constitute attractive areas), service operators may have to overlap within this area.

4.2.2 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 government (e.g., municipality) tasked with issuing tenders for service (policymakers)
- Transport planners supporting policymakers
- Travelers (end-users)

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

4.2.3 Data needs

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

- Selection of the type of service to be analyzed
- Geofenced districts
- Number of available service operators
- Optimal fleet size in each district
- Demand factors in each district
- Expected daily demand in each district
- Operating cost per vehicle per minute
- Expected revenues per minute of rent
- Average users walking speed
- Mean trip duration (in minutes)

Similar to the first tool, the second tool will be able to analyze various shared-mobility services, including bike, scooter, kick scooter, and car-sharing services.

4.2.4 Data outputs

The outputs of the tool (indicatively) include:

- Definition of sub-areas
- Fleet assigned to each sub-area per operator
- KPIs for each area (e.g., related to level of service, operating costs, profitability)

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, the third tool 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 and Urban Vehicle Access Restrictions. The tool will analyse and correlate various data sources providing information about traffic intensity, weather conditions, air quality, and events. The ultimate purpose is 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 utilization (e.g., linear regression, logistic regression, artificial neural networks, deep neural networks)

4.3.2 Targeted users

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

- Data providers (incl. traffic management centres)
- Departments of local government (e.g., municipality) responsible for the enforcement of traffic restriction policies (policymakers)
- Transport planners supporting policymakers

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

4.3.3 Data needs

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

- 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

4.3.4 Data outputs

The outputs of the tool (indicatively) include:

- Traffic-related forecasts
- Air quality-related forecasts

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 discretization 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 (policymakers)
- Off-street parking managers (private service providers)
- Transport planners supporting policymakers
- Traffic police
- Drivers

However, among the above stakeholders the ones that belong to the targeted users of the current tool are transport planners and policymakers. 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 generalized 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 will be 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 behaviors 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 discretization 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 second tool is depicted in Figure 12.

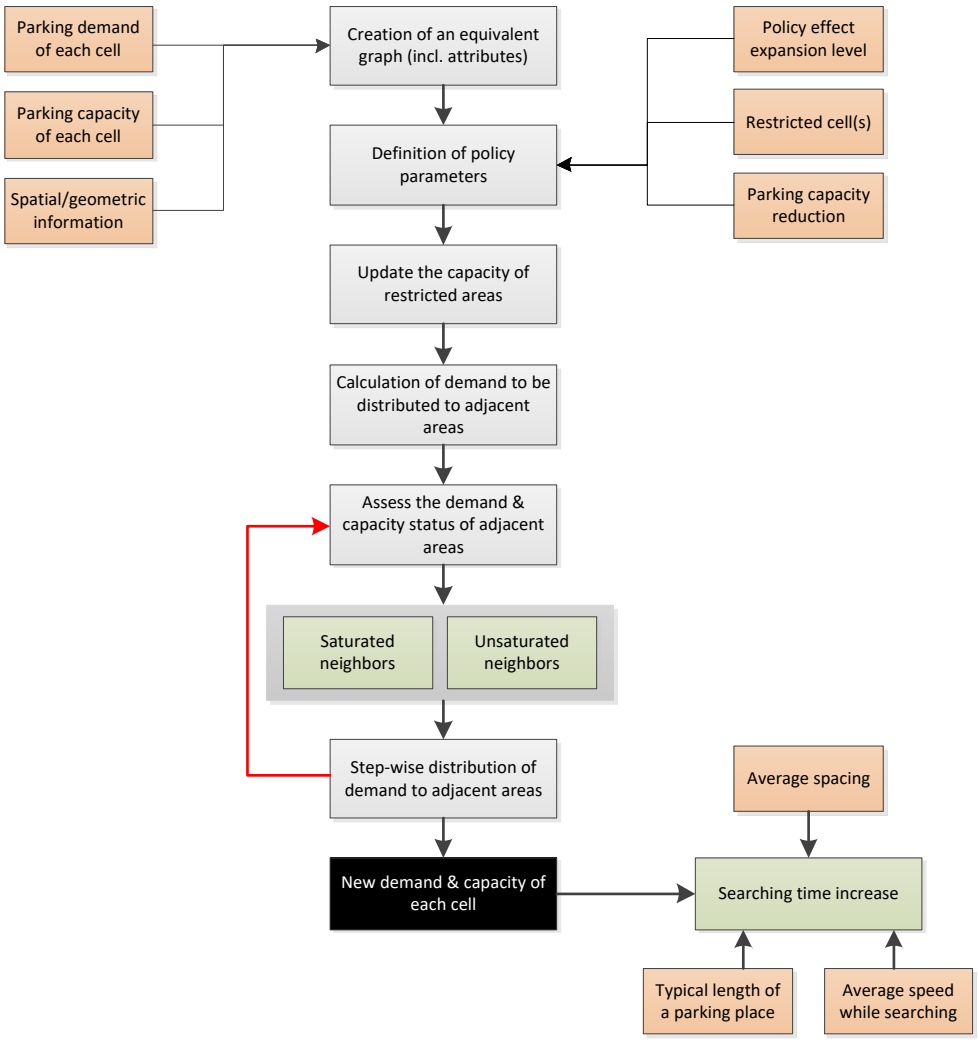


Figure 12: Computational flow of the fourth tool of nuMIDAS toolkit

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 13). By that means, it becomes readily manageable to a define an adjacency matrix A_{ij} and, thus, identify the neighbors 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).

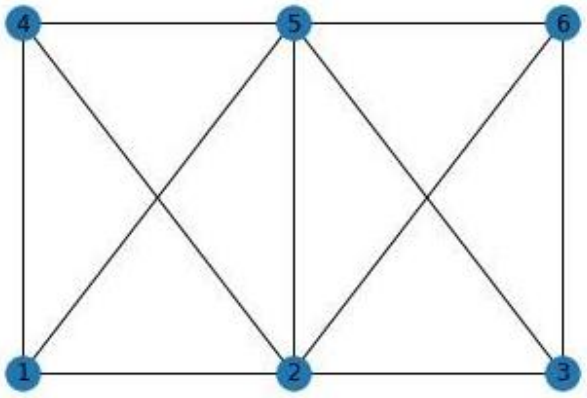


Figure 13: Structure of equivalent graph

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 neighbors 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 summarized in the pseudocode included in the following block.

Step 0: Identification of saturated and non-saturated neighbors
Condition 1: Non-saturated neighbor(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 neighbors 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 neighbors 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 neighbors
        Condition 1.2.1: Non-saturated neighbors 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 neighbors 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 neighbors 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 neighbors 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 neighbors 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 neighbors of the nth level are reached. The condition based on which the tool continues this exploratory process is that all neighbors are saturated. Provided that even one neighbor is assessed as non-saturated, this translates that PD^* have been already attracted by the neighbors of the (n-j)th level (Figure 14).

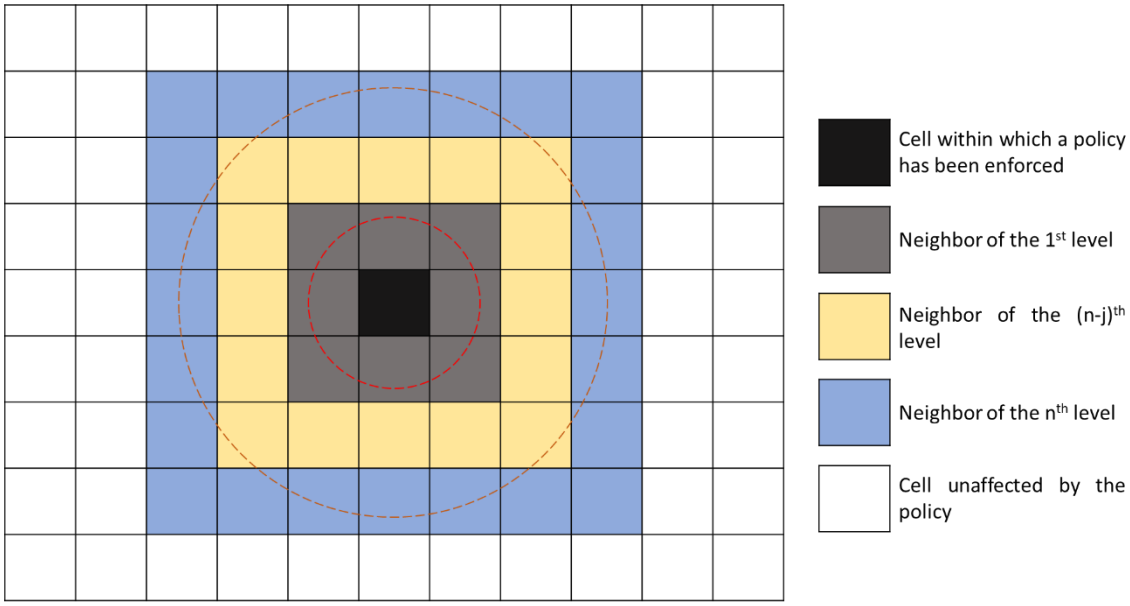


Figure 14: Neighbours notation

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 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 3.7.0. The dependencies of the code are as follows:

- Networkx library
- Numpy library
- Pandas library
- Matplotlib.pyplot library

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

- User_Input: is the class including the code for retrieving all the required input from the user.
  - user_input: is the function through which the user is prompted to provide input to the tool.
  - check_user_input_dimensions: is the function which checks whether the number of cells for which the demand and capacity is provided is the same with the number of nodes of the equivalent graph.
  - check_user_input: is the function which checks whether the format of the user input is valid and as expected.
  - select_area_percentage_expansion_level: is the function through which the user provides information about the cells in which a policy parking restriction will be applied, as well as, the percentage of parking reduction capacity and the level of the policy expansion effect.
- Add_Attributes_to_Graph: 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).
- Neighbors: is the class including the code for identifying both the saturated and unsaturated neighbors of a graph's node.
  - neighbors_found: is the function which identifies the neighbors that a node of the graph has.
  - none_active_neighbors: is the function which identifies the saturated neighbors that a node of the graph has.

- active_neighbors_found: is the function which determines the unsaturated neighbors a node of the graph has, based on the difference between the initial demand and capacity values.
- active_neighbors_found_new: is the function which determines the unsaturated neighbors 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_neighbors_found_new_mulitple: is the function which determines the unsaturated neighbors 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.

- Calculations: is the class through which the distribution of the excess demand is applied to the neighbors 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.
  - two_areas_selected: is the function which executes either one_closed_area_calculations or one_area_selected_common functions and updates the demand and capacity values after the implementation of the parking restriction policy on each policy effect expansion level.
  - one_closed_area_calculations: is the functions which distributes the demand to the neighbors of a node or n number of nodes when the nodes are both in the neighbors list of each one.
  - one_area_selected_common: is the functions which distributes the demand to the neighbors of a node
  - add_demand_to_active_nodes: is the function through which both the distributed demand and the weighted distributed demand are added to the unsaturated nodes of the graph.
- Main: is the function in which all the necessary processes are conducted.

The results of the code quality control per class, by utilizing the methodology described in Section 3.2, are included in Tables 8, 9, 10, and 11.

Table 8: Code quality control results regarding the first class of the 4th tool (User_Input)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	user_input function	CC index	A (2)	low - simple block
	check_user_input_dimensions function	CC index	A (2)	low - simple block
	select_area_percentage_expansion_level function	CC index	C (13)	moderate - slightly

				complex block
Maintainability	User_Input class	Maintainability index	A (66.58)	Very high maintainability
Raw metrics	User_Input class	LOC	188	total # lines of code
	User_Input class	LLOC	81	total # logical lines of code[9]
	User_Input class	SLOC	80	total # source lines of code[10]
	User_Input class	comments	21	total # comment lines
	User_Input class	multi	4	total # lines representing multi-line strings
	User_Input class	blank	84	total # blank lines
Hal metrics	User_Input class	n_1	9	total # distinct operators[11]
	User_Input class	n_2	18	total # distinct operands[12]
	User_Input class	N_1	13	total # operators
	User_Input class	N_2	26	total # operands
	User_Input class	Vocabulary	27	$n = n_1 + n_2$
	User_Input class	Length	39	$N = N_1 + N_2$
	User_Input class	calculated_length	103.59	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	User_Input class	Volume	185.44	$V = N \log_2(n)$

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

	User_Input class	Difficulty	6.50	D = (n_1/2)x (N_2/n_2)
	User_Input class	Effort	1205.36	E = D x V
	User_Input class	Time	66.97	T = E/18 seconds
	User_Input class	Bugs	0.06	B = V/3000

The cyclomatic complexity of the User_Input class is ranked as A (5.0). Hence, it can be considered as a well-structured and stable piece of code.

Table 9: Code quality control results regarding the second of the 4th tool (Add_Attributes_to_Graph)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	add_diagonal_edges_new function	CC index	A (5)	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
Maintainability	Add_Attributes_to_Graph class	Maintainability index	A (72.10)	Very high maintainability
Raw metrics	Add_Attributes_to_Graph class	LOC	124	total # lines of code
	Add_Attributes_to_Graph class	LLOC	48	total # logical lines of code[13]
	Add_Attributes_to_Graph class	SLOC	48	total # source lines of code[14]
	Add_Attributes_to_Graph class	comments	18	total # comment lines
	Add_Attributes_to_Graph class	multi	4	total # lines representing multi-line strings

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

	Add_Attributes_to_Graph class	blank	54	total # blank lines
Hal metrics	Add_Attributes_to_Graph class	n_1	5	total # distinct operators[15]
	Add_Attributes_to_Graph class	n_2	23	total # distinct operands[16]
	Add_Attributes_to_Graph class	N_1	25	total # operators
	Add_Attributes_to_Graph class	N_2	50	total # operands
	Add_Attributes_to_Graph class	Vocabulary	28	$n = n_1 + n_2$
	Add_Attributes_to_Graph class	Length	75	$N = N_1 + N_2$
	Add_Attributes_to_Graph class	calculated_length	115.65	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Add_Attributes_to_Graph class	Volume	360.55	$V = N \log_2(n)$
	Add_Attributes_to_Graph class	Difficulty	5.44	$D = (n_1/2)x (N_2/n_2)$
	Add_Attributes_to_Graph class	Effort	1959.52	E = D x V
	Add_Attributes_to_Graph class	Time	108.86	T = E/18 seconds
	Add_Attributes_to_Graph class	Bugs	0.12	B = V/3000

The cyclomatic complexity of the Add_Attributes_to_Graph Class is ranked as A (5.0). Hence, it can be considered as a well-structured and stable piece of code.

Table 10: Code quality control results regarding the 4th tool (Neighbors Class)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	neighboors_found function	CC index	A (3)	low - simple block
	none_active_neighboors function	CC index	C (12)	moderate - slightly

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

				complex block
	active_neighboors_found function	CC index	A (3)	low - simple block
	active_neighboors_found_new_mulitple function	CC index	A (4)	low - simple block
	active_neighboors_found_new function	CC index	A (3)	low - simple block
Maintainability	Neighbors class	Maintainability index	A (63.79)	Very high maintainability
Raw metrics	Neighbors class	LOC	183	total # lines of code
	Neighbors class	LLOC	75	total # logical lines of code[17]
	Neighbors class	SLOC	74	total # source lines of code[18]
	Neighbors class	comments	20	total # comment lines
	Neighbors class	multi	4	total # lines representing multi-line strings
	Neighbors class	blank	85	total # blank lines
Hal metrics	Neighbors class	n_1	11	total # distinct operators[19]
	Neighbors class	n_2	33	total # distinct operands[20]
	Neighbors class	N_1	30	total # operators
	Neighbors class	N_2	60	total # operands

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

	Neighbors class	Vocabulary	44	$n = n_1 + n_2$
	Neighbors class	Length	90	$N = N_1 + N_2$
	Neighbors class	calculated_length	204.52	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Neighbors class	Volume	491.35	$V = N \log_2(n)$
	Neighbors class	Difficulty	10.0	$D = (n_1/2)x (N_2/n_2)$
	Neighbors class	Effort	4913.49	E = D x V
	Neighbors class	Time	272.97	T = E/18 seconds
	Neighbors class	Bugs	0.16	B = V/3000

The cyclomatic complexity of the Add_Attributes_to_Graph Class is ranked as B (6). Hence, it can be considered as a well-structured and stable piece of code.

Table 11: Code quality control results regarding the 4th tool (Calculations Class)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	Calculations function	CC index	B (6)	low - well-structured and stable block
	one_area_selected function	CC index	C (12)	moderate - slightly complex block
	two_areas_selected function	CC index	E (38)	high - complex block, alarming
	found_nodes_with_specific_demand function	CC index	A (3)	low - simple block
	one_closed_area_calculations function	CC index	F (44)	very high - error-prone, unstable block
	add_demand_to_active_nodes function		B (7)	low - well-structured and stable block
	one_area_selected_common function		A (3)	low - simple

				block
Maintainability	Neighbors Class	Maintainability index	A (23.63)	Very high maintainability
Raw metrics	Neighbors Class	LOC	671	total # lines of code
	Neighbors Class	LLOC	314	total # logical lines of code[21]
	Neighbors Class	SLOC	311	total # source lines of code[22]
	Neighbors Class	comments	29	total # comment lines
	Neighbors Class	multi	4	total # lines representing multi-line strings
	Neighbors Class	blank	327	total # blank lines
Hal metrics	Neighbors Class	n_1	15	total # distinct operators[23]
	Neighbors Class	n_2	165	total # distinct operands[24]
	Neighbors Class	N_1	156	total # operators
	Neighbors Class	N_2	315	total # operands
	Neighbors Class	Vocabulary	180	$n = n_1 + n_2$
	Neighbors Class	Length	471	$N = N_1 + N_2$
	Neighbors Class	calculated_length	1274.05	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Neighbors Class	Volume	3528.66	$V = N \log_2(n)$

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

	Neighbors Class	Difficulty	14.32	D = (n_1/2)x (N_2/n_2)
	Neighbors Class	Effort	50524.04	E = D x V
	Neighbors Class	Time	2806.89	T = E/18 seconds
	Neighbors Class	Bugs	1.18	B = V/3000

The cyclomatic complexity of the Calculations Class is ranked as C (17). Hence, it can be considered as a well-structured and stable piece of code that may be further enhanced in the future.

4.4.7 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 12-15). Table 12 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 13, on the other hand, includes both the demand- and capacity-side inputs of the above mentioned (4*4) area. Finally, Table 14 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 neighbors of these cells.

Table 12: 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 13: 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 14: 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 15-17. Table 15 provides the average searching time induced by all road users before the enforcement of the restriction policy. Table 16 presents the outputs of the tool in terms of distributing the excess demand into the neighbor areas of Areas 3-4. It is noted that the demand to capacity ration in the neighbor 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 neighbor areas to neighbors of the n^{th}

level. Finally, Table 17 presents the updated average searching time induced by all road users after the enforcement of the restriction policy.

Table 15: 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 16: 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 1717: 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 bast test 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 18 presents the outputs of the tool in terms of distributing the excess demand into the neighbor areas of Areas 3-4, this time including 2nd level neighbors (highlighted with light blue color). Similarly, Table 19 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 neighbors.

Table 18: 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 19: 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 utilizes 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 15 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 17 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[25].

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

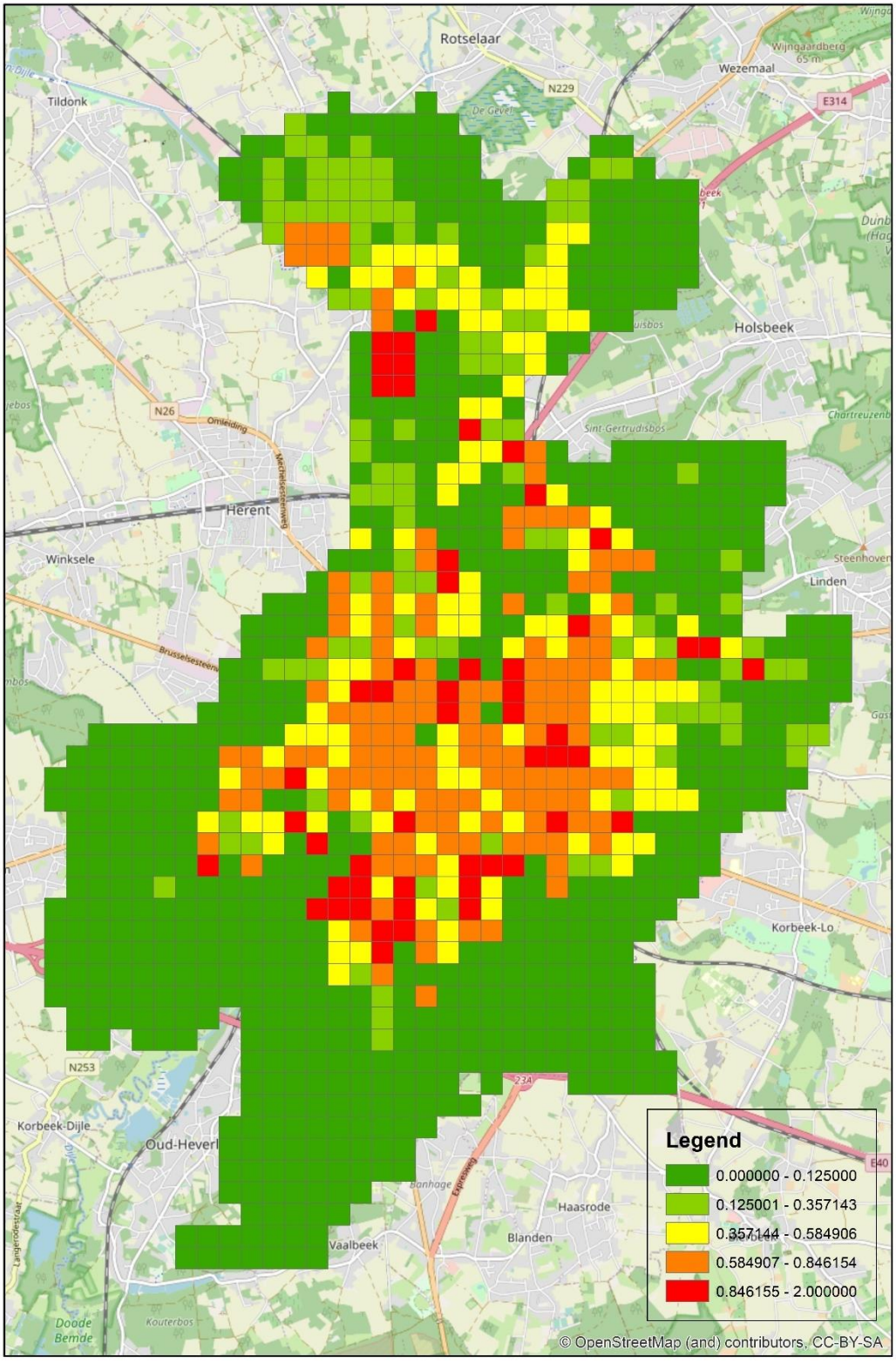


Figure 15: Parking demand to capacity ratio during the morning rush hour in Leuven.

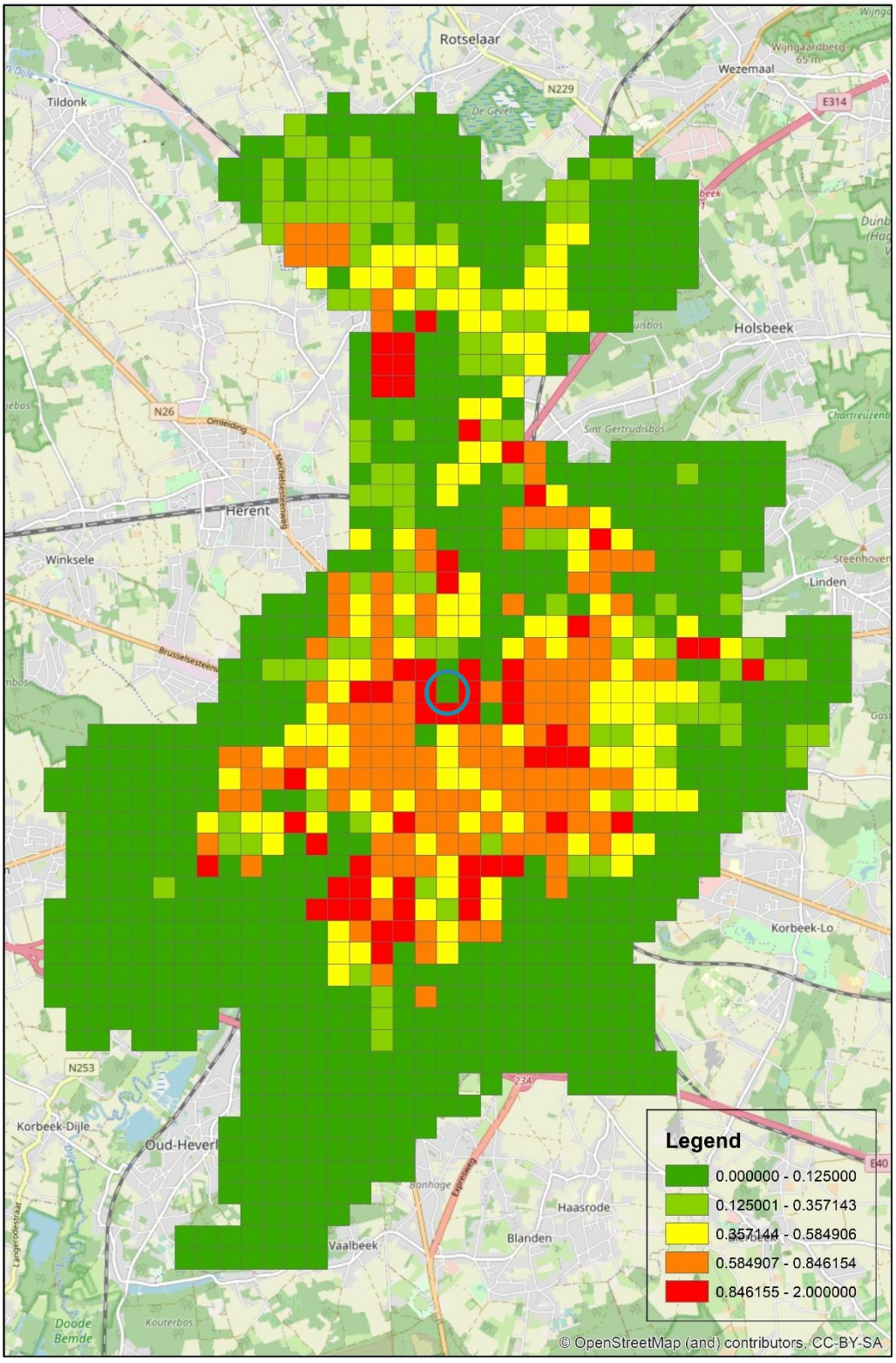


Figure 16: Updated parking demand to capacity ratio after the enforcement of a restriction policy in the city centre of Leuven (2nd test scenario of the fourth tool)

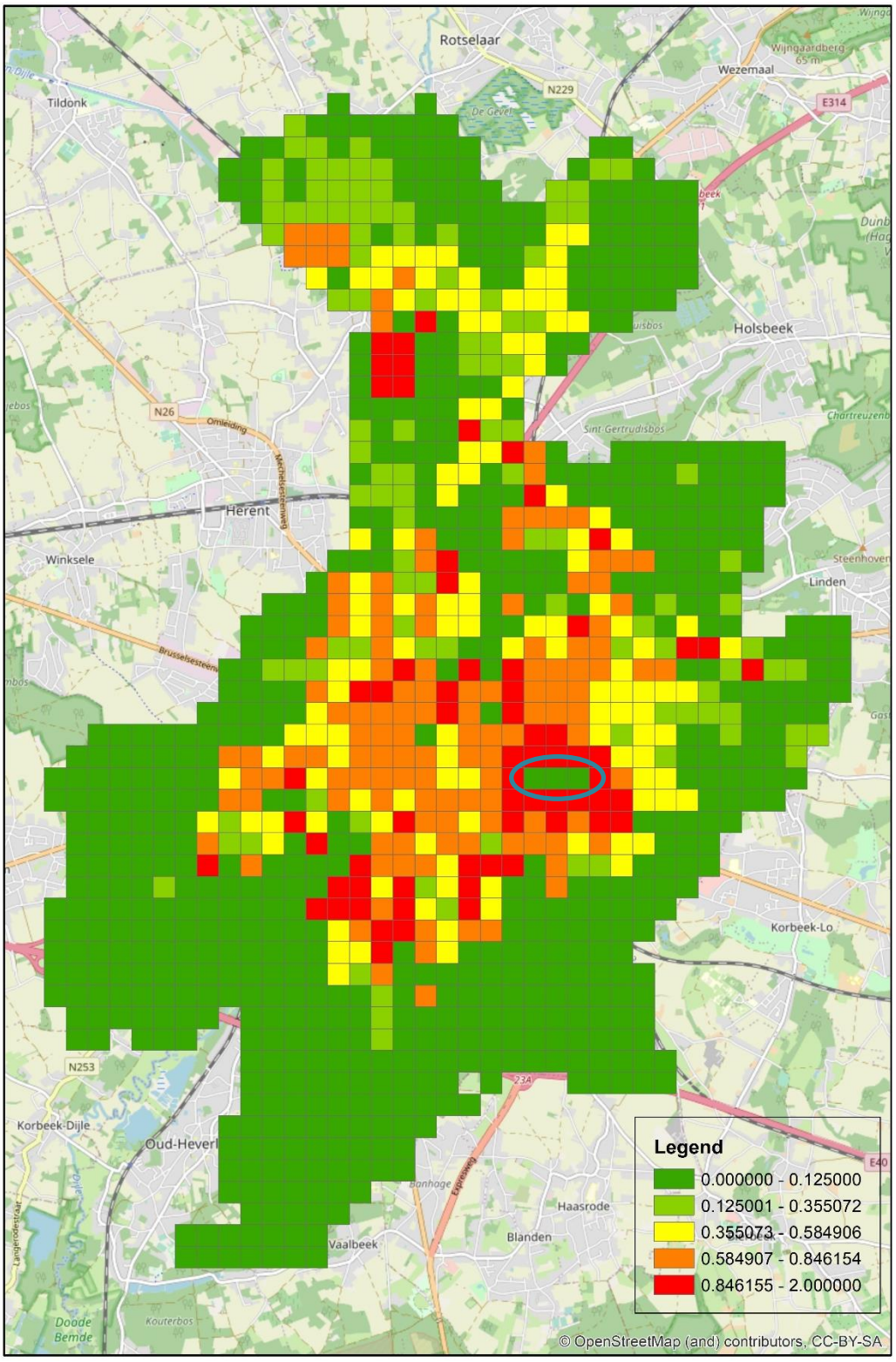


Figure 17: Updated parking demand to capacity ratio after the enforcement of a restriction policy in Leuven East (2nd test scenario of the fourth tool)

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.

The algorithmic framework of the current tool is presented to its full extent. However, for data availability completeness issues its application takes places upon sample data stemming from the pilot city of Thessaloniki. Its full application will be based on the data stemming from the pilot city of Barcelona, wherein ANPR systems are installed and operated. Sample data from the pilot city of Thessaloniki are also based on point-to-point detections systems, i.e., Bluetooth detectors.

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 (policymakers)
- Transport planners supporting policymakers

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 prioritization of relevant investments (e.g., development of park & ride facilities).

4.5.3 Data inputs

In contrast to the tools described in the previous sections, the input provided to the current tool only include values imported from a file or database. Such input includes the following:

- 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)

The first input mentioned above is provided once during the initialization of the current's tool operation. The second input mentioned above is provided in the tool in a real-time or tactical basis depending on the desired time scale of the produced OD table. Finally, the third input mentioned above is addressed as semi-static and required to be updated at acceptable time intervals.

4.5.4 Data outputs

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

- Skim Matrix
- OD table
- 3D OD table (i.e., augmented with vehicle-related information derived from an external database)
- Data Analytics

4.5.5 Computational flow

The computational flow of the fifth tool is depicted in Figure 18. 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 analyzed 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[26]. 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 tested, 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 utilized 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 time frame of which is within the limits defined by the skim matrix. After the identification of the time feasible trips, the 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 utilize data included in an external database to create a 3D OD table that will include the number of trips between each

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

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 prioritization of relevant investments.

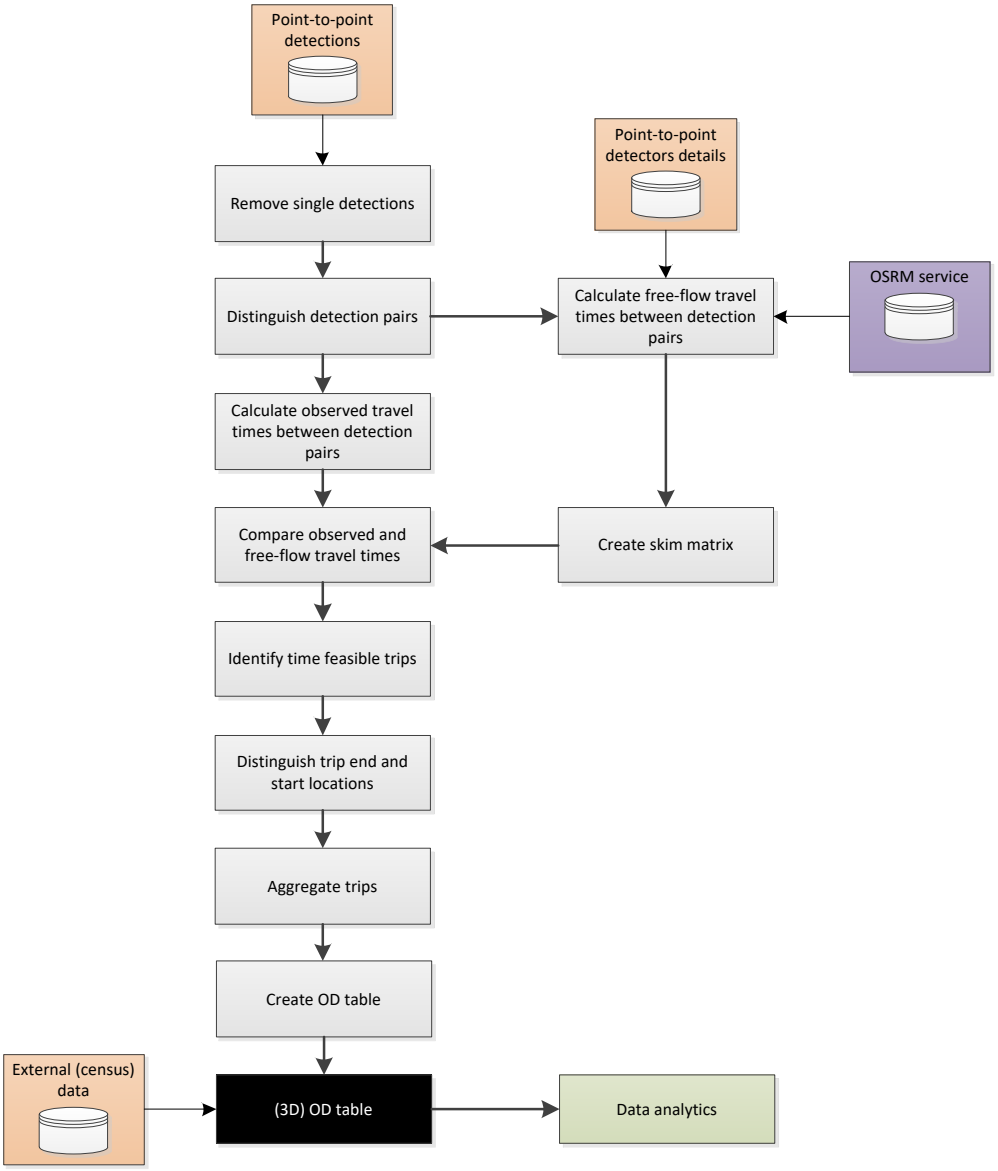


Figure 18: Computational flow of the fifth tool of nuMIDAS toolkit

4.5.6 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 3.7.0. The dependencies of the code are as follows:

- Math library
- Numpy library
- Pandas library
- Requests library

- Json library
- Sklearn scikit-learn library

The fifths 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.
- user_input: is the function which prompts the user to select one of the available operations (identify and remove single detections, distinguish camera pairs and calculate observed travel times, estimate travel times, create OD table, calculate 3D OD table and perform data analytics) and checks whether the format of the user input is valid and as expected.
- convert_to_minutes: is the function which takes as input a value in hours:minutes:seconds format and converts it to minutes format.
- remove_single_detections_to_all: is the function which takes as input all point-to-point detection records for several days and by using vehicle (pseudo-anonymized) identifiers deletes records corresponding to vehicle that have been detected only one time during each day.
- identify_camera_pairs_and_collect_travel_times: is the function which takes as input the edited/cleaned point-to-point detection records for several data, identifies for each day the detection pairs, as well as collects the observed travel times between these pairs.
- travel_times_osrm: is the function which takes as input the identified point-to-point detection pairs and, through information related to the geographical location of detectors, a query is made to the OSRM service with the aim of retrieving the free-flow-based travel time between each pair. The retrieved values are inserted in a skim matrix the values of which are scaled up to account for congestion effects.
- calculate_od_matrix: is the function which takes as input both the edited point-to-point detection records and the travel times which have been estimated through the previous function. Based on this input, the function identifies the time feasible paths for each detected vehicle and creates an Origin-Destination (OD) table.

The results of the code quality control, by utilizing the methodology described in Section 3.2, are included in Table 21.

Table 20: Code quality control results – 5th tool (UC5)

Category	Evaluation element	Metric	Rank (Score)	Interpretation
Cyclomatic Complexity (CC)	Main function	CC index	A (5)	low - simple block
	user_input function	CC index	A (1)	low - simple block
	convert_to_minutes function	CC index	A (1)	low - simple block
	remove_single_detections_to_all function	CC index	C (19)	moderate -

Category	Evaluation element	Metric	Rank (Score)	Interpretation
				slightly complex block
	identify_camera_pairs_and_collect_travel_times function	CC index	B (10)	low - well-structured and stable block
	travel_times_osrm function	CC index	B (8)	low - well-structured and stable block
	calculate_od_matrix function	CC index	D (30)	more than moderate - more complex block
Maintainability	Entire UC5 .py file	Maintainability index	A (29.95)	Very high maintainability
Raw metrics	Entire UC5 .py file	LOC	754	total # lines of code
	Entire UC5 .py file	LLOC	352	total # logical lines of code[27]
	Entire UC5 .py file	SLOC	351	total # source lines of code[28]
	Entire UC5 .py file	comments	35	total # comment lines
	Entire UC5 .py file	multi	4	total # lines representing multi-line strings

[27] Logical lines of code may be defined as lines of code including executable expressions.
[28] 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	365	total # blank lines
Hal metrics	Entire UC5 .py file	n_1	19	total # distinct operators[29]
	Entire UC5 .py file	n_2	153	total # distinct operands[30]
	Entire UC5 .py file	N_1	107	total # operators
	Entire UC5 .py file	N_2	207	total # operands
	Entire UC5 .py file	Vocabulary	172	$n = n_1 + n_2$
	Entire UC5 .py file	Length	314	$N = N_1 + N_2$
	Entire UC5 .py file	calculated_length	1191.09	$n_1 \log_2(n_1) + n_2 \log_2(n_2)$
	Entire UC5 .py file	Volume	2331.85	$V = N \log_2(n)$
	Entire UC5 .py file	Difficulty	12.85	$D = (n_1/2)x (N_2/n_2)$
	Entire UC5 .py file	Effort	2997.09	E = D x V
	Entire UC5 .py file	Time	1665.06	T = E/18 seconds
	Entire UC5 .py file	Bugs	0.78	B = V/3000

The average cyclomatic complexity of the UC1 code file is ranked as B (6.125). Hence, it can be considered as a well-structured and stable piece of code.

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

4.5.7 Demonstration and testing

Demonstration and testing in the context of current tool is achieved through sample data stemming from the pilot city of Thessaloniki. These data include 557.180 records from Bluetooth detectors covering a typical workday (07/02/2022). After removing single detections, the number of records gets equal to 543.278. The derived OD table includes 1.328 OD pairs and 78.638 trips. Figures 19-20 depict the desire lines from two specific detector locations[31] to the entirety of detector locations.

The full extent of the current tool's outputs will be provided in the next version of the current deliverable building upon data stemming from the pilot city of Barcelona, wherein ANPR systems are installed and operated.

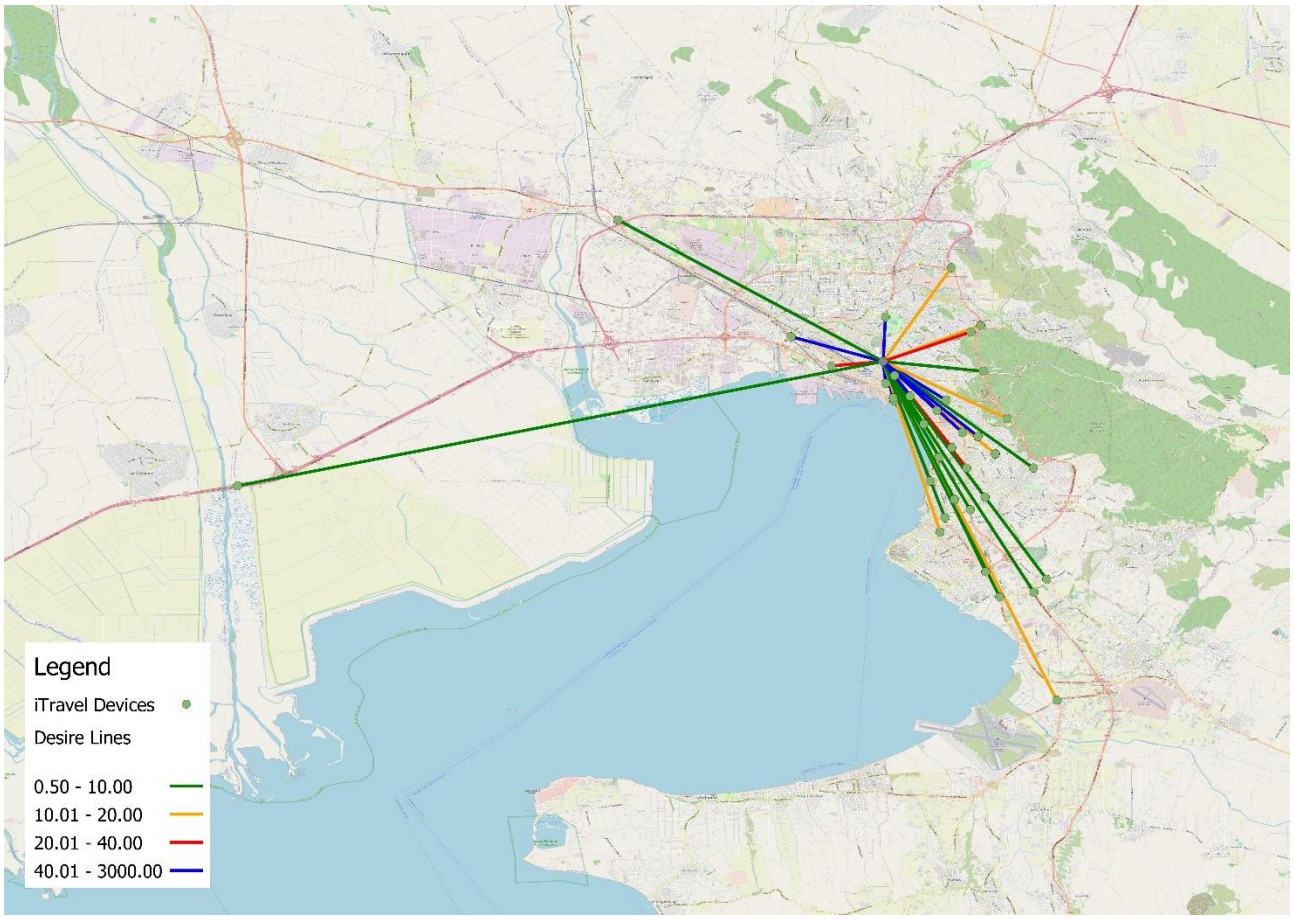


Figure 19: Desire lines derived from the OD table estimated by using data from the pilot city of Thessaloniki (O: detector installed within the city centre, D: all installed detectors)

[31] The first location is within the city centre, while the second location is the Airport of Thessaloniki (SKG).

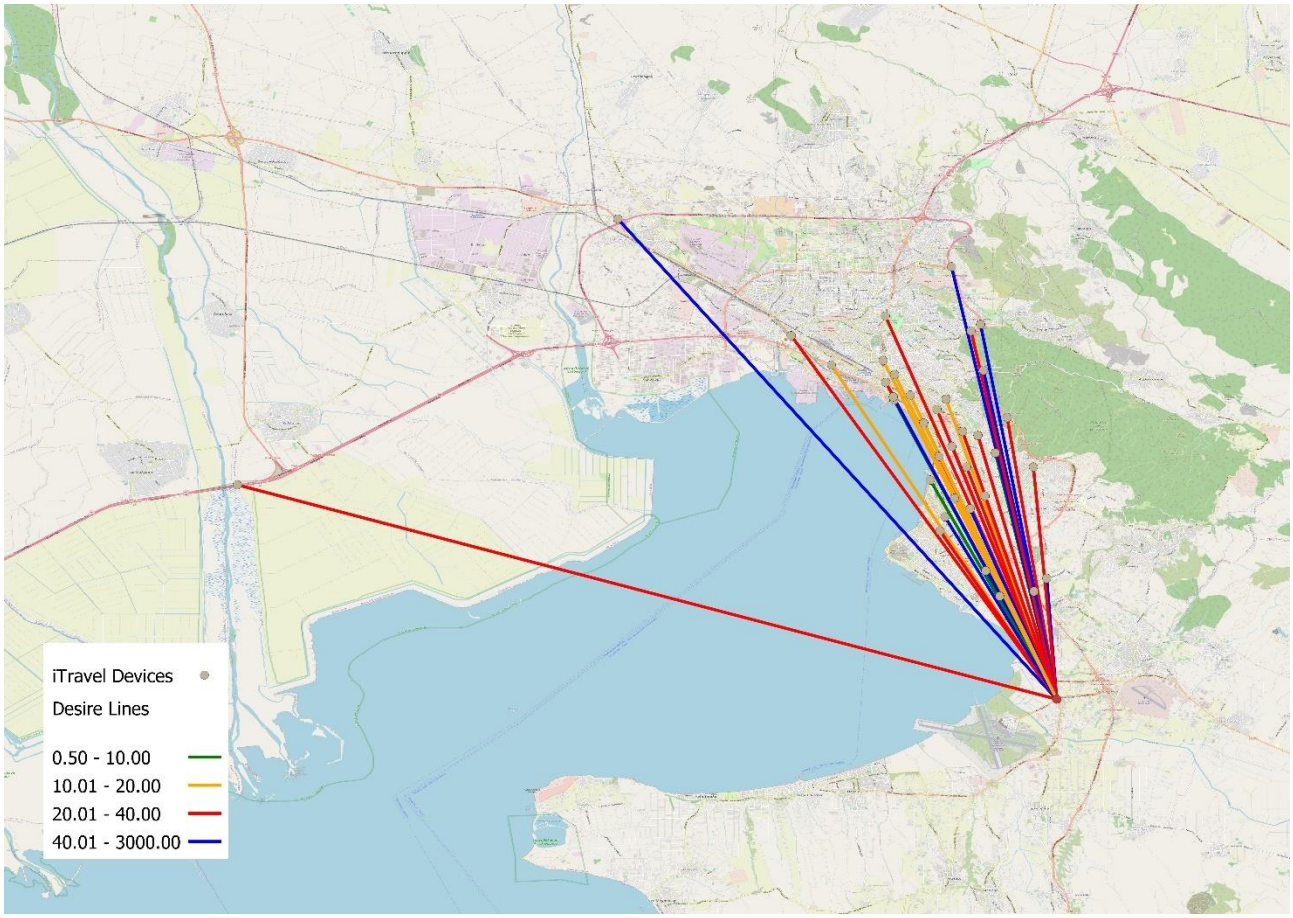


Figure 20: Desire lines derived from the OD table estimated by using data from the pilot city of Thessaloniki (O: detector installed at SKG, D: all installed detectors)

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 different traffic management scenarios enabled by intelligent transport systems and capable of improving the road conditions. The scope of the sixth tool to be integrated into the nuMIDAS toolkit involves the data-driven assessment of traffic management scenarios, including both conventional traffic management measures (e.g., traffic lights control or provision of information through Variable Message Signs) as well as novel and advanced technologies, such as C-ITS enabled dynamic traffic management (e.g., personalised provision of dynamic warnings, information, and guidance to drivers of connected vehicles).

The information utilized by traffic managers may be divided into (a) incident-related, (b) weather-related, (c) speed-related, (d) field device-related, (e) work zone-related, and (f) event-related. 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 proactive 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 tested in several circumstances, while in the latter the strategies and measures are by nature more experimental. The entirety of information and traffic management approaches mentioned above will be considered by the sixth component of the nuMIDAS toolkit.

4.6.2 Targeted users

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

- Data providers (incl. ITS service providers)
- Department of a municipality responsible for the enforcement of traffic restriction policies (policymaker)
- Traffic Management Centers
- Transport planners supporting policymakers

4.6.3 Data inputs

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

- Database of traffic management scenarios (including trigger and target KPI values)
- Currently active traffic management scenario
- Historical data concerning speed, traffic flow, and travel time
- Real-time or synthetic data concerning speed, traffic flow, and travel time

4.6.4 Data outputs

The outputs of the tool (indicatively) include:

- KPIs indicating the performance of activated traffic management scenarios

5 Concluding remarks

This deliverable aims to describe the 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. For the methods and tools that are in full development a description of their computational flow is outlined. In the context of the relevant subsections, an illustrative diagram is delivered emphasizing 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 showcase that the methods and tools in full development perform as expected and produce rational results. The content of this document will be enriched in its next version by repeating the analysis stated above for the entirety of the toolkit's components.

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.