POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

ALGORITHMS AND METHODS FOR THE DESIGN AND DEVELOPMENT OF INTELLIGENT, CONTEXT AWARE AND SUSTAINABLE MOBILITY SERVICES

Doctoral Dissertation of:
Alessio Pagani

Supervisor:
Prof. Francesco Bruschi

Tutor:
Prof. Donatella Sciuto

The Chair of the Doctoral Program:
Prof. Andrea Bonarini

2017 – Cycle XXX

Abstract

PUBLIC transport plays at least two key roles in the structure and functioning of a city: it enables mobility for everyone, besides their availability of a private means of transport, optimizing the economical, social and environmental cost of movement. In this context, the interest towards the applications of ICT in public and private urban transport has grown significantly over the last few years. Ideally, users should be provided with the most precise and up-to-date information at any given time, with the most accurate forecasts about their travel, and should be assisted in making real-time decisions maximizing their satisfaction (which is, in turn, a complex function of kinematic measurable and psychological perception). In the field of transit design and planning, however, transport agencies do not exploit the inherent potential of the increasing volume of data being generated by user devices and applications and work mostly in open loop when planning services. In the field of user interfaces with transportation, on the other hand, continuous, highly context-aware, real-time interaction can still be found only in a very limited number of cases, mostly in private transportation. One of the main issues in actually developing both data-driven, closed loop planning tools and assistive, portable, continuously interacting applications is getting to know the transports system state (equations of motion of the means, position of the users on the means). In the majority of cities the most temporally accurate data currently available is yet only the estimated departure time of the next train or bus at the stops.

Both these informational dimensions are being thoroughly revolutionized, in their perspectives and actual possibilities, by the flood of digitally available data concerning the transport operation and user state.

There is a relevant and potentially disruptive consequence of this perspective, when taken to the extreme: if adequate relevant information is available to transport agencies in real-time, it will be possible for them to dynamically adjust service parameters, and immediately assess their effects, in order to dynamically improve service quality in a closed-loop fashion. This would open up the possibility, among the others, to exploit self-adjusting behaviors that perform small variations (experiments) on transit schedule parameters, seeking to optimize some service metrics (e.g., mean travel time) at the

lowest cost possible, in real-time.

One noteworthy and remarkable observation, that is central for the motivation of the present work, is that the information to be provided to the users and the one for the transit agencies are far from being disjoint. In both cases, the ability to precisely determine the users travel context and in particular to localize them on identifiable means of transportation, is pivotal.

Aim of this thesis is to propose a general framework able to provide all the information required to develop multimodal, real-time, continuous and context-aware applications. From a transit agency perspective, it could be possible to evaluate metrics valuable to duly assess service quality, such as origin-destination matrices annotated with mean travel time, number of connections taken, mean waiting times at connections, number of passengers that traveled between two points in a given time interval.

Summary

THE quantity of data generated by public transport agencies, users and public transport oriented applications is constantly growing. This information is not fully exploited though, especially in the public transports field: just a few of the existing applications make use of this information and no service is able to offer a dynamic, real-time, individually-customized interaction.

Aim of the research is to exploit all the information already available in each city, such as open data, transport means timetables, actual arrival times upon each stop, crowd-sourced data and social network news feeds, in order to design innovative tools and services for the public transportation. Projectual aim of the work is the design of a *Public Mobility Platform*, a set of methods and tools to gather all the possible information available and process it.

The **Public Mobility Platform** manages all the information about the public mobility, such as the vehicles, the traveling users, the state of the transportation network and road network. This platform allows the design of user oriented planning and assistance services increasing the perceived quality of the public transport service and offering a better travel experience, reducing traveling time and improving perceived comfort, even in case of disruptions. Furthermore, this innovative platform can also supply public transport agencies with detailed information such as number of traveling passengers, commutation times and routes.

The first addressed problem revolves around the reconstruction of the position and the estimated motion law of the different means of transportation in a given area, starting from available information, such as ETA at the stations and users position sampling. To solve the problem, a framework using a state-based Bayesian approach to the reconstruction of the transit system state from limited publicly available knowledge was designed. The system is general and effective also in the presence of various real world kinds of noise, such as information blackouts.

To enable the most advanced proposed features, the Public Mobility Platform has a specific module to seamlessly exploit location of users to estimate which means they are on-board. Obtaining this information poses two main challenges. The first one is the estimated association of generic agents (the users) to the vehicle they are rid-

ing, taking into account noise, limited precision of the sensors available (e.g., GPS, accelerometers) and possible presence of multiple means of transport in the same street or area. The second challenge is understanding the travel state of the agents, that is if they are actually on the vehicle or not (e.g., they are just walking on the street close to the bus). The proposed solution is based on a Particle Filter with a model that includes the probability for each user to be on a selected vehicle. This solution is effective in presence of noisy data and identifies correctly when the users are actually on a vehicle or not. Thanks to underlying model, this approach is also convincing when it has to identify on which specific vehicle each user is traveling. This information is of great value for the transit agencies and it is essential to develop innovative social applications and interactions.

To validate the algorithms and the designed platform an ad hoc simulator of the public transport network was developed. The system was tested and validated using the means of transport simulator and, as a case of study for real applications, it was used to reconstruct the state of the means of transport of the City of Milan using arrival times data provided by the Milan public transport consortium.

A further improvement was achieved integrating in the Public Mobility Platform a Knowledge Discovery from Data system that first gathers data from car sharing sites and applications, and then processes them to estimate interesting metrics such as travel time and vehicle flows in the urban areas at different times and in different days. The information gathered can be processed in real-time, to estimate instant traffic, and can be exploited to perform deeper analysis, using historical data. This information was also used to analyze vehicles availability as a function of time in different zones and to show how the results can be applied to travel time estimation, car stockout forecast and multimodal travel planning.

A further experimental module of the platform was designed to predict urban mobility scenarios and to explain them. The main topic is relative to the private public transportation (e.g., taxis and car-sharing) and includes two main techniques based on explainable deep learning: the first one is focused on the design of interpretable deep neural networks using features engineering techniques to design feature-based neural networks. The second one revolves around the automatic detection of geographical features using a deep neural network based on car routes input.

The platform and the designed services were released as a set of API and web applications and were then tested in real case scenarios, in collaboration with the TIM Joint Open Lab.

Contents

1 Introduction **1**
- 1.1 User Information Needs . . . 1
- 1.2 Services and Information . . . 2
- 1.3 Gamification and Social Applications . . . 3
- 1.4 Data-driven Intelligent Planning . . . 4
  - 1.4.1 Data Availability . . . 4
- 1.5 M.I.E. Project . . . 5
- 1.6 Thesis Structure . . . 5

2 State of the Art **7**
- 2.1 Public Mobility State . . . 8
  - 2.1.1 Definition of *Public Mobility State* . . . 8
  - 2.1.2 Public Transport State Reconstruction . . . 9
  - 2.1.3 User contextualization . . . 10
- 2.2 Arrival Time Unreliability . . . 11
- 2.3 Existing Applications . . . 12
  - 2.3.1 Public Transport Trip Planners . . . 12
  - 2.3.2 Road State Reconstruction Applications . . . 16

3 Public Mobility Platform **19**
- 3.1 Conceptual Framework . . . 19
- 3.2 Data Fusion . . . 20
- 3.3 Public Transport Network . . . 21
- 3.4 Road network . . . 21
- 3.5 User Contextualization . . . 22
- 3.6 Network Features Classification . . . 22
- 3.7 APIs . . . 22
- 3.8 Communication Infrastructure . . . 22
- 3.9 Simulator . . . 23
- 3.10 Platform Implementation . . . 24
  - 3.10.1 Data Used in the Current Implementation . . . 24

3.10.2 Docker Containers 24

4 Data Collection Techniques and Definition of the Required Data Structures 27
4.1 Creation of the Transport Network Graph 28
4.1.1 Representation of the Public Transport Vehicles on a Time-Expanded Graph 29
4.2 Static Timetables 30
4.3 Real-Time Waiting Times 31
4.4 Car Sharing 31
4.4.1 Data sources 31

5 Reconstruction of the Public Transportation Network State 33
5.1 Proposed Model 33
5.1.1 Initialization 34
5.1.2 Definition of the Network State 34
5.1.3 Model 35
5.1.4 Emitting and absorbing stations 35
5.2 Kalman Filter 36
5.2.1 Unscented Kalman Filter Prediction 36
5.2.2 Unscented Kalman Filter Correction 37
5.2.3 Vehicles Arrival Time Prediction 37

6 Knowledge Discovery from Car Sharing Data for Traffic Flows Estimation 39
6.1 Description 40
6.2 Data Gathering 40
6.3 Implementation 41
6.4 Data aggregation 41
6.5 Data processing 41

7 User Contextualization 43
7.0.1 Definition of the User State 43
7.0.2 Implementation 44
7.0.3 Resampling 45

8 Classification of the Network Features 47
8.1 Methodology 48
8.1.1 Features Engineering 48
8.1.2 Network Based Input 48
8.2 Network Topology 49
8.2.1 Selecting number of layers and neurons 49

9 Time of Arrival Cumulative Probability in Public Transportation Travel Assistance 51
9.1 General Approach 52
9.2 Probabilities Definiton 54
9.2.1 Probability to Catch a Vehicle 54
9.2.2 Arrival Time Probability 57
9.2.3 Monte Carlo Simulation 57
9.3 Fields of Application 57

9.4 Use Cases . . . 58
9.4.1 Routes with Low Frequency Lines . . . 58
9.4.2 Numerical Example . . . 58

10 Analysis, Experimental Results and Real Case Scenarios **61**
10.1 Public Transport Network State Reconstruction Module . . . 61
10.1.1 Experimental results . . . 62
10.1.2 Arrival Times Predictions . . . 63
10.2 User Contextualization Module . . . 64
10.2.1 Experimental Results . . . 65
10.3 Real Scenario Full Stack Experiments . . . 67
10.3.1 Robustness of the Approach . . . 68
10.4 Road Network State Reconstruction Module . . . 68
10.4.1 Data aggregation . . . 69
10.5 Network Behaviors Classification . . . 72
10.5.1 Taxi Dataset . . . 73
10.5.2 Modular Network Experiments . . . 76
10.5.3 Network Based Input Experiment . . . 77

11 Applications **81**
11.1 Trip Planning and Assistance Application . . . 81
11.1.1 Geocoding . . . 82
11.1.2 Routes Calculation . . . 82
11.1.3 Description of the Optimal Route Computation Algorithm . . . 82
11.2 Development of Engagement Mechanisms and Gamification Techniques to Ease the Interaction with Users . . . 85
11.2.1 Gamification . . . 85
11.2.2 Gamification in the Transport System . . . 86
11.3 Traffic Flows Estimation . . . 87
11.4 Cars Stockout Detection . . . 89

12 Conclusions **91**
12.1 Further Work . . . 93

Bibliography **95**

CHAPTER 1

Introduction

1.1 User Information Needs

Travelers have different information needs that change when they are planning a trip and when they are undertaking it. Three different stages were identified in a journey, roughly in conformity with three location types: pre-trip, wayside, and on-board [34]:

Pre-trip : the user plans a travel that will start in hours or days. Planning a trip consists of exploring all the possible combinations to reach a given destination. The most significant data in this phase are the scheduled timetables.

Wayside : at this stage the user is at, or closed to, the public transport station. Real-time information becomes valuable, as it can be used for real-time planning and travel support.

On-board : the trip is started and the user is traveling. Real-time information would be maximally valuable, and the users theirself could be a source of information.

While different services are already available to satisfy the *pre-trip* information needs, a huge margin of enhancement appears to be present concerning the *real-time* needs of the two last phases.

Focus of the work is on the **on-board** needs: in other words supporting the users while they are on-board, the trip is started and they are traveling. Real-time information would be maximally valuable and the users themselves could be a source of information. The planner should consider more constraints such as the traffic status and disruptions made available in close-real-time and the choices already taken user (e.g., a new fastest route is available but the user is already on a bus). Travel assistance could include: alerts on actions to take (e.g., getting off the train at the next stop), changes in

estimated quantities (e.g., a drop in the likelihood to make a given connection, causing and update of the estimated time of arrival), current status of the travel (distance from destination, number of stops, updated arrival time variations, etc.), new route options, in case of disruptions/traffic jams or when better alternative paths arise.

At the present day, several services allowing to plan a trip based on scheduled timetables are available. Some of them also offers specific real-time information, *wayside* as well as *on-board* (an analysis of the current applications is presented in Section 2.3).

Notable features are missing, though. First of all, it is unclear whether existing applications make full use of real-time information when searching for the fastest path (or, rather, run search algorithms on scheduled timetables and then *annotate* the result with real-time info bits). Examples of potentially viable, missing features are related to the possibility of highly context aware social interactions. For instance, to clarify with an example, consider a metrics able to tell you how many of somebody's Facebook friends are on his same line at a given moment.

Moreover, operating data from travel assistance apps could be fruitfully exploited to provide transit agencies with precise metrics, such as the number of passengers that traveled between two points in a given time interval, how many connections they had, how much time they spent waiting for connections. Such information would allow to measure transit performances with unprecedented precision and effectiveness.

It is crucial to observe that to provide such level of informed real-time interaction it would be necessary to track the users down to the individual means of transport they are traveling on. As of now this information is not automatically available to the existing travel planning (Moovit [47] uses it in its *EnRoute* feature, but it seemingly requires users' active collaboration in specifying which means they are on).

To obtain this information, it is first necessary to know the state of the transit network, that is position, current and future, of all the traveling means. Attempts to obtain such precise users location, as described in [33], are based on instrumentation of the transit means with IoT equipment installed on board.

1.2 Services and Information

There is a clear need of **Advanced Traveler Information Services** (ATISs) [13] potentially resulting in mobile, multimodal, dynamic and personal travel information services.

The literature also suggest that travelers are particularly using ATISs to be informed about their favored mode of transport, but they wish to be informed not only about travel times and costs of alternatives but also about more *soft* characteristics such as convenience, comfort and privacy.

Here below an analysis of the information required by different significant transit information services.

Static trip planning can rely on scheduled timetables only. If so, it will be unable to compute the fastest route according to the current actual situation, nor to give the user real-time information.

Real-time annotated trip planning is slightly more powerful: the routing is made with the scheduled information, but the user is then fed with related real-time annotations (such as a delay in the departure time of a train).

Information required: estimated departure time of the means.

Real-time trip planning uses the current available real-time data to search the best traveling options. The service can for instance provide suggestions about faster routes suddenly become available.

Information required: estimated position, current and future, of all the transport means.

Advanced social features requires knowledge of the present position of the means, and of the position of the users on each means (the *contextualization* of users on means).

Feedback to transit agencies requires knowledge of past positions of both means and users.

Currently, the availability of transport real-time information is very uneven, varying greatly between cities and transport agencies. According to a census of the transit data available worldwide [72], agencies publishing static schedules also disclose estimated departure times (in variable formats) significantly often, but provide position of the means very seldom.

The reasons why positions are often not provided are multiple, including privacy, technical difficulties, and convenience: estimated departure time is much more valuable, to the users, than the position of the means, making their publishing seemingly unworth the effort.

Anyway, such lack is a barrier in defining and developing new services, like planners that use the real-time data, offer advanced social features, and generate meaningful feedback data for transit agencies.

Since the complete network state (present and future positions of means) is so enabling, it makes sense to devise ways to obtain it from more accessible information, such as estimated departure times.

In the proposed framework, a data fusion approach that aims at reconstructing the information needed to develop a new generation of ATISs from all the information publicly available (e.g., transit schedule, estimated departure times of the next means at the stops, roads speed, traffic flows) is proposed. The system reconstructs the equation of motion of the transit means and can be extended to take advantage of other observations, such as sensors of passage and sporadical measures of users positions, and to estimate the position of the users on the means.

1.3 Gamification and Social Applications

The concept of **gamification** implies the application of dynamics characterizing the gaming process (e.g., scores, levels, prizes) in actual contexts instead of ludic, in order to enhance commitment and competitively, stimulating to seek for solutions to real problems [41].

Gamification aims to involve people in everyday activities through interactive processes. It has many objectives and all of them can be reached by changing users routine and behaviors: through gamification, in fact, multiple results are achievable such as fidelization, recruiting, problem solving. Examples of possible gamification applications enabled by the framework proposed in this work can be: play Pokemon Go [43] and

catch Pokemon positioned specific vehicles, challenge passengers on the same means of transport at some games (e.g., play Ruzzle [40] with other passengers), leave messages in specific vehicles or stations.

In the mobile devices big-data field, the gamification techniques are a key tool to stimulate users to supply data and information both in a non-active way (e.g., just through the use of a smartphone app) and in an active one (e.g., when the user is asked to supply information through push services).

Contextualized social applications include all the new social applications that can be proposed in the transportation field using the estimated position of the user. These applications include sharing and organizing routes with friends (e.g., choose a route to meet one of your Facebook friends), share preferences with other passengers on the same means of transport.

1.4 Data-driven Intelligent Planning

Operating data from travel assistance apps could be fruitfully exploited to measure transit performances with unprecedented precision and effectiveness: for instance it would be possible to provide transit agencies with precise metrics, such as the commutation time, the number of passengers that traveled on a given line in a given time interval, how many cnnections they had, how much time they spent waiting for connections.

Using machine learning techniques, travel agencies will be able to analyze the data collected in order to improve the service offered to their users. Using Inverse Reinforcement Learning (IRL) techniques [2] it is possible to understand from data what are the motivations that drive users to choose some routes instead of others. Such information is really important to address what-if analyses [27], where a travel agency can predict the effect of introducing changes to its service (e.g., adding a transport line, or modifying an existing one, or raising the frequency of transport means) with greater accuracy than what can be achieved by standard simulators. The same approach can be exploited both in the offline scenario, where the travel agency introduces changes to improve the effectiveness of its service, and in real-time, where it is possible to intervene to manage unexpected issues on the transport network (e.g., dynamically increasing the bus frequency on a line when the number of users gets larger than normal).

1.4.1 Data Availability

Currently, the availability of transport real-time information is very uneven, varying greatly between cities and transport agencies. According to a census of the transit data available worldwide [72], agencies that publish static schedules also disclose estimated departure times (in variable formats) significantly often, but provide positions of the means rarely. The main reason is that estimated departure time is much more valuable to the users than the position of the means, making their publishing seemingly not worth the effort.

Anyway, such lack is a barrier in defining and developing new services, like planners that use the real-time data, offer advanced social features, and generate meaningful feedback data for transit agencies.

Since the complete network state (present and future positions of means) is so enabling, it makes sense to devise ways to obtain it from more accessible information,

such as estimated departure times.

In this work a data fusion approach is proposed. It aims at reconstructing, exploiting Bayesian methods, the information needed to develop a new generation of ATISs from all the information publicly available (transit schedule, estimated departure times of the next means at the stops, user position data), and to generate information that transit agencies can exploit to implement data-driven service scheduling both offline and in real-time.

1.5 M.I.E. Project

This research is part of the cluster project M.I.E. (*Mobilità Intelligente Ecosostenibile*) funded by MIUR within the actions of the Technological National Cluster on Technologies for the Smart Communities. In this context, the contribution is in the sustainable intelligent mobility systems field with the aim of defining analysis and design tools for the development of innovative transport services, in order to increase the level of information, reliability, quality of experience, integration of different modalities (public, private, shared vehicles) and social interaction among travelers. Aims of the M.I.E. project satisfied by the Public Mobility Platform are to provide all the information required to develop a multimodal, real-time, continuous and context-aware travel assistant and to provide transit agencies with precise metrics, such as the number of passengers that traveled between two points in a given time interval, how many connections they had, how much time they spent waiting for connections. Such information would allow to measure transit performances with unprecedented precision and effectiveness.

1.6 Thesis Structure

The thesis structure is the following:

- Chapter 1: thesis introduction.
- Chapter 2: the state of the art in public mobility transport is discussed and the main challenges introduced. Subsequently the already available applications in the fields of trip planning, public transport state reconstruction and road state reconstruction are analyzed.
- Chapter 3: the conceptual framework is introduced and an overview of the modules is proposed. The general architecture and the current implementation are discussed.
- Chapter 4: the data sources are listed, the data fusion process and the required data structures are illustrated.
- Chapter 5-6-7-8: the core modules of the Public Mobility Platform are discussed in details: the problems of the reconstruction of the public mobility state, the user contextualization, the reconstruction of the road network state and the features classification are here analyzed and the current solutions and techniques implemented are described.
- Chapter 9: the estimated time of arrival cumulative probability concept is introduced along with its implementation and advantages.

- Chapter 10: all the experimental results and the performances both with the simulator and in real cases are shown.
- Chapter 11: possible new mobility applications enabled by the innovative technology of the Public Mobility Platform are here proposed.
- Chapter 12: the work is briefly summarized and the achievements are illustrated.

CHAPTER 2

State of the Art

Designing assistive, portable, continuously interacting applications requires to know precise information about the transports system state (e.g., position and equations of motion of the means, position of the users on the means). In many cities the most temporally accurate data available are the estimated departure time of the next train or bus at the stops, thus the information regarding the transport systems state is not directly available but has to be reconstructed using the available data.

Reconstruct this information is a complex procedure that requires to design a set of methods able to gather the available data and to elaborate and propose the structured needful information.

The first challenge in designing such an infrastructure regards the ability to locate the vehicles using the publicly available data, that is generally fragmented and inaccurate. To be useful in this field of applications, the localization of the vehicles can not be referred only to their absolute position (e.g., GPS coordinates) but instead it has to concern the estimation of their relative position in the transport network. Along with the relative position of the vehicles themselves, to provide interesting information and to compute accurate forecasts, another fundamental information is their travel speed in the network. Estimate the travel speed of the vehicles in the different parts of their route and how it changes during the day requires further information that, like for the position, is not currently directly available. Thus, a demanding part of the work revolves around the design of agorithms for the estimation of this fundamental information.

In literature some approaches to localize the vehicles and their speed can already be found (Section 2.1.2), however the more advanced approaches proposed are not specific for the public transportation, they depend on the availability of the GPS of the vehicles and they can predict the veichles position just some seconds in advance. The available works focused on the public transport require historical data to work properly

and do not react in real-time to the changes in the network. In this work the position of the public transport network is reconstructed in real-time, providing predictions in the order of tens of minutes. Since the GPS position of the vehicles in the public transport field is not publicly available, a framework that exploits only the heterogeneous data sources (including indirect observations) publicly available is proposed.

When considering the design of advanced applications for public transportation, the position of the vehicles and the travel speed in the segments of the lines are not sufficient as it would be for private transportation applications. While position and travel speed of the roads would be enough to compute the more convenient route and to provide a good estimation of the time of arrival in private transportation applications (e.g., a planner that recommends the best route to reach a desired destination by car), in public transportation another fundamental information is required: the context of the users. The context of the users can be defined as their state in the transport network, that is, if the users are already on a vehicle (and which exact vehicle), they are still walking or they are waiting at a station. The current approaches in this field (a review is proposed in Section 2.1.3) currently require an instrumentation of the vehicles using dedicated hardware (e.g., bluetooth beacons). In this work a method that does not require hardware instrumentation but only the users (even inaccurate) position to estimate their state in the transport network is proposed.

In this chapter, after a review of the literature, the concept of *public mobility state* is then defined and formalized, including all the information regarding the position of the vehicles, the speed of the segments and the context of the users. The problem of the arrival time unreliability and how it badly affects the currently available public transport trip planning services is then introduced. In particular, the limitations of the services based on the estimated time of arrival are highlighted and a general algorithm to compute the arrival time as cumulative probability is introduced. This technique can be used to compute the routes and the commutation times that consider, along with the other constraints, also the delay probability of the vehicles based on the current state of the network.

In the last section of this chapter the available tools and applications in the public transportation planning and assistance field are reviewed, with particular focus on applications that propose real-time features. Their limitation are discussed and new application that overcome these limitations using the proposed framework are introduced.

2.1 Public Mobility State

In this section the most interesting approaches available in literature to reconstruct one or more dimensions of the *public mobility state* are critically analyzed and discussed. An overview of the methods used to overcome the current limitations and to reconstruct the more innovative features of the public mobility state is then proposed.

2.1.1 Definition of *Public Mobility State*

The **public mobility state** can be defined as all information concerning the public transport mobility, including all the means of transport position and speed, the lines, the roads, the traveling users and some related information. The public mobility state can

be divided in three interacting dimensions: the *public transport network state*, the *road network state* and the *users state*.

The **public transport network state** represents the current position the means of transport and their equations of motion (that is the prediction, in the future, of their position).

The **road network state** represents the current state (e.g., speed, traffic flows) of the roads in a given area.

The **users state** is composed by the position of the traveling users (e.g., gathered using a mobile phone application), their destination and, when on a vehicle, the association with the means of transport they are traveling on.

2.1.2 Public Transport State Reconstruction

The reconstruction of the public transport state revolves around the ability to accurately estimate the current position of the means of transport in the transport network and their speed. In literature, as explained in the survey [49], state-based models are widely reported in Travel Time Prediction (TTP) and state reconstruction because they are capable of handling congested traffic situations and noisy data [11].

Among the different state-based models, the approaches based on a *Kalman filter* [42] and its underlying model are the most commonly used. Compared to other state-based models, the Bayesian filters (and in particular the Kalman filter) allow to manage noise in the data, which is extremely relevant in online learning tasks for short-term prediction problems.

Approaches Based on Kalman Filter

Given a set of measurements coming from different sources and a model that describes them as functions of some state variables, a Kalman filter optimally estimates the current state, taking into account observations reliability.

Some significant features of this filter, with respect to the application considered, are the possibility to integrate heterogeneous data sources, to obtain a continuous model evolution notwithstanding the discontinuities in the observations, and the ease of making predictions by simply simulating the model with the estimated state.

In literature there are some examples of application that use the Kalman filter to reconstruct the position or the arrival times of some vehicles.

Huifang Feng, Chunfeng Liu, Yantai Shu and Oliver W. W. Yang in their work [24] show that a Kalman filter is successfully applied to predict the moving vehicle's future location information with the aid of GPS in a VANETs. They reconstructed the position and speed of the vehicles using the current GPS position predicting the position in the close future (some seconds). However in public transport systems the GPS position of the vehicles is not publicly available, and the predictions have to be at least in the order of tens of minutes. These limitations are overcome in this work proposing a general approach that uses heterogeneous data sources (including indirect observations) and proposing a model that allows to predict the travel time of the means of transport for the following hours.

Steven I-Jy Chien and Chandra Mouly Kuchipudi in their study [12] used the Kalman filter to perform a travel time prediction based on real-time and historic information collected by road side terminals installed on the New York State Thruway. Specifically,

they selected intervals of 5 minutes each in peak hours and off peak hours and used the average travel times of some selected days to predict the travel time of the same intervals in the following days. In another experiment the average travel time of tagged vehicles in each time interval was treated as the true value to predict the travel time in the next time period. Results reveal that using previous time interval data is generally less erroneous. However, the idea has some drawbacks: using the data from the previous time interval may not always be feasible, especially if the time interval in the experiment is short. Very few travel time samples can be obtained for prediction under this situation. Predicting travel time only for the next time interval would be the primary limitation while using only data collected from the previous time interval. The advantage of using historic data over the link-based model is data availability, allowing prediction at any given time, but at the expense of prediction accuracy under congestion situations and not predictable disruptions.

Similarly, M. Deeshma and Ashish Verma [17] presented a study to predict travel time of buses in Bangalore city. The authors measured travel time of vehicles directly using simple stop watch and then predicted the travel time using a Kalman filtering technique. The authors proposed 2 scenarios: the first scenario used the data of the same trip to predict the next road section travel time. The second scenario used the data of the same day, a week before the input. The output is the predicted travel time. It was found that the first scenario provided the best results for the study site during selected time blocks. This approach gives a general idea of the travel time of the vehicles but it is not accurate, especially on short travels.

In this work, a general real-time system based of a Kalman filter is proposed: instead of just using some recorded travel times to predict the future travel times, the state (position and speed) of all the means of transport is reconstructed using the publicly available data. This technique can be used to predict the travel time of all the vehicles in the public transport network and the arrival time of the passengers. Moreover, this approach does not depend on the availability of a single data source (e.g. GPS position or recorded information about travel time) but different data sources can be used.

2.1.3 User contextualization

One of the most challenging part of this work revolves around the contextualization of the users in the public transport network.

Yu Zheng et al. in their article [78], propose an approach to automatically infer transportation modes, including driving, walking, taking a bus, and riding a bicycle, from raw GPS logs based on supervised learning. But this approach can discriminate the transportation mode but not the exact vehicle the user is riding.

The only available experiments in the user contextualization in public transport vehicles exploit the beacon technology [4]: this solution uses micro-location, contextual communication and presence detection based on iBeacons to locate the users. However, this approach requires dedicated hardware to be installed in the public means of transport and mobile devices compatible with the iBeacons technology.

In this work a framework to contextualize the passengers on the specific vehicles is proposed. Similarly to the public transport state reconstruction, a state-based approach able to filter the noise in the input data is required. Moreover, it does not have to require dedicated hardware instrumentation. Even in this case the Bayesian filtering

approaches are the ones that offer the best solutions in order to design a system that offers the ability to manage noisy data and to provide a continuous model: some significant features of this family of filters, with respect to the application considered, are the possibility to integrate heterogeneous data sources, to obtain a continuous model evolution notwithstanding the discontinuities in the observations, and the ease of making predictions by simply simulating the model with the estimated state. Due to the high nonlinearity of this specific problem, the approach used in this work is based on a *particle filter*.

The **Particle Filter** (PF) methods are recursive implementations of Monte-Carlo based statistical signal processing, the PF objective is to estimate the posterior density of the state variables given the observation variables [48].

The PF algorithm does not require the noises to be Gaussian and the posterior probabilities are represented by a set of randomly chosen weighted samples, in this work different type of particles and different weights are used: 2 types of particles are defined, *walk particles* and *vehicle particles*, with different weights, as explained in details in Chapter 7.

2.2 Arrival Time Unreliability

P. Rietveld et al. [61] used data on unreliability of individual transport modes to simulate the degree of unreliability in the total chain for various parts of the day (morning peak, off-peak, and Sundays). They found that about 40-45% of travelers arrive according to schedule, about 30% has a delay of 0-5 minutes; the rest (25-30%) has longer delays. The average travel time of routes with connections is about 10% higher than the scheduled travel time. Most of the extra travel time is spent as waiting time on platforms because unreliability implies that travelers miss connections.

Considering the high probability of delays just discussed, it is clear why the current public transportation trip planners are not able to provide accurate time of arrival estimations and they may also propose non optimal routes (e.g., routes with high probability to miss a connection). To exploit all the possible information and improve ETA accuracy, in the more advanced trip planners this value is calculated using static data and then updated using real-time data as soon as they are available. Even in these applications, while this value conveys much information with as little data as possible, it is valuable insofar the user catches all the proposed connections in the travel and, more generally, it lacks information on the effects of haphazard factors like traffic jams or disruptions. In this thesis the arrival time unreliability and how it affects the current applications accuracy is addressed, in Chapter 9 the estimated time of arrival is proposed as a generalized concept: the *Estimated Time of Arrival Cumulative Probability (ETACP)*. The **ETACP** for a route is defined as the cumulative probability to arrive at the desired destination within a given time, considering the specific characteristics and sources of variability of a route (e.g., number of connections, traffic probability in a line, walking time between connecting stations).

2.3 Existing Applications

2.3.1 Public Transport Trip Planners

An analysis of the services offered by the existing most widespread travel planners is proposed in this section, reviewing their features. In particular, the planning features are described and the most popular ways the major public transportation trip planners display the estimated travel time and/or the estimated time of arrival are shown. As an example, at the same hour of the day, the best routes between the same two points of the City of Milan are requested.

It is also shown that none of the existing approaches seems to offer advanced travel assistance, highly context-aware social interactions and they do not generate feedback data for transit agencies.

ATM GiroMilano

The web service GiroMilano, powered by ATM, is a local multimodal travel planner that uses means of transport of the city of Milan: ATM (undergrounds and buses) and Trenord (trains). It visualizes the public transport stations, the GuidaMi car-sharing and BikeMi bike-sharing service stations. GiroMilano needs as input the origin and destination of the travel, it permits to exclude some means of transport and it gives the user the possibility to choose a route using different criteria (fastest, with less transfers, with shortest walking path). The desktop version does not integrate real-time data, while the smartphone version provides real-time arrival time at the user closest stations but not a travel planner service. It proposes multiple routes and, for each of them, it shows the estimated travel time (Fig. 2.1) considering the timetables and the frequency of the vehicles in each line.

Figure 2.1: *Possible routes and estimated travel time proposed by ATM GiroMilano.*

CityMapper

Citymapper [14] is a travel planner available in many metropolitan areas in the world, including Milan and Rome. In Italy it combines buses, undergrounds and trains managed by ATAC Roma, COTRAL Roma, ATM Milano, Trenord and Trenitalia. In the urban areas it provides combined routes with bike-sharing services (e.g., BikeMi). It also computes positions and routes using open data provided by Car2Go and Uber companies. It uses real-time data to show means of transport arrival time at the closest

stations and car-sharing/bike-sharing positions. Similarity to Giromilano, CityMapper proposes multiple routes and the estimated travel time (Fig. 2.2) using the timetables and the frequency of the vehicles in each line.

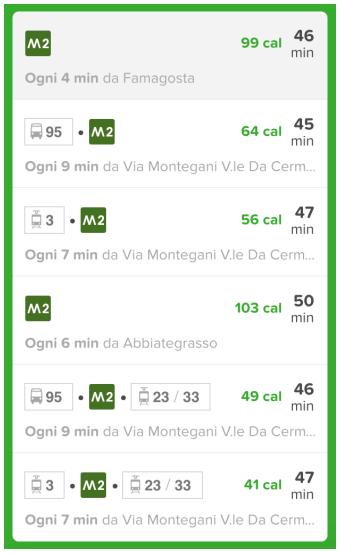

Figure 2.2: *Possible routes and estimated travel time proposed by CityMapper.*

Moovit

Moovit [47] offers real-time public transit information and GPS navigation across transit modes, including buses, ferries, undergrounds, trains and trams. Users can access a live map and view nearby stops and stations based on their current GPS location, as well as plan trips across transportation modes based on real-time data.

This application differs from traditional public transit applications because it is community-driven and integrates official public transit data from transit operators with real-time data collected from users via crowdsourcing. Moovit also created transit data for areas in which no data is officially available by deploying a *community editor* and allowing volunteer editors to generate schedule and map data to be loaded into the platform.

Moovit uses speed and position of the users on the means of transport to improve trip plan results based on current conditions, and share this data with the user community. Using the *En Route* functionality, the user can choose the line they are riding and be notified about the estimated arrival time and the upcoming stops. Moovit uses speed and position of the users on the means of transport to improve trip plan results based on current conditions, and share this data with the user community. In addition users can also actively send reports including reasons for delays, overcrowding, satisfaction with their bus driver, and wifi availability.

Moovit does not disclosure the underlying model and mechanism for its EnRoute feature, but according to some observations and experiments, it likely uses the GPS position of the user, together with an estimation of the travel times of the routes. It does not seem to know the *identity* and position of the particular means. Moovit [47] proposed multiple routes and in addition to the average estimated travel time, it shows

the real-time estimated time of arrival (Fig. 2.3).

Figure 2.3: *Possible routes, estimated travel time proposed by Moovit.*

Like the previous services, Moovit uses static timetables to compute the routes but it also annotates the results using, where available, the real-time waiting times at the stations.

Google Transit

Google's own public transport route planner. Transit [31] calculates routes, transit times and cost, and can compare a trip using public transportation with one using a private car or the Uber service. It is fully integrated into Google Maps and provides routes using all the data coming from providers that publish data in the GTFS format.

In selected cities, Google also provides a service called Live Transit that offers real-time informations about the incoming vehicles. This service alerts the user when a vehicle is arriving a bit earlier than scheduled and notifies when the means of transport are departing from a particular station for a chosen route. This service is currently available in Chicago, Seattle, San Francisco, the UK, the Netherlands, and Budapest [30].

It is not clear whether Transit uses real-time information to compute the routes. Real-time annotation is provided only where they are provided in GTFS real-time by local transport agencies. Google Transit is, currently, the most advanced service and it has two visualization modes. The first one, shown in Fig. 2.4, is similar to the one proposed by Moovit and it proposes multiple routes with the estimated travel time and estimated time of arrival computed using the timetables and, where available, the GTFS data in real-time.

The second one, called Schedule Explorer and shown in Fig. 2.5, proposes a graphical visualization of the routes showing on a timeline the estimated travel time and the estimated time of arrival relative to each route.

Hyperpath

A different technique is used by Hyperpath [18]. This platform uses an hyperpath approach for path suggestion and the optimal hyperpath is found according to an adaptive

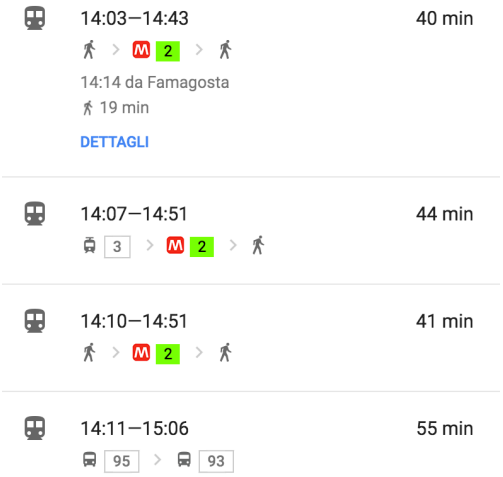

Figure 2.4: *Possible routes, estimated travel time proposed by Google Transit.*

Figure 2.5: *Possible routes, estimated travel time proposed by Google Transit using Schedule Explorer.*

indifferent diversion rule: traveler boards the first arriving vehicle of a set of attractive lines (indifferent optimal strategy). The suggestion can be given according to an adaptive-intelligent diversion rule, which uses also the real-time available predictive information (intelligent optimal strategy). It shows the estimated travel time considering the timetables and the frequency of the vehicles in each line.

According to the presented review, none of the existing approaches seem to offer advanced travel assistance, highly context-aware social interactions and they do not generate feedback data for transit agencies. One of the aims of this thesis is to provide the necessary infrastructure to enable the development of such applications. An example of an innovative application was developed and proposed in Chapter 11.1.

2.3.2 Road State Reconstruction Applications

The major providers currently involved in the road state reconstruction (i.e., actual speed of a road, traffic flows, roads congestion) are reviewed in this section.

TomTom

TomTom [71] produces traffic data using information gathered from public sources and probe vehicles that use TomTom devices. This system provide reliable estimations on the highways but it is not very accurate in the urban areas due to the limited number of probe vehicles.

InfoBlu

Infoblu [38] aggregates the information about the highways in the north of Italy, similarly to TomTom they use probe vehicles to estimate the traffic in the highways and in the major extra urban streets.

Google Traffic

Google Traffic [68] estimates the road traffic calculating the speed of users along the streets. To do this, Google analyzes GPS-determined locations exploiting data available from a large number of mobile phone users, processing the incoming raw data about mobile phone device locations. Google Traffic has a good accuracy even in the urban areas and provides travel time estimations using historical data. With the acquisition of *Waze* [29], Google Traffic now integrates also crowdsourced data.

Inrix

Inrix [39] is the company exploiting the major number of different sources: it uses fixed probes, probe cars and crowdsourced data. Inrix provides information to several different public administrations.

Differently from the current services, the work proposed in this thesis does not require any dedicated hardware (e.g., GPS sensors, speed detection systems) or a large number of users (e.g., crowdsourced data) to work properly. This system can evaluate the traffic congestion where Google Traffic and other services are not available and can be integrated in application like the previously mentioned ones to improve their road network state estimation accuracy. In the proposed Public Mobility Platform, the road network state can be used to improve the accuracy of the real-time estimated travel time

of the massive public means of transport that travel on roads (e.g., buses) and to evaluate alterative routes using private public means of transport (e.g., car sharing services).

On the application side, *Petrovska and Stevanovic* in their work [54] provide an automated and interactive visualization tool for congestion analysis in real-time that uses live traffic congestion data from Google Maps traffic layer with the aim of reducing the traffic congestion on roads and of providing important data which can help road traffic management. In this kind of services, the methods used in this work can be used to estimate traffic, travel time and cars flow using a dataset publicly available online.

CHAPTER 3

Public Mobility Platform

Designing innovative services in the public transportation field requires an accurate knowledge of the public mobility state and of the traveling users. As explained in Chapter 2.1.1, the *public mobility state* is defined as the whole information about the *public transport network state*, the *road network state* and the *users state* in the transport network.

The *public transport state*, the *road network state* and the *users state* are reconstructed using the more effective techniques, creating a general framework for the reconstruction of the *public mobility state*. As discussed in Chapter 2, this information can be reconstructed using a state-based approach that exploits Bayesian filtering techniques to integrate and manage the heterogeneous, noisy, fragmented data sources publicly available. This framework, the data fusion and storage process and the API server compose the platform that, hereafter, will be referred as the **Public Mobility Platform**.

In this chapter the architecture of the Public Mobility Platform is introduced along with a briefly description of the conceptual modules that compose it.

3.1 Conceptual Framework

In this section the conceptual framework is presented, the platform is composed of six main modules, as shown in Fig. 3.1:

- Data fusion: in this module the data are collected from heterogeneous data sources, elaborated and stored.
- Public transport network: in this module lines and stops states are reconstructed and the position and velocity of the vehicles are estimated using the data supplied by the data fusion module and the information relative to the *road network*. The

accuracy is then improved using the position of the user elaborated in the *user contextualization* module.

- Road network: similarly to the *public transport network* module, in this module the state of the roads of the city is reconstructed using the information elaborated in the *public transport network* and the information extracted processing the car sharing data.
- User contextualization: in this module the users of public transports are contextualized in the transport network. Information about the position, the vehicles on which they are traveling, their routes, and the state of the specific vehicles (e.g., occupancy of the vehicle) are provided.
- Network features classification: this framework analyzes the *road network* and the *public transport network* and classifies the data. It is used to identify and explain the network behaviors and to automatically detect characterizing features.
- API: This module exposes to external services and tools all the information regarding the public mobility state.

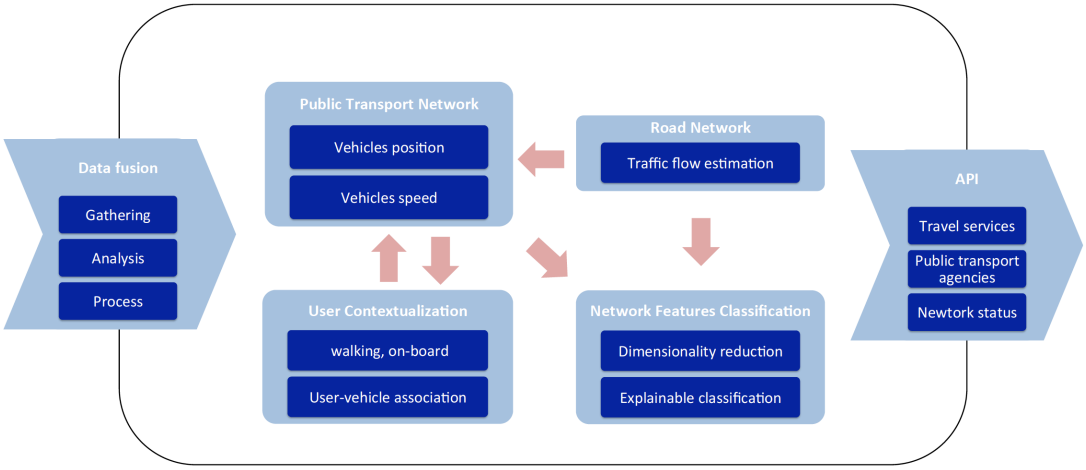

Figure 3.1: *Public mobility platform*

3.2 Data Fusion

The **data fusion** module gathers, elaborates and stores in customized data structures all the data collected from the heterogeneous sources available in different cities. The platform is not dependent from a single source and can thus be used in different cities according to the available data.

Examples of data sources that can be used are the following:

- Timetables: supplied by transport agencies.
- Open data: supplied by transport agencies and public authorities.
- Arrival times on platforms: supplied by transport agencies in real-time.

- Crowd-sourced data: supplied by transport network users through dedicated applications.
- Social networks real-time information: supplied by transport agencies on different social networks.
- Sensors and/or probes: located along the lines.
- Web sites: to scrap real-time information and car sharing information.

Although many transport agencies own also the GPS position of the vehicles, this information is not usually disclosed for privacy and security reasons hence it is not considered in the current implementation. The platform is potentially able to manage also this kind of information, where and when available, but it is designed to provide accurate information in absence of these data.

The data collected from different sources are processed, cleaned and structured according to the state of the art data mining techniques, the whole information is stored afterwards in a database and exposed to the other modules.

The data gathering and processing implementation is further described in Chapter 4.

3.3 Public Transport Network

The **public transport network** module aims at the public transport state reconstruction, that is the position of each vehicle and the estimated velocity of each vehicle on each line. This module employs a Bayesian based approach, the *Kalman Filter*. As described more in details in Chapter 5, given a series of observations (input data) and a model describing the function of some state variables, the Kalman Filter allows to estimate in an accurate way the current state, considering the reliability of the different sources and thus filtering the noisy information. This module uses the defined model to provide information relative to the position and speed of the vehicles on each line and can be used to estimate, per each vehicle, the arrival time to the next stations.

3.4 Road network

The **road network** module aims at the reconstruction of the real-time state of the road network, that is the estimation of current traffic congestion and travel speed. All the information relative to the roads speed and possible disruptions is gathered, elaborated, stored and exposed to the other modules. The information used to reconstruct this state is provided by two main sources:

- Massive public transport vehicles: the speed of the public vehicles that travel on road, such as buses and trams, estimated in the *public transport network* module.
- Private public transport vehicles: the traffic flows detected using *data knowledge from data* techniques, using private public transport services (i.e., the car sharing services).

The details of the implementation of the public transport network module are illustrated in Chapter 5, while the estimation of the traffic flows is described in Chapter 6.

3.5 User Contextualization

The **user contextualization** module associates the traveling users to the public transport network. It detects whether the users are actually on a vehicle or not (e.g., somebody walks along a vehicle) and, for those actually on a vehicle, it determines the exact mean they are on.

The GPS position of the users matched with the closest vehicle is not sufficient as it does not define the user state (*walking* or *on the vehicle*). Moreover, it does not define on which specific vehicle the users are traveling on when there are different public vehicles in the neighborhood. To overcame these limitations, an implementation of the Particle Filter created to discriminate the users state and to estimate on which vehicles they are potentially on has been designed. In the proposed implementation the *particles* of the Particle Filter are divided in two main groups: *walking particles* and *vehicle particles*. These particles are distributed around the users and their surrounding vehicles positions, propelled according to the laws of motion defined in the module and updated, according to their actual distance from the observed user's position.

More technical details and the implementation of this module are described in Chapter 7.

3.6 Network Features Classification

The **network features classification** module allows to classify behaviors of the public transport and road networks, in order improve some characteristics (e.g., car sharing allocation) of the public transport system.

This module uses various implementations of *artificial deep neural networks* to identify selected behaviors learning form examples provided by the public transport and road networks. To understand, appropriately trust, and effectively manage the behaviors, an innovative *explainable deep learning* approach has been designed and tested. Two main techniques have been used: a classification based of feature engineering and one based on spatial characteristics of the network (e.g., interesting areas or road).

The implementation details about this technique are described in Chapter 8.

3.7 APIs

The Public Mobility Platform exposes, through this service, a set of APIs supplying information on stations, lines, waiting times upon each station and the state of the networks. This information is exposed in order to allow the development of innovative mobility support services by external developers.

The list of the currently exposed APIs is described in Table 3.1.

3.8 Communication Infrastructure

The overall quality and reliability of communication infrastructure is a critical factor for the design of innovative transport services: such services require a modern, well developed and reliable telecommunication network. In fact, the capabilities of the communication infrastructure are crucial to enable the collection of data from the field

Table 3.1: *API exposed*

API Address	Description
<*service_url*>*/get_lines*	It provides the ID of all the lines available in the database.
<service_url>/station/<station_ID>	It provides all the information concerning a single station (e.g., station code, GPS position, waiting time for the next vehicle).
<service_url>/line/<line_ID>	It provides an ordered list of all the stations composing a requested line.
<service_url>/rtline/<line_ID>	It provides the ordered list of all the stations composing the requested line and the updated waiting time for the first vehicle arriving to each station.
<service_url>/state/<line_ID>	It provides, for the requested line, the estimated position of the trains and the velocity of each segment of the line.

Description of the APIs exposed by the Public Mobility Platform.

to reconstruct the public transport network state and to provide the updated real-time information with latencies that make the received information usable.

In Italy, where the proposed platform has been tested, the ICT Infrastructure includes [36]:

- 159 mobile cellular subscriptions per 100 people.
- 95-100 % broadband and mobile broadband coverage.

Generally speaking, an excellent coverage of the considered area is fundamental in order to let the system work properly. On the other hand, thanks to the limited quantity of data transmitted, the network speed is valuable but it is not an essential parameter: a 3G connection is deemed adequate.

3.9 Simulator

In order to obtain a set of synthetic data emulating a set of means of transport moving on different lines, an ad-hoc simulator has been developed. This simulator makes it possible to define the main characteristics of the transport network, such as the average speed of the means of transport and the average waiting time at each stop. Moreover, to test the robustness of the *public transport network* module, it allows to create interesting use cases generating noisy data obtained perturbing the vehicles (e.g., introducing noise in the model) or specific corner cases, such as two buses moving together on the same line one just after the other. To test the accuracy of the *user contextualization* module, the simulator allows to simulate several users walking, waiting at the stations and traveling on the vehicles of the network.

The output of the simulator is the list of expected arrival times (expressed in minutes) of the upcoming bus for each stop of each line, which is the exact same data available

in most of the cities that public information about their transport network state. To test the user contextualization module, the simulator produces a list of GPS like traces.

To evaluate the performances of the designed modules of the platform, the simulator can also provide the real state of the transport network (i.e., the location of each mean of transport) and the real state of the users (i.e., users walking / onboard, specific vehicles on which they are traveling). The accuracy is then evaluated comparing this state with the one reconstructed by the Public Mobility Platform.

The simulator is written in Python [25] exploiting the SimPy framework [66], a process-based discrete-event simulation tool.

3.10 Platform Implementation

The Public Mobility Platform has been developed implementing several services hosted on a Linux server. Each service is executed in a dedicated Docker container to guarantee the isolation and correct execution, even in case of migration to other servers. Each container communicates with the others through apposite URIs.

An additional Docker container host the web server in charge to dispatch all the information regarding the public mobility state through APIs.

3.10.1 Data Used in the Current Implementation

The current implementation of the Public Mobility Platform has been developed in Milan, as discussed more in details in Chapter 4, using two main sources of data:

- Open data supplied by the Municipality of Milan: data concerning stations, lines, timetables. Thanks to these data the position and the sequence of the stations on each line is reconstructed, as well as the schedule for the vehicles that still have to depart.

- Waiting time at the stations/stops: data about the waiting time for the next vehicle upon. These data are employed to reconstruct in real-time the position of the vehicles, in order to offer to the user a real-time planning and assistance service while traveling.

Using the developed travel planning and assistance application (Section 11.1) the users' GPS position is collected, in order to contextualize them in the public transport network.

3.10.2 Docker Containers

Docker is an open-source platform for automatic application deployment inside a software environment. Docker provides an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux, allowing independent containers to coexists on the same instance.

Figure 3.2 shows the complete architecture design: the docker containers, in red, with the running services and the platform inputs and outputs in green.

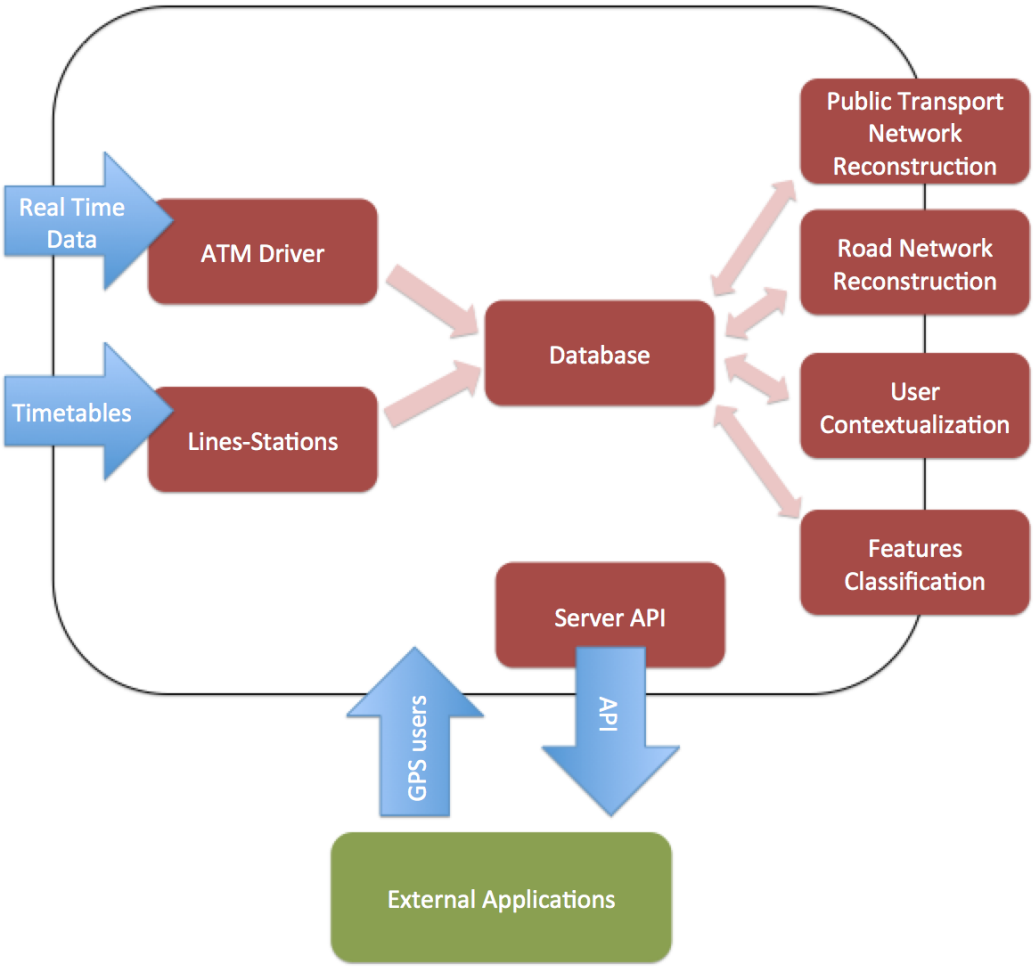

Figure 3.2: *Docker containers on a Linux server*

ATM Driver Container

This container manages the real-time information about the City of Milan provided by of the ATM consortium. This information, received as a data stream, contains detailed data about waiting time for the first available vehicle arriving at each station on each line. The data are parsed and stored in a structured form. All the data containing old or useless information are removed (e.g., night line figures during the day).

This container runs a Java service receiving the updated data stream through a socket. The information is processed and stored in the container named *Database*.

Lines-Stations Container

This container collects, cleans and processes the data relative to lines, stations and timetables provided as open data by the Municipality of Milan. The data are processed in order to obtain information concerning the GPS location and the identification code for each station and the ordered list of stations on each line (a single station can be matched to multiple lines). In addition, the timetables for each line are available. All

these data are stored in the database container.

Database Container

This container executes an instance of the database where all the collected information is stored. These data are then exposed for further processing and, when new real-time data are available, notified to the interested modules.

Public Transport Network Container

This container hosts the public transport network reconstruction module of the Public Mobility Platform. When any state update is computed, the position and velocity of the vehicles is stored in the database container.

Road Network Container

This container hosts the road network reconstruction module of the Public Mobility Platform. When any state update is computed, the information regarding the roads state in the network is updated in the database container.

User Contextualization Container

This container hosts the user contextualization module of the Public Mobility Platform. When new GPS traces are received, they are processed using the updated public transport network state, stored in the database and returned to the requiring external application.

Features Classification Container

This container runs all the services of the classification module of the Public Mobility Platform. This module process data and returns interesting analysis, as explained in Chapter 8.

Server API Container

The Server API container is the interface with the database container and the other interactive containers that provide the APIs required to enable the external innovative applications and tools.

CHAPTER 4

Data Collection Techniques and Definition of the Required Data Structures

Data gathering is a complex and elaborated phase, since information must be extracted from different sources using different techniques depending on the specific provider.

Generally the data sources are divided in three types [20, 67]:

- **Structured data** are data already tagged and sorted.
- **Semistructured data** have fixed fields but contains separate data elements, that make it easy to tag and structure them.
- **Unstructured data** are the most difficult to analyze because they have not fixed fields or path.

Depending on the source, the data can be gathered using public or private API or through a data scraping process. The data originating from heterogeneous sources is then merged to a common structure using Schema Matching and Data Transformation techniques [50] typical of data fusion processes.

In this chapter all the data used in the current implementation of the Public Mobility Platform are described and all the techniques used to acquire and process them are introduced. Afterwards the structures used to represented these data are illustrated.

The current implementation utilizes the GTFS timetables and the real time waiting times at the bus stops provided by ATM (the public consortium that manages the public transport in Milan) and the car sharing data available in Milan. To facilitate the use of this information the public transport network data are represented using time-expanded graphs while the road network data are represented using directed graphs. These structures, in fact, do not require any further processing when a route computation or update is required and are easily updatable as soon as new information is available.

4.1 Creation of the Transport Network Graph

The acquired information is elaborated in order to create a *time-expanded graph* of the public transport network in Milan. As explained in details in the following section, this structure allows to represent the arrival time to each station of all the vehicles and to update it with real time data.

A **graph** [75] is a set of elements, called nodes, that can be connected by lines, called edges. Formally, a graph is an ordered pair $G = (V, E)$ of sets, where V is the set of the nodes and E the set of the edges, such that the elements of E are couples of elements of V.

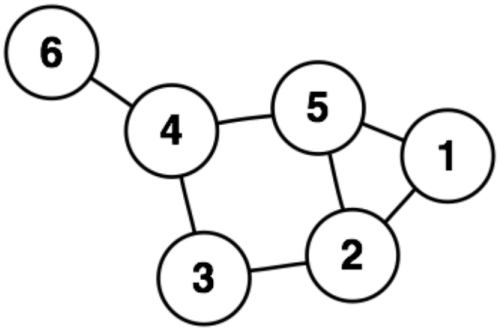

Figure 4.1: *Example of a graph with 6 nodes and 7 edges*

An example of a graph composed of 6 nodes and 7 edges is shown in Fig. 4.1 and described as follows:
$G = (V, E)$
$V = [1, 2, 3, 4, 5, 6]$
$E = [(1, 2), (1, 5), (2, 3), (2, 5), (3, 4), (4, 5), (4, 6)]$

In the *flow models*, like the collective public transport network, usually it is useful to highlight how the single units of the flow are moving in the network. In these situations the use of *time-expanded graphs* [19] is required: in these graphs the nodes are replicated at different time instants and suitable edges represent the flow at different times.

The **time-expanded graphs** have peculiar characteristics that differentiate them from the classic graphs:

- The edges are labeled with the transit times that specifies how long does it take to go from a node to the following (these values may change over time).
- The transit time on a line changes according to the time currently shown on a line.

The last property is crucial in systems like the one proposed because it allows to update the graph in real-time. However, using a time-expanded graph highly increases the complexity of finding the best routes. In fact, with these graphs it is not possible to apply the classical methods to find the shortest paths between two points.

In the designed Public Mobility Platform is proposed a particular representation of a time-expanded graph to describe the network and an on-purpose implementation of the Dijkstra's algorithm to calculate the best routes.

4.1.1 Representation of the Public Transport Vehicles on a Time-Expanded Graph

The public transport time-expanded graph is first created using the available timetables in GTFS format described in Section 4.2. This graph contains 2 different types of nodes:

- *Station node*: it represents a physical stop or station of a public transport line.
- *Stop node*: it represents a stop of a specific ride on a line of the public transport (the time a ride arrives to that node). For each physical stop or station these nodes are replicated at different time steps, one for each ride of the day, as scheduled in the timetables.

The *station nodes* are connected to all *stop nodes* corresponding to the same physical stop or station with a directed edge. The edges that connect a *station node* to a *stop node* have a variable weight (equal or greater than 0) that represents the vehicle pick-up estimated cost.

The *stop nodes* are connected with a directed edge to the corresponding *station node* and to the following *stop node* in a public transport line. The weight of the edges that link a *stop node* to the relative *station node* is a variable weight (greater than 0) that represents the vehicle drop-off estimated cost. The edges that connect a *stop node* to a *station node* are always greater than 0 to prevent infinite loops between a station and one of the relative stop nodes.

Differently from the usual time-expanded graphs, every *stop node* is labeled with a timestamp showing the estimated arrival time of the vehicle in that node. This implies that the weight of an edge is not showed on the edge itself because this information is already contained in the *stop node* it is connected to (the weight is the difference between the starting and arriving nodes of an edge). Due to this particularity of the *stop nodes* the cost of a path is not given by the sum of the weights of all the edges included in the path but it is already explicated in the last node of the path. Moreover, since the *stop nodes* are labeled with absolute timestamps, it is possible to instantly invalidate all the nodes (and the relative incoming edges) with an expired timestamp (the timestamp is prior to the current time or the possible arrival time of a user to that node).

Single Line Non Time-Expanded Graph

In a non time-expanded graph, nodes represent the stations along a public transport line while edges represent the connections between a station and the next with the relevant travel times. An example of single line non-expanded time graph is shown in Fig. 4.2.

Figure 4.2: *Example of non-expanded time graph*

Single Line Time-Expanded Graph

The graph in Fig. 4.3 shows the representation of the same graph proposed in Fig. 4.2, but with a time-expanded graph. The blue nodes represent the *station nodes* while the green nodes represent the *stop nodes* replicated for 3 different rides and labeled with the expected arrival time.

In the time-expanded graph used, the *stop nodes* are replicated per each vehicle traveling on a single line and each of the nodes is labeled with the estimated arrival time of the vehicle at that specific *stop node*. Each *stop node* is connected with a bi-directional edge to the relevant *station node*.

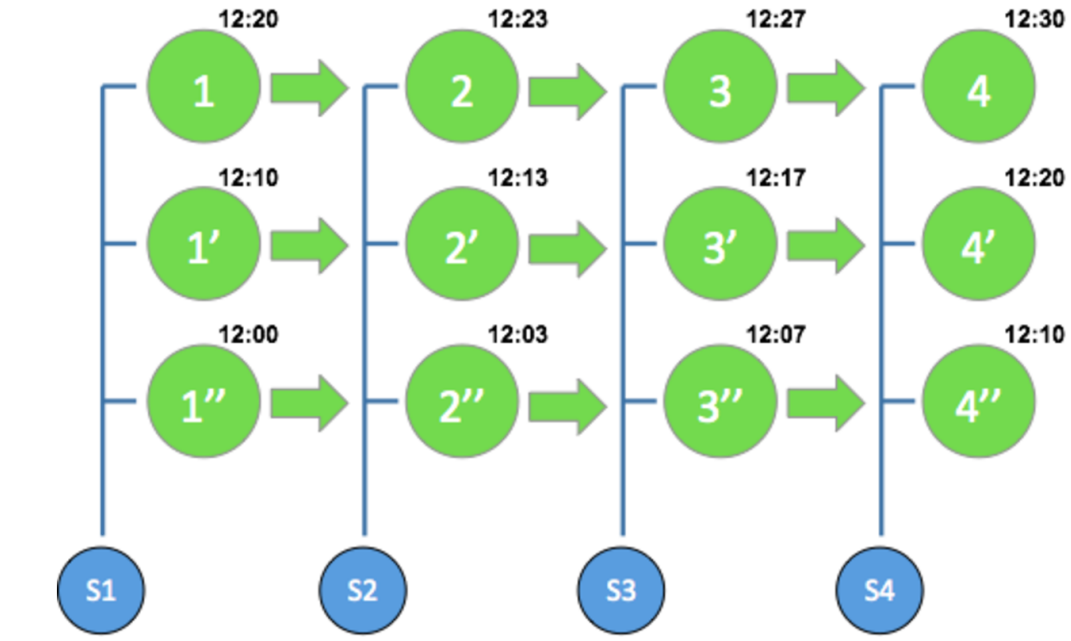

Figure 4.3: *Example of time-expanded time graph*

4.2 Static Timetables

The data relative to the public transport timetable are available online, provided by the Municipality of Milan through the agency AMAT (Agenzia Mobilità Ambiente e Territorio) and supplied in *General Transit Feed Specification* or *GTFS* format [59].

GTFS is a global standard format used to supply information about public transport timetables and the relevant geographic localization. The GTFS format allows a public transport company to publish as *feed* the information concerning the timetables in a standard format. Each feed is made up of a series of text rows describing in a detailed way information such as the stops on each line, the routes, the time schedule of the line and so forth.

These data are elaborated to reconstruct all the lines of the public transport network in a directed graph and stored in the *data fusion module* of the Public Mobility Platform. This structure is the base for the computation of the public transport routes, on this

graph the real time information is annotated.

The data provided by AMAT include all the information of the ATM public lines relative to the buses, trams and subways. The considered tables are:

- Stops table: all the stations and all the stops of the public transport lines in Milan are contained in this table.
- Routes table: the list of the public transport lines in Milan with the relevant information per each line is contained in this table.
- Stop times table: all the information about arrival time of each vehicle at each stop is contained in this table.
- Trips table: the list of all the scheduled trips per each line is contained in this table.
- Calendar table: the days in which the line is in service and the validity period are contained in this table.

4.3 Real-Time Waiting Times

The real-time information is fundamental in order to reconstruct the public mobility state accurately and thus to design effective innovative applications. In the current implementation, this information is guaranteed by the real-time waiting times relative to the next arriving vehicle, for each line, at the public transport stops/stations.

The real-time waiting times are, in Milan, provided by ATM, the local consortium for public transportation. The updated data are sent to the platform every 30 seconds, with information regarding the waiting time of the next vehicle (for each line) to each station. This information is parsed, cleaned and used to annotate the time-expanded graph of the public transport, thus adding the real time information required to design the innovative services.

4.4 Car Sharing

The car sharing data are used to estimate the traffic flows in the road network. The data are scraped from the car sharing providers web sites: the data have to be gathered from heterogeneous data sources, merged, normalized and stored in an unique consistent database. The process includes outliers detection and removal (e.g., fake cars used by some providers for testing purposes), and fix of wrong or imprecise data (e.g., car reserved but not used).

4.4.1 Data sources

The three main car sharing provider in Milan were selected and the data relative to the cars were gathered, cleaned and stored in a database. For each provider the data are gathered every 15 seconds and stored in a database, the extracted data are:

- GPS position (latitude and longitude).
- Fuel level.
- Plate.

- Status of the car (interior and exterior).

Plus some information added by the scraping scripts:

- Name of the provider.
- Timestamp.

These data include only the information of the car available (rentable) at that time, while no information about the currently rented car is available.

The identified providers, selected considering the major number of cars available in Milan, are:

- **Car2Go** [10]: it provides a specific API with JSON structured data.
- **Enjoy** [21]: it is accessible through private http GET request to an internal server, authentication and city selection with cookies.
- **Twist** [9]: it provides unstructured data in a JavaScript file.

Data are gathered every 15 seconds using different techniques: *Car2Go* data are requested using the provided public API while the other two providers do not offer any API, hence, the data are gathered using scraping techniques.

CHAPTER 5

Reconstruction of the Public Transportation Network State

In this chapter the current implementation of the *public transportation state reconstruction* module is presented in details. This module exploits the static (timetables) and real-time (waiting times at the stations) information in order to reconstruct the position and the speed of the vehicles in the public transportation network. This information is used to provide the more accurate possible estimation of the time of arrival and to annotate the time-expanded graph of the public transport network.

The proposed model and the chosen approach are firstly introduced, then the current implementation is described. The results and the real case experiments are discussed in Chapter 10.

5.1 Proposed Model

As discussed in Section 2.1.2, the Bayesian filters have good performances in situations where a model is required and the input data are noisy. The state of the transport network is represented using a model that describes a public transport line and the traveling vehicles. In particular, the model of each line is independent from the others and consists of the positions of all the vehicles of that line and the average speed of each line section. A *line section* can be defined as the list of roads connecting two consecutive line stops.

In order to reconstruct the state of the public transport network, an algorithm based on the Kalman filter was designed. The transport network was modeled as set of independent lines, so that each line of the public transport network has both its own filter and its own state.

5.1.1 Initialization

To correctly reconstruct the state of a single line of the transport network, each stop has to be assigned to the corresponding line. To perform this task, the static timetables usually available as open data or provided by the local public transport consortiums are typically sufficient. The information extracted from these static timetables is used to initialize the state of the public transport network and to organize it into lines; the geo-referenced location of each stop was extracted and an ordered list of stop for each line was created. Then the system starts processing the data coming from the measurements in order to populate the data of this structure, associating the information gathered with the corresponding line stop.

5.1.2 Definition of the Network State

Since each line is treated as independent from the others, in this section are proposed all the relevant features of a single line. In particular, the model of each line includes:

Constants

- s_L, which is the number of stops of line L;
- m_L (equivalent to $s_L - 1$), which is the number of sections of line L;
- n_L, which is the number of vehicles that cover line L each day;
- $y_{L_1}, \ldots, y_{Ls_L}$, which are the locations (distances in meters from the first station) of the s_L stops of line L. For the sake of simplicity, it can be assumed that the distance between each couple of adjacent stops is constant. Using real distances do not improve the accuracy of the predictions of the module, but it is useful to obtain estimations of the the real average speed of each line section.

State Variables

- $x_{L_1}, \ldots, x_{Ln_L}$, which are the positions (relative to the locations of the line stops) of the n_L vehicles traveling on line L. A value smaller than 0 indicates that the vehicle still has to start its trip, while a value greater than y_{Ls_L} (used here as a proxy of the line length) indicates that the vehicle has reached his destination. In all the other situation, the vehicle is traveling and its position corresponds to a stop or to an intermediate location between two consecutive stops.
- $v_{L_1}, \ldots, v_{Lm_L}$, which correspond to the average velocities of the sections of line L. In this work, all the vehicles traveling at the same time on the same line section are assumed to travel at the same average speed.

Observations

- $z_{L_1}, \ldots, z_{Ls_L}$, which are the waiting times (expressed in minutes) seen at each stop of line L. These values are updated after receiving new observations and represent the main input of the model.

5.1.3 Model

The proposed Kalman-based approach exploits the following model in order to predict the value of the state variables (the position of each vehicle and the average speed of each line section) starting from the input observations (the waiting time for the next vehicle at each stop).

In particular, the position of each vehicle x_{L_i} at time $t + \Delta t$ is computed in the following way:

$$x_{L_i}(t + \Delta t) = x_{L_i}(t) + \Delta t * v_{L_{x_i}}(t) + \epsilon_x$$

where a non negative value of the position ($x_{L_i} \geq 0$) corresponds to a traveling vehicle, while a negative value ($x_{L_i} < 0$) identifies a vehicle that still has to start its trip.

The speed of each segment v_{L_i} remains constant in the model (it is changed, according to the observations, in the $correction$ phase):

$$v_i(t + \Delta t) = v_i(t) + \epsilon_v$$

ϵ_x and ϵ_v are the process noises that, in this scenario, can be set to 0 without loss of generality of the model.

Finally, in order to compute the actual gain of the Kalman filter, an output of the model has to be associated to each observation, thus making it possible to evaluate discrepancies between the model state and the real-world measurements. To this end, for each stop S the first vehicle V that will arrive in S is selected and the current location of V and the average velocity of the section V is traversing to compute the estimated waiting time of stop S are used. In particular, each observation z_{L_1} is associated to the time required by the vehicle in position $p_{z_{L_1}}$ to get to the next stop:

$$p_{z_{L_1}} = y_{L_1} - \max_{j: x_{L_j} < y_{L_i}} (x_{L_j})$$

5.1.4 Emitting and absorbing stations

Since the n_L vehicles are not traveling on line L at the same time, handle correctly the status of the idle ones is indispensable. For this reason, the model has been integrated with a couple of *virtual stops* per line, one at the beginning and one at the end of the line itself.

The *virtual stop* at the beginning of the line is called **emitting station** and it is used to *hold* all the vehicles still in the depot. Starting from the emitting station, the vehicles move very slowly towards the first stop of the line, and the model takes care of letting them reach this stop exactly at the expected departure time. However, since the framework is fed with real-time data, it is able to modify the speed of the *virtual section* between the emitting station and the first stop in order to perfectly match the input data coming from the actual measurements.

The second *virtual stop* is called **absorbing station** and it is placed after the last stop of the line. The line section between the last stop of the line and the *absorbing station* is

characterized by an extremely low speed since is basically used to *hold* all the vehicles that have already completed their itinerary. Since there are no actual measurements coming from the *absorbing station*, the speed of this section is never modified.

These 2 stations are fundamental in the model defined. The *emitting station* allows to exploit the information coming from the observations in the first station and ensuring the vehicles to reach it at the desired time. The *absorbing station*, on the other hand, allows to virtually remove the vehicles arrived to destination from the line and let the Public Transport Network State Reconstruction module perform optimally.

5.2 Kalman Filter

As described in Section 5.1, the proposed approach exploits as input the waiting time obtained from each stop and computes the positions of all the vehicles traveling on the different lines of the transport network, as well as the average speed of all their sections. Unfortunately, the conversion from times to positions requires a non-linear computation, thus the traditional implementation of the Kalman filter cannot be employed (it requires both the model end the observation functions to be linear). In the frame of non-linear filters, the two most common algorithms are the textitExtended Kalman Filter (EKF) [70] and the *Unscented Kalman Filter (UKF)* [76].

The **Extended Kalman Filter** takes advantage of the linearization of the original non-linear filter dynamics around the previous state estimates. With some preliminary experiments it has been found out that the EKF performs well only when there is a single vehicle at a time in the line, otherwise some issues start arising around stops. In particular, the problems are related to situations where a vehicle V passes a stop S, and thus the display of S start showing the new arrival time (the one of the following vehicle), but in the model V is located just before S (the error could also be minimal): in this case, the model associates the new arrival time to V, thus pushing it back in order to match the observed data. As a consequence, also all the other arrival times are associated to the wrong vehicle, thus causing all the vehicles to jump backward in the line. To solve this problem it is possible to reduce the weights of the observations, consequently increasing the importance of the weights of the model. This solution partially mitigates the issue, but it also reduces the accuracy of the predictions.

On the other hand, the **Unscented Kalman Filter** captures the mean and the covariance of the estimate by using a sampling technique known as the unscented transform: a method for calculating the statistics of a random variable which undergoes a non-linear transformation. This technique, instead of calculating the partial derivatives, uses a set of carefully chosen sample points around the mean, the *sigma points*, and propagates them through the non-linear system, from which the posterior mean and covariance of the vehicles positions and segment speeds are then recovered.

5.2.1 Unscented Kalman Filter Prediction

The state of the proposed Kalman filter is piecewise linear: in fact, the status of each vehicle of the public transportation network is function of its current position and of the average speed of the section it is currently traversing.

For this reason the *transition matrix A* used in the prediction of the new state of the sigma points cannot be constant, but it has to be redefined at every time step in

accordance with the current position of the vehicles, depending on the position they move with a different speed.

5.2.2 Unscented Kalman Filter Correction

A function that predicts the waiting times seen at each stop/station using the predicted position of the vehicles is defined. These values are compared with the real waiting times (the observations) and used in the UKF algorithm to correct the predictions.

5.2.3 Vehicles Arrival Time Prediction

The designed model describes the position of each vehicle and the average speed of all the sections of each line of the transport network. Thanks to this model, the Kalman filter can also be used as a predictor by simply running the update algorithms without new observations, thus letting the state of the system autonomously evolving. In other words, if the algorithm is executed forward in time, the Kalman update phase keep on predicting the position of the vehicles, but the correction phase is ineffective since there are no observations.

This technique is used to predict the departure time and the arrival time of each vehicle for each station of the line it is traversing, thus making it possible to elaborate paths that make use of ride coincidences that will happen in the future.

CHAPTER 6

Knowledge Discovery from Car Sharing Data for Traffic Flows Estimation

The newly introduced car sharing services are an unexploited source of data that could be used to estimate the state of the *road network* as well as to provide new interesting analysis on urban mobility.

The Public Mobility Platform includes a *road network module* that exploit this sources of data: the information relative to the car sharing services gathered in the data fusion module is processed to estimate interesting metrics such as travel time and vehicle flows in the urban areas at different times and in different days. The information gathered can be processed in real-time, to estimate instant traffic, and can be exploited to perform deeper analysis, using historical data.

An accurate knowledge of the *road network state* is fundamental to elaborate efficient strategies in urban transport management and planning.

Currently, the major suppliers reconstruct the road network state using dedicated data collection methods [60] such as GPS sensors (floating car data [73]), mobile cell towers, access detectors, speed detection systems and, recently, crowdsourced data.

The **road network state** is defined as the state of the traffic on the urban streets and can be expressed as [57]:

1. Travel time: time needed to travel along a road segment of length L.
2. Speed: average speed in a road segment L completed in time TL.
3. Delay: difference between the effective travel time and the same travel with no traffic.

In this chapter an approach that exploits a *knowledge discovery from data* technique to extract new information from car sharing data is proposed. This new information is

used to estimate the road network state without requiring probes or sensors. These data can be elaborated, using data processing techniques [26], to estimate the travel time exploiting the historical information.

The approach is then tested in a real case scenario, gathering data from the major providers available in Milan and interesting analysis are shown, such as the availability of cars in the different parts of the city during the day, the travel time and the average vehicles speed and fuel consumption.

Knowledge discovery from data (KDD) is the process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [23].

[5, 64] outlines nine steps major steps in KDD:

1. Define purpose of process.
2. Generate subset data point for knowledge discovery.
3. Removing noise and expired data and handle missing data fields.
4. Find useful properties to present data depending on the analysis context.
5. Map purposes to a particular data mining methods.
6. Choose data mining algorithm and method for searching data patterns.
7. Research patterns in expressional form.
8. Returning any steps 1 through 7 for iterations.
9. Use information directly, combining information into another system or simply enlisting and reporting.

6.1 Description

Current services that estimate the state of urban road network use external, on purpose, instruments such as probe vehicles or passage / speed sensors. The designed knowledge discovery module uses an approach to estimate such state without using any external instrument but only exploiting data already available online, produced by the car sharing providers. This information is processed to provide an estimation of the state of the road network in the cities where other services (with dedicated resources) are not available and can be used, as an additional source, to improve the accuracy of state of the road network in the cities where other services are already available.

6.2 Data Gathering

Data gathering is one of the more complex and elaborated phases of the road network module: the information must be gathered and processed from different sources using different techniques depending on the specific provider.

Data are collected and stored every 15 seconds using different techniques, depending on the specific provider. Details about the data sources and the data gathering process have been presented in Chapter 4.4.

6.3 Implementation

Scripts for the information gathering are written in **CoffeeScript** [15] using **Node.js** [51], an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices and **Express.js** [22], a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

As main database it has been used MongoDB [46], a document-oriented database that is proven to be efficient with this kind of data and effective even for big data problems [45].

6.4 Data aggregation

As explained in [55] aggregation tools have the ability to organize data in a way that it can be quickly searched, analyzed and efficiently utilized.

In this work three types of aggregation have been considered: per provider, per time and per zone, the results are shown in Chapter 10.4.

6.5 Data processing

Data processing is defined as the *Collection and manipulation of items of data to produce meaningful information* [26].

The car sharing data, stored in the form of *car entry* are processed to detect *travels*. A **travel** is defined as a pair of car entries: each **car entry** include the car plate, GPS position of the car, the timestamp, the fuel level and other informations. A travel is created each time a vehicle (identified by its plate) disappears and reappears after some time, excluding cars that reappear in the same position (these are considered cars booked but not used, thus excluded from the travel list).

A car disappears from the list of the available vehicles (updated each 15 seconds) when it is booked or rented by a user and it reappears when the user completes the trip.

For each travel some meaningful informations are extracted, such as:

- **Travel time**: interval of time elapsed from when a car is booked or rented and when it is released.
- **Average speed during the day**: rate between the distance traveled and the travel time. Since the distance traveled is not known, it is estimated using the route proposed computing the best path between the origin and the destination.
- **Fuel consumption per km during the day**: the fuel consumed per km during the different hours of the day.

CHAPTER 7

User Contextualization

The **User Contextualization** is the complex procedure of understanding when the users are actually on a means of transport or not (e.g., they are just walking on the street close to a bus) and connecting them with the right vehicle they are traveling on, even in presence of multiple means of transport in the same street or area. The quite straightforward task consisting in using the user's GPS position to associate them with the closest mean of transport is not satisfactory since it can not discriminate the status of a user (i.e., walking or onboard) or the actual vehicle in case of multiple vehicles nearby.

In this chapter the module used to contextualize the users in the public transport network is discussed in details. The user contextualization module proposed exploits a Bayesian approach, a Particle filter with two types of particles: the *walking particles* and the *vehicle particles*. These particles are spread randomly around each vehicle and around the user position and are propagated according to their proximity with the user observed position.

To contextualize the users the information about the state of the public transport provided by the *public transport network* module described in Chapter 5 was used: as already discussed, this module exploits a state-based Bayesian approach to the reconstruction of the transit system state (equations of motion of the means, position of the vehicles) from limited available knowledge (e.g., estimated departure time at the stops, data from users phone). The system is general and effective also in the presence of various real-world kinds of noise, such as information blackouts.

7.0.1 Definition of the User State

In this section are defined all the relevant features that compose the *user state* and a model to describe the traveler state is proposed. The **user state** includes the *user position* and the *vehicles state*.

In particular, the model of each **user position** includes:

- $User\ position$: longitude and latitude of the user.
- $User\ action$: walking or onboard.

While the **vehicles state** includes, for each vehicle in the network:

- $Vehicle\ position$: latitude and longitude of the vehicle.
- $Line\ number$: the line number or code of the vehicle.

At every new data available, the user contextualization module is updated with the following observations:

- $User\ position$: last valid latitude and longitude of the user.
- $Vehicles\ state$: position and corresponding line number of all the vehicles currently traveling on the public transport network.

7.0.2 Implementation

When a new user is added to the system, a new user contextualization process is initialized. Particle filtering methodology exploits a set of particles to represent the posterior distribution of some stochastic process given some noisy and/or partial observations. In this implementation, two main sets of particles are designed to distinguish between the walking and the onboard state of the user:

- $Vehicle\ particles\ (VP)$: these particles represent a *user onboard state* and they are initialized with a default weight W. Each particle is assigned to a specific vehicle on a specific line. If two vehicles are close there are particles with similar position but assigned to different vehicles.
- $Walk\ particles\ (WP)$: these particles represent a *user walking state* and are initialized with a default weight $W/2$, half of the weight of the vehicle particles (to boost the evolution of vehicle particles, as explained later).

In the proposed implementation, at every step some WPs are spread around the position of the user and some VPs around all the currently traveling vehicles. Their weight is then updated according to the distance of the observed user position:

$$error = (|usr_{lat} - prt_{lat}| + |usr_{lng} - prt_{lng}|)^2$$

$$particle\_weight = e^{-error/sigma}$$

Where usr_{lat} and usr_{lng} are the last received coordinates (latitude and longitude) of the user being contextualized while prt_{lat} and prt_{lng} are the coordinated of the specific particle. The $sigma$ value influence the capability of the filter to handle the relation between the GPS noise and the accuracy of the estimation. In the current implementation it has been chosen, considering several experimental tests, the value $sigma = 0.9^2$.

Even when a user is on a vehicle, some WPs are considered valid and this can reduce the effectiveness of the filter in detecting the real state of the user. To prevent

this behavior, at the WPs is assigned a lower initial weight: this boosts the evolution of the VPs when a user is close to a vehicle for a certain amount of time improving the accuracy of the user contextualization module.

According to the particles, the users can be estimated in two states: *walking*, users that are walking in the street or waiting at the station, or *onboard*, users that are on a specific vehicles.

Users are mapped as **walking** when the majority of their particles is of type *walk* (WPs). Users are mapped as **onboard** when the weighted majority of their particles is of type *vehicle* (VPs).

For users mapped as onboard, it is also possible to estimate the specific vehicle they are traveling on: on one hand, if the VPs all belong to the same specific vehicle, then the users are assigned to that vehicle. On the other hand, if the VPs belong to different vehicles, the percentage of every vehicle in the set is calculated and all the VPs that are assigned to vehicles present with a percentage of less than a selected threshold have been removed (in the current implementation the threshold is set, after some experimental tests, to 20%). If the remaining VPs all belong to the same specific vehicle then the users can still be classified and they are assigned to that vehicle, otherwise it can be identified the state of the users as *onboard*, but it cannot be estimated the exact vehicle they are traveling on.

7.0.3 Resampling

Resampling [32] is a technique used to generate new particles from posterior distribution function prepared in previous steps. In this approach are created the new particles using the **Multinomial resampling** (MR), also called Simple Random Resampling [35]. It consists of drawing the new particles from a weighted distribution of the particles.

Since the MR approach could evolve only a certain type of particles (e.g., a walking particle close to a station has to become a vehicle particle when a user catches the vehicle), at every step it substituted a percentage of the particles with new generated particles. This process, called **rejuvenation** [8], allows us to have a mix of all the possible types of particles. The new particles are created following the initialization process described in the previous section. Experimental tests have shown that, in this process, a rejuvenation rate of 10% optimizes the balance between the importance of the old particles (to keep track of the current estimated state) and the need of new ones (to explore new possible states or vehicles).

CHAPTER 8

Classification of the Network Features

The public transport and road networks contain interesting patterns that can be used to extract valuable information. This information can be used to improve the offered services and to predict the behaviors of the network (e.g., car sharing allocation, stockout detection). The **network feature classification** module is an experimental tool of the Public Mobility Platform designed to classify these behaviors and to explain why the occur.

One of the best techniques to recognize patterns and classify them is using *deep learning* approaches [6,65]. Deep learning main advantage over other machine learning algorithms is its capacity to extrapolate new features from a limited set of features contained in a training set. That is, it will search for and find other features that correlate to those already known.

The ability of deep learning to create features without being explicitly told means that it is possible to save sometimes months of work by relying on these networks. It also means it is possible to work with more complex feature sets than other techniques.

One of the main limitations of the deep learning systems is the machine's current inability to explain their decisions and actions to human users, for this reason in this module are designed an tested systems whom focus is on *explainable deep learning* techniques. In this section various experimental explainable deep learning implementations are proposed in order to identify selected behaviors, learning form examples provided by the public transport and road networks. To understand, appropriately trust, and effectively manage the behaviors, an innovative *explainable deep learning* [63] approach was designed and tested. Two main techniques are proposed: a classification tool based on input features engineering and another based on the automatic detection of spatial characteristics of the network (e.g., interesting areas or roads).

8.1 Methodology

8.1.1 Features Engineering

In the first proposed approach, the possible features are manually identified in a preprocessing phase using features engineering techniques [74]. A separate neural network is then designed for each feature. Each of the designed subnetwork is fed only with the subset of data related to it, the output of each subnetwork is then used as an input of the final neural network, whose task is to merge and weight the previous output and identify the classes.

The network explanation is obtained analyzing the output of the single subnetworks and their weight in the final network.

8.1.2 Network Based Input

The automatic extraction geospatial features from a graph is useful to analyze behaviors characterized by spatial information. In the case of road networks, the chosen paths can be represented using the network graph of the area to analyze and used to classify the samples. The roads of city can be represented as a directed graph, where the edges are the roads and the nodes the conjunctions. Thus, a trip can be described as an ordered list of nodes where the first node is the starting point and the last node is the arrival point.

However, representing the full network graph of a city can be memory intensive task and consequently the learning process of a neural network can be computationally very intensive. For these reasons several innovative design are proposed, tested and compared trying to identify the best compromise between the size of the neural network input and the quality of the information representation.

The geographic features of a ride are represented as an ordered graph, where the nodes are single points in space defined by their latitude and longitude, while the edges are the ways that connect each node to the others.

All the path representations use the *nodes vector* of the city. The **nodes vector** is the vector containing all the nodes included in the road network of the city. In this vector all the nodes are set to 0, except the one crossed by the path we want to represent, that are set to 1.

Different techniques are used to represent the path: in the **full static node network** all the crossed nodes are included in the *nodes vector*. On the other hand, in the **full temporal node network** each crossed node is represented in a new *nodes vector*, therefore adding a temporal order in the input. The input is, therefore, no longer a vector but instead a sparse matrix, where each column represents a sequential node in the path.

Since the first and last nodes of a path can be particularly interesting, an alternative version of the network based input approach is designed inserting the last node in first column of the nodes matrix. In this version the last node is always in the first column of the nodes matrix while the first node is always in the second column. The deep neural network can take advantage of this information to learn the particular cases in which first and last nodes are considered relevant.

To reduce the computational work and to optimize the learning process, two more techniques to riduce the size of the input preserving the temporal information are exper-

imented. The first technique uses a fixed temporal window and only the first k' and the last k'' nodes are included in the input, the latter group the consecutive nodes in clusters of fixed dimension. Clearly in these approaches part of information is lost: in the first case the central part of the path, in the second one the granularity of the temporal sequence is reduced.

8.2 Network Topology

The **topology** of a neural network refers to the way the neurons of the different layers are connected, the most widespread topology in deep learning problems is the fully connected, three-layer, feedforward network [77]: all the neurons of a layer in the network are connected to all neurons in the following layer, and the activations of the output neurons constitute the output of the whole network. However, in this kind of networks, it is hard to understand what happens inside the neural network, and why a network classifies an entry in a certain class.

8.2.1 Selecting number of layers and neurons

One hidden layer is usually sufficient for the large majority of problems, even if there are some situations in which performance improves with a second (or third, etc.) hidden layer.

Selecting the optimum number of hidden layers neurons is a hard task, since there are no mathematical approaches to choose them. However, there are some empirically-derived rules-of-thumb, of these, the most commonly relied on is the following: the optimal size of the hidden layer is usually between the size of the input and size of the output layers. A rough approximation can be obtained by the geometric pyramid rule proposed by Masters [44]. For a three layer network with n input and m output neurons, the hidden layer would have sqrt(n*m) neurons.

Starting from these guidelines, each one of the subnetworks and the main deep neural network of both the proposed approaches were designed, then the number of layers and neurons was optimized analyzing the learning performances.

CHAPTER 9

Time of Arrival Cumulative Probability in Public Transportation Travel Assistance

In travel planning and assistance applications, the Estimated Time of Arrival (ETA) is defined as a single time value that indicates the expected time it will take for a traveler to reach the desired destination. While this value conveys much information with as little data as possible, it is valuable insofar the user catches all the proposed connections in the travel and, more generally, it lacks information on the effects of haphazard factors like traffic jams or disruptions. In the Public Mobility Platform the estimated time of arrival is proposed as a generalized concept: the Estimated Time of Arrival Cumulative Probability (ETACP) for a route is defined as the cumulative probability to arrive within a given time, considering the specific features and sources of variability of a route (e.g., number of connections, traffic probability in a line, walking time between connecting stations). ETACP is useful in when for a user it is not important to choose the fastest route, but the one that guarantees with higher probability to reach the desired destination within a given time (e.g., take a train, arrive to a office that is closing soon). How to compute ETACP in real case scenarios is shown and innovative ways to effectively represent its information content to the users are introduced.

Current travel planning applications for public transportations propose routes that, given some constraints (e.g., mode of transport, number of connections, cost), minimize the estimated time of arrival (ETA). The routes are, nevertheless, characterized by different variables and contingencies that can affect the arrival time, such as the traffic, variable waiting times at the stations, the number of connection, the delay probability of a line, the real time state of the network.

Considering these sources of variability, the fastest route could not be the best choice for a traveler, since a route that allows to arrive at destination within a certain time with high probability could be preferable to a route that could arrive earlier but with a

considerable probability to miss a connection or get stuck in a traffic jam.

ETA alone, then, does not contain all the information logistically or even only psychologically relevant to travelers. For these reasons, the adoption, in travel planning algorithms, of generalization of the scalar metric ETA is proposed: **Estimated Time of Arrival Cumulative Probability** or **ETACP**, defined as a function that describes the cumulative probability to arrive at destination within a given time, taking into account the relevant sources of variability.

The **estimated travel time** or **ETT** is the time required by a vehicle or desired to reach a given destination [28], it is calculated as:

$$ETT = (length\,of\,the\,segment) \,/\, (vehicle\,speed)$$

For routes with multiple public transportation lines, the **route estimated travel time** or **RETT** is defined as the sum of the ETT of every vehicle plus the waiting time at the intermediate stations.

$$RETT = \sum_{i=1}^{l} (waiting\,time\,at\,station\,i + ETT_i)$$

with l defined as the number of public transport lines to ride in the proposed route.

ETT and RETT can be computed using the static timetables, considering the vehicle frequency in each line and using real time data.

The **estimated time of arrival** or **ETA** is a measure of when a vehicle is expected to arrive at destination considering a given departure time [7, 56]. in public transportation is used to compute estimated arrival times depending on either static timetables or through real time measurements such as traffic or disruptions.

ETA is obtained adding the RETT at the departure time.

$$ETA = (departure\,time) + RETT$$

9.1 General Approach

The estimated travel time of a vehicle depends on several variables and possible unexpected events, thus the time of arrival could be actually described as function that describes the probability of arrival at a given time, where the ETA is the point with the highest probability. The probability to arrive before the ETA is usually low while the probability to arrive after the ETA depends on a set of variables and could be far from negligible.

Some typical arrival time probability functions (Fig. 9.1) are bell shaped, with the highest point at ETA, a small left tail (probability to arrive before the ETA) and a more conspicuous right tail (probability to arrive after the ETA). The cumulative probability distribution of the time of arrival defined as ETACP, Estimated Time of Arrival Cumulative Probability, represents the probability to arrive to destination within a given time.

A **route** is defined as the ordinated chain of public transport lines that compose a path from its origin to its destination. Each line includes the line name, the mode of transport and the departure and arrival stations.

When a route is composed of one line only the ETACP depends only on the variables of the line and can be easily computed. On the contrary, when a route is composed of

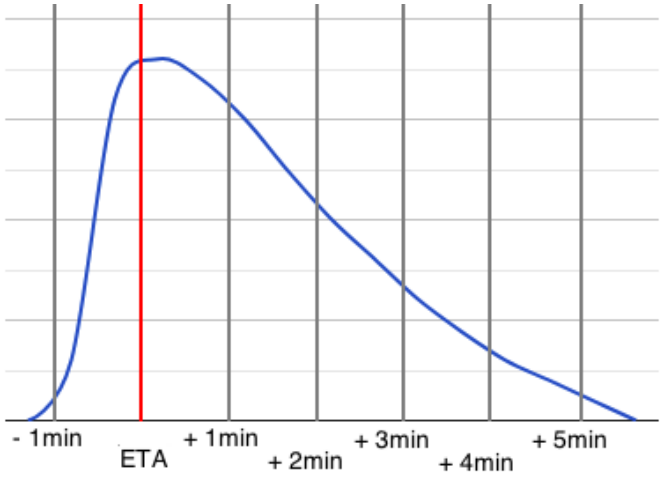

Figure 9.1: *Example of time of arrival probability distribution of a public transport line.*

more than one line, the ETACP depends on the arrival time probability of the first line plus, for each of the following lines, the probability to arrive to the interchange station before the vehicle of the following line and the time of arrival probability to destination of that vehicle.

As a simple example to clarify the concept, Fig. 9.2 shows a route with two lines; some upcoming vehicles and just a few simple variables (waiting time at the station and possibility of delay on a line) are considered.

First, let us compute ETA and its probability: the earlies ETA possible is $Tc.1$ and it is accomplished when the user catch the first arriving vehicle in Line A ($Va.1$) and in Line B ($Va.2$). This probability $P_{Tc.1}$ depends on the probability to arrive at the departure station before time $Ta.1$, the probability of delay of the vehicle $Va.1$ on Line A, the waiting time at the interchange station ($Tb.3$ - $Tb.1$), and the delay probability of the vehicle $Vb.1$ on Line B:

$P_{Tc.1} = P(T_{arrive\,to\,departure\,station} < Ta.1) * P(T_{arrive\,to\,interchange\,station} < Tb.3) * P(T_{arrive\,to\,destination\,station} < Tc.1)$

and:

$P(T_{arrive\,to\,departure\,station} < Ta.1) =$ probability to reach the departure station before Ta.1

$P(T_{arrive\,to\,interchange\,station} < Tb.3) = P(Tb.1 + T_{walk\,to\,station} < Tb.3) + P(Va.1\,delay)$

$P(T_{arrive\,to\,destination\,station} < Tc.1) = P(Vc.1\,delay)$

When the probability $P_{Tc.1}$ is lower than 100%, other possibilities are considered: if the connection in the interchange station is lost, the following vehicle on line B ($Vb.2$) can be caught and with a high probability the time of arrival at destination will be $Tc.2$. On the contrary, if the vehicle at the departure station $Va.1$ is missed, the second upcoming vehicle on Line A ($Va.2$) will be caught with a high probability. Reaching the interchange station with vehicle $Va.2$, it can be caught the second vehicle on Line B ($Vb.2$) arriving to the destination station at $Tc.2$. However, there is a low probability

to catch the vehicle $Vb.2$ because of the low waiting time $Tb.4$ - $Tb.3$ at the interchange station and because of the possibility of delay of vehicle $Va.2$. On the contrary, there is a high probability to catch the vehicle $Vb.3$ and to arrive to the destination station at time $Tc.3$ (which is the safest ETA).

As for $P_{Tc.1}$ the probability to arrive at destination at $P_{Tc.2}$:

$P_{Tc.2} = P_{Va.1\,and\,Vb.2} + P_{Va.2\,and\,Vb.2}$

$P_{Va.1\,and\,Vb.2} =$
$P(T_{arrive\,to\,departure\,station} < Ta.1) * P(Tb.2 < T_{arrive\,to\,interchange\,station} < Tb.4) *$
$P(T_{arrive\,to\,destination\,station} < Tc.2)$

$P_{Va.2\,and\,Vb.2} =$
$P(Ta.1 < T_{arrive\,to\,departure\,station} < Ta.2) * P(T_{arrive\,to\,interchange\,station} < Tb.4) *$
$P(T_{arrive\,to\,destination\,station} < Tc.2)$

Similarly, the other probabilities are estimated and the ETACP is composed. The system continues to calculate new arrival times $Tc.x$ until the cumulative probability to reach the destination at time $Tc.x$ is 100%.

Simulating a route with the described characteristics, a trip frequency of 15 minutes for both lines A and B ($Ta.2$ - $Ta.1$ = $Tb.4$ - $Tb.2$ = 15 min) and a waiting time of 2 minutes at the interchange station ($Tb.2$ - $Tb.1$ = $Tb.4$ - $Tb.3$ = 2 min) have been obtained the results shown in Fig. 9.3: in 9.3 A the probability to arrive to destination in one of the three possible ETAs (estimated time of arrival of the vehicles in the last line of the chain) is shown while in Fig. 9.3 B the ETACP function that shows the probability to arrive within a given time is illustrated.

While a traditional trip planner would propose as ETA the time $Tc.1$, with no other information and with a high probability of error, using ETACP is it clear that the probability to arrive within $Tc.1$ is only 20%, the probability to arrive within time $Tc.2$ is 75% and it is almost certain to complete the travel within time $Tc.3$. This information is very useful for travelers when they are planning a trip, especially when they have severe time constraints (e.g., take an airplane).

9.2 Probabilities Definiton

In this section the two types of probability that compose the ETACP, the probability to catch a vehicle (i.e., to be at the station before the vehicle) and the probability to arrive to destination at a given time, are discussed in more details.

9.2.1 Probability to Catch a Vehicle

The probability to catch a vehicle P_{cv}, that is the probability to arrive to the station before the vehicle, depends on the estimated waiting time wt at the station. wt is calculated considering the time remaining before the vehicle arrival, the walking distance time to the station and, if the route is composed of more than one lines, the time to arrive at the interchange station of the vehicle in the previous line:

$wt = remainingbeforethevehiclearrival - walkingdistance - timetodestinationofthepreviousveh$

A connection is considered as *safe*, thus with $P_{cv} = 100\%$, if the waiting time at the station is more than a chosen threshold (in the current implementation of the Public

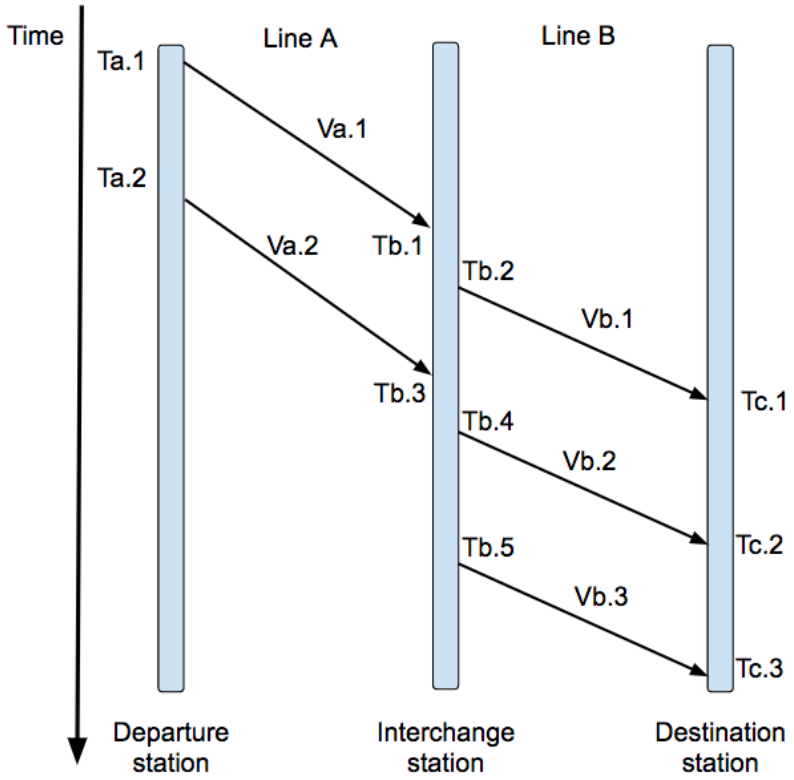

Figure 9.2: *Example of a route with a connection and 2 vehicles on each line.*

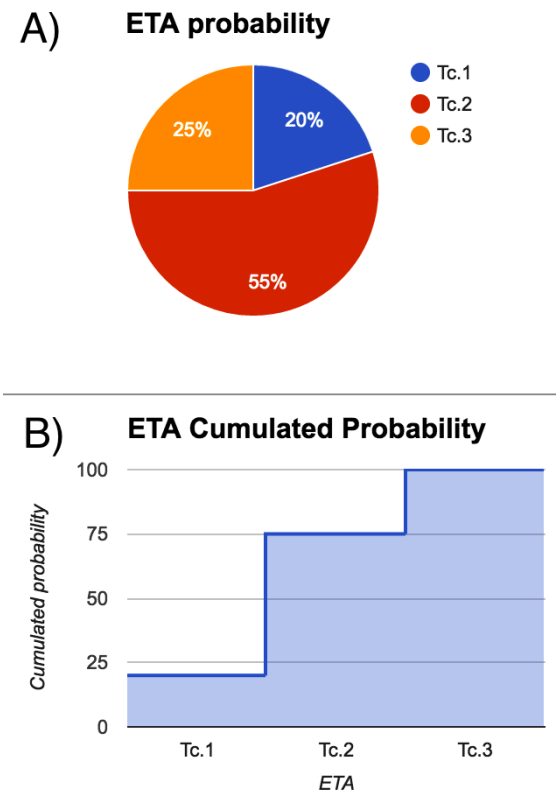

Figure 9.3: *Probability (A) and cumulative probability (B) to arrive at time Tc1, Tc2 and Tc3.*

Mobility Platform has been set to 10 minutes), otherwise it is computed using a function of the waiting time with a logarithmic behavior.

9.2.2 Arrival Time Probability

The probability for a vehicle to arrive to destination at the ETA, as discussed in the Section 9.1, is described as a *bell curve* with a long tail at its right. The probability to arrive within the ETA and the delay probability both depend on several factors, among which are:

Hour of the Day (e.g., peak, off-peak, weekend)

Peak hours are considered subject to higher probability of delay, due to the high passengers number and road-traffic intensity.

Delay Probability in the Line

It is higher for vehicles (e.g., buses) that share paths with private transportation than for vehicles with private lines (e.g., undergrounds).

Number of Intermediate Stops

The higher the number of intermediate stops, the higher is the probability to accumulate delays.

Real Time Vehicles Position

The real time position of the vehicles allows to adjust ETA in real time. The *public transport network* module is in charge of providing this information.

9.2.3 Monte Carlo Simulation

Another technique to estimate the arrival time probability is to use Monte Carlo methods [62]. This technique uses repeated random sampling to estimate the probability of events when it is difficult to use other approaches. Determine the arrival time probability is possible using a public transportation simulator (e.g. MATSim [16]) to simulate users that catch/miss the vehicles and some random line dependent delays on the public transportation lines. Executing the simulation several times, a precise approximation of the RETT and thus of the ETACP can be obtained.

9.3 Fields of Application

Applications of ETACP in the public transportation services are useful for both for the users, improving the trip planning services, and the public transport agencies.

Trip Planning and Assistance Services

ETACP can be integrated in all the current trip planning applications (that use static and / or real time data) to propose routes in an innovative way, proposing for example the first and last quartiles of the $ETACP$. These platforms would thus propose new interesting routes: users not only could choose the fastest or cheapest routes but they could

also evaluate routes with the highest cumulative probability to arrive within a certain time. For example, a user that has to be in a place within a given time T could prefer a slightly slower route that guarantees to arrive to destination with a high $ETACP_T$ (probability to arrive before time T) than a faster route with a lower $ETACP_T$.

Public Transportation Agencies

Public transportation agencies can use the ETACP to improve the evaluation of commutation time on the routes with multiple lines. This service can also be used when the transportation agencies have to update their timetables, performing what-if analysis, applying optimization policies and avoiding schedule changes that leads to low $ETACP_{Tmax}$ (within a reasonable time $Tmax$) for the most common paths and flows in the city.

9.4 Use Cases

In this section some common use cases are illustrated where the application of the ETACP can be particularly useful and interesting.

9.4.1 Routes with Low Frequency Lines

In routes with low frequency lines (e.g., buses during the night, regional trains) the ETACP has a remarkable importance, since missing a connection can increase arrival time significantly. In these cases in fact, even if the ETT is low and the $ETACP_{ETT}$ is quite high, it is interesting for a user to be informed that in case of unexpected events the ETT can increase more than what is usually expected. In scenarios where a user has, for example, to take an airplane or to be in a given place at a given hour, it is much more valuable to choose a route that guarantees to reach destination within that time than with the fastest route.

As an example, imagine a urban route with a line with frequency one hour: if for some reason (even with low probability) a connection is lost or there is a delay in one of the lines before the low frequency line, the ETA increases of one hour or more. In these situations it can be useful to notify the user that there is a (small) probability that chosen route can be completed in an unreasonable time.

9.4.2 Numerical Example

In this section, as an example, an application of the ETACP service to evaluate two simulated similar routes used to reach a given destination in Milan is proposed (Fig. 9.4). The selected destination can be reached using two different routes: *Route1* includes *Line A* and *Line B* while *Route2* includes *Line C* and *Line D*. The timetable of each Line is shown in Table 9.1. Supposing to start the trip at 13:55, the current trip planners would propose as the best route the *Route1* since, according to the timetables, it is the fastest one: in the best case the destination is reached at 14:45, while with *Route2* at 14:55. However, *Route1* include 2 risk factors: the first vehicle of Line A departs in 5 minutes and the first vehicle of Line B departs only 3 minutes after the incoming vehicle of Line A has reached its destination. If, due to a disruption or any other reason, one of the vehicles is not caught, the destination is going to be reached at 15:10 or 15:30 (at

Table 9.1: *Timetables of the lines A, B, C, D*

Line ALine A		Line BLine B	
Departure	Arrival	Departure	Arrival
14:00	14:27	14:30	14:45
14:30	14:57	14:55	15:10
		15:20	15:35
Line CLine C		Line DLine D	
Departure	Arrival	Departure	Arrival
14:05	14:35	14:40	14:55
14:15	14:45	14:50	15:05
		15:00	15:15

least 25 minutes after the suggested ETA). On the other hand, *Route2* has a longer ETT (14.55 instead of 14.45), but with less risky connections and longer waiting times.

To clarify the concept, in Fig. 9.5 are reported the ETACP of *Route1* and *Route2*: *Route2* has a ETACP of 95% to reach the arrival point at 15:05, while *Route1* has a similar probability only 25 minutes later, at 15:30.

Figure 9.4: *Proposed routes to travel from Departure Point to Arrival Point.*

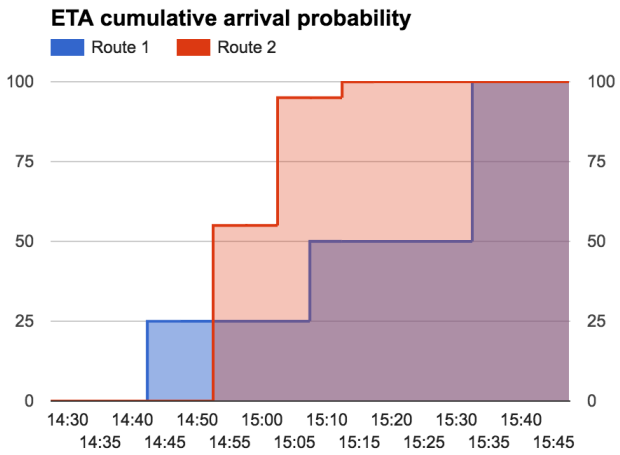

Figure 9.5: *Cumulative ETA probability of Route1 and Route2*

CHAPTER 10

Analysis, Experimental Results and Real Case Scenarios

It this chapter the relevant experimental results and the performances of the Public Mobility Platform are discussed. The platform is firstly validated using the simulator, then tested in real cases scenarios: the experiments were placed in Milan using the publicly available data of transport agency of Milan (ATM) and the major car sharing providers (Car2Go, Enjoy and Twist).

10.1 Public Transport Network State Reconstruction Module

To validate the public transport network module, both simulated and real data are used.

In order to test the designed platform synthetic data emulating a set of means of transport moving on different lines are used. These data are generated using the designed simulator, adding different kinds of noise to test the robustness and the reliability. The simulator was configured to produce as output the list of expected arrival times (expressed in minutes) of the upcoming bus for each stop of each line, which is the exact same data available in most of the cities that public information about their transport network state. The simulator also provides the state of the transport network (i.e., the location of each means of transport) in order to evaluate the accuracy of the proposed state reconstruction platform.

On the other hand, to validate the proposed approach also in real cases, the real-time data provided by ATM, the local consortium of the public transport of Milan, are collected. ATM publishes the updated real-time arrival times every 30 seconds. These data are continuously gathered, filtered (e.g., filtering out expired or missing observations and replacing them with default values) and used to feed the public network state reconstruction module.

10.1.1 Experimental results

In both cases the proposed public transport network module was able to correctly reconstruct the state of the transport network and to accurately predict the position of the different means of transport in the next future (tens of minutes). The results obtained using the ATM real data, which is for sure the most interesting case, are here presented. In particular, here are presented the results obtained from data relative to line 23 of the Milan public transport network. In Figure 10.1 are represented these data with the following conventions:

- *big green dots* indicate that a vehicle has reached a particular stop;
- *small green dots* correspond to vehicles approaching a particular stop (ETA at the stop $<$ 1 min);
- *small colored dots* correspond to ETA $>$ 1 (their color ranges from yellow to red, depending on the actual value of the ETA);
- *small black dots* have been used to identify missing data (no information about the ETA at the stop is available).

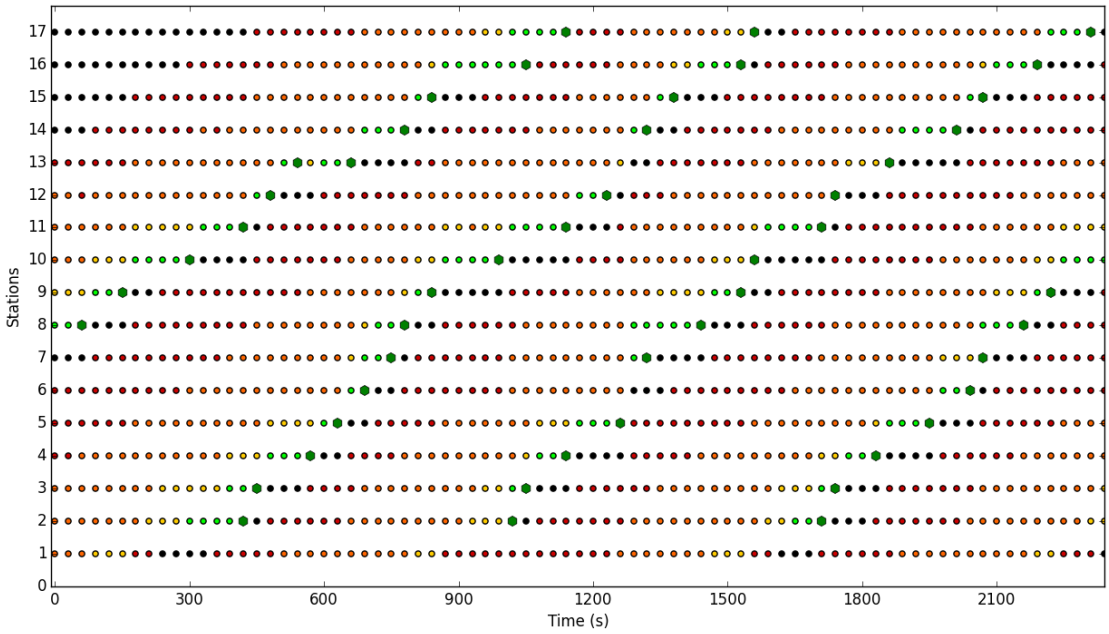

Figure 10.1: *Real-time data relative to the ETA of the stops of line 23 of the Milan transport network*

As it is possible to see in Figure 10.2, which shows how data shown in Figure 10.1 are used to identify and track moving vehicles on line 23, with the proposed approach is possible to automatically obtain a description of the transport network which is compatible with real world data, even in stops characterized by noise or missing data. In particular, Figure 10.2 shows that the proposed approach is able to track with a very high accuracy all the vehicles, each one represented by one of the solid lines, through the different snapshots of the ETA of each stop coming every 30 seconds.

Analyzing the data of different lines over a couple of hours of service has proven that the reconstruction of the public transport network state is characterized by a quite

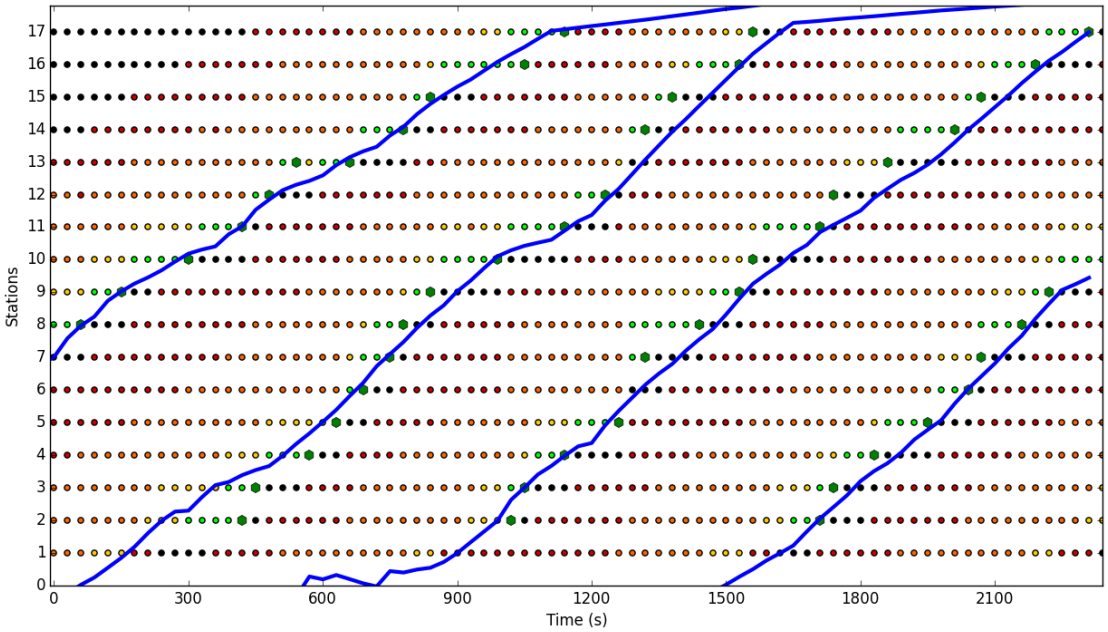

Figure 10.2: *Reconstruction of the position of the vehicles of line 23 of the Milan transport network*

high accuracy: the mean error is around 30 seconds with a variance of 0.59 and the maximum error recorded is around 3 minutes.

10.1.2 Arrival Times Predictions

As discussed in Section 5.2.3, the proposed Public Transport Network module can also be used as a predictor at any instant of time by simply letting it evolve without any observations, thus exploiting only the current state of the network (and eventually its history) to estimate the future position of each vehicle of the network.

In Figure 10.3 $time$ 1200 s is supposed to be the current time, thus all the information gathered from $time$ 0 s to $time$ 1200 s is available, as long as the positions of the vehicles estimated by the model included in the module and represented in the figure as the solid blue lines on the left. The predictions of the future positions of the vehicles are represented in Figure 10.3 by the dashed light blue lines on the right. Obviously the information plotted in black and white on the right side of the picture Figure 10.3 is not available when the predictions are generated, but it has been included it in the picture in order to validate the accuracy of the predictions, which are in fact very close to real state of the network in the next future. Several of this tests, both in simulated and real case scenarios, were executed in order to verify that, once the model learns the average speed of each section of the line, the predictions are very accurate and can thus be used to estimate the arrival time of each vehicle at the next few stops.

Moreover, as soon as new data are available (i.e., the updated ETA of each stop), the proposed module can use them to refine its predictions, as shown in Figure 10.4, generally increasing their accuracy over the time.

By performing predictions about the state of the transport network in the next 20 minutes after each update of different lines over a couple of hours of service, the observed accuracy is quite high, and their error has a mean value of 1 minute with a

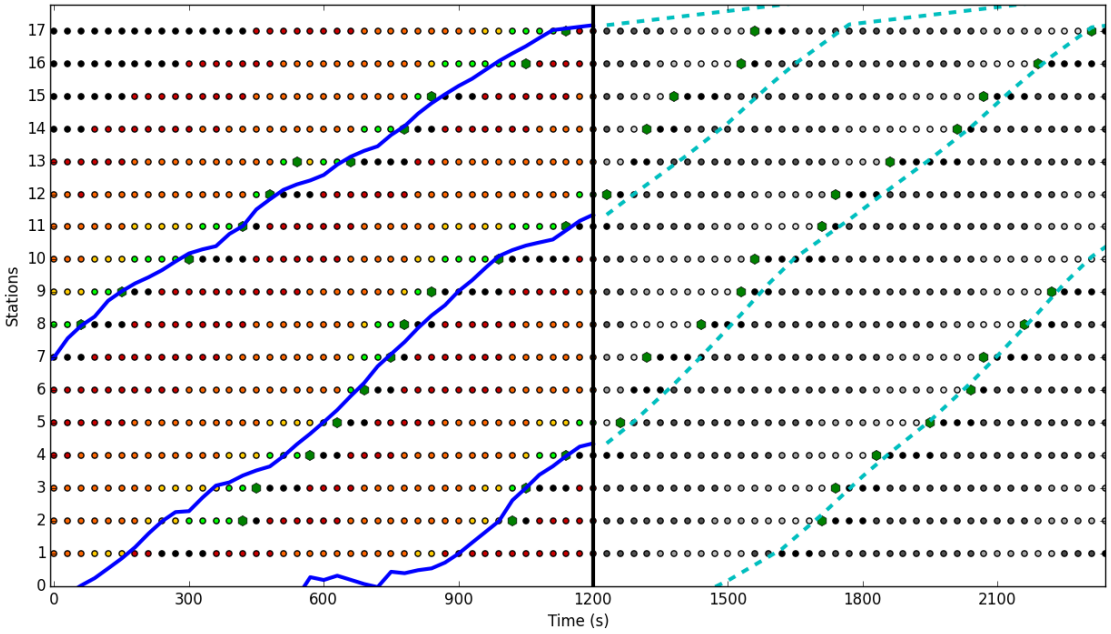

Figure 10.3: *Prediction of the arrival time at each station for the vehicles on a line in Milan*

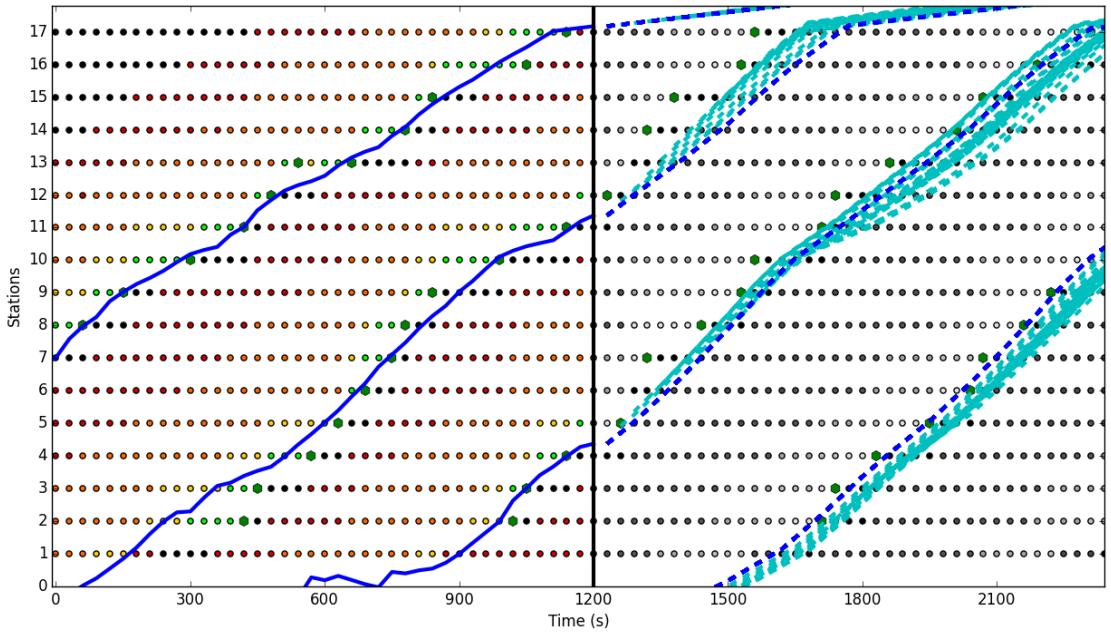

Figure 10.4: *Step-by-step predictions of the arrival time at each station for the vehicles on a line in Milan*

variance of 0.99. The maximum prediction error was of 3.5 minutes.

10.2 User Contextualization Module

In this section the effectiveness of the user contextualization module is discussed and the experimental results are illustrated. To validate the proposed user contextualization module, both simulated and real data coming from the transport agency of Milan, were

used.

The simulator can be configured to produce as output the GPS position of potential users perturbed with a Gaussian noise comparable to the real error of the GPS detected by the smartphones, these data are used to evaluate the user contextualization module.

Similarly, to validate the user contextualization platform, the GPS of real users was collected during several trips in Milan traveling on the public means of transport.

10.2.1 Experimental Results

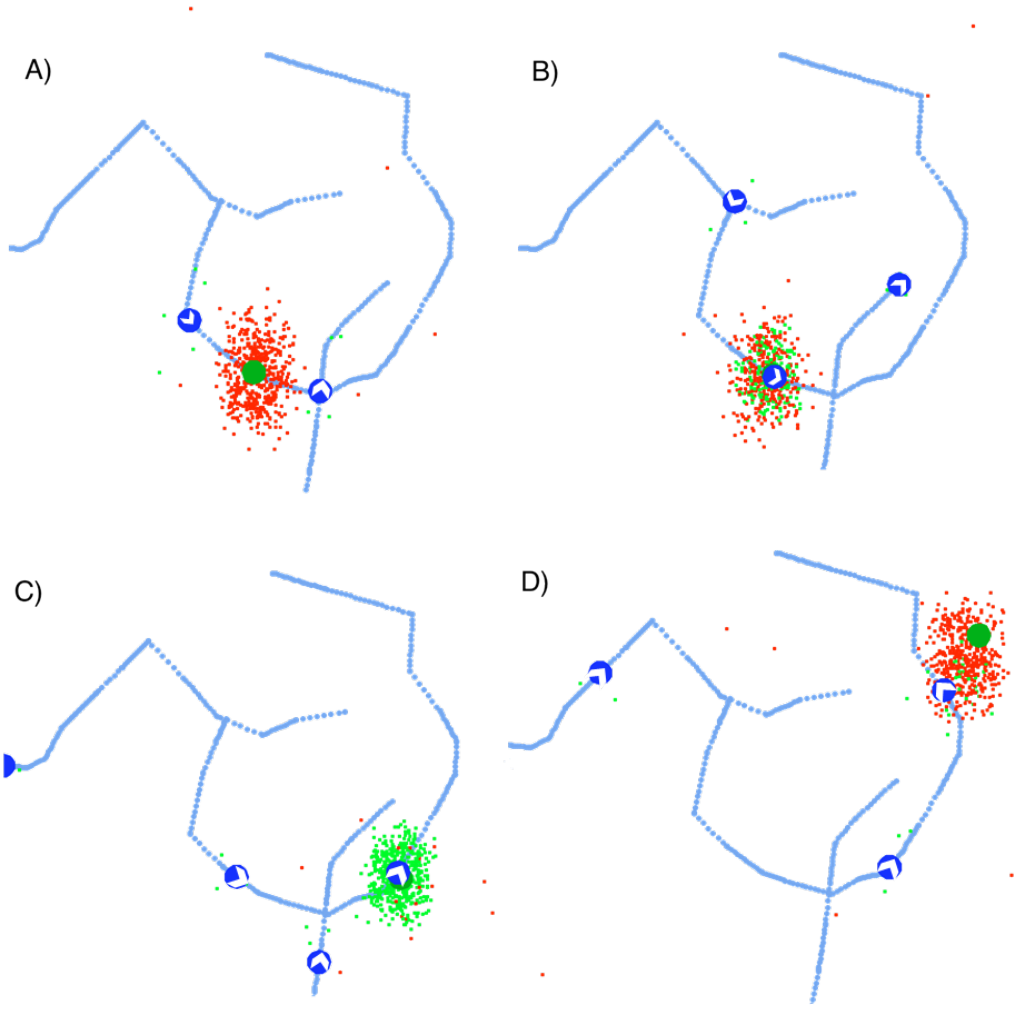

Figure 10.5: *Example of user contextualization with simulated data in a scenario with 3 different lines.*

The user contextualization platform was tested with the data provided by the simulator and then validated using real time data. Figure 10.5 shows a simulation of a user traveling on a line and has the following conventions:

- *light blue lines:* track of the vehicles of the different lines;
- *big blue dots:* vehicles traveling on the corresponding lines;

- *big green dot:* estimated user position (weighted mean of the particles);
- *small red dots:* walking particles (WP) at the current step, as described in Chapter 7.0.2;
- *small green dots:* vehicle particles (VP) at the current step, as described in Chapter 7.0.2.

In Graph A of Figure 10.5 the user is waiting at the station, the majority of the particles are WPs and just a few VPs are spread around the vehicles to explore if the user is close to that vehicle. Graph B shows a train approaching the station close to the user and the VPs start spreading: almost half of the particles are WPs and the others are VPs. In Graph C the majority of the particles are VPs, that means that the user is correctly estimated on the vehicle (the vehicle particles are always associated to a specific vehicle). In Graph D the user has reached the desired destination and the majority of the particles switch back to WPs.

As shown, the proposed approach can detect the users that are walking and ones that are traveling on vehicles, identifying the specific vehicles and lines. Several tests in simulated and real case scenarios showed the user contextualization module estimated the user correctly with around 96% of accuracy.

Degenerate Case

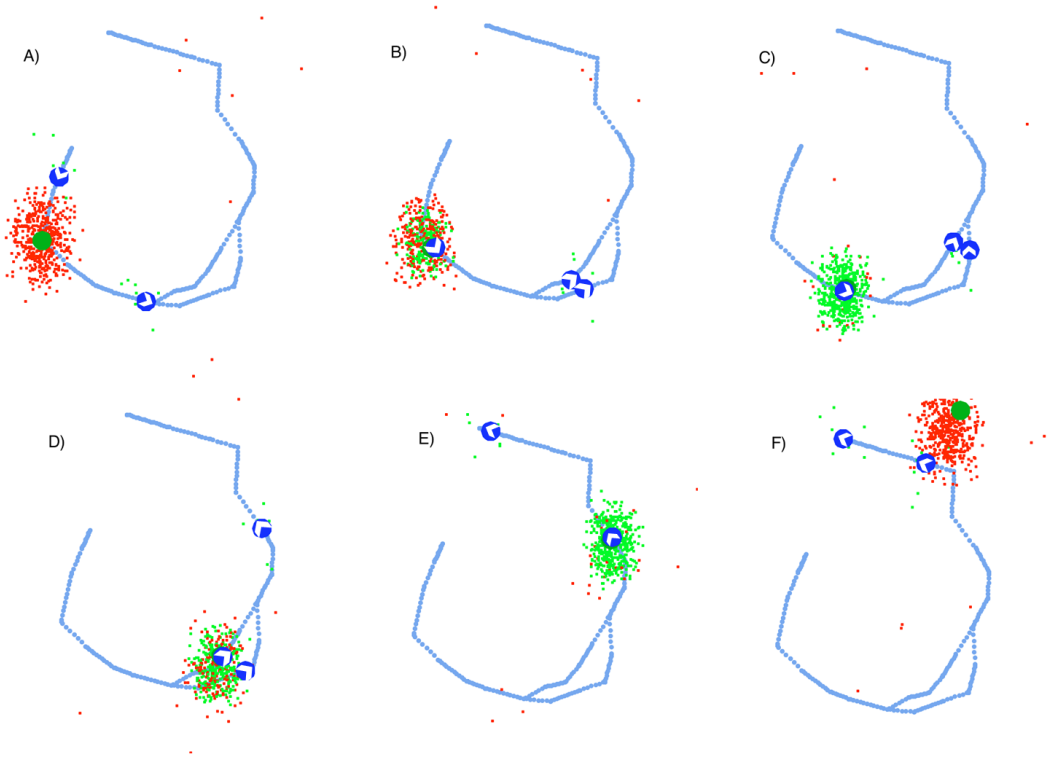

Figure 10.6: *Example of user contextualization with simulated data in a scenario with 2 partially overlapping lines A (internal) and B (external).*

Effectiveness of the proposed process in case of vehicles of different lines traveling closely (i.e., one next to the other) on same lane as then been evaluated. For this reason,

as shown in Figure 10.6, a scenario was simulated where two lines (A and B) are firstly overlapped, then separated and eventually overlapped again. Graph A of Figure 10.6 shows a user waiting for a vehicle on line A at the station (majority of walking particles, like in Figure 10.5). In Graph B two vehicles of two different lines (A and B) are approaching the station: the particles are half WPs and half VPs (almost half of them belonging to vehicle A and half to vehicle B). In Graph C the user is traveling and the majority of the particles are VPs, but almost half of the particles still belong to vehicle on line A and the other half to vehicle on line B: it is still not possible to accurately estimate on which vehicle the user is (the only known information is when the user is on a vehicle belonging to line A or B). In Graph D the lines are separated, the majority of the particles belongs now to line A (the one the user is traveling on) and it is now possible to correctly estimate the vehicle of the user. In Graph E the lines are overlapped again but, since the majority of the new particles are resampled from particles belonging to vehicle on line A, the majority of the particles still belongs to vehicle A and the correct estimation of the user is still possible. Graph F shows that the user has reached the desired destination and the majority of the particles switch back to WPs.

10.3 Real Scenario Full Stack Experiments

To generally evaluate the performances of the Public Mobility Platform, the state reconstruction module and the user contextualization module were used together to conduct several experiments in the City of Milan. In particular, in different tests of some hours each, the arrival times at the stations provided by the local public transport consortium were stored and, at the same time, the GPS position of some users traveling around the city using the public transport network was recorded.

The arrival times were used to reconstruct the public transport network state and, at the same time, the reconstructed state and the GPS position of the users were used to map the users on the vehicles they were traveling on. Several experiments were conducted, considering 2 main classes of test: in the first one the line is assumed to be known (e.g., automatically if users used one of the travel planner applications related to the platform to compute the route, or manually, asking them the line number) and the contextualization module had to identify only the specific vehicle. In the other class, there was no information about the line and the platform had to identify both the correct line and the specific vehicle.

Among the different tests, here a typical traveling scenario is here proposed: the analyzed case is a route that consists of a trip on a tram of the line 23, with a vehicle of line 33 very close to it in the first part of the route (line 23 and line 33 share the first stations of their path).

The first experiment, where the user chosen line was known, the platform estimated the correct state (*walking* / *onboard*) with an accuracy of 96%.

The second experiment, where there was no information about the state of the users and their route, the platform estimated the correct state with an accuracy of 92% and the correct vehicle with an accuracy of 71%. In the first part of the route, with one vehicle of line 23 and one of line 33 very close to each other, the vehicle estimation was not accurate and the estimation of the vehicle was sometimes switched from one vehicle to the other. In the second part of the route, with only one vehicle, the vehicle

Table 10.1: *Robustness to noise in user position with known line*

Noise (decimal degrees)	correct estimation
No	96%
0.001	96%
0.002	94%
0.003	87%
0.004	79%
0.005	51%

Average percentage of correctly estimated steps (state and vehicle) during several travels in Milan, known the line the user is traveling on.

Table 10.2: *Robustness to noise in user position with unknown line*

Noise (decimal degrees)	correct estimation	correct state, wrong line
No	72%	21%
0.001	69%	20%
0.002	56%	31%
0.003	55%	22%
0.004	49%	25%
0.005	39%	40%

Average percentage of correctly estimated states and steps (specific vehicles and states) during several travels in Milan, unknown the line the user is traveling on.

estimation was instead quite accurate.

Similar performances were obtained in other full stack experiments, with usually higher accuracy where there were not vehicles of different lines traveling close to each other.

Nevertheless, for all the services that do not require real-time user contextualization (e.g., offline route analysis), a backpropagation [53] algorithm can be used to correctly estimate the specific vehicle even in the first part of the route.

10.3.1 Robustness of the Approach

The experiments were repeated adding a Gaussian noise to the user GPS observations to test the robustness of the proposed approach, for example to simulate observations using less accurate mobile tracking systems (e.g., using cell triangulation to save battery power). The experiments of the previous section were reproposed (known or unknown line) adding to the user GPS position a Gaussian noise with a standard deviation of 0.001, 0.002, 0.003, 0.004 and 0.005 decimal degrees (0.001 is approximately 100 meters).

The results, shown respectively in Table 10.1 and Table 10.2, prove that this approach is effective even in presence of small and medium noise, while the performances drops in presence of extreme noise.

10.4 Road Network State Reconstruction Module

Like in the previous sections, the study focus the Milan Metropolitan Area, that is the City of Milan and the surrounding areas organically connected where the car sharing

services are available. The services considered are the currently major car sharing providers: *Car2go*, *Enjoy* and *Twist*.

The car sharing data are gathered, using the available API or through data scraping techniques, from the websites of the operators. A dedicated server recorded the position of each car over the course of several days.

In the latest version of the Public Mobility Platform the Twist provider was removed because it is not active anymore. In the following sections this provider is proposed anyway for analysis purposes.

10.4.1 Data aggregation

As explained in [55] aggregation tools have the ability to organize data in a way that they can be quickly searched, analyzed and efficiently utilized.

All the analysis and process are executed using Pandas [52], a Python library providing high-performance, easy-to-use data structures and data analysis tools.

Three types of aggregation were considered: per provider, per time and per zone.

Aggregation per Provider

Enjoy and Car2Go are widely used and their customers show similar usage patterns, with Enjoy being the provider with the major number of cars hired most of the times (also due to the fact that at the time of the study it was the provider with the highest number of cars). Twist is much less widespread, and its vehicles are almost always available but not used.

In each of the following graphs, the highest line is the sum of the three providers while the other three lines represent the three providers: the red line is for Enjoy cars, the azure one is for Car2Go cars and the blue one for Twist cars.

Aggregation per Time

The data are firstly divided in two groups, **weekday data** (form Monday to Friday) and **weekend data** (Saturday and Sunday), and then aggregated per hour. The hired vehicles are calculated, for each provider, subtracting the available vehicles from the total of the vehicles.

Results are shown in Figure 10.7. During the weekdays it is easy to identify two usage peaks, one form 10:00 am to 12:00 am (more than 300 car used at the same time) and the other from 8:00 pm to 22:00 pm (more than 400 car used at the same time). On the other hand, there are just a few vehicles used (less than 60 car used at the same time) during the night, especially from 4:00 am to 8:00 am. During the weekend the car are used more uniformly, even during the night, with an average of 200 cars used at the same time between 5:00 am and 11:00 am and an average of 420 cars used at the same time during the rest of the day. The information about the hired cars is correlated with the traffic on the city roads, thus it is the first indicator for the reconstruction of the road network state.

The data of the two previous groups are divided in three subgroups, based on the GPS of the car, as shown in Fig. 10.8 and listed here:

- **City center**, the so called *cerchia dei Bastioni*, that delimits the historic center of Milan.

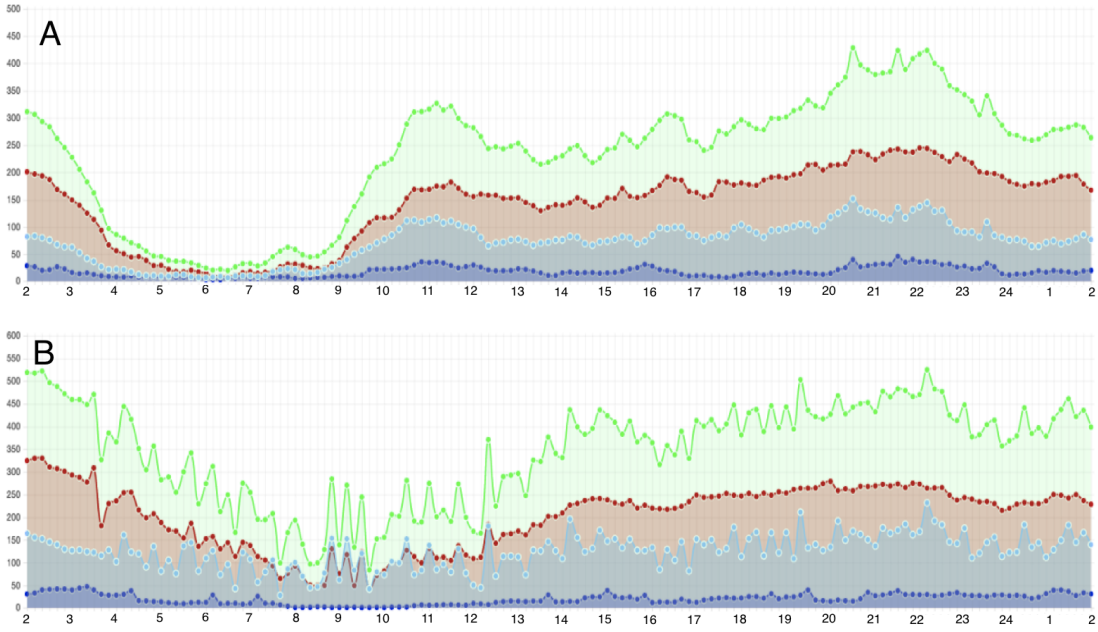

Figure 10.7: *Average cars hired per hour, during the weekday (A) and during the weekend (B).*

- **Circumvallation**, the area between the city center and the so called *circonvallazione filoviaria* that delimits the historic urban area of Milan.
- **Outskirts**, all the new urban area of Milan outiside the *circumvallation*.

Aggregation per Zone

To interpret the data, the city was clustered in three areas, characterized by their urban function: the city center contains the majority of the offices and the shops, the area between the city center and the circumvallation includes shops, offices and houses, while the outskirts are the more popular area and includes the majority of the residential zones.

In Figure 10.9 the availability of cars in the city center during the weekdays (a) and the weekend (b) is shown: in the weekdays there is good availability of cars (at least 30 cars) between 7:00 am and 5:30 pm, while there is scarce availability during the rest of the day. Weekend situation is similar, with a good availability during the day (from 10:00 am to 5:00 pm) and low for the rest of the time.

In Figure 10.10 the availability of cars in the circumvallation during the weekdays (A) and the weekend (B) is shown: both in the weekdays and in the weekend there is a good availability of cars, with more than 200 cars always available. This proves that many people moves in this area of the city.

In Figure 10.11 the outskirts, the biggest area of the city, are considered, during weekdays (A) and the weekend (B). the majority of the cars is obviously here with the highest concentration during the night (more than 900 cars available from 10:30 pm to 9:30 am during the weekdays and form 4:30 to 12:30 during the weekends), when people came back from work or from the bars and restaurants in the city center.

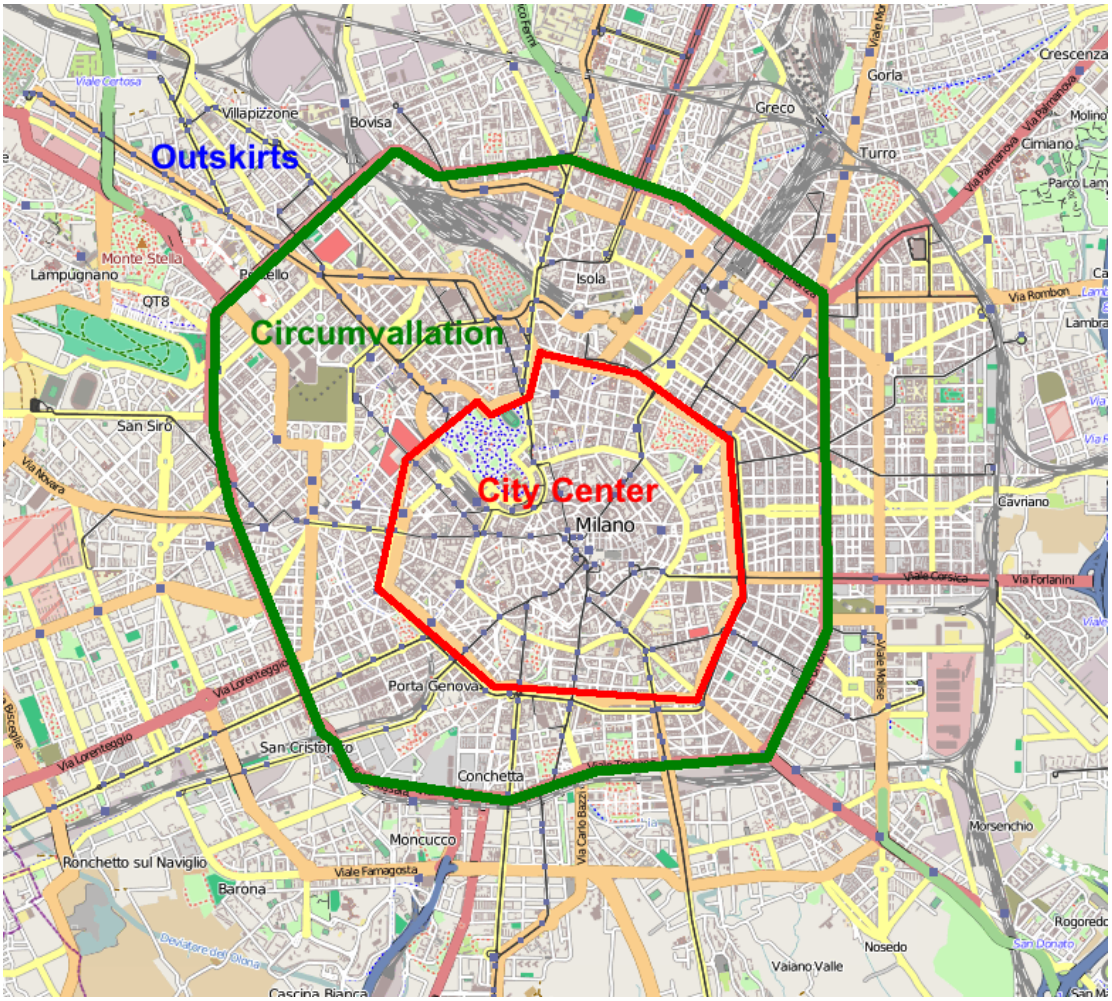

Figure 10.8: *Division of the city in the three major areas: city center, circumvallation and outskirts*

These analysis are used to determine the cars flows, that are, as expected, in the morning from the outskirts to the center and in the evening from the center to the outskirts. To determine the **cars flows** accurately in the Public Mobility Platform, a similar analysis was done dividing the city in more zones considering the main city roads and the major points of interest.

Travel Time

The **travel time** is calculated as the difference between when a user book / rent a car and when the user completes the travel. The computed travel time thus include also the time from when the car is booked and when the travel actually starts.

Figure 10.12 shows the cumulative travel time: approximately 10% of the travels ends in 10 minutes, half of the travels are concluded in half an hour, 75% of the travels end in 1h. The majority of the travels (about 40% of the total) last between 20 and 40 minutes.

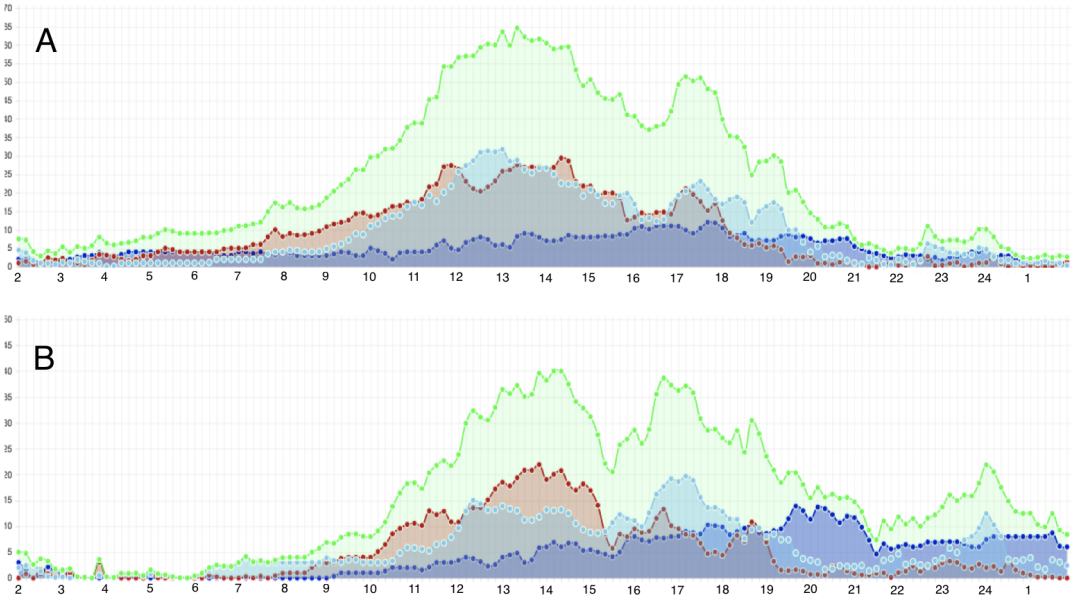

Figure 10.9: *Average number of cars available in the city center during the weekdays (A) and the weekend (B)*

Average Speed and Fuel Consumption

To compute average speed and fuel consumption of each travel, the *distance traveled* is firstly computed as the length of the best route proposed by a route planner (there is no information about the real routes). The **average speed** is defined as the difference between the distance traveled and the travel time and the **fuel consumption per km** is defined as the difference between the fuel consumption and the distance traveled.

Even if the results are rough, due to the approximations and the quality of the data (especially for the fuel), in Figure 10.13 are shown clear trends in the average speed and fuel consumption: the blue line shows the averages for the weekdays, while green line shows the averages for the weekends. It is clear that the average speed is much higher during the night (triple fo the day hours), the average speed remains high until 6:30am during the weekdays and until 9:00am during the weekends. On the contrary, the fuel consumption is higher during the day, with 2 peaks: one at 13:00pm and one at 17:00pm.

10.5 Network Behaviors Classification

In this section the initial results of the different innovative deep neural network designs proposed in Chapter 8 are discussed. Training deep learning networks requires a big amount of data and collecting them with the Public Mobility Platform requires a considerable amount of time. For this reason the experiments discussed here use an already available online dataset provided by the City of New York. New experiments will be executed when enough data from the public transport and road network will be available. Purpose of this section is to provide an overview of the techniques tested and evaluated (without discussing the specific implementation details) and to demonstrate

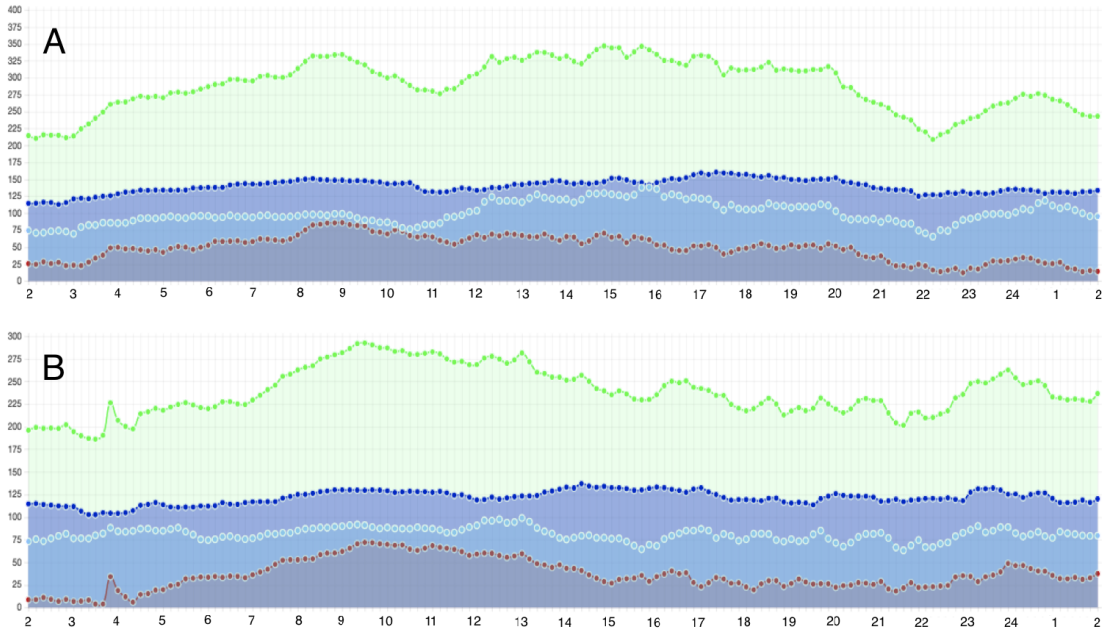

Figure 10.10: *Average number of cars available in the circumvallation during the weekdays (A) and the weekend (B).*

the possibility to use deep neural networks in mobility scenarios and to automatically explain the classification results.

In the current experiments the different network designs are trained to predict the tips of the taxi rides. The objective is to detect the features (if they exist) that determine the tip value beside the trivial ones, such as distance and time.

10.5.1 Taxi Dataset

To test the different approaches the **yellow and green taxi trip records** available online (`http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`) is used. This dataset includes fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data used in the attached datasets were collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP).

Each record contains the following fields:

- tpep_pickup_datetime
- tpep_dropoff_datetime
- passenger_count
- trip_distance
- pickup_longitude

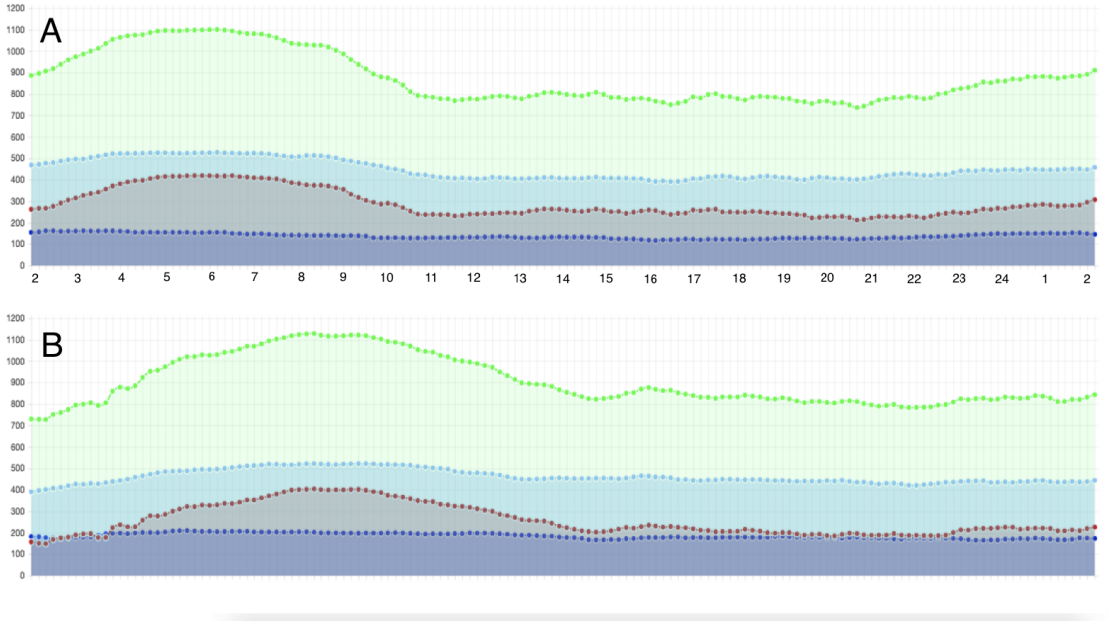

Figure 10.11: *Average number of cars available in the outskirts area during the weekdays (A) and the weekend (B).*

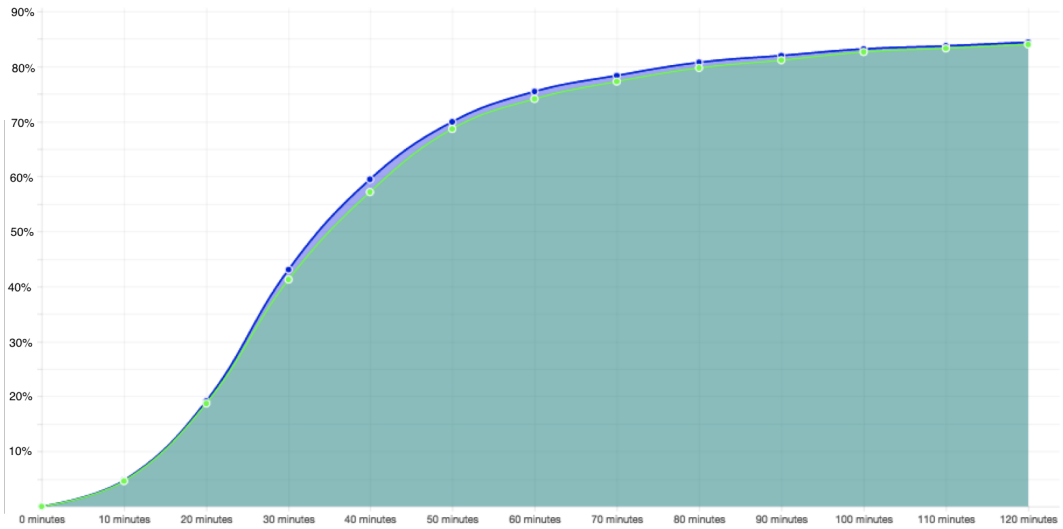

Figure 10.12: *Cumulative travel time.*

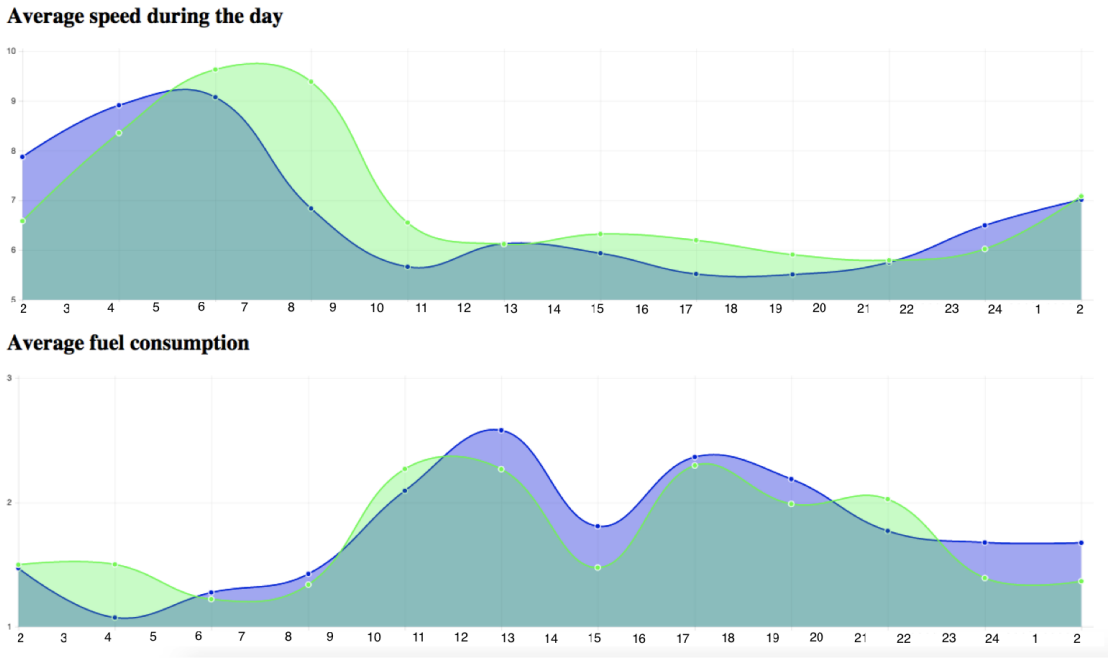

Figure 10.13: *average speed (A) and average fuel consumption (B) during the day.*

- pickup_latitude
- dropoff_longitude
- dropoff_latitude
- fare_amount
- extra
- mta_tax
- tolls_amount
- improvement_surcharge

Even if in these approaches it is not strictly necessary, in our experiments the inputs were normalized using **feature scaling** techniques. The input standardization improves the training speed and it reduces the chances of getting stuck in local optima [3], moreover weight decay and Bayesian estimation can be done more conveniently.

Any scaling that sets to zero the mean or median or other measure of central tendency is likely to be a good, and robust estimators of location and scale [37]. For this reason, in this approach, the inputs are standardized to [-1,1].

Tip Classes Quintiles

To classify the tips, the dataset is divided into fifths and classified accordingly.

Five class corresponding to the 5 quintiles are defined:

- Class 1: 1st quintile.

- Class 2: 2nd quintile.
- Class 3: 3rd quintile.
- Class 4: 4th quintile.
- Class 5: 5th quintile.

Route Computation

In the studied dataset the only geographical information regards the GPS coordinates of the pick up and drop off points, to test this approach the possible route of each taxi entry is thus estimated computing the shortest path from the departure node to the arrival node. As a reference graph, the graph provided by **OpenStreetMap** is used, and for each taxi ride the nodes with the closest coordinates to the departure and arrival points are selected. To compute the optimal path an implementation of the Dijkstra's algorithm is used.

The optimal path detected in the graph is represented as a vector that includes all the nodes of the city, where the crossed nodes are set to 1 and the others to 0.

Another studied approach concerns the usage of an adjacency matrix to represent the ride path: however, this representation is highly demanding from the memory and computational points of view. Moreover, it does not scale very well (i.e., it grows quadratically in function of the graph size) and thus it can not be used in big networks with several nodes.

The nodes vector approach, on the other hand, increases linearly with the number of nodes, allowing the exploitation of bigger graphs. Moreover, as already discussed, nodes vector can also be used to represent conveniently temporal information.

10.5.2 Modular Network Experiments

In the current implementation 4 main sub-networks (NSN) were identified: departure NSN, arrival NSN, ride information NSN and fare NSN. The departure and arrival NSNs (Fig. 10.15) use 3 input neurons, corresponding to 3 data fields: latitude, longitude and time. The ride information NSNs (Fig. 10.16) requires 2 input fields: passenger count and trip distance. The fare NSNs (Fig. 10.17) requires 5 input fields: fare amount, extra, mta tax, tolls amount and improvement surcharge.

Each sub-network has 1 neuron in the output layer, these outputs are used as input in the general neural network with one or more hidden layers (an example in Fig. 10.14). The output layer of the general neural network predicts the tip class.

The analysis of the output layer of the sub-networks is used to explain the predicted tip class.

Results

Each sub-network used a hidden layer with 3 neurons, the output are used as input on another neural network with a hidden layer with 3 neurons. After several experiments with different parameters, the network was trained using 5000 samples per class and 10000 learning iterations. With this configuration, the modular neural network correctly estimates around 80% of the tips. More designs were tested increasing the number of neurons and of layers obtaining similar performances.

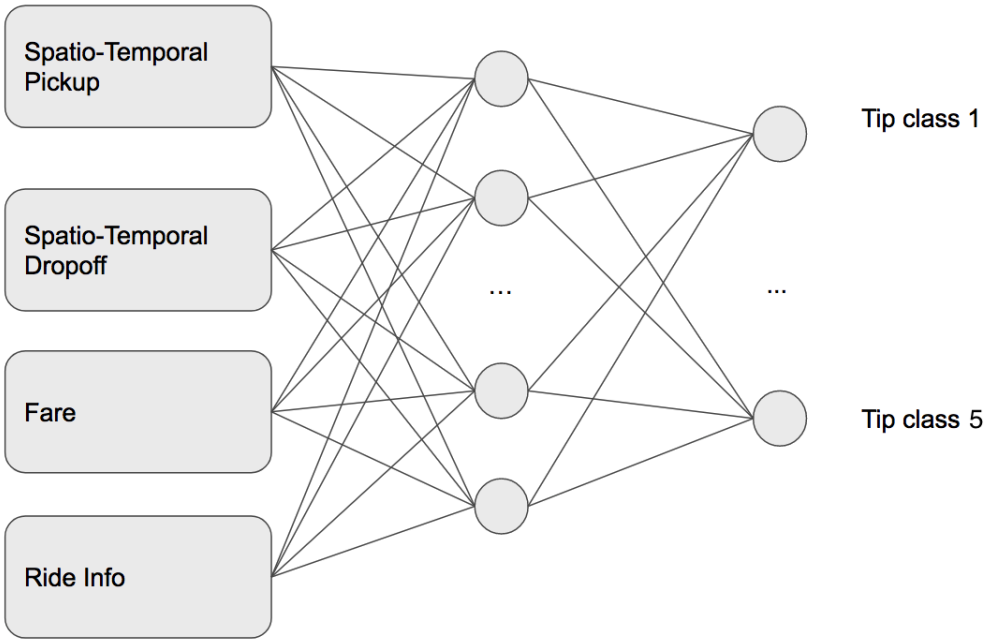

Figure 10.14: *The main network.*

The developed explanation script, in charge of providing an explanation of the classification based on the input features, generally confirmed, as easily predictable, that the tip value strongly depends on the cost of the ride (i.e., the fare neural sub-network). Even if the results were somehow trivial, an innovative explainable approach based on engineered features was successfully tested and evaluated.

10.5.3 Network Based Input Experiment

In the last section the input features were previously identified and selected, on the other hand, in this section a technique to classify the tips without engineered features and using only network information is proposed. Purpose of this approach is to test innovative techniques in the explainable deep learning field applicable to all the graphs.

Purpose of the **network based input approach** is to automatically extract interesting explainable features from the routes of the taxis and classify the tips accordingly. As explained in 8.1.2, the input of these deep neural networks is a *nodes vector*. The **nodes vector** is the binary vector containing all the nodes in a graph, with one or more nodes crossed by a path set to 1 and the others to 0.

Several deep network designs with different input were tested and below discussed. The number of hidden layers in the neural networks and the number of neurons in each layer are not discussed here, since the optimal values are dependent on the type of input graph and the dataset, thus not relevant in this general analysis.

Full static node network

The **full static node network** is the simplest design: the input is the nodes vector with all the nodes crossed by a path set to 1, thus with no temporal information. Using this

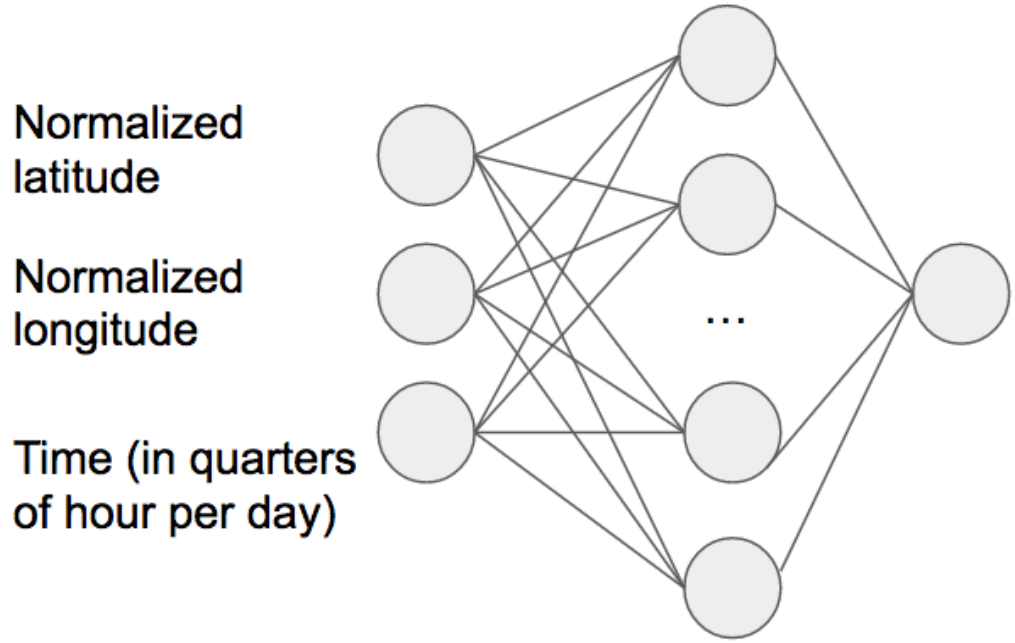

Figure 10.15: *Spatio-temporal feature sub-network.*

deep neural network, the best classification accuracy obtained in the same dataset of the previous experiments is around 50%.

Full Temporal Node Networks

In the **full temporal node network** the nodes vector is replicated n times, composing a sparse nodes matrix where each column corresponds to a specific node crossed by a path of maximum length n. Each column thus contains only one node set to 1, this representation therefore includes the temporal order in which the nodes are crossed in a path.

The **full static node network with end point first** is an alternative representation of the *full temporal node network* where the end point of a path is in the first position in the nodes matrix. This representation can be useful when the last point in a path is more relevant that the others and having it in a fixed position allows the deep neural network to learn this information (in the other full temporal node network the end point position depends on the length of the path).

Another alternative implementation tested is the **full temporal node network with only first k' and last k" nodes**. In this implementation, instead of representing all the path, only the first and the last k nodes are included in the input nodes matrix. This implementation increase the learning speed of the network but, clearly, the information regarding the central part of a path is lost. However in some types of networks this information is not as relevant as the beginning and the end of the path.

The first design has a classification accuracy of around 60%, while the second and third perform a bit better (respectively 67% and 64% of accuracy) suggesting that the

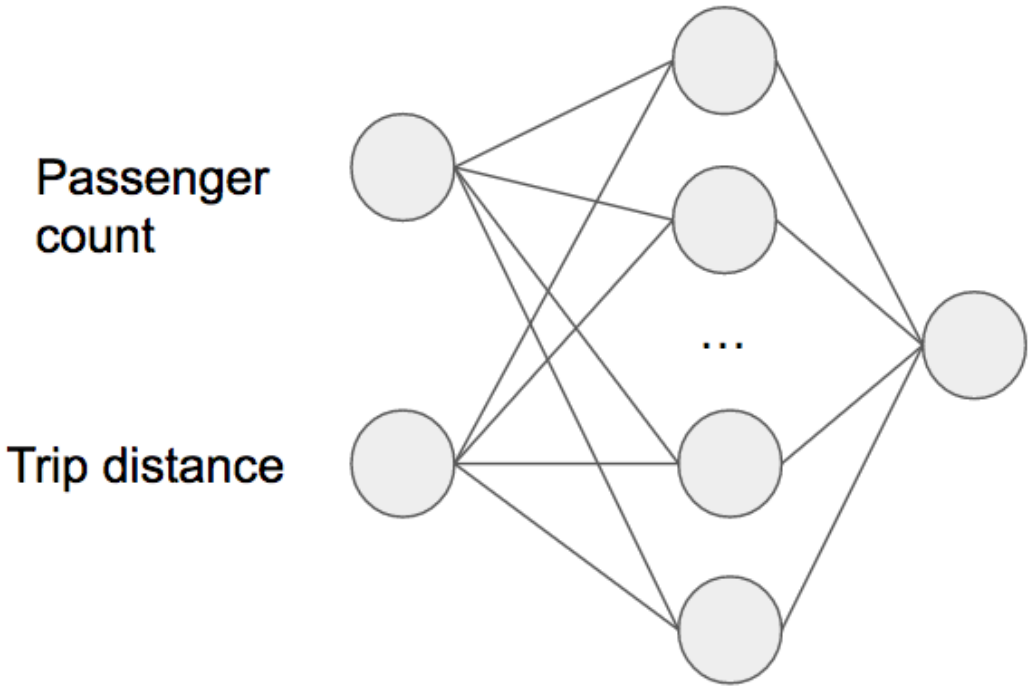

Figure 10.16: *Ride information sub-network.*

drop off point of a taxi ride influences the tip class and the starting and arrival points are more important indicators than the rest of the ride.

Clustered Temporal Node Network

The **clustered temporal node network** exploits nodes clustering to improve the learning speed. The nodes in a path are divided in temporally ordered groups and each nodes vector in the input matrix contains k nodes. In this approach the fine grained temporal information is lost, but a rough (depending on the size k of the clusters) temporal information of the whole path is conserved. If $k = 1$ this implementation is equivalent to a full temporal node network, while if $k = length\ of\ the\ path$ this implementation is equivalent to the full static node network. This approach with a reasonable size of k has performances similar to the ones of the full temporal nodes network but with a higher learning speed and a lower memory usage.

Explainable Input Based Network

Analyzing the weight of the neurons allows to identify the parts of the network which influence the classification of the tips. From the experiments it is deduced that the length of the network generally influence the tip class. Other factors that influence the tip are certain pick up and drop off points, such as airports and famous points of interest in the city.

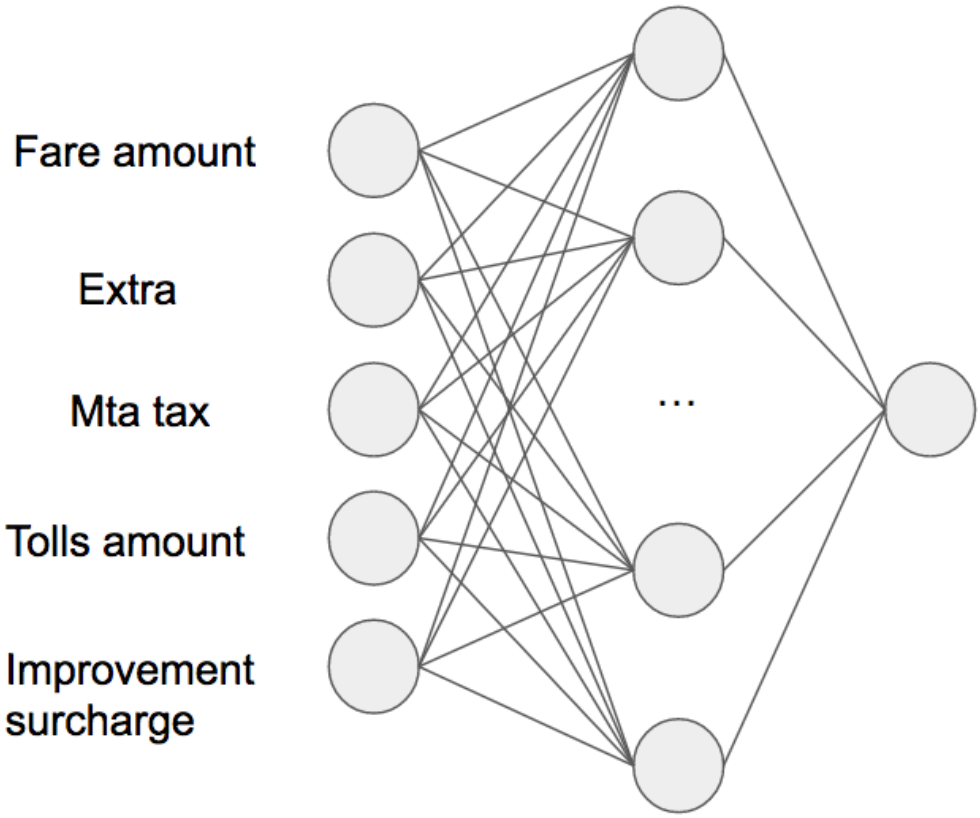

Figure 10.17: *Fare feature sub-network.*

CHAPTER 11

Applications

In this chapter are proposed some examples of innovative public transport applications in the context of the smart mobility already developed, or that can be developed, using the Public Mobility Platform.

11.1 Trip Planning and Assistance Application

Aim of this section is showing how it is possible to realize a multimodal real-time travel assistance application using the information provided by the Public Mobility Platform.

The **trip planning and assistance** application allows to compute the optimal journey to reach the desired destination at a certain time. These journeys are optimized considering the most advanced generated information, such as, the dynamic real-time computation and constantly update of the information about connections, waiting times and disruptions.

An innovative feature of the application is the possibility of managing disruptions of the public transport service in real-time, replanning the routes and proposing the best solutions.

Given a starting point and a destination and using the updated information provided by the Public Mobility Platform, this application proposes the best route considering a number of parameters such as user's preferences in terms of type of mean (e.g., no underground lines), walking segments (e.g., change only if the stations are less than 100 meters one from the other), specific lines or stations (e.g., the user knows that there is a problem at a particular station, so the application can be manually configured to exclude journeys traveling through that specific station).

The trip planner and assistance application consists of a web app dedicated to the public transport users that offers an innovative planning and assistant service before

and while traveling. This app features the following relevant functions:

- Path planning based on the whole available real-time information.
- Detailed visualization of the route including stops and line changes.
- Contextualized real-time assistance during the trip, constantly updated arrival time, possible real-time deviations from the previously-planned itinerary in case of disruptions.
- Removal of station and lines not to be considered in the planning.
- Social network and gamification activities for the users traveling on the same vehicle.

11.1.1 Geocoding

To determine the departure and the arrival of a route, given the addresses in text format, a geocoding service based on open source and cooperative technologies and systems was developed.

The cartographic information were gathered using OpenStreetMap (OSM) [1]: a collaborative open source project with the main purpose of creating maps and cartography.

Beyond the great accuracy, the geographical data available on the OSM are also license-free (Open Database License). In other words, it is possibile to use them freely for any purpose with the only conditions of quoting the source and use the same license for all the studies derived from OSM data. Another important characteristic of OSM is that everybody could contribute enriching or editing the data, improving the accuracy and the continuously updating of the maps.

11.1.2 Routes Calculation

Using the previously described geocoding service, the departure and destination addresses are converted into GPS coordinates and all the potential arrival and departure stations are selected choosing among the ones in the range of the departure and the arrival points: the range width is configurable according to the maximum walking distance the user is prone to cover.

The trip departure and arrival nodes are added to the time-expanded public transport network graph (as described in Chapter 4.1) connecting them to the possible departing and arrival stations previously detected. An optimal route computation algorithm (described in Section 11.1.3) is executed on this graph to estimate the best itineraries from the departure node to the arrival one and the best options in that precise hour of the day are shown.

11.1.3 Description of the Optimal Route Computation Algorithm

The chosen algorithm is customized version of the Dijkstra's algorithm created to work on the designed graph.

Path Computation Algorithm

On the public transport graph a modified version of the well-known Dijkstra's algorithm [69] is used to compute the optimal paths. To each node is associated a label whose value can change during the iterations (representing the value of the best path found to reach the node until that iteration) and at end of the algorithm represents the minimum arrival time to reach the node. At the beginning, the origin node is assigned a label equal to departure time of the user. All stop nodes with timestamp lower than the departure time of the user are neglected since they correspond to events already happened (i.e., rides already carried out). At each iteration, the node with minimum label v is selected, its label becomes definitive and the labels of all its successors are updated. If the successor is a station node w, its label $L[w]$ is updated according to the usual rule of Dijkstra's algorithm:

$L[w] = minL[w], L[v] + c_{vw}$

where c_{vw} is the cost of the edge (v, w) and $L[v]$ is the label of node v. Differently from Dijkstra's algorithm, the label of each stop node is not computed through the previous formula, since it coincides with its timestamp. The algorithm ends when the label of one station node in N_A becomes definitive.

Alternative Routes in Case of Disruptions or Emergencies

One of the innovative features of the trip planning and assistance application, is its ability to manage unexpected events, delays and service disruptions concerning public transportation in real-time. It allows to model a range of service disruptions, much wider than those possible with approaches currently proposed by transit agencies.

An example of typical disruption is the temporary inaccessibility of a station, with the railways still operative, due to external events such as weather (e.g., entrances flooding for a thunderstorm), public order (e.g., access prohibited for security reasons) or concerning the service (e.g., emergency maintenance activities). In these cases the platform offers the possibility to plan routes passing through this station, without considering it as a starting or arrival point. Moreover, to manage the event of sudden station closures (e.g., due to damages, strikes), the route planning application offers the possibility to exclude one or many stations in the itinerary calculation.

The application allows to manage different types of possible disruptions:

1. Station fully closed: both the access and the train transit are forbidden.
2. Station access closed: entrances and exits to the station are closed but transit through the stations possible.
3. Trip cancelled: a vehicle on a line is suppressed.

According to these types of disruptions, the application manages the problem proposing alternative paths in accordance with certain conditions.

In the proposed graphic representations, stop nodes are marked with numbers while station nodes are indicated with the letter S: in case of disruption with a station completely closed (Fig. 11.1), the relevant station nodes ($S3$) are closed and also the relative stop nodes (3 and $3'$) are deleted from the public transport network graph before the itinerary computation.

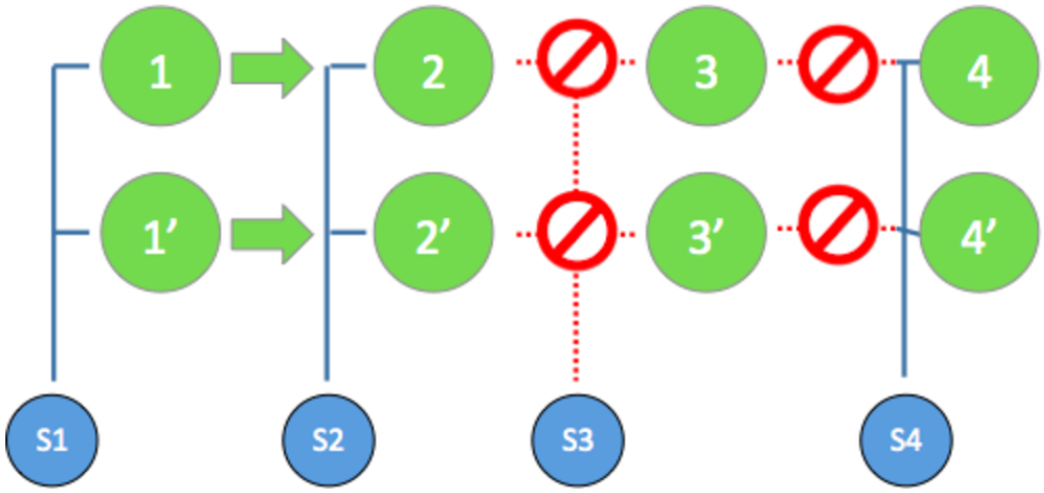

Figure 11.1: *Station closed and vehicles circulation forbidden*

In the second type of disruption, only access and/or exit are blocked and, as shown in Fig. 11.2, the relevant station node ($S3$) is deleted but the stop nodes (3 and $3'$) are still considered in the public transport network graph as it is possible to transit through the station (but it is not possible get in/out at this station).

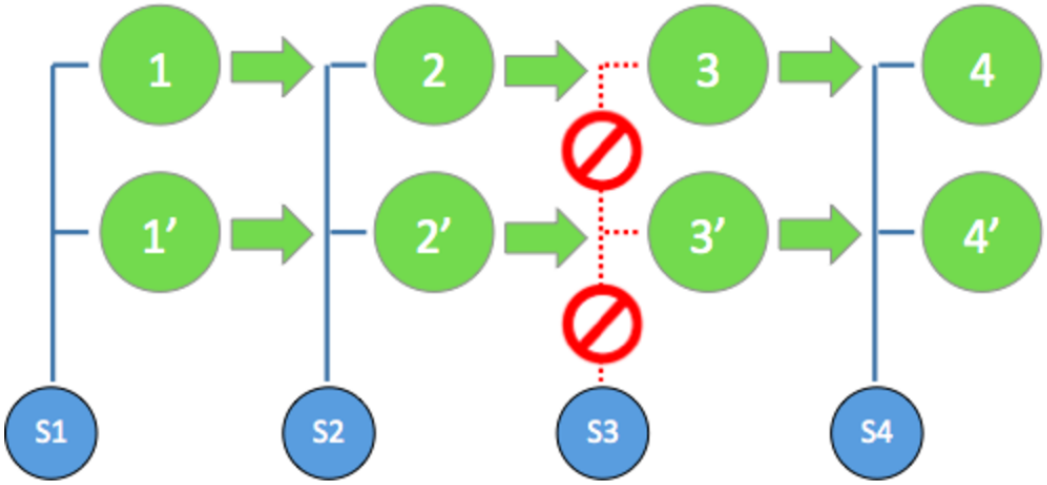

Figure 11.2: *Station closed but vehicles circulation permitted*

In the last type of disruption, shown in Fig. 11.3, the trip is cancelled, the station nodes remain (Sx) available and only the stop nodes connected to that specific trip are cancelled ($1'$, $2'$, $3'$, $4'$).

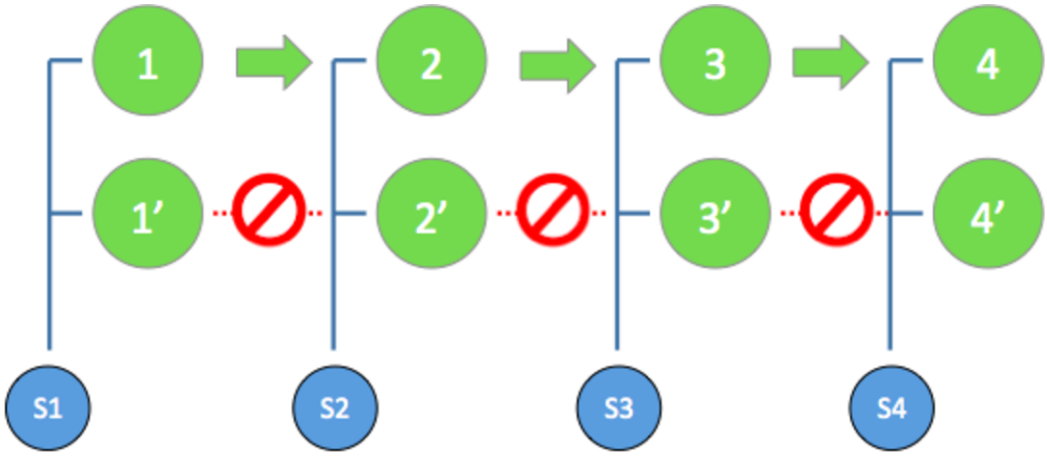

Figure 11.3: *Ride cancelled*

11.2 Development of Engagement Mechanisms and Gamification Techniques to Ease the Interaction with Users

The importance of the data coming from users' mobile devices is crucial for many advanced services. It has been shown how the creation of analysis models could allow the implementation of tools useful for the administration and for the mobility system managers to better understand users' needs and thus offering them services fulfilling these requests. Some of the most relevant gamification techniques, to encourage the users to use provide such data, are here described: they are conceived to turn users into active actors supplying information, both from classic sources such as mobile-phone related data and data networks. In order to better understand the potential applications of the gamification, this section offers a complete overview of the variety of available techniques to educate users, even not strictly related with data gathering process.

11.2.1 Gamification

The concept of **gamification** revolves around the application of dynamics characterizing the gaming process (e.g., scores, levels, prizes) in actual contexts instead of ludic, in order to enhance commitment and competitivity, stimulating the seek of solutions for real problems [41]. Gamification aims at involving people in everyday activities through interactive processes. Gamification has many objectives and all of them can be reached by changing users routine and behaviors: through gamification, in fact, multiple results are achievable such as fidelization, recruiting, problem solving.

Beside the basic fidelization, it is now possible to introduce and design ludic methodologies to stimulate the active behavior of the users increasing their awareness on certain themes. A recent study about the influence of gamification on society [58] states that the ludic component could ease the comprehension of the actual world promoting the application of good behaviors and incentivizing people in adopting positive conducts. A gamified product offers objectives to reach, levels to complete, competitions with other users, sharing of own successes, rewards earnings and it stimulates everybody's need of competing.

Games can be also applied to improve or making more bearable certain experiences which are usually considered not pleasing and, above all, they can also succeed educational goals by leading users to good behaviors and habits.

In the mobile devices big-data field, the gamification techniques are a key tool to stimulate users to supply data and information both in a non-active way (e.g., just through the use of a smartphone app) and in active one (e.g., when the user is asked to supply information through push services).

Objectives and Techniques

Gamification techniques allow to get the users involved: from mere passive users of a service to a proactive actors able both to supply information (that implies focusing on specific targeted users) and, on the other hand, to be influence the users in their activities (to promote good habits and behaviors).

Here below some of the most widespread and applied gamification techniques:

- Scores/Credits: score-collection aims to improve users participation. Even without an actual value associated with the scores, users are lead to make certain actions to increase their score balance. By dividing these scores in categories, it is possible also to make users act differently, with specific goals. Scores are necessary to win prizes so that the users have the feeling that the time spent collecting scores is well-spent.

- Levels: through each level, according to the score gained, it is possible to rank users. The level introduce a new goal to reach, often through the summing up of scores/credits. Each completed level corresponds to the achievement of certain privileges and advantages that can be signaled and underlined in the user's profile. This process makes users more focused as they have an actual graphic visualization of the hierarchical level achieved.

- Virtual prices and virtual goods: the accumulation of scores/credits can give the users the chance of winning prizes and/or swapping and buying virtual goods, increasing their involvement.

- Rankings: rankings, along with levels, are the most effective method to sort and filter game users. This sorting process can be driven by different features such as the time spent on the game, the level reached, the points/credits gained and the performances in general. The users lead to spend more and more time playing to improve their position in the ranking, monitoring their and competitors' results. Competition is a very powerful mechanism when it is strictly linked to the will of standing out in a certain community.

11.2.2 Gamification in the Transport System

The application field of gamification is wide and various and could extend in healthcare and wellbeing industry, e-commerce, education and e-learning and many more. Concerning the mobility and the transport field, it is possible to detect some types of gamification techniques:

- Informative

- Guide simulators
- Planning games
- Scenarios design
- Education
- Involvement

Most of the games are merely a combination of two or more of the above mentioned types and the list is clearly not complete. Almost all the games include an educational component which is quite often the main goal of the game itself. In the end, the game represents just a single component of the services, usually associated to social network paradigms.

Only the last type is typically related to mobile devices data gathering: examples of possible gamification applications that can be developed using the Public Mobility Platform are: play Pokemon Go [43] and catch Pokemon positioned specific vehicles (e.g., only if you are on a bus you can find and catch that Pokemon), challenge passengers at some games (e.g., play Ruzzle [40] with other passengers), leave messages in specific vehicles that can be read only by other people that will catch the same vehicle.

Contextualized social applications include all the new social applications that can be proposed in the transportation field using the estimated position of the user. These applications include sharing and organizing routes with friends (e.g., choose a route to meet one of your Facebook friends), share preferences with other passengers on the same means of transport (e.g., music preferences).

11.3 Traffic Flows Estimation

Concerning the use of car sharing data, an interesting tool for the estimation of traffic flows is here proposed. This service can compute the travel time and the traffic flows without any need of probe cars, detectors or crowdsourced data. This tool can be used to predict the travel time at any given time of any given day and to improve the estimated state of the road and public transport networks. This tool is designed to estimate the travel time at a given hour between two points of the city using real-time and historical data. It is then used to improve the estimated road and public transport network state. All the travels started within a given radius r from the starting point and ended within a radius r from the destination point are grouped and the estimated travel time is calculated as the average travel time of each travel (Sec. 10.4.1). The data considered can include all the data of the travels occurred in the same day of the week at the same hour (used for statistical analysis) or only the travels in the current day (used for real-time traffic estimation).

In Figure 11.4 is shown an example of estimation of a travel time between two points of the city: the tool looks for the cars that traveled from point A to point B at that hour and calculates the average travel time.

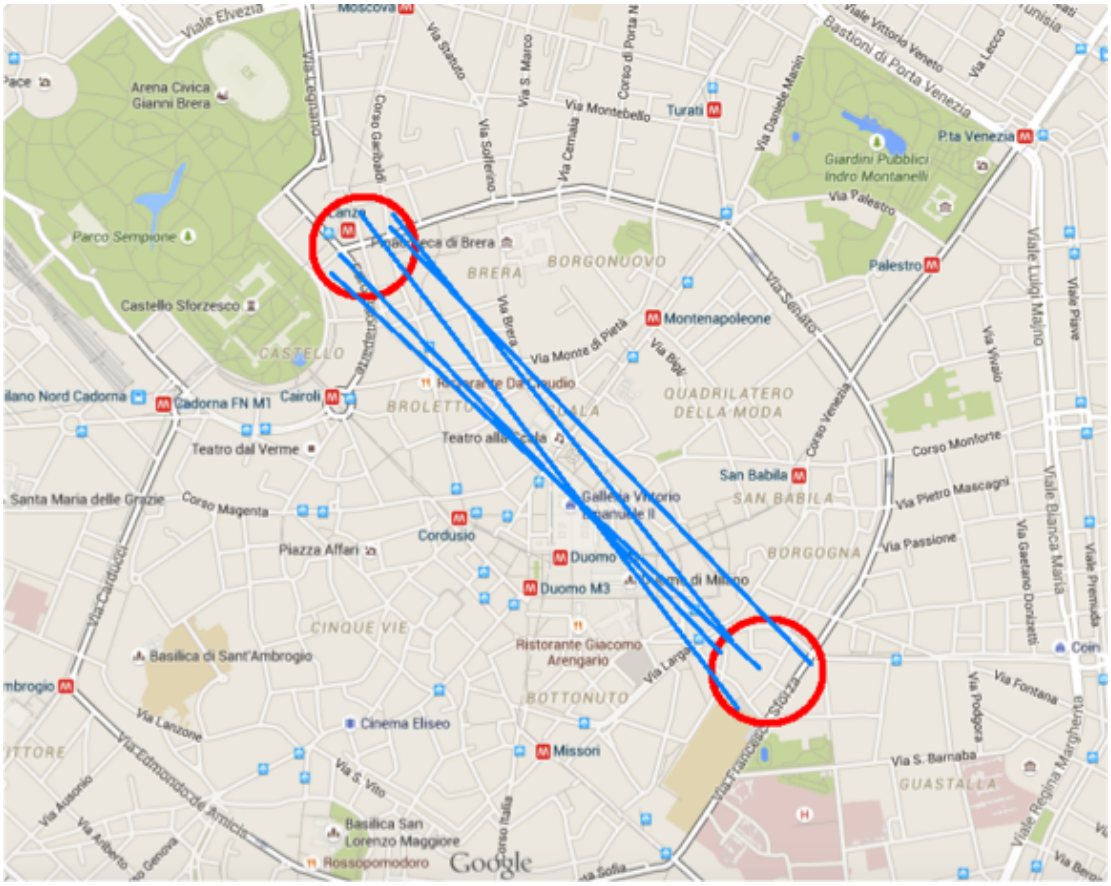

Figure 11.4: *To estimate the travel time from point A to point B the average time of similar travels is computed selecting all the travels that start inside a circle of radius r with center in A and end in a circle with radius r and center in B that occurred at the same time in the same day in the last weeks.*

11.4 Cars Stockout Detection

Another interesting application relative to the car sharing services concerns the detection of areas with a high request of cars but with no cars available (stockout). This allows to detect zones where the number of vehicles is decreasing even though they are still very requested. This information is very useful for the providers as they would be able to take action by moving the fleet according to the requests with an increase of profits: for example by relocating the cars into these specific areas or offering discounted price to users moving towards less populated, but requested, areas.

CHAPTER 12

Conclusions

Aim of this thesis was to provide the necessary infrastructure to enable the development of innovative, context-aware services and applications in the field of the public transportation. A set of methods and tools was designed to create the structures and the information required. These instruments were integrated in a general framework able to reconstruct and update in real-time the *public mobility state*. As discussed in Section 2.1.1, the *public mobility state* can be defined as the intersection of three state dimensions:

- The **public transport network state** represents the current position of each means and by their equations of motion (that is the prediction of their future position).
- The **road network state** represents the current state (i.e., travel speed, traffic flows, congestion) of the roads in a given area.
- The **users state** is composed by the position of the traveling users (e.g., gathered using a smartphone application), their destination and, when on a vehicle, the matching with the means of transport they are traveling on.

These three states are not directly available, thus they are reconstructed using publicly available data, such as open data, transport means timetables, actual arrival times upon each stop and users' GPS position. Each state is elaborated in a different conceptual module and, together with the *data fusion module* and the *API server module*, they compose what in this thesis was defined as the *Public Mobility Platform*.

The **Public Mobility Platform** manages all the information regarding the public mobility, the vehicles, the traveling users, the state of the transportation network and road network. This platform enables the design of user oriented planning and assistance services increasing the perceived quality of the public transport service and offering a

better travel experience, reducing traveling time and improving perceived comfort, even in case of disruptions. Furthermore, this innovative platform can also supply public transport agencies with detailed information such as number of traveling passengers, commutation times and routes.

The **public transport network reconstruction module** was designed to reconstruct the position and the estimated motion laws of all the means of transportation in a given area, using information typically available, such as estimated time of arrival of the vehicles at the stations and users position sampling. The proposed framework uses a state-based Bayesian approach for the reconstruction of the transit system state from limited publicly available knowledge. In Chapter 10.1 the proposed general approach was proven to be effective both in reconstructing the state of the transport network and in exploiting it to predict the position of the public transport vehicles in the near future, also in the presence of various real world kinds of noise, such as information blackouts.

The **user contextualization module** integrates the location of users in the reconstructed public mobility state in order to estimate when the users are on a means of transport. Obtaining this information posed two main challenges. The first one was to understand the travel state of the agents (the users), that is when they are actually on the vehicle or not (e.g., they are just walking on the street close to a bus). The second challenge revolved around the association of these generic agents to the vehicle they are riding, taking into account noise, limited precision of the sensors available (e.g., GPS, accelerometers) and possible presence of multiple means of transport in the same street or area. The proposed solution is based on a Particle Filter with a model that includes the probability for each user to be on a selected vehicle. In Chapter 10.2 the system was proven to be effective in presence of noisy data and to identify correctly when the users are actually on a vehicle or not. Thanks to underlying model, this approach is also convincing in the identification of the specific vehicle each agent is traveling on.

The **road network reconstruction module**, through a knowledge discovery method that processes the real-time position of the car sharing vehicles available, estimates interesting metrics, such as travel time and vehicle flows in the urban areas at different times and in different days. A technique was designed to estimate the road network state (travel time and traffic flows) that does not require any dedicated sensor or probe, but it only uses information (historical and in real-time) about the car sharing vehicles movements during the day. This technique can be used to reconstruct the road network in the cities where other services (with dedicated resources) are not available and can be used, as an additional source, to improve the accuracy of road network state in the cities where other services are already available. The reconstructed road network state (i.e., the actual speed on the roads, the traffic congestions) can be provided to the external applications in order to estimate the routes for the private public transport services (e.g., car sharing). Moreover, this state is integrated with the public mobility state in order to improve the speed accuracy of the vehicles that travel on roads shared with the private transportation. In Chapter 11, the full knowledge discovery from data approach implementation was illustrated and interesting applications were proposed in the field of car stockout forecast and multimodal travel planning.

A further module of the platform was designed to predict urban mobility scenarios and to explain them. The main topic is relative to the private public transportation (e.g., taxis and car sharing) and includes two main techniques based on explainable

deep learning: the first is focused on the design of interpretable deep neural networks, using neural sub-networks to represent engineered input features. These sub-network are used to explain how these features determine the predictions. The second technique revolves around the representation of geographical informations about travels using road graph and feature selection. This module requires a big amount of data to work properly, for this reason a publicly available dataset was selected to test it. The results are encouraging and will be applied to analyze the public transport and road network as soon as enough data will be available.

An ad hoc simulator was developed to test the set of methods and tools and the Public Mobility Platform. In Chapter 10 the accuracy and the robustness of the approach is demonstrated, even in case of missing data, extremely inaccurate GPS position of the users and unlikely corner cases. Afterwards,

A working implementation of this platform was developed for the city of Milan and, in collaboration with TIM Joint Open Lab, it was evaluated, proving to have excellent performances even in real case scenarios. Nevertheless, the system is general an can be exported in other cities and easily adapted to the real-time data available.

In this thesis an innovative platform that enables interesting innovative public transport services was proposed. From a user point of view, the public transport applications can shift from a planning paradigm to an assistance one, where information is provided during the whole trip and not just before the beginning. These new applications can provide the maximum precision and accuracy considering the context of the users (e.g., the estimated time of arrival is constantly updated, taking into account the precise means the users are traveling on, they can play context-aware games and share news with the other passengers). From the transit agencies perspective, it is possible to evaluate metrics valuable to duly assess service quality, such as the number of connections taken, mean waiting times at connections and commutation time. If adequate relevant information is available to travel agencies in real time, the new generation of transport services could dynamically adjust service parameters, and immediately assess the effects, in order to dynamically improve service quality in closed-loop fashion.

12.1 Further Work

In current implementation of the Public Mobility Platform, the most accurate source of data employed is the waiting time at the station. However, this kind of data has a margin of error in the order of tens of seconds and the values are updated every 30 seconds. One approach to improve the accuracy could be the integration of more accurate data sources such as fixed probes, webcams and crowdsourced data. These sources, integrated through a data fusion process with the waiting times, could lead to a further improvement in the precision of the public transport state estimation, improving at the same time the performances of the user contextualization module and the ETACP reliability.

On the other hand, the contextualization of the users is currently obtained exploiting their GPS position provided by mobile applications. Even though this information is viable, it is not the only metric that can be provided by a smartphone. Other potentially valuable information can be exploited: the integrated accelerometer and barometer, for instance, could be used to detect the state of the users. The accelerometer can be

employed to design tools to detect whether users are walking or they are traveling on a vehicle. Moreover, they can detect when users, and thus the associated vehicles, are moving and when they are stationary (e.g., at a station/stop). This information could be crucial when integrated in the public transport reconstruction module, to improve the reconstruction accuracy of the vehicle position and speed. Other tools, based on the barometer integrated in the smartphone, can be used to evaluate where the users are located: indeed, trough these tools, the position of the users can be tracked underground or overground.

The accelerometer and barometer can be combined to generate an identifiable trace, that can be defined as the *traveler signature*. Users with a similar traveler signature can be clustered in the same vehicle, even when the GPS is inaccurate or unavailable, enabling more accurate context-aware and social applications.

Considering the public transport system itself, the Public Mobility Platform could open up the possibility, among the others, to exploit self-adjusting behaviors performing small variations on transit schedule parameters, seeking to optimize some service metrics (e.g., mean travel time, connections) at the lowest cost possible, in real-time. A visionary challenge would be the design of public transport services without predefined lines. A public transport system that dynamically adapts the network and the stops to the origin and destination of passengers.

Bibliography

[1] Open street map. *http://www.gamification.it/gamification/gamification-e- obiettivi-principali/*.

[2] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[3] S. Aksoy and R. Haralick. Feature normalization and likelihood-based similarity measures for image retrieval. *Special Issue on Image and Video Retrieval*, 22(5):563–582, 2001.

[4] Onyx Beacon. Large scale ibeacons network guides visually impaired people to use the public transportation service, 2015.

[5] E. Begoli and J. Horey. Design principles for effective knowledge discovery from big data. *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA) Joint Working IEEE/IFIP Conference*, 2012.

[6] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *eprint arXiv:1708.08296*, 2017.

[7] M. Cameron and A. Brown. Intelligent transportation system mayday becomes a reality. *TAerospace and Electronics (NAECON 1995), IEEE pp. 340-247*, 1995.

[8] K. R. Canini, L. Shi, and T. L. Griffiths. Online inference of topics with latent dirichlet allocation. *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[9] Twist Car and Volkswagen. Twist car sharing, 2015.

[10] Car2Go. Car2go car sharing, 2015.

[11] U. Carrascal. A review of travel time estimation and forecasting for advanced traveler information systems. Master's thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2012.

[12] S. I-Jy Chien and C. Mouly Kuchipudi. Dynamic travel time prediction with real-time and historic data. *Journal of Transportation Engineering*, 2003.

[13] C. G. Chorus, Eric J. E. Molin, and B. Van Wee. Use and effects ofadvanced traveller information services (atis): A review of the literature. *Transport Reviews*, 2006.

[14] citymapper.com. Citymapper, 2016.

[15] CoffeeScript. Coffeescript, 2015.

[16] MATSim Community. Matsim, 2013.

[17] M. Deeshma and A. Verma. Travel time modeling for bus transport system in bangalore city. *The International Journal of Transportation Research*, 2015.

[18] Università di Torvergata. Hyperpath, 2016.

[19] E. Köhler , K. Langkau , M. Skutella. *Time-Expanded Graphs for Flow-Dependent Transit Times*. McGraw-Hill, 2002.

[20] C. Eaton, D. Deroos, T. Deutsch, G. Lapis, and P.C. Zikopoulos. Understanding big data: Analytics for enterprise class hadoop and streaming data. *Mc Graw-Hill Companies, 978-0-07-179053-6*, 2012.

[21] ENI. Enjoy car sharing, 2015.

[22] ExpressJS. Expressjs, 2017.

[23] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *American Association for Artificial Intelligence, AI Magazine, pp. 37- 54*, 1996.

[24] H. Feng, C. Liu, Y. Shu, and O. W. W. Yang. Location prediction of vehicles in vanets using a kalman filter. *Wireless Personal Communications*, 2014.

[25] Python Software Foundation. Simpy, 2016.

[26] C. French. Data processing and information technology (10th ed.). thomson. p. 2. 1996.

[27] M. Gavanelli, M. Milano, A. Holland, and B. O'Sullivan. What-if analysis through simulation-optimization hybrids. In *ECMS*, pages 624–630, 2012.

[28] S. Ghosh, T. Lee, and T. S. Lee. Intelligent transportation systems: New principles and architectures. *CRC Press*, 2000.

[29] Google. Waze, 2015.

[30] Google. Google live transit, 2016.

[31] Google. Google transit, 2016.

[32] N. J. Gordon, D. J Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings-F*, 1993.

[33] M. Handte, S. Foell, and S. Wagner. The unscented kalman filter for nonlinear estimation. *IEEE Internet of Things Journal*, 2016.

[34] J. Hine and J. Scott. Seamless, accessible travel: usersv́iews of the public transport journey and interchange. *Transport Policy 7*, 2000.

[35] J.D. Hol. Resampling in particle filters. *Linkoping*, 2004.

[36] HSBC and GrantThorton. Doing business in italy, 2015.

[37] B. Iglewicz. *Understanding Robust and Exploratory Data Analysis*. Robust scale estimators and confidence intervals for location. 1983.

[38] Infoblu. Infoblu, 2013.

[39] Inrix. Inrix, 2010.

[40] MAG Interactive. Ruzzle game, 2016.

[41] J. J. Lee, J. Hammer. Gamification in education: What, how, why bother? *Academic Exchange Quarterly*.

[42] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME, Journal of Basic Engineering pp. 35-45*, 1960.

[43] Niantic Labs. Pokemon go, 2016.

[44] T. Masters. *Practical neural network recipes in C++*. 1993.

[45] MongoDB. Big data: Examples and guidelines for the enterprise decision maker. *MongoDB White Paper*, 2016.

[46] MongoDB. Mongodb, 2017.

[47] moovitapp.com. Moovit, 2016.

[48] P. Del Moral. Non linear filtering: Interacting particle solution. *Markov Processes and Related Fields*, 1996.

[49] L. Moreira-Matias, J. Mendes-Moreira, J. F. de Sousa, and J. Gama. Improving mass transit operations by using avl-based systems: A survey. *IEEE Transactions on Intelligent Transportation System*, 2015.

[50] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps: Resolving inconsistencies at schema-, tuple-, and value-level ieee data eng., vol. 29, no. 2, pp. 21-31. *Confluence The Next Generation Information Technology Summit (Confluence), IEEE*, 2006.

[51] NodeJS. Nodejs, 2017.

[52] Pandas and NumFocus. Pandas, 2017.

[53] Z. Pang, D. Liu, N. Jin, and Z. Wang. Neural network strategy for sampling of particle filters on the tracking problem. *International Joint Conference on Neural Networks*, 2007.

[54] N. Petrovska and A. Stevanovic. Traffic congestion analysis visualisation tool. *International Conference on Intelligent Transportation Systems (ITSC), IEEE*, 2015.

[55] P. A. Prakashbhai and H. M. Pandey. Inference patterns from big data using aggregation, filtering and tagging-a survey. *Confluence The Next Generation Information Technology Summit (Confluence), IEEE*, 2014.

[56] D. A. Propp and C. A. Rosenberg. A comparison of prehospital estimated time of arrival and actual time of arrival to an emergency department. *The American Journal of Emergency Medicine 9 (4): 301-303*, 1991.

[57] C. A. Quiroga. Performance measures and data requirements for congestion management systems. *Transportation Research Part C: Emerging Technologies*, 2000.

[58] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME, Journal of Basic Engineering*, pages 35–45, 1960.

[59] R. Wade. Welcome to google transit: How (and why) the search giant is remapping public transportation. *Community Transportation: 3.*, 2012.

[60] M. Rahmani, H. N. Koutsopoulos, and A. Ranganathan. Requirements and potential of gps-based floating car data for traffic management: Stockholm case study. *International Conference on Intelligent Transportation Systems ITSC, IEEE*, 2010.

[61] P. Rietveld, F. R. Bruinsma, and D. J. van Vuuren. Coping with unreliability in public transport chains: A case study for netherlands. *Transportation Research Part A: Policy and Practice, Volume 35, Issue 6, Pages 539-559*, 2001.

[62] C.P. Robert and G. Casella. Monte carlo statistical methods. *Springer-Verlag (second edition)*, 2004.

[63] W. Samek, T. Wiegand, and K. Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[64] S. Sargiroglu and Duygu Sinanc. Big data: A review. *Collaboration Technologies and Systems (CTS), IEEE*, 2013.

[65] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks. 61: 85?117. PMID 25462637*, 2015.

[66] Team SimPy. Simpy, 2016.

[67] S. Singh and N. Singh. Big data analytics. *International Conference on Communication, Information & Computing Technology Mumbai India, IEEE*, 2011.

[68] K. Subramanian and R. Srikanth. Now, apps for live traffic feed. *The Hindu*, 2016.

[69] T. H. Cormen and C. E. Leiserson and R. L. Rivest and C. Stein. *Introduction to Algorithms (Second ed.) - Section 24.3: Dijkstra's algorithm*. MIT Press and McGraw?Hill, 2001.

[70] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. 2005.

[71] TomTom. Tomtom, 2012.

[72] transitwiki.org. Publicly accessible public transportation data, 2016.

[73] Shawn M. Turner, William L. Eisele, Robert J. Benz, and Douglas J. Holdener. Travel time data collection handbook. *Research Report 07470-1F, Texas Transportation Institute The Texas A&M University System College Station*, 1998.

[74] Analytics Vidhya. Feature engineering: How to transform variables and create new ones? *Analytics Vidhya*, 2015.

[75] V.K. Balakrishnan. *Graph Theory (1st ed.)*. McGraw-Hill, 1997.

[76] E. A. Wan and R. Van der Merwe. The unscented kalman filter for nonlinear estimation. *IEEE - Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC)*, 2000.

[77] Geoffrey I. Webb, Eamonn Keogh, Risto Miikkulainen, and Michele Sebag. *Practical neural network recipes in C++*. 1993.

[78] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W. Ma. Understanding transportation modes based on gps data for web applications. *ACM Transactions on the Web (TWEB)*, 4(1), 2010.