Erschienen in: Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems : GIS '16. - New York, NY : ACM Press, 2016. - 82. - ISBN 978-1-4503-4589-7
https://dx.doi.org/10.1145/2996913.2997007

MoTrIS: A Framework for Route Planning on Multimodal Transportation Networks

Theodoros Chondrogiannis
Free University of Bozen-Bolzano
tchond@inf.unibz.it

Johann Gamper
Free University of Bozen-Bolzano
gamper@inf.unibz.it

Roberto Cavaliere
IDM Südtirol, Bozen-Bolzano
roberto.cavaliere@idm-suedtirol.com

Patrick Ohnewein
IDM Südtirol, Bozen-Bolzano
patrick.ohnewein@idm-suedtirol.com

ABSTRACT

In this paper, we present MoTrIS, a service-oriented framework which enables spatio-temporal query processing on multimodal networks that are composed of a road network and one or more schedule-based transportation networks. MoTrIS provides a remote access API, which allows for the development of applications that require the processing of routing queries on multimodal networks. We discuss the architecture of MoTrIS and we elaborate on each of its individual components. The *data input* module allows for the import of data from various sources into a spatial-enabled relational database. The *network* module builds a multimodal network by combining a road network with one or more transportation networks. The *timetable* module stores and queries the schedule for each transportation mode. The *query processing* module enables the execution of queries over the multimodal network. The *visualization* module exports the results into a visualizable format. Finally, we present a web application which allows users to create, modify and test advanced spatio-temporal services, and we demonstrate all the necessary steps for a user to build such a new service.

CCS Concepts

•**Information systems** → **Geographic information systems;** *Network data models;*

Keywords

Route Planning;Multimodal Networks;Query Services

1. INTRODUCTION

Route planning services have become very popular over the past decade. The increased availability of road and public transportation network data has triggered the development of a variety of new applications, e.g., various forms of routing such as finding the shortest path, optimal location queries and reachability analysis to name a few examples. Furthermore, apart from traditional platforms that offer routing services, e.g., GoogleMaps and Bing, various platforms have emerged that incorporate the concept of *Mobility-as-a-Service* [2]. Graphhopper[1] offers basic routing services via an API that can be used for the development of more advanced routing-based applications. The project OSRM/MoNav [8] obtains data from Open Street Maps and allows the processing of distance and shortest path queries on road networks. TransDec [6] is a real-world data-driven framework, which obtains data from sensors and analyzes the traffic conditions and historical trajectory data to improve the quality of the results. CrowdPlanner [11] also employs historical information and recommends routes by taking into consideration the preferences of the users. For transportation networks, Graphast [9] is a framework which enables processing of time-dependent spatio-temporal network queries [3].

In contrast to route planning for road and transportation networks, route planning on multimodal transportation networks has yet to be studied in depth. The multimodal route planning problem seeks journeys combining schedule-based transportation (e.g., buses, trains) with unrestricted modes (e.g., walking, driving). This problem is significantly harder than its individual components. Recent approaches [5] apply constraints on the multimodal network in order to exclude sequences of transportation modes that are impossible for the user to take. Another line of work [1, 4] includes methods which compute trips that comply with a set of user-defined preferences and constraints, e.g., no transfers, least possible walking, etc. Regarding systems that support query processing on multimodal networks, ISOGA [7] enables the computation of isochrones on multimodal networks for reachability analysis.

In this paper, we present and demonstrate MoTrIS, a **M***ultim***o***dal* **Tr***ansport* **I***nformation* **S***ystem*. MoTrIS differs from existing frameworks as it tackles the challenge of combining different types of networks into a single multimodal network and enables querying and visualization of multimodal transportation networks. The main goal of MoTrIS is to allow the efficient processing of queries, such as shortest path and distance queries, which can be employed as basic building blocks for creating more advanced solutions to different types of problems on multimodal transportation networks, e.g., journey and itinerary planning. In MoTrIS, developers can choose a road network and select the modes of transportation which will be associated with the chosen road network. Then, MoTrIS creates a new instance/service, where the selected street and transportation networks are combined into a single multimodal network. Users can then submit queries via a public API and, hence, employ the query services directly in their applications. We show that MoTrIS is highly extensible; new algorithms can be easily integrated to allow the processing of more types of queries.

[1]https://graphhopper.com

SIGSPATIAL'16 October 31 - November 03, 2016, Burlingame, CA, USA

ACM ISBN 978-1-4503-4589-7/16/10.

DOI: http://dx.doi.org/10.1145/2996913.2997007

The rest of the paper is organized as follows: In Section 2 we present the architecture of MoTrIS along with the description of its modules. In Section 3 we describe three demonstration scenarios: importing data, creating new services and processing queries. Section 4 concludes the paper and outlines future work.

2. MoTrIS FRAMEWORK

MoTrIS is a service-oriented framework designed to support applications that depend on multimodal transportation networks. The framework allows users to create customized services over specific regions and employ those services directly in their applications. Figure 1 illustrates the architecture of MoTrIS, which is composed of four core modules: the *network* module represents the multimodal network; the *timetable* module handles the transportation network data, i.e., the timetable and the availability of each transportation mode; the *query processing* module includes algorithms for processing queries either on the multimodal network or on each of its individual components; and the *visualization* module produces the results of each query execution. To enable easy access to query services, MoTrIS provides an API which allows users/developers to integrate customized routing services into their applications. Apart from the core modules, Figure 1 also illustrates the *data import*, which is responsible for importing road and transportation network data into PostGIS[2], a spatial-enabled RDBMS, as well as the *web application*, which allows users to create new services and run sample queries to test the services.

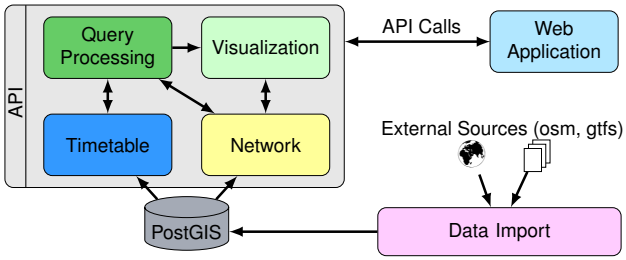

Figure 1: System architecture.

2.1 Data Import & PostGIS

The *data import* module is responsible for importing road and transportation network data from various data sources into PostGIS. In particular, we obtain road network data from *Open Street Map*[3] (OSM). The road network data from OSM is transformed into a routable format and stored in a single relational table in PostGIS. Each tuple represents an edge of the road network along with related information about the edge and its two adjacent nodes, i.e., edge length, source and target node location, etc. Together with the road network, a polygon that marks the boundaries of the region associated with the road network is stored in PostGIS.

To employ services over smaller regions, the module allows the users to specify sub-regions of already downloaded road networks. Polygons representing sub-regions are stored in PostGIS in a hierarchical fashion, as illustrated in Figure 2. After specifying a polygon that represents a sub-region, the module extracts the edges of the road network that lie inside the polygon. Then, PostGIS creates a view which models only the part of the road network that is bounded by the given polygon. Figure 2 illustrates an example, where we have the entire network of Italy obtained from OSM and stored in table *italy_streets*. The polygons representing various cities and states are stored in table *road_network*. To obtain the sub-network for the city of Bolzano, we extract all the edges that lie entirely inside the respective polygon. Then, we create the view *bolzano_view* which contains only the extracted edges and represents the sub-network for the city of Bolzano. Note that the extracted edges may not always form a connected sub-network but several sub-networks. Since sub-regions are not always defined with the road network in mind, it is possible that, in order to reach certain parts of the defined region, the user has to travel outside the region. To address this problem, the data import module comes with an optional verification process which extracts only the largest sub-network and, hence, ensures that every sub-network stored in a view is a connected graph.

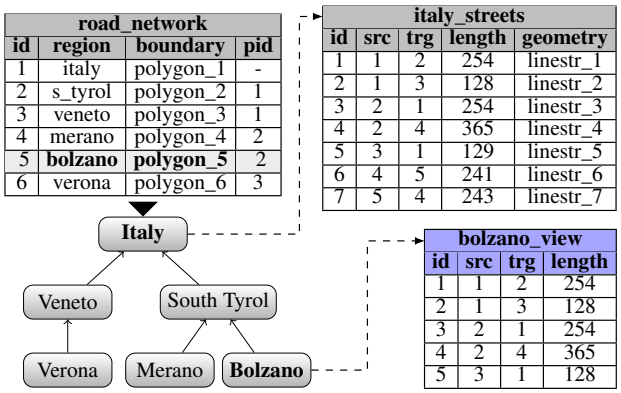

Figure 2: Road network extraction and materialization.

For transportation network data, we first create a set of relational tables in PostGIS that match the General Transit Feed Specification[4] (GTFS). Due to lack of space, we outline only the key GTFS entities for building a transportation network: *stops*, which are individual locations where vehicles pick up or drop off passengers and can be associated with several routes; *routes*, which are groups of trips that are displayed to riders as a single transportation service, i.e., a bus line; *trips*, which store sequences of two or more stops that are on the same route; and *stop_times*, which store the time that a vehicle arrives at and departs from individual stops for each trip. The module creates additional columns for spatial data types. For example, the location of a stop is stored both as two separate float numbers (representing longitude and latitude as dictated by GTFS) and as a *Point* data type.

Finally, the data import module associates each imported transportation network with a road network. The associated road network is either provided manually or is determined automatically as the network bounded by the smallest polygon which also bounds the given transportation network. For each stop in the transportation network, we determine the closest point p of the road network (the black point along the road network in Figure 3) and retrieve edge $e(u, v)$ which contains p. We compute weights $w_u(u, p)$ and $w_v(p, v)$, which represent the length of the segment of e from u to point p and the length of the segment of e from point p to v, respectively. We store in PostGIS these two links for each bus stop as a single tuple $\langle stop\_id, e\_id, u\_node\_id, v\_node\_id, w_u, w_v \rangle$. Since the regions are stored hierarchically, the transportation network associated with a given region can be directly employed for services on all regions that are ancestors of the given region in the

[2]http://postgis.net
[3]https://www.openstreetmap.org/
[4]https://developers.google.com/transit/gtfs/reference

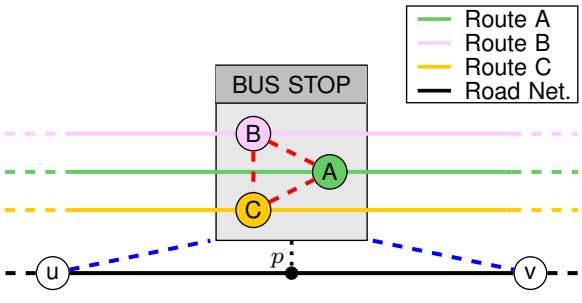

Figure 3: Multimodal network.

hierarchy. For example, given the polygon hierarchy in Figure 2, a transportation network associated with Bolzano can be directly employed for services over South Tyrol and Italy.

2.2 Network Model

The *network* module is responsible for constructing and maintaining the multimodal network graph for each service in main memory. To construct a multimodal network, we first obtain from the database the road network for a specific region by querying the respective view. Then, we model each transportation network following the time-dependent model [10]. First, for all routes, we extract from the trips the stops that each route serves. For each distinct pair of a route and a stop we create and add a node to the transportation network graph. For example, in Figure 3 there are three routes passing through the bus stop. Hence, there are three nodes A, B and C, one for each route, added to the transportation network graph. Next, we extract from the trips all pairs of consecutive stops and we add edges to the transportation network graph connecting pairs of nodes associated with stops that are on the same route. Each edge is assigned with a travel time function, which is executed on query time and determines the weight, i.e., the time required to cross the edge.

Finally, we add link edges to complete the construction of the multimodal network. Naturally, a stop can be associated with more than one nodes. The number of nodes associated with a given stop is the same as the number of routes that pass through the given stop. We add transfer link edges between all nodes associated with the same stop (red dashed lines in Figure 3). The weight of each link edge represents the time required to switch from one route to another. By default the transfer time is zero, unless it is specified explicitly in the input GTFS timetable. Next, for each bus stop, we load from the database the precomputed links to nodes u and v of the road network. For each node n associated with a stop, two link edges $e(u, n)$ and $e(n, v)$ with weight w_u and w_v, respectively, are added to the network graph, thereby connecting the transport networks with the road network. In Figure 3, each of the two blue dashed lines represents three link edges, each connecting a road network node with one of the nodes A, B, C of the transportation network, which are all associated with the same stop.

2.3 Timetable

The *timetable* module is responsible for storing in main memory and querying the schedule of each transportation network. Given a timestamp and an edge of some transportation network, the timetable determines the weight of the given edge, i.e., the travel time needed to cross the edge at the given time. For example, given a bus network, the timetable module stores for each edge e a set of entries $\{n_s, n_d, t_s, t_d\}$, where n_s is the source node, n_d is the target node, t_s is the departure time of the bus from the stop associated with source node and t_d is the arrival time of the bus at the stop associated with target node. Given a timestamp t, i.e., arrival time of the user at the stop associated with n_s, the timetable determines the bus that the user should take to reach n_d. The selected bus is the one which arrives at the stop after t, i.e., $t_s \geq t$ and the travel time $t_d - t$ required to reach node n_d, including the waiting time, is minimum.

2.4 Query Processing

The *query processing* module executes queries over the multimodal network. Currently, shortest path queries of the form $q(G, p_s, p_t, t, M)$ are supported, where G is a multimodal network, p_s and p_t are the source and destination query points on the map, t is the starting time of the journey and M is the allowed modes of transportation. First, the module maps the query points to the multimodal network in the same way it maps bus stops when constructing the multimodal network. To compute the shortest path on the multimodal network, a modified version of Dijkstra's algorithm is used. During the expansion of edges on a transportation network, the module queries the timetable to obtain the weights of the edges. Naturally, only edges of the allowed modes of transportation M are expanded.

We have designed the query processing module in a way that it is easily extensible. The users access the algorithms in the query processing module via an API interface, which provides the users with all the required calls for submitting queries and retrieve results. However, the multimodal network is accessed in an abstract way and cannot be modified by the algorithms that are implemented in the query processing module. By designing the module with extensibility in mind, we aim for MoTrIS to become a solid framework for further research that is not limited to route planning services on multimodal networks. More algorithms for processing different types of queries can be added to our framework as long as the implementation of these algorithms does not require the modification of the network model.

2.5 Visualization

The *visualization* module is responsible for producing the results in a visualizable format after a query is successfully executed. Given the result of a shortest path query, first the module retrieves from the network model the geometries for all the edges contained in the shortest path. The module produces a response in GeoJSON[5] format, which can be visualized directly by most external map services, e.g., Google Maps, Mapbox, OpemStreetMap, etc. Apart from the route itself, the module includes in the response additional information about the result, such as distance, trip duration for each mode, cost of the trip, etc., provided that the information is available in the dataset.

2.6 Web Application

The *web application* allows users to create, modify and test services. Users can create a new service by selecting the region/road network on which their service will be applied, the transportation modes that shall be supported by the new service and the queries that will be available. The application allows the visualization of the resulting multimodal network. For each route in the transportation network the application displays the links connecting the transportation with the street network. Moreover, to get a clear picture of the query results, a testing interface for submitting queries and visualizing the results is provided. For the visualization the module employs Mapbox[6].

[5]http://geojson.org
[6]https://www.mapbox.com

3. DEMONSTRATION SCENARIOS

This section demonstrates how to use our framework for building customized routing services. We will show how to import data into the PostGIS database, how to create a new service and, how to test the new service by running sample queries. The query chosen for this demonstration is the shortest path query between two locations on a multimodal network.

Data Import. We collected road network data for the Province of South Tyrol from OSM and bus schedule data from the local bus operator SASA[7]. In our demonstration we will show all the steps required to import the collected data into our PostGIS database. The demonstration will highlight the database schema used to store both the collected road network data and the bus-schedule. Furthermore, we will present the verification process along with a visualization of each imported network.

Create New Service. We will use the web application of MoTrIS to create a new service for the city of Bolzano and enable the processing of shortest path queries over the multimodal network. To create a new service, the user first defines a region over which he/she wants the new service to be functional and then selects transportation modes from a list of available transportation networks. MoTrIS extracts from the database the road network for the requested region as well as the timetables for the requested transportation networks, and constructs then the multimodal network for the new service. We will demonstrate how the multimodal network graph is constructed by combining the road network and the transportation networks, and we show a visualization of the resulting multimodal graph.

Running Queries. Finally, we will show how the user can submit shortest path queries to MoTrIS and visualize the results using the web interface. The application asks the user to define a starting location, a destination, the date and the starting time (or the desirable arrival time) of his trip. Figure 4 illustrates a screenshot of the web interface of MoTrIS for testing query services. More specifically, the visualization of a shortest path query over a multimodal network for the city of Bolzano is shown. The blue lines represent the parts of the route that the user crosses on foot while the orange lines represent the parts that are crossed by bus.

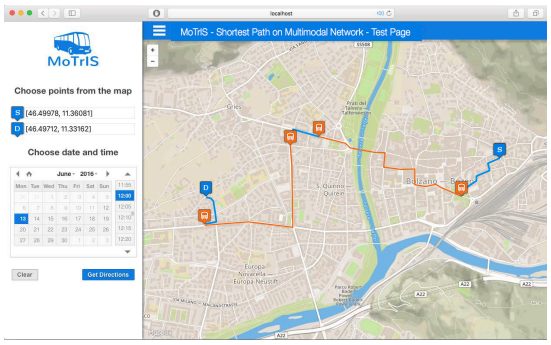

Figure 4: Sample query result visualization.

4. CONCLUSION AND FUTURE WORK

We have presented MoTrIS, a service-oriented framework which provides support for building applications on multimodal networks. Our framework combines street and transportation networks into a single multimodal network and provides query services via a public API. In addition, MoTrIS comes with a web-interface which enables the users to create, modify and test their own customized services. In the future, we plan to extend MoTrIS with an additional service to monitor the timetable information and support real-time updates on the schedule. For example, MoTrIS will be monitoring the location of buses in real-time, determine the real arrival time at each bus stop and compute the travel time of each trip accordingly. Furthermore, we plan to use MoTrIS as basis for further research on more efficient algorithms for various problems on multimodal networks, such as the analysis and evaluation of transportation networks. Such tools will support city planners and transportation scientists to identify problems and evaluate the usability of existing transportation systems and will support the development of more efficient solutions.

[7]http://sasabus.org/opendata

5. REFERENCES

[1] H. Bast, M. Brodesser, and S. Storandt. Result Diversity for Multi-modal Route Planning. In *Proc. of the 13th ATMOS Workshop*, pages 123–136, 2013.

[2] R. Cavaliere, P. Ohnewein, M. Windegger, and S. Kirchlechner. The path towards mobility-as-a-service: A case study from south tyrol, italy. In *Proc. of the ITS European Congress*, 2016.

[3] C. F. Costa, M. A. Nascimento, J. A. F. Macedo, and J. C. Machado. A^*-based Solutions for KNN Queries with Operating Time Constraints in Time-Dependent Road Networks. In *Proc. of the 15th IEEE MDM Conf.*, pages 23–32, 2014.

[4] D. Delling, J. Dibbelt, T. Pajor, D. Wagner, and R. F. Werneck. Computing Multimodal Journeys in Practice. In *Proc. of the 12th SEA*, pages 260–271, 2013.

[5] D. Delling, T. Pajor, and D. Wagner. Accelerating multi-modal route planning by access-nodes. In *Proc. of the 17th ESA*, pages 587–598, 2009.

[6] U. Demiryurek, F. Banaei-Kashani, and C. Shahabi. TransDec: A spatiotemporal query processing framework for transportation systems. In *Proc. of the 26th IEEE ICDE*, pages 1197–1200, 2010.

[7] M. Innerebner, M. Böhlen, and J. Gamper. ISOGA: A System for Geographical Reachability Analysis. In *Proc. of the 12th Int. Symp. on Web and Wireless Geographic Informatin Systems*, pages 180–189, 2013.

[8] D. Luxen, N. Gmbh, and C. Vetter. Real-Time Routing with OpenStreetMap data Categories and Subject Descriptors. In *Proc. of the 19th ACM SIGSPATIAL GIS Conf.*, pages 513–516. ACM, 2011.

[9] R. P. Magalhães, , J. Machedo, C. Ferreira, L. Cruz, G. Coutinho, and M. Nascimento. Graphast : An Extensible Framework for Building Applications on Time-dependent Networks. In *Proc. of the 23rd ACM SIGSPATIAL GIS Conf.*, pages 4–7, 2015.

[10] E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. Efficient models for timetable information in public transportation systems. *Journal of Experimental Algorithmics*, 12(2.4):2.4:1–2.4:39, 2008.

[11] H. Su, K. Zheng, J. Huang, H. Jeung, L. Chen, and X. Zhou. CrowdPlanner : A Crowd-Based Route Recommendation System. In *Proc. of the 30th IEEE ICDE*, pages 1144–1155, 2014.