International Journal of Intelligent Transport System Research manuscript No.
(will be inserted by the editor)

Semantically Enriched Multi-Modal Routing

Thomas Eiter · Thomas Krennwallner · Matthias Prandtstetter · Christian Rudloff · Patrik Schneider · Markus Straub

Received: date / Accepted: date

Abstract We present an innovative extension to routing: *intention-oriented routing* which is a direct result of combining classical routing-services with Semantic Web technologies. Thereby, the intention of a user can be easily incorporated into route planning. We highlight two use cases where this hybridization is of great significance: *neighborhood routing*, where a neighborhood can be explored (e.g. searching for events around your place) and *via routing*, where errands should be run along a route (e.g. buying the ingredients for dinner on your way home). We outline the combination of different methods to achieve these services, and demonstrate the emerging framework on two case studies, with a prototype extending in-use routing services.

1 Introduction

While many services exist for both motorized private transport (MPT) and public transport (PT) and to some extend to multimodal transport, there are still many restrictions to the functionalities of routing services.

One such restriction is that routing services only offer simple routes from a given starting point to a supplied end point along one route. Many services like the router offered by Google or PT routers available through PT providers (e.g., http://anachb.at in Greater Vienna) give the shortest route and some alternatives from one specified point on a map or address to another; sometimes intermediate points can be prescribed, such that trips with several legs can be found. Another restriction is that routing is bound to addresses and PT stops but to the best of our knowledge, does not account for the *intention* behind a user's trip; that is, more flexible routes can be determined with information about intended activities in the input. The research project *MyITS* (My Personalized Intelligent Mobility Service) addresses these restrictions by fostering an *intention-oriented routing* approach. This approach, which we present in this article, extends and connects existing routing methods with Semantic Web technologies — that facilitate reasoning — to accommodate user intentions.

This approach, which has been implemented in a prototype on top of existing in-use routing services, opens up the possibility for a range of novel routing services. In this article, we shall concentrate on two such services.

The first one is *semantic multi-modal (SMM) neighborhood routing*. While several uni-modal neighborhood routers exist (e.g., http://www.tom-carden.co.uk/p5/tube_map_travel_times/applet/, last viewed Sept. 2012) and some services give geographical neighborhoods around PT stations as the reachable area (e.g., http://www.mapnificent.net/, last viewed Sept. 2012), no service actually uses multimodal routing, i.e., includes for example the region reachable by foot around transit locations within a given time interval. The approach presented here searches for multiple, semantically determined locations within a certain travel time from a starting point (i.e., within an *isochrone*) using intermodal routing. This neighborhood routing does not offer the traveler different routes to a single location but the choice of locations plus intermodal routes to those locations, e.g., all supermarkets within a 5 minutes journey by foot, bike, PT, MPT, or a combination of these modes.

As a second use case of intention-oriented routing, we consider *via routing*, which offers the possibility to search for a route with specific start and end points but gives the users

T. Eiter, T. Krennwallner, and P. Schneider
Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9–11, A-1040 Vienna, Austria
E-mail: {eiter,tkren,patrik}@kr.tuwien.ac.at

M. Prandtstetter, C. Rudloff, and M. Straub
AIT Austrian Institute of Technology GmbH, Mobility Department
Giefinggasse 2, 1210 Vienna, Austria
E-mail: first.last@ait.ac.at

the chance to add an intermediate activity to their route. Corresponding *intermediate* or *via points* are automatically determined, on a semantic basis, and appropriate routes through these points are computed which are then presented to the user. For example, the user may specify that a pharmacy offering homeopathic products has to be located along a route from work back home (for her intended shopping stop).

At the heart of our intention-oriented routing approach lies the innovative combination of two technologies in a new routing tool. One is advanced routing, served by an efficient algorithm that considers not only one mode of transport (MOT) but all available MOTs during the same routing request, resulting in intermodal routes. The other is Semantic Web technology, which provides the data backbone for semantic information used to search for semantically-enriched *points of interest* (POIs); e.g., for Chinese restaurants.

The remainder of this article is organized as follows. In Section 2, we introduce the technologies that are applied in our enriched routing tool with the relevant background and existing work on the subject. The details of the interaction of routing and semantic search as well as details of the route ordering models are given in Section 3. Afterwards we consider the novel routing services; we describe in Section 4 SMM neighborhood search and in Section 5 via routing and illustrate it on a use case. In Section 6.2 we then consider an experimental evaluation of our prototype on in-use data setting. In the final Section 7, we conclude with a discussion and an outlook on future work.

2 Background Technology

In this section, we present the methods necessary to achieve an intention-oriented routing service: path, neighborhood and corridor computations as well as Semantic Web technologies and geospatial databases. Although all of these technologies are (still) active research fields on their own, only the combination of them leads to an innovative routing approach providing a valuable service to its users.

Path Computations. The problem of finding a (shortest) route from a given origin to a given destination is defined on an (un)directed graph where costs are assigned to each edge (arc); see Moore [19]. In the general case (where costs for edges/arcs are real numbers) the algorithm of Bellman and Ford [7] has to be applied whose runtime is in $O(n^3)$, with n denoting the number of nodes in the graph. However, with respect to transportation science, the edge costs given are in the most cases non-negative for all edges of the graph. Therefore, Dijkstra's algorithm [9] can be applied which solves the shortest path problem using a simple implementation in time $O(n^2)$.

Neighborhood Computation. Although the term neighborhood is widely used, in this paper the term is used in its geographic meaning: a region around a point where each point lying inside the area can be reached within at most δ minutes. We also write in short δ-neighborhood. Since we assume that all edge costs given in our transportation network are non-negative, a δ-neighborhood can be easily computed via Dijkstra's algorithm [9] by aborting the algorithm as soon as every unfinished node has costs larger than δ. All nodes finished so far, i.e. all nodes for which the minimal path from the origin has already been computed, are those points lying within the δ-neighborhood of the origin.

Corridor Computations. Although the methods presented above are theoretically sufficient to compute shortest routes as well as neighborhoods in the given setting they are not sufficient to handle real-world location based data: in most cases WGS84 (or other geo-referenced) data sets, which are often obtained via GPS, are not directly assigned the edges and/or nodes in an underlying graph. To determine whether a POI, e.g. a pharmacy, is located along a route (or can be reached within at most δ minutes, it is necessary to map this POI to edges and nodes of the graph. There are several methods to do this: e.g. by classical map-matching [24]. These methods suffer, however, from the problem that each time an update is applied to the setup (e.g. map update) the mappings for all POIs have to be (re)computed in the worst case.

Therefore, we decided to apply corridor methods which compute a hull enclosing all edges and/or nodes in question. E.g. for a given route the enclosing hull (or corridor) is computed by applying a buffer method, where a buffer is "a geometry that contains all points whose distance from [the route] is less than or equal to [a predefined threshold]" (also referred to as the *dilated hull*, see also http://www.postgis.org/docs/ST_Buffer.html). While these methods apply well to routes (see e.g. Fig. 1a), they are not satisfactory for neighborhoods since artifacts may occur. For example, blocks of buildings might be marked as unreachable although they are in fact reachable, cf. Fig. 1b. Therefore, we investigated beside the dilated hull other hull computation methods giving convex hulls and non-convex hulls for the neighborhood computation. The convex hull computation is one of the standard hull computation methods in literature. An apparent advantage of it is the existence of efficient algorithms, e.g., *Graham's scan*, with the lower bound of $O(n \log n)$, where n is the set of points in the plane to be enclosed (see [2]). However, for certain applications the convex hull does not provide desired results. In particular the convex hull for long routes or corridors may contain regions that are not sensible search regions for the semantic search (e.g. the convex hull of a route along a large ring road would contain most of a city). Several non-convex hull computation were introduced to obtain *finer* shapes. One of this methods are α-shapes, which were introduced by Edelsbrunner et al. [11] as a generalization of the convex hull. The construction of an alpha-shape is

(a) Buffering around a route. (b) Enclosed "unreachable" areas. (c) Public transport.

Fig. 1: Corridors (filled area) for a route, and a neighborhood using public transport. Unreachable areas are shown in white.

performed by an intersection of all closed discs with a given radius that contain the set of points in the plane.

Our investigations show that the convex hull is the fastest evaluation, however it is not feasible for car and public transport corridors, as false positives can appear frequently (e.g., the yellow points in Fig. 3b). For non-convex hulls, we applied the χ-shapes method of Duckham et al. [10] (with the lower bound of $O(n \log n)$). Compared to the dilated hull, we still observed more false positives for public transport corridors, as this algorithm tends to create a single polygon from the underlying points. However, these shortcoming could be overcome, if the points are previously clustered, and then the χ-shapes are calculated (c.f., [10]) . Please note that in case of public transport, the corridor might consist of non-connected and/or overlapping regions, see Fig. 1c, since public transport stations act as sub-origins. Hence, the dilated hull method appears to be the most appropriate.

Semantic Web Technologies and Geospatial Databases. Official data sets (provided by governments for public use) and collaborative projects like OpenStreetMap (OSM) are becoming large sources of spatial data, where geospatial databases are the backbone for storing and querying these data (cf. [8, 16]). These databases often have the drawback that querying them is complicated, inference mechanisms are not common, and extending them to the new data sources is difficult, due to extracting, transforming, and loading steps. In response, Semantic Web technologies are becoming more interleaved with geospatial databases [4, 21], which should lead to an easier integration and querying of spatial data.

Due to the lack of space we cannot give a full introduction to Semantic Web technologies. We refer to the seminal article of Berners-Lee et al. [3] for an outline of the ideas behind the Semantic Web. In particular, ontologies are used for modeling knowledge domains, by expressing relations between terms with a restricted vocabulary and by modeling them as a taxonomy. In the Semantic Web context, the standard modeling language *OWL* [17] with its (formal) logical underpinning of Description Logic (DL) plays a central role. The vocabulary of a DL consists of *individuals* (*instances, objects*), *concepts* (*classes*), and *roles* (*attributes*). A *knowledge base* consists of a *terminological box*, which contains axioms about relations between classes and roles, and an *assertional box*, which contains factual knowledge about individuals (cf. [1]).

As shown in the MASTRO system [5], these technologies are more than an advancement of the WWW. Therewith, ontologies are applied as a feasible way to integrate data sources through a global schema expressed as them. The integration, also called *global-as-view* approach [18], is a set of mapping assertions which maps source schema to a global schema by first-order logic (FOL) queries (in practical terms, roughly speaking SQL queries).

3 Methods

Although the methods applied within this project (routing and semantic search technologies) are well-understood on their own, the proposed combination of these techniques is rather novel and may lead to computational expensive tasks if not done properly. We propose to perform the incorporated methods in sequential order, meaning that either a semantic search constrained by the routing results (route-constraining) or route computations are performed based on the result set returned by the semantic search (post-filtering). In a final step, the routing tool orders the routes based on a utility maximization approach that takes the preference of the users for certain modes within a multimodal route into consideration. Using mixed logit modeling, this allows for personalizing the ordering of the routes by calculating individual level parameters once a user has used the router multiple times.

In more detail, on the routing side we use an algorithm that is not only capable to compute shortest routes, but as

mentioned above (intermodal) isochrones and furthermore *corridors* around the routes that are used as input regions to a semantic search.

On the semantic side, flexible data integration and querying of multiple data sources is a key ingredient of our approach; autonomous data vendors provide heterogeneous data that is used to answer queries associated with the intention of the user. Using various data sources of substantial size gives the opportunity to find intended POIs, which may fall into multiple classes ranging from rather generic ones like "restaurant" to more detailed ones such as "pizzeria". Moreover, we can exploit the structure of the taxonomic information that is implicitly stored in the data sources by making them concrete in a formal ontology. The resulting *ontology-based data access* can be used to answer broad queries like "restaurants with Italian cuisine," that should return pizzerias and classical Italian restaurants, to be used for route selection. Furthermore, it is feasible to seamlessly add new data from fresh sources, which allows to generate more accurate answers for the extended domain.

Route-constrained Semantic Search. In a route-constrained semantic search, the query itself is constrained by the results of a preceding routing phase. This approach is especially helpful whenever either the number of candidate POIs returned by an unconstrained search is rather large or if the constraints stated by the user with respect to the routing are rather hard (e.g. only POIs no farther away than 5 minutes). Hence this technique is mainly applied for neighborhood routing (cf. Sec. 4). In addition, it is also applied for via routing if the number of candidate POIs is rather large, such that finding an appropriate via point along a convenient route is easy (cf. Sec. 5).

Post-Filter Semantic Search. In contrast, in a post-filter semantic search first an unconstrained search that returns all candidate POIs is performed. The POIs are then subsequently processed in routing. Since for each of the obtained POIs routes have to be computed, this technique is only applied for via routing (cf. Sec. 5) if the number of candidate POIs is rather small. This approach, however, imposes no limitations on the semantic search as well as the routing procedure.

Method Selection and Route Choice. Since our framework supports two possible sequential executions of its tasks, we need a way to determine which one should be applied for a concrete user request (this only applies for via routing). For this purpose, a simple determination heuristic is applied. First we perform an unconstrained semantic search and obtain a set of candidate POIs. Instead of the cardinality of the set, database statistics regarding the concept and role assignments could be used. If the cardinality of this set is above a predefined threshold a route-constrained search is applied. Otherwise we continue with the post-filtered procedure.

Independent of the method applied (route-constrained or post-filtered), in most cases more than one possible route is returned (mainly because in most cases the number of candidate POIs is larger than one). Therefore, it is further necessary to rank all obtained routes which can be done using straightforward evaluation strategies like (shortest) travel time or (shortest) distance. In MyITS a more complex procedure described below was implemented.

Route Ordering. While the user has the last choice in the chosen system architecture, the ordering of the routes should make the search for the preferred route easier. To support the user, the routes are ordered using a random utility maximization approach. Here the utilities are determined from properties of the route between the start point, POI and end point of the route. These include the travel time spent in different modes, number of changes between modes and waiting time at the transfer point. Further properties could be added, concerning for example the quality of the POI (how much does the POI match the search criterion originally entered by the user), the position of the POI in the route and other route properties for certain modes, like the amount of time traveled on bike infrastructure or number of left turns. However these are not considered in the model of this prototype.

To estimate a mixed logit model for mode choice, routes actually chosen by the users have to be collected. This is done by adding the possibility to choose a favorite route from the set of routes provided by the routing application. The favorite route together with the other routes in the choice set are then stored in a database. The variables are calculated for all routes and the utilities

$$V_{ni} = \sum_{k=1}^{m} \beta_k x_{nik} + \varepsilon_{ni} =: \beta' X_{ni} + \varepsilon_{ni} \quad ,$$

where x_{nik} are the m variables calculated from route i of user n; ε_{ni} is the extreme value distributed unobserved utility; and $\beta_k \sim N(\mu_k, \sigma_k^2)$ are normally distributed parameters with mean μ_k and standard deviation σ_k^2. Distributed parameters are used to model heterogeneity in the population of users, like different preference levels for the use of buses. The model is estimated by maximizing the negative simulated log-likelihood given by

$$SLL = \sum_{n=1}^{N} \sum_{i=1}^{J} d_{ni} \ln \check{P}_{ni} = \sum_{n=1}^{N} \sum_{i=1}^{J} d_{ni} \ln \frac{1}{R} \sum_{r=1}^{R} \frac{e^{\beta^{r\prime} X_{ni}}}{\sum_{j=1}^{J} e^{\beta^{r\prime} X_{nj}}} \quad ,$$

where $\check{P}_{ni} = \frac{1}{R} \sum_{r=1}^{R} \frac{e^{\beta^{r\prime} X_{ni}}}{\sum_{j=1}^{J} e^{\beta^{r\prime} X_{nj}}}$ denotes the simulated probability of user n to chose route i. Here, $\check{P}_{ni}$ is simulated by drawing R times, where $\beta^r, r = 1, \ldots, R$ denote the draws from the joint distribution $N(M, \Omega)$ with covariance matrix Ω of the β_i and $d_{ni} = 1$ if route i is chosen by user n and 0 otherwise.

Once ten routes are collected for some user, individual level parameters are estimated for that user. While the method

works as soon as one route is available for the user, ten was chosen to minimise biases introduced through routes that do not fall into the users usual behavior.

The individual level parameters give information where in the population distribution $N(M,\Omega)$ of the parameters the preferences of this particular user lie. The estimation process for the individual level parameters works by taking R draws β^r from the population distribution $N(M,\Omega)$ of the parameters. The simulated mean $\check{\beta}_n$ of the individual level parameters depends on the sequence of T chosen alternative $y_n = \langle y_{n1},\ldots,y_{nT}\rangle$ and the corresponding variables $X_n = \langle X_{n1},\ldots,X_{nT}\rangle$ of the individual is given by the weighted sum

$$\check{\beta}_n = \sum_{r=1}^{R} \frac{P(y_n|X_n,\beta^r)}{\sum_{s=1}^{R} P(y_n|X_n,\beta^s)} \quad ,$$

where the probability of the chosen alternatives y_{nt} is given by

$$P(y_n|X_n,\beta) = \prod_{t=1}^{T} \frac{e^{\beta' X_{ny_{nt}t}}}{\sum_j s^{\beta' X_{njt}}} \quad ,$$

where $X_{ny_{nt}t}$ denotes the variables calculated for the route chosen by user n in choice situation t. The ordering of the routes can also be done for the individual in a similar way by using the simulated probabilities $\check{P}_{niT+1}(X_n,y_n,M,\Omega)$ of a new choice situation with variables X_{nT+1} given by

$$\check{P}_{niT+1}(X_n,y_n,M,\Omega) = \sum_{r=1}^{R} \frac{P(y_n|X_n,\beta^r)}{\sum_{s=1}^{R} P(y_n|X_n,\beta^s)} \frac{e^{\beta^{r\prime} X_{niT+1}}}{\sum_{j=1}^{J} e^{\beta^{r\prime} X_{njT+1}}}.$$

For more details on the properties and calculations of individual level parameters as well as mixed logit models see [23]. While the procedures and data-bases were implemented and the first estimated parameters show for example a preference for the use of trams over buses and a preference for shorter time walking, the parameter estimates are not included here, as the sample is small and due to the limited data collection not representative.

Ontology-Based Data Integration. In our case the Ontology-Based Data Integration [5] is provided by an OWL 2 QL *Upper Level Ontology* (ULO), which acts as a global schema and is tailored to geospatial data sources and to MyITS-specific sources (e.g., the restaurant guide of Falter, see http://www.falter.at/web/wwei/). As shown in Fig. 2, the top level of the ULO is built on already defined work with GeoOWL (see http://www.w3.org/2005/Incubator/geo/XGR-geo/) However, for our needs the (spatial) *Feature* concept of GeoOWL is too general, so we introduced a more detailed categorization for the second level based on the nine top-features of GeoNames (cf. http://www.geonames.org/) such as *AdministrativeFeature*, *AreaFeature*, *BuildingFeature*, etc. The third level is build mainly from OSM categories as which are relevant for the MyITS system. The ULO is designed to be extended, in case we include new data sources. A further extension of the first level are the concepts *QualAttribute* and *Geometry*, where

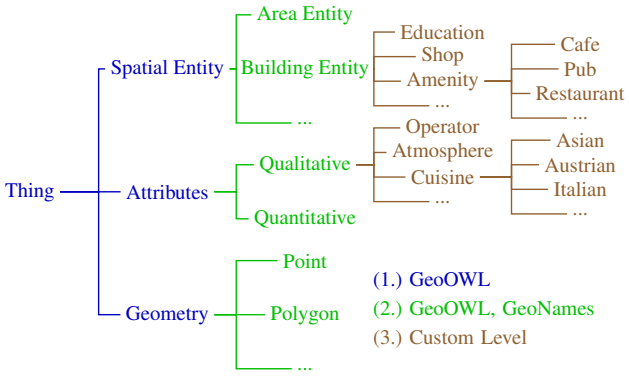


Fig. 2: Simplified Upper Level Ontology

the former is meant to extend a spatial feature with additional qualitative attributes and the later is meant to define the geometry of a spatial feature.

Mapping of Spatial Instances. The mapping between spatial instances (objects) in the database (representing the POIs) and the ontology can either be done on-demand by FOL queries (as in MASTRO) or can be calculated preliminarily and materialized in an extension of the assertional box (similar methods are used in annotations engines as KIM [20]). We adapted the latter for the following reasons:

(1) Rewriting is difficult, as domain specific heuristics are crucial for the assignment of objects to appropriate concepts and are hardly expressible in FOL queries (e.g., guessing the type of cuisine from the name of a restaurant).
(2) A part of the mapping relies on external computation sources like geometry and string metrics engines.
(3) Data cleansing (e.g., duplicates) and inconsistency management is much easier to handle, if the mapping is materialized.

At present, we use a custom mapping framework, but plan to apply a rule-based framework based on HEX-programs [12] for the purpose of declaratively defining the materialization. Furthermore, with HEX-programs default negation, recursion, and constraints can be straightforwardly expressed.

Having several data sources with similar objects assigned (e.g., between OSM and Falter), the query result may contain large amounts of duplicates. For our prototype, we have for simplicity disregarded duplicate elimination (which causes no loss of results), for which suitable methods can be developed.

Spatial Query Answering. The enriched ULO can be accessed, at the system level, by *spatial conjunctive queries (SCQ)*, which extend conjunctive queries with spatial predicates (e.g. intersects, contains, next to). In such queries, informally instances can be located with spatial objects whose relationships are determined. By rewriting techniques, and in

exploiting the *PerfectRef* algorithm [6], SCQs can be rewritten to a union of conjunctive queries. Under certain syntactic conditions, a 2-stage evaluation—evaluation of the ULO part of the query (over the instances stored in a database) followed by filtering via spatial relations—is possible, which makes this approach attractive for practical realization. For accessing the instances, the SCQs are rewritten into SQL queries, which are evaluated on the combined DB of all data sources. Note that these SQL statements can become very complex, so that normal databases (e.g., Postgres) fail to optimize or even evaluate them. For this purpose, we decompose the different CQ which are linked by spatial predicates, and evaluate the spatial relations (internally) in-memory. We refer to Section 6.2, which illustrate possible SCQs and their evaluation time.

For keyword-based query answering, concepts of the ontology are labeled with keywords. On query evaluation, the keywords which the user enters are mapped to concepts and roles from the ontology; an *auto-completion* service aids the user to compensate lack of domain knowledge. Based on the keyword structure, a feasible CQ is generated and extended with spatial predicates to SCQs; in that, we use a specific *meta-model* that is stored in the ontology. An excerpt of the ontology is shown in Fig. 2; the concept *SpatialFeature* intuitively says that the instance has spatial features, which is extended by the sub-roles of *hasQualValue* with qualitative values, which are asserted to sub-concepts of *QualValue*. Furthermore, the instance is represented by a geometry, asserted to sub-concepts of *Geometry*. However, also normal role assertions for qualitative attributes are considered (e.g., a restaurant with Italian food and a guest garden).For more details we refer to [13].

4 Neighborhood Routing

The focus of the first use case is the so-called *neighborhood routing* where a user desires to explore the neighborhood for a specific class of POIs. We see several applications for this case, e.g.:

- a tourist looking for a coffee shop after visiting an sight,
- a resident looking for a particular event (e.g. concert) after dining, or
- a newcomer checking reachable supermarkets by a certain modality in his/her area.

Method. More formally, as input we consider a start coordinate A, a POI query Q as described in the previous section, a modality choice M, and a distance threshold δ in minutes. Note that the modality choice may consist of multiple modes of transport if inter-modal routes are accepted. Then the following procedure is applied:

(1) compute the δ-neighborhood of A considering modality M, resulting in a graph I of all reachable edges and nodes;
(2) based on graph I, calculate a corridor C by either the convex (or non-convex) hull or the dilated hull (buffer) operator using a computational geometry engine;
(3) perform a semantic search on the knowledge base using the corridor C as spatial restriction, leading to a set of POIs; and
(4) the final result is built by calculating routes for a selection of POIs (e.g. the closest ones).

At this point it is assumed that at least one POI is located in the neighborhood. If this is not the case, there are basically two possibilities how to proceed: the user is informed that there is no matching POI in the δ-neighborhood. Alternatively, the threshold δ is iteratively enlarged until at least one matching POI can be reached. The user is then informed that the proposed POI is outside the originally intended δ-neighborhood. However, the second case may result in an infinite loop if the number of iterations is not limited.

Case Study. Assume that an opera buff, who does not favor the second act of Richard Wagner's Parsifal, wants to have a snack at one of Vienna's famous Würstlstände (a kind of Viennese hotdog booth). Between the second act, she looks for any Würstlstand reachable in 5 minutes by foot or taxi (cf. Fig. 3a and 3b showing the 5-minutes-neighborhoods for walking (a) and by taxi (b).)

Our case study is based on the POIs of greater Vienna contained in OSM. In Fig. 3a we can see the search results for "Würstlstand" filtered by a convex resp. dilated hull are similar to each other. Fig. 3b shows that 40 POIs are in the convex hull (colored points), only 30 POIs are covered by the dilated hull (darkest points); 10 POIs are not reachable at all (white points). It is noticeable that depending on the distance and modality the shape of the neighborhood may become more tree-like, having the effect that the proper reachable areas are covered more accurately by dilated hulls. More detailed investigations reveal that six of the differing yellow points shown in Fig. 3b are in a neighborhood of Vienna with limited car access.

5 Via Routing

In the *via routing* use case, we are focusing on determining a route between a given origin-destination pair via some POI, which is dynamically determined by a semantic search. We have identified several applications for this case, e.g.,

- a tourist wants to eat in a restaurant on his way to the hotel, which offers a particular cuisine. ,
- a commuter needs to stop at a pharmacy, or
- a person invited for dinner needs to buy some flowers.

Method. Formally, given the origin A, destination B, POI query Q, mode choice M, and distance threshold δ in minutes, the task is to generate a route from A to B using M via one

(a) 5 minutes walking distance (b) 5 minutes taxi ride

Fig. 3: 5-minutes-neighborhood (dark edges), convex hull (dashed area), dilated hull (filled area), and Würstlstände (different points) starting at Vienna opera house

particular POI within a δ-neighborhood that is included in the answers to Q. In contrast to neighborhood routing, the procedure applied within this case may vary depending on the size of the result set $\mathscr{P}$ of POIs obtained during a first (unrestricted) semantic search. For a small number of results, the procedure will continue with

(1) computing a via route for each $P \in \mathscr{P}$, i.e., a route from A to P followed by a route from P to B, and then
(2) analogously to neighborhood routing, the obtained routes will be sorted and presented to the user.

However, if the cardinality of $\mathscr{P}$ is too large, the following approach will be applied:

(1) compute a few alternative routes $\mathscr{R}$ from A to B optimizing a defined criterion, e.g., shortest travel time;
(2) compute a corridor C for $\mathscr{R}$;
(3) obtain a new result set $\mathscr{P}'$ of POIs by semantic search constrained to C, i.e., disregard POIs located outside of C;
(4) compute for each $P \in \mathscr{P}'$ a route from A to P followed by a route from P to B; and
(5) sort the obtained set of routes and present it to the user.

In the second case, it is possible that despite a large result set $\mathscr{P}$, we have no POI located within corridor C. As a consequence either the width of the corridor is iteratively enlarged or additional alternative routes are computed using (for example) an adapted optimization criteria.

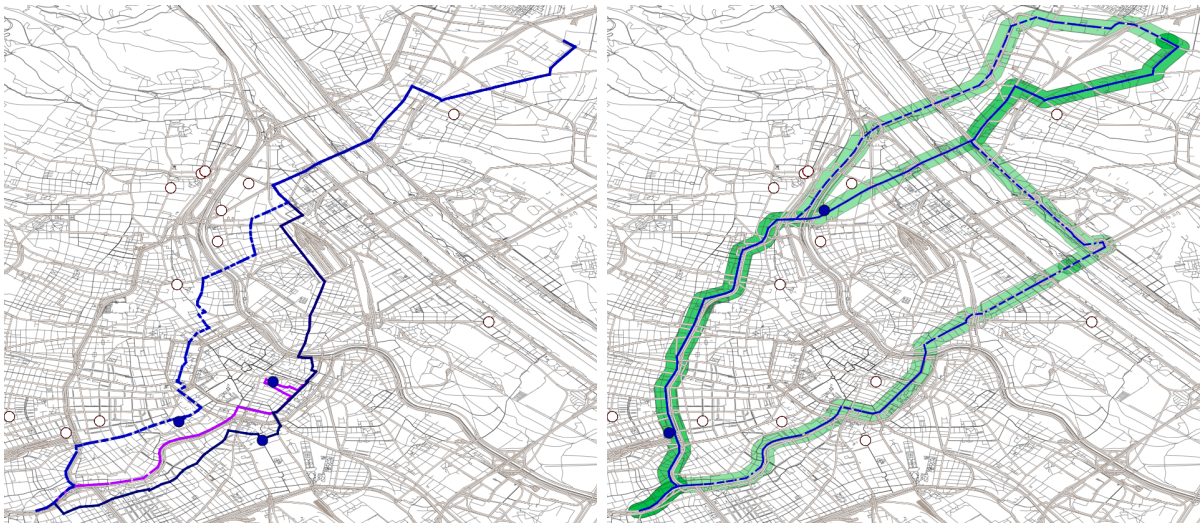
Case Study. Take as an example a father of two children, who is late from work and has to buy some Chinese food for his family on his way home. For this purpose, he is looking for a Chinese restaurant reachable by car, having a parking lot in front of the door and selling take-away food.

Since the number of Chinese restaurants in Vienna with a parking lot is around 17, both described methods could be applied. For the post-filtered method, we evaluated the semantic search for the concept Restaurant with Chinese cuisine. Then, three candidate POIs were chosen for calculating a via route, resulting in six partial routes. In Fig. 4a the results are represented after checking if the candidates were within the distance threshold. The second approach is illustrated in Fig. 4b, where three alternative routes were calculated, which composed the base for calculating three corridors. We only considered the dilated hull for the corridors, since the convex hull for the three routes would cover almost the entire city. In contrast to the previous use case, the buffer size for the dilated hull was chosen 3 times higher, otherwise solely restaurants directly on the routes would have been chosen. Finally, the corridors where applied as a filter for the semantic search, resulting in two possible results.

The larger the set of possible POIs is growing, the more efficient is the second approach, since only a few alternative routes have to be calculated. On the other hand, when less POIs exist, the second approach is inefficient, because alternative routes or the corridor might have to be recalculated several times.

6 Implementation and Experiments

Within this section, we give (a) an overview of the chosen system architecture, which was realized in an experimental prototype (proof-of-concept) as well as (b) the computational



(a) Routing via three distinct restaurants (b) Two restaurants in the computed corridor

Fig. 4: The results of both methods for the ride home with Chinese restaurants as white spots with the best one shown darker

results validating, that the chosen methods have enough potential to be applicable in a real-world setting.

6.1 System Architecture

A modular architecture is given by the fact that multiple (research) organizations participated in the project and certain components are *legacy systems* as the PT router. For the experimental prototype these architecture enabled us to have a fully distributed system with three independent servers. The main components of the created system are: the *user interface* in form of a website (UI), the *routing service* (router) and the *query answering* (QA). In addition, several supporting components like databases were incorporated into the system architecture, cf. Fig. 5. Following, the individual system components are described in more detail.

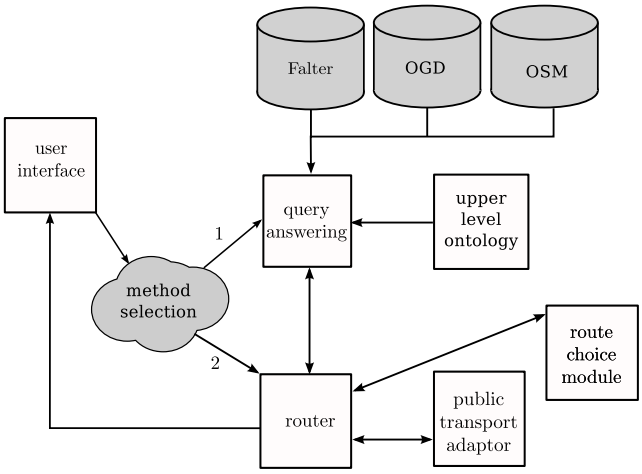


Fig. 5: System architecture as used for the proof-of-concept

User Interface. The UI forms the entry point for the user, cf. Fig. 5. All requests are given to the system and answers, i.e. routes and proposed POIs, are presented to the user via it. It is a JavaScript-enabled website using jQuery and OpenLayers with Spring and Apache Tomcat as middleware.

Router. The routing algorithm is implemented in Java 7, the web service is a Java RESTful service (`javax.ws.rs`) and runs on a Ubuntu 10.04 (x86_64) server with an Intel Xeon W3550 CPU (Quad-Core, 3.06GHz) and 12GB of RAM. Each request to the server is handled in a single thread, but threading is used to work on several requests at a time. At this time a standard Dijkstra algorithm is used for computing shortest travel time routes. The travel times for motorized individual transport are based on historical time-series data available in-house [14]. Travel times for walking and cycling are computed using standard travel speeds (1.34m/s for walking, 4.2 m/s for cycling). Finally, the travel times for public transport are based on average travel times between stations available through the public transport authority.

While different routes are computed by the routing service for all four modes of transport, some additional steps have to be performed after the final routes from/to the POIs have been obtained: First, it is necessary to order the routes according to the route choice model described in Section 3. Second, the routes for public transport have to be adapted via the official public transport router provided by the public transport company to align departure and arrival times with the up-to-date time-tables, cf. also Fig. 5.

Query Answering. Our keyword-based query answering approach runs on a Ubuntu 10.04 (x86_64) system with an

Intel Xeon CPU (Quad-Core, 2.53GHz) and 16 GB of RAM. It has been developed in Java 6 and uses PostGIS 1.5.1 (for PostgreSQL 9.0) as the spatial-extended database. Further, the parameters `shared_buffers` and `work_mem` of the PostgreSQL database were increased to utilize available RAM.

For the rewriting of *spatial conjunctive queries* (SCQ), we adapted OWLGRES 0.1 [22] to obtain the result of the *PerfectRef* algorithm, i.e., the *perfect rewriting*, of a conjunctive query and the *upper level ontology* (ULO). We implemented the evaluation of spatial predicates (e.g., "contains") in two different ways, one by using the native query evaluation of PostGIS, and the other as a built-in component of our query evaluation algorithm. For the first, we use the PostGIS functions, e.g., `ST_Contains(x,y)`, and for the second, we apply the functions of the JTS Topology Suite from http://sourceforge.net/projects/jts-topo-suite/. Note, that in our experiments, we solely consider the second approach, as in our setting it scales better (c.f. [13]).

The following three data sources building the backbone of the query answering part were used, cf. Fig. 5: First, we incorporated POIs like pharmacies, supermarkets, and parks contained in OSM. Second, data made available like public toilets, recycling collecting points, and playgrounds by the Viennese local governments via *open government data* (OGD) was included. Third, we had the possibility to include data provided by the local weekly magazine "Falter", which mainly covers restaurants, concerts, events and other information on entertainment offers.

Method Selection. The method selection decides which method (either route-constrained or post-filtered search) should be applied. Currently, this selection is implemented on the same machine as the query answering component. In the current version, the heuristic always applies route-constrained search for neighborhood routing while a post-filtered search is applied for via routing. To reduce computation times for the latter, we additionally restrict the semantic search by a bounding box such that routes to all existing POIs covered by the databases are avoided.

As indicated in Fig. 5, the method has to decide whether query answering or routing should be performed first (either path 1 or path 2). If routing is chosen first, corridors are provided to the query answering component, which uses them to spatially constrain the semantic search. Otherwise, an unconstrained semantic search is performed. As soon as some candidate POIs have been selected, these are given to the router and the final routes are computed, ordered and adapted (in the case of public transport routes). These routes are then directly returned to the user interface.

Table 1: Benchmark metrics for the upper level ontologies

	scenario	S_1	S_2
upper level ontology	concepts	324	324
	roles	32	32
	maximal depth	7	7
spatial instances	in OSM	$\approx 70\,000$	$\approx 8\,200$
	in OGD	$\approx 7\,200$	—
	in Falter	$\approx 3\,700$	$\approx 3\,700$
after annotation	instances	$\approx 28\,000$	$\approx 5\,200$
	concepts	$\approx 28\,000$	$\approx 5\,200$
	role assertions	$\approx 26\,700$	$\approx 15\,500$

6.2 Experiments

Scenarios, Modes of Transport, and Queries. For our experiments, we used two scenarios S_1 and S_2 for *greater* and *smaller Vienna*, respectively. The metrics of the resulting ULO are shown in Table 1. Since multi-modality was one focus of this work, we performed the experiments with four different modes of transport (MOT): *Foot*, *Bike*, *Car*, and *Public Transport*. The first three MOT graphs have roughly 32000 nodes and 90000 edges each. As the actual data is not publicly available, the size of the underlying graph for *Public Transport* is unknown.

We defined eight different queries varying in complexity (with respect to query answering), where we highlight the following four queries:

- Q_1 : (*pharmacy*) returns all pharmacies (many and well distributed over Vienna);
- Q_2 : (*bio supermarket*) returns all organic supermarkets (a few in Vienna);
- Q_5 : (*italian cuisine, guest garden, wlan, child friendly*) returns the instances which serve Italian cuisine (including pizzerias, etc.), have a guest garden, provide WLAN, and are child-friendly;
- Q_8: (*italian cuisine, guest garden, next to, atm, next to, metro station*) returns the restaurants with Italian food and a guest garden, whereby these instances are next to an ATM and a metro station.

The four queries represent the following classes of queries: a simple (concept) query with high selectivity, a simple (concept) query with low selectivity, a complex query with a large rewriting, and a rather complex spatial query. Q_1 and Q_2 have a rewritten query size of 1 (where we measure the size of a SCQ as the number of atoms in it); Q_5 has the largest query size of 255; Q_8 adds the evaluation of the spatial relation *next to*, which filters the three separate evaluations (*italian cuisine, guest garden*, *atm*, and *metro station*).

Experimental Setup. Based on a scenario S, a mode of transport M, and a query Q, we performed two performance tests: one for neighborhood routing (NR) and one for via

Table 2: Benchmark reference data for Neighborhood Routing (NR) and Via Routing (VR) performance tests

Scenarios (2)	S_1, S_2
Modes of transport (4)	*Foot*, *Bike*, *Car*, *Public Transport*
Queries (8)	$Q_1, \ldots, Q_8$
NR start coordinates (8)	$A_1^{\mathrm{NR}}, \ldots, A_8^{\mathrm{NR}}$
NR distance thresholds (5)	450, 900, 1350, 1800, 2250 seconds
VR origin-destination pairs (8)	$(A_1^{\mathrm{VR}}, B_1^{\mathrm{VR}}), \ldots, (A_8^{\mathrm{VR}}, B_8^{\mathrm{VR}})$
Total number of benchmark instances per scenario	
Neighborhood Routing	1280
Via Routing	256

routing (VR). The benchmark sets for neighborhood routing consist of instances of the form $\langle S, M, Q, A^{\mathrm{NR}}, \delta \rangle$, where A^{NR} is the starting coordinate for computing a δ-neighborhood for M. The benchmark sets for the via routing tests consist of instances of the form $\langle S, M, Q, A^{\mathrm{VR}}, B^{\mathrm{VR}} \rangle$, where $(A^{\mathrm{VR}}, B^{\mathrm{VR}})$ is a origin-destination pair.

For the neighborhood routing benchmark sets we use five increasing distance thresholds δ. The starting point A^{NR} are from a predefined set of eight POIs. In case of the via routing benchmark sets, we considered a predefined set of eight origin-destination pairs $(A^{\mathrm{VR}}, B^{\mathrm{VR}})$. For both NR and VR performance tests, we have chosen the set of starting points as well as the set of origin-destination pairs randomly from our log of historical user queries.

We performed 30 independent runs for each benchmark instance for measuring the response time performance of the system. All observations presented in the following are based on trimmed means with top and bottom runtimes for an instance removed.

Table 2 conveniently summarizes the reference data for our two performance tests. In total, we performed 76800 runs on NR instances over both S_1 and S_2, and 7680 runs on VR instances over S_1. The focus of S_2 is on the evaluation of query evaluation with DBs of different size. Therefore, we only consider the NR, because the challenging part of the evaluation lies the combination of an ontological and spatial query, which does not occur with the fixed bounding box of the VR.

Detailed results of all performance tests, POI names and their WGS84 coordinates of all chosen points including maps are given in Online Resource 1. We summarize their outcome in this article in Fig. 6 and 7.

General Result Observations. In general, it can be observed that the system architecture as given is capable of answering most of the explored experimental setups in acceptable times. I.e., 95% (resp. 67%) for NR and 96% (resp. 65%) for VR of the requests could be answered in less than five (resp. three) seconds. Although this performance is already convincing, it has to be highlighted that public transport requests have to be seen differently as all public transport routes have been obtained via the officially available public transport route server leading to sometimes exceptional long computation times. Therefore, we focus in the further discussion on MOTs *Foot*, *Bike* and *Car* only. Note that the different locomotion speeds for the different MOTs imply that the size of the isochrones generated for NR is increasing from *Foot* over *Bike* to *Car*.

Detailed Observations for Neighborhood Routing. We show here the results for the NR performance tests as average runtimes over the start coordinates $A_1^{\mathrm{NR}}, \ldots, A_8^{\mathrm{NR}}$ for each M and δ under scenarios S_1 and S_2 in Fig. 6. Fully detailed results of the experiments are documented in Online Resource 1.

Having a look at the total response times for NR (cf. Fig. 6a), it can be observed that they increase with growing neighborhood size. On the one hand, this behavior is explained by the increasing effort necessary for computing the corridors (isochrones have to be transmitted from the routing server to the query answering server and about 35% of the corridor computation times is consumed by the buffer calculations using the JTS Topology Suite). On the other hand, query times are also dependent on the neighborhood size, cf. Fig. 6b: we can see that the query times start to grow (fast) but flatten for larger neighborhoods. This behavior indicates that the corridor filter in the SQL selection has a strong impact on the performance which saturates when reaching a certain limit. While the query times for Q_1 and Q_2 are relatively low (they do not differ observably, hence selectivity has no influence on the query time), the query times for Q_5 are rather high which is apparently caused by the query size itself. Interestingly, Q_8 shows the largest deviation of query times, which can be explained by (a) the in-memory evaluation of intermediate results and (b) the three separate evaluations (producing the intermediate results), which forces the database system to render and filter the corridor three times.

With respect to the contribution to total response times, query times for Q_1 and Q_2 make up about 22% and 28%, respectively; query times for Q_5 lie at about 66% and query times for Q_8 vary from 65% (*Foot* with 450s) to 37% (*Car* with 1800s) of the total response times. Response times for the corridor calculation with respect to the total time is for Q_1 41%, for Q_2 48%, for Q_5 26%, and for Q_8 29%. Individual corridor calculations vary between 5% (*Foot* with 450s) and 70% (*Car* with 2250s).

The times for isochrone computations and final route determination need no further discussion, since classic Dijkstra computations are performed that are independent of the queries and/or selected POIs.

Detailed Observations for Via Routing. The results for VR with S_1 are shown in Fig. 7 and documented in full-length in Online Resource 1. Each category of the bar chart in Fig. 7

represents an instance with independent origin-destination pair.

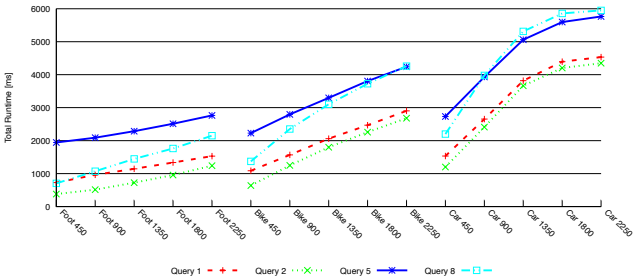
As shown in Fig. 7b, we have observed for all queries except Q_5 an identical behavior regarding query time. The queries, even with arbitrary different start- and end-points, have all an evaluation time between 138ms and 700ms. Regarding the total time, query answering for Q_1, Q_2, and Q_8 consumes in average 14%, 37%, and 19% of the total time. The difference between Q_1 and Q_2 can be explained due to the 10 (resp. 3) calculated routes. These results are indeed encouraging, as the semantic search performs usually below 1s.

We highlight that in query Q_1, at *Car 3 and Car 6* no routing is done as the endpoint is in a zone with no vehicular traffic. Another exception is Q_5, which consumes avg. 85% of the total time. We explain this by the size (and structure) of Q_5, which drives the query optimizer of PostgreSQL 9.0 to its limits. Then, in comparison to the other queries Q_5 has outliers for all modalities. The reason for this lies in the chosen start- and end-points, which are (far) to the SW and NE of Vienna, resulting in a bounding box containing all POIs of the database. The query optimizer seems to filter first the query joins from Q_5 (as WLAN and child-friendly), hence already reducing the number of possible POIs, and then evaluates the expensive spatial containment. For all other queries, first the spatial containment for numerous POIs is checked before the query joins are evaluated.

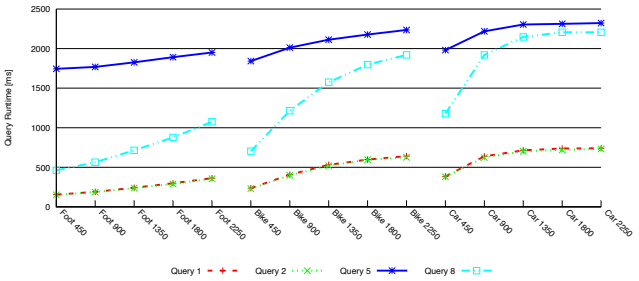
Note, that we have implemented only the post-filtered search, therefore the corridor calculation is simple an enlarged bounding box of start- and end-point. This bounding box is negligible, as it is calculated in $<$1ms.

Summing up, we observed encouraging results for NR and VR regarding the combination of semantic search and routing, which could be even more efficient if deployed on the same server, as the corridor transfer time would disappear.

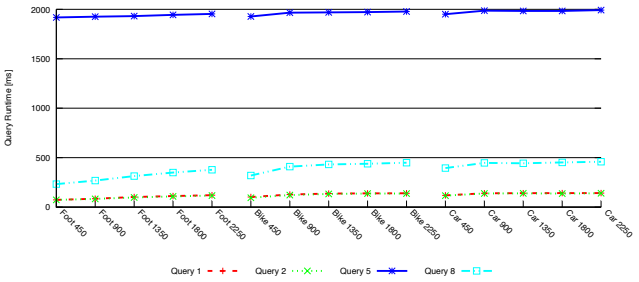
Detailed Observations with Respect to the Scenarios. The query answering times for NR comparing S_1 and S_2 are shown in Fig. 6c and documented in full-length in the Appendix. In this case, we have observed that the queries for the smaller scenario S_2 show an almost linear behavior for all modalities and durations. Furthermore, the expected impact of the size of the underlying databases can be seen. Especially, Q_8 suffers from increasing database size which can be explained by the fact that for this type of query all steps have to be performed multiple times (depending on the number of "next to" terms given).



(a) Total time of NR over S_1



(b) Query time of NR over S_1



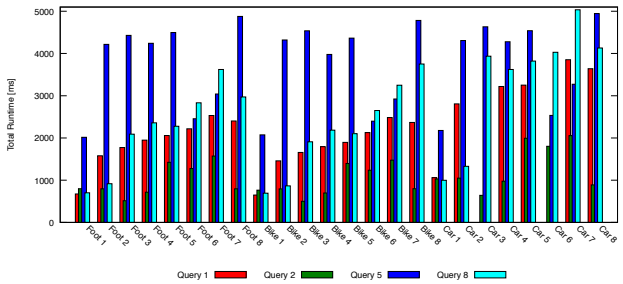
(c) Query time of NR over S_2

Fig. 6: Neighborhood routing (NR) with scenarios S_1 and S_2

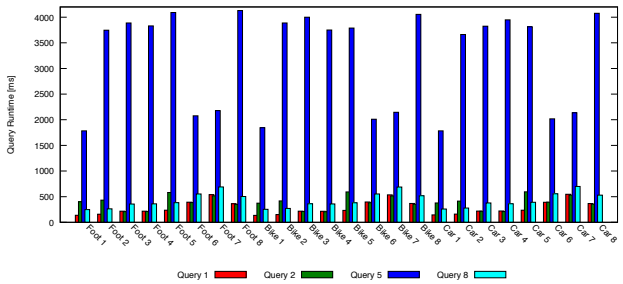
7 Discussion and Outlook

In this paper we proposed a novel *intention-oriented routing* approach, arising from the combination of two state-of-the-art technologies: routing and semantic web technology. We developed a general framework incorporating the central methods and components and presented the application of them on two use cases: *neighborhood routing* and *via routing*. Both methodologies are executed sequentially in the framework leading to two possible strategies: route-constrained and post-filtered semantic search. While the former consists of a semantic search constrained by the results of a previously performed routing phase, the latter computes routes for all results obtained via an unconstrained semantic search. The shape and composition of the corridors is of essential significance such that a proper result set meeting the user's intention can be generated. We primarily investigated two corridor computation methods, namely *convex hull* and *dilated hull computation*, both leading to distinct results for neighborhood routing. Furthermore, it could be observed that convex hulls are not meaningful with respect to via routing, where more complex hull computations need to be performed, e.g. alpha-shapes.

At present, we focused only on via routing via one specific point. However intention-oriented routing with multiple



(a) Total time of VR over S_1



(b) Query time of VR over S_1

Fig. 7: Via routing (VR) with scenario S_1

via points which are automatically determined is imaginable, e.g. for finding a route from home to work via a supermarket, a pharmacy, and a laundromat. It can be shown by a reduction from the well-known *Generalized Traveling Salesman Problem* that this problem is NP-hard, i.e., no polynomial time algorithm is likely to exist. However, in the presence of a sequence the problem can be solved via dynamic programming methods, cf. [15].

Algorithms currently applied in online routing services are based on a shortest route algorithm, where either the shortest or fastest route is presented to the traveler. However, when travelers decide on a route, in particular in PT, additional criteria are used to choose a particular route. This was addressed by ordering the routes using a random utility maximization approach. Furthermore individual level parameters were used to personalize the route ordering. However an extended data collection effort as well as further test of the system need to be performed before a conclusive statement can be made about route preferences of people in Vienna as well as the difference of preferences for different users.

We finally performed some benchmark experiments to get a deeper insight in the performance of the individual components and the inter-dependency of them. We could show that in all cases the total response times of the system are acceptable in terms such that an individual could explore his/her neighborhood or plan his/her routes by using the proposed system. If further performance enhancements are undertaken (e.g. migrating the complete system onto one physical machine), the system would be competitive compared to other route-planning systems already available now with respect to response time while functionality is obviously much higher.

Future research can be along different tracks. For the semantic search, the extension of our introduced spatial query language (e.g., with a cluster operator) could be investigated. Further, the mapping framework, could be realized on a rule-based framework as HEX-programs, which also could deal with duplicate elimination. Since we considered a limited set of data sources, new sources would bring an extension of the ontology for a particular domain, e.g., a movie domain if the sources are extended with cinemas.

For this paper, we applied a route-constrained search for neighborhood routing and a post-filtered search for via routing. For the neighborhood routing the route-constrained search is preferable, because the steps for creating a constraining corridor have to be calculated in any case. The isochrone is a by-product of the path computation and the dilated hull needs to be determined every time, as we do not apply classical map-matching. For the via routing both methods are suitable, however we only applied the post-filter search, as we did not develop suitable methods for generating alternative routes yet, which is necessary for creating a constraining corridor. Future work includes an in-depth analysis for situations where one method is superior to the other, which would provide the basis for developing heuristics that help to find a cut-off point for choosing a particular method. These heuristics would depend on parameters such as query, result and corridor size (as shown in the experiments in Section 6.2). A further result of this study would be a cost model that, given the input parameters, provides a useful estimation of the parameters. For instance, the result size can be estimated based on the ontology statistics of assigned instances. E.g., for the query “Italian restaurant”, we could derive that many instances are assigned to the concept “Restaurant” and “Italian Cuisine”, and if the estimated corridor size is small, the route-constrained method should be applied.

One major task still open is the enhancement of route-constrained semantic search such that *alternative* routes and corridors used for constraining the search. This means that in addition to the, e.g., fastest route, other fast (but not fastest) routes are computed and used for defining the corridor constraining the search. This task is, however, a very difficult one since even the definition of a valuable alternative is complex.

To validate the route ordering approach a larger, more representative data set needs to be collected. Furthermore, a test scenario should be developed that can be used to analyze if individual level parameters and personalized route suggestions help to improve the acceptance of a routing service and can be used further to influence users towards a more sustainable travel behavior. On the side of choice models several extensions should be considered. First, it will be important to include all modes into one model to be able to compare and

simultaneously order multimodal and uni-modal routes for all modes. Second, it is important to include the POI into the choice process. This can be done by estimating combined destination and route choice models. A second approach would be to combine ordering of the points done in the semantic search step with route choice models.

Acknowledgements Supported by the Austrian Research Promotion Agency (FFG) project P828897, and the Austrian Science Fund (FWF) projects P20841 and P24090-N23. We also thank our project partners Rosinak & Partner ZT GmbH, Fluidtime Data Services GmbH, ITS Vienna Region, new turn / DI Klaus Heimbuchner, and Falter Verlagsgesellschaft m.b.H. for their valuable input and support.

References

1. Baader, F., Horrocks, I., Sattler, U.: Description logics. In: S. Staab, R. Studer (eds.) Handbook on Ontologies, pp. 21–43. Springer (2009). DOI 10.1007/978-3-540-92673-3
2. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer (2008)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **284**(5), 34–43 (2001)
4. Bishr, Y.A.: Geospatial semantic web: Applications. In: Encyclopedia of GIS, pp. 391–398. Springer (2008). DOI 10.1007/978-0-387-35973-1_514
5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The mastro system for ontology-based data access. Semantic Web **2**(1), 43–53 (2011). DOI 10.3233/SW-2011-0029
6. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. Journal of Automated Reasoning **39**(3), 385–429 (2007) DOI 10.1007/s10817-007-9078-x
7. Cormen, T.H., Leiserson, C.E., Rivest, R.I., Stein, C.: Introduction to Algorithms, Second Edition. MIT Press, Cambridge (2001)
8. DeMers, M.N.: Fundamentals of geographic information systems (4. ed.). Wiley (2008)
9. Dijkstra, E.W.: A Note on Two Problems in Connection with Graphs. Numerical Mathematics **1**, 269–271 (1959)
10. Duckham, M., Kulik, L., Worboys, M.F., Galton, A.: Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. Pattern Recognition **41**(10), 3224–3236 (2008) DOI 10.1016/j.patcog.2008.03.023
11. Edelsbrunner, H., Kirkpatrick, D.G., Seidel, R.: On the shape of a set of points in the plane. IEEE Transactions on Information Theory **29**(4), 551–558 (1983) DOI 10.1109/TIT.1983.1056714
12. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer Set Programming. In: 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), pp. 90–96. Professional Book Center (2005)
13. Eiter, T., Krennwallner, T., Schneider, P.: Lightweight spatial conjunctive query answering using keywords. In: 10th Extended Semantic Web Conference (ESWC 2013), *LNCS*, vol. 7882, pp. 243–258. Springer (2013) DOI 10.1007/978-3-642-38288-8_17
14. Graser, A., Ponweiser, W., Dragaschnig, M., Brändle, N., Widhalm, P.: Assessing traffic performance using position density of sparse FCD. In: 15th International IEEE Conference on Intelligent Transportation Systems (ITSC 2012), pp. 1001–1005. IEEE (2012) DOI 10.1109/ITSC.2012.6338704
15. Hu, B., Raidl, G.R.: Effective neighborhood structures for the generalized traveling salesman problem. In: 8th European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2008), *LNCS*, vol. 4972, pp. 36–47. Springer (2008) DOI 10.1007/978-3-540-78604-7_4
16. Jones, C.B.: Geographical Information Systems and Computer Cartography. Prentice Hall (1997)
17. Krötzsch, M., Patel-Schneider, P.F., Rudolph, S., Hitzler, P., Parsia, B.: OWL 2 web ontology language primer. Tech. rep., W3C (2009). URL http://www.w3.org/TR/2009/REC-owl2-primer-20091027/
18. Lenzerini, M.: Data integration: A theoretical perspective. In: 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002), pp. 233–246. ACM (2002) DOI 10.1145/543613.543644
19. Moore, E.: The shortest path through a maze. In: H. Aiken (ed.) Proceedings of an International Symposium on the Theory of Switching, April 2–5 1957, Part II, The Annals of the Computation Laboratory of Harvard University Volume XXX, pp. 285–292. Harvard University Press, Cambridge, Massachusetts (1959)
20. Popov, B., Kiryakov, A., Kirilov, A., Manov, D., Ognyanoff, D., Goranov, M.: Kim - semantic annotation platform. In: ISWC 2003, pp. 834–849 (2003)
21. Rodríguez, M.A., Cruz, I.F., Egenhofer, M.J., Levashkin, S. (eds.): GeoSpatial Semantics (GeoS 2005), *LNCS*, vol. 3799. Springer (2005) DOI 10.1007/11586180
22. Stocker, M., Smith, M.: Owlgres: A scalable owl reasoner. In: 5th OWLED Workshop on OWL: Experiences and Directions (OWLED 2008), CEUR-WS, vol. 432. CEUR Workshop Proceedings (2008)
23. Train, K.: Discrete Choice Methods with Simulation. Cambridge University Press (2003)
24. Wu, D., Zhu, T., Lv, W., Gao, X.: A Heuristic Map-Matching Algorithm by Using Vector-Based Recognition. In: International Multi-Conference on Computing in the Global Information Technology (ICCGI 2007), pp. 18. IEEE (2007) DOI 10.1109/ICCGI.2007.3