Int J Data Sci Anal (2016) 2:29–44
DOI 10.1007/s41060-016-0030-1

REGULAR PAPER

On querying and mining semantic-aware mobility timelines

Stylianos Sideridis[1] · **Nikos Pelekis**[2] · **Yannis Theodoridis**[1]

Received: 15 July 2016 / Accepted: 13 October 2016 / Published online: 1 November 2016

Abstract The explosion of available positioning information associated with the inferred or user-declared semantics of the respective locations has contributed in what is called the Big Data era by posing new challenges to the mobility data management and mining research community. Motivated by a series of challenges posed in Pelekis et al. (SIGKDD Explor 15(1):23–32, 2013), in this paper, we present a unified framework for the management and the analysis of LifeSteps, i.e., data objects about human mobility including both (raw) spatio-temporal trajectories and their semantic counterpart. In particular, we provide solutions for developing real-world semantic-aware moving object database and trajectory data warehouse systems and we devise respective query processing algorithms. Our experimental study on realistic synthetic data including synchronized raw (i.e., GPS log) and semantic (i.e., diaries) information verifies the effectiveness and efficiency of the proposed framework.

Keywords Mobility data warehouses · Semantic trajectories · Pattern queries · Semantic mobility network

1 Introduction

Ubiquitous positioning devices enable the generation of huge volumes of location information on a continuous basis. The trend in modern *Location-based Services* (LBS) and *Location-based Social Networking* (LBSN) applications that make use of such data is not only to focus on raw movement information collected by those devices, but also to target at the behavioral rationale and motivation of movement, in order to provide added value and enriched services.

A semantically annotated trajectory, hereafter called in short *semantic trajectory*, is an alternative representation of the (raw) spatio-temporal motion path of a moving object as this is logged by the positioning device. Instead of a sequence of space-time information, in a semantic trajectory the motion is represented as a sequence of semantically meaningful episodes, typically *stops* (e.g., at home, at office, for shopping) and *moves* (walking, driving, etc.), a representation which results in detecting homogenous fractions of movement [9]. Extracting and managing semantics from (raw) trajectory data is a promising channel that leads to significant storage savings. Of course, as already declared in [13], it is not only a matter of the database size; maintaining semantic (textual, for the purposes of this paper) information turns out to be quite useful in terms of context-aware movement analysis. In fact, semantic-aware abstractions of motion enable applications to better understand and exploit on human mobility: for instance, identify those locations where some activity (work, leisure, relax, etc.) takes place, infer how long does it take to get from one place of interest (POI) to another (e.g., from home to office) using a spe-

This paper is an extended version of the DSAA'2015 Long Presentation paper "Hermes^{sem}: a Semantic-aware Framework for the Management and Analysis of our LifeSteps" [14].
Source code, datasets etc. for Hermes^{sem} and Hermoupolis are available at http://infolab.cs.unipi.gr/{hermes|hermoupolis}.

✉ Nikos Pelekis
npelekis@unipi.gr

Stylianos Sideridis
siderste@yahoo.gr

Yannis Theodoridis
ytheod@unipi.gr

[1] Department of Informatics, University of Piraeus, Piraeus, Greece

[2] Department of Statistics and Insurance Science, University of Piraeus, Piraeus, Greece

cific transportation means, conclude about the frequency of an individual's outdoor activities, calculate indices related to environmentally friendly or sustainable mobility, and so on. Given such analytics, future semantic-aware LBS/LBSN applications can be established, including location sharing and ranking, recommendation according to travel and socio-demographic similarity (e.g., for dynamic ridesharing purposes).

Services that take advantage of the knowledge that is hidden in spatio-temporal-textual sequences are already available (e.g., Foursquare check-ins). However, there are other emerging application domains and tools that could benefit from such analytics. For instance, in the maritime and aviation domain there are recent efforts aiming at novel mobility forecasting, based on semantically re-constructed synopses of raw data collected by surveillance data sources [5,6].

More specifically, in the maritime domain, raw AIS (Automatic Identification System) signals transmitted by vessels and collected by AIS receivers are transformed to semantic trajectories where episodes in this domain correspond to identified critical phenomena in the vessels' routes ('stop,' 'turn,' 'drift,' 'move slowly,' etc.), or more complex events ('suspicious delay,' 'possible rendezvous,' 'approach fast,' 'pick package', etc.) [10,11]. Respectively, in the aviation domain, surveillance Automatic Dependent Surveillance–Broadcast (ADS-B) signals are processed in order for aircrafts' routes to be transformed to sequences of characteristic points where certain events take place ('push-back/towing,' 'taxiing,' 'take-off,' 'initial climb,' 'landing,' etc.). In both of the above application domains, semantics can be further enriched if linked with external data sources, such as weather reports and static geographic databases (seaports, airports, protected regions, etc.).

Transportation experts can benefit greatly by exploiting all this information in a unified and easy to use framework. For example, areas of unusual vessel behavior or passages through areas of special interest (like biodiversity zones) could be identified and so appropriate actions could be taken. Analogous study would be possible for airways. Even movement prediction in near real time is possible where the framework extensibility allows the implementation of advanced prediction algorithms. Traffic scientists, already using frameworks to analyze spatio-temporal movement, would additionally have the ability to semantically connect movement patterns and behaviors and come to crucial conclusions regarding, for instance, the traffic policy of either a small- or a large-size city.

Finally, recent advances in the simulation domain make use of mining analytics that extract collective behavioral patterns which are synthetically re-played so as to allow realistic simulation of moving objects mobility [15]. Using this technique, experts or companies are able to apply various movement scenarios and make appropriate decisions (e.g., to prioritizing some pathways over other or advertiser to place their advertisement in congestion areas).

Although conceptually strong, such a representation of mobility lacks a robust DBMS support in order to realize the above model and give life to novel applications and services based on this. In this paper, we present a unified framework that provides efficient and effective storage, indexing mechanisms, and query language able to support Semantic Mobility Databases (SMD). More specifically, the merits and contributions of our proposal are summarized below:

- Following the successful MOD engine paradigm, we design a datatype system and its associated query language, coined Hermes^{sem}, for the representation and management of SMD engine into an extensible DBMS architecture. Furthermore, we present a design for modeling Semantic Mobility Databases Cubes (SMDC).
- We propose efficient access methods for semantic trajectories, called SemTB-tree and Sem3DR-tree (extensions of the well-known TB-tree [17] and 3DR-tree [19], respectively, designed for (raw) trajectory data), for the hybrid indexing of both the spatio-temporal and the semantic (i.e., textual) component.
- We develop efficient query processing algorithms upon the proposed indices in order to support a useful query type at the SMD level, called *spatial-temporal-textual pattern* (ST^{2}P) query, as well as algorithms for efficiently feeding SMDC from SMD.

The paper is structured as follows: Sect. 2 presents related work. Section 3 provides background information. Sections 4 and 5 present the core of the paper, namely the realization of the proposed SMD and SMDC, respectively. Section 6 evaluates Hermes^{sem}. Finally, Sect. 7 concludes this paper.

2 Related work

Modeling, management, and knowledge discovery aspects on (raw) spatio-temporal trajectories of moving objects have been exhaustively researched in the past two decades [16], including plenty of algorithms and systems, spreading from data management [3] to data mining [1]. On the other hand, semantic mobility data management is a relatively new entry in the research agenda. Models for semantic trajectories include [9,13], while techniques for extracting semantic trajectories from raw ones have been also proposed recently [22]. In [16], the interested reader may find a survey of relevant models and techniques.

According to the state-of-the-art model [9], a semantic trajectory is defined as a sequence of episodes, labeled either as 'Stops' or 'Moves', each associated with appropriate meta-

data (tags). Technically, Stops are places (points or regions) where the object remains *static* (whatever definition is used to realize this behavior) and Moves are the parts of the object's trajectory in between two Stops, i.e., where the object is *moving*. This model was extended in [13] in order to enable the management of such data to extensible database architectures, as well as to support their modeling and analysis at various scales and/or spatio-temporal granularities.

Traditionally, aggregated information from DBs is stored in a Data Warehouse (DW), in the form of data cubes [2]. Data cubes are views of a DW, used for multi-dimensional analysis, the so-called OnLine Analytical Processing (OLAP). The data cube paradigm has been extended to support spatial [4] and (raw) trajectory DWs [7,8,18], involving spatial, temporal, and thematic dimensions as well as spatial, spatio-temporal, and numerical measures. A DW model for semantically-enriched mobility data, called Mob-warehouse, was proposed in [20] to enrich trajectory data with domain knowledge by following the so-called 5W1H model (Who, Where, When, What, Why, How). In [5], another semantic model tailored to open, linked data were proposed. In [13], a graph-based representation of mobility-aware data cubes was proposed.

Apart from the inclusion of semantics at the conceptual level, one of the challenges raised in [13] was the necessity to efficiently support the management and analysis of such data. Thus, [13] sketched the big picture of an envisioned three-tier framework, as follows: (i) at the bottom-layer, a traditional MOD lies, being in charge of the raw mobility data and supported by well-known access methods and query functionality [8,16]; (ii) at the middle-layer, it is the SMD that provides novel datatypes, indexing methods, and operators extending MOD query languages for querying and analyzing mobility data from a semantic perspective; (iii) at the top layer, the application interface provides users with querying and analysis functionality on either MOD or SMD, via simple SQL.

This paper presents a data management and analysis framework, which is a realization of that vision. To the best of our knowledge, this approach is novel and provides a valuable tool in the hands of the researchers in the field.

3 Background

Formally, the (*raw*) *trajectory* τ of a moving object is defined as a tuple (*o-id*, *traj-id*, T), where *o-id* (*traj-id*) is the identifier of the moving object (the specific trajectory of the moving object, respectively) and T is a 3D polyline consisting of a sequence of N+1 pairs (p_i, t_i), $0 \leq i \leq N$, where p_i is a 2D point (x_i, y_i) in the plane and t_i is a timestamp, assuming linear interpolation between two consecutive pairs (p_i, t_i) and (p_{i+1}, t_{i+1}). In turn, a (raw) trajectory defined as above can be partitioned into a sequence of (raw) sub-trajectories; formally, a (*raw*) *sub-trajectory* τ' of a (raw) trajectory τ is defined as a tuple (*o-id*, *traj-id*, *subtraj-id*, T'), where *o-id* (*traj-id*, *subtraj-id*) is the identifier of the moving object (the specific trajectory and sub-trajectory of the moving object, respectively) and T' is the portion of T between two timestamps, t_i and t_j, $t_i < t_j$. We are now able to define their semantic variants, namely mobility timelines (for trajectories) and LifeSteps (for sub-trajectories, respectively). As an example, Fig. 1 illustrates a mobility timeline consisting of seven (7) LifeSteps.

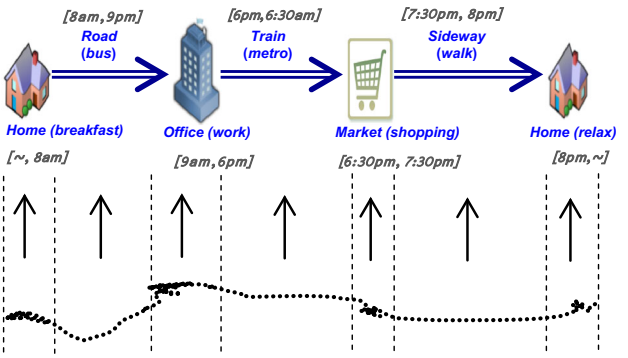

Fig. 1 A mobility timeline consisting of 7 LifeSteps

Definition 1 (*LifeStep*) A LifeStep ls corresponds to a sub-trajectory τ' and is defined as a tuple (LifeStepID, LifeStepFlag, MBB, tags, T-link), where LifeStepID is the identifier of the LifeStep, LifeStepFlag is a flag taking values from set {'Move', 'Stop'}, MBB is a tuple (MBR, t_{start}, t_{end}) corresponding to the 3D approximation of τ', with MBR being the 2D enclosing rectangle of the spatial projection of τ' in 2D plane and [t_{start}, t_{end}] being the 1D temporal projection of τ' in 1D timeline, tags is a set of keywords, describing the corresponding activities and semantic annotations related to this portion of movement, T-link is a link to τ'.

Definition 2 (*Mobility timeline*) A mobility timeline τ^{sem} of a moving object is defined as a triple (o-id, timeline-id, T_{LS}), where o-id (timeline-id) is the identifier of the moving object (the mobility timeline of the moving object, respectively) and T_{LS} is a sequence of LifeSteps belonging to the same trajectory τ and being successive in time, i.e., $ls_i[t_{end}] = ls_{i+1}[t_{start}]$.

Looking at the above definitions, it is clear that the content of a MOD (raw trajectories) and the respective of a SMD (mobility timelines) do not share much in common. This means that existing MOD engines, such as Hermes [16] and Secondo [8], cannot be used as-is in order to handle a SMD. For instance, queries like "Find people crossing the city center on their way from office back to home" or "Find people driving more than 20 km on their way from home to office" or

"Find people spending more than 1 hour daily for bring-get activities of their children at schools" cannot be easily supported by the above MOD engines since they make strong use of semantics. Nevertheless, these queries are typical examples of "what-if" analysis in the transportation domain and, as such, they ask for efficient support in DBMS.

The above discussion results in a categorization of queries over MOD/SMD in at least three types [13]. Q1-type queries like range, nearest-neighbor, enter, crossover raw trajectories have been extensively studied in the MOD literature. On the other hand, Q2-type queries (that involve semantic trajectories only) and Q3-type queries (that involve both raw and semantic trajectories) are innovative and they cannot be considered as straightforward variations of Q1-type ones. Moreover, Q4-type queries include advanced operations, such as pattern queries, over SMD. A typical example is the following: "Find people who follow the home–office–*–gym pattern", where we search for LifeSteps including the specific sequence (with the wildcard '*' denoting 'any').

As a step forward, we introduce the notion of *Semantic Mobility Data Cube* (SMDC), where aggregated data should not only expose interesting measures with respect to the chosen dimensions via a relational format, but they should also encapsulate the spatial topology and its intrinsic relationships. To succeed this ambitious goal, we exploit on the so-called *Semantic Mobility Network* (SMN) [13], a dynamic graph representation of the semantic mobility timelines. A nice interesting characteristic of a SMN, which we consider as a novel characteristic of our approach in the mobility management and analysis domain, is that this graph-based design is data-driven and unifies all the mobility-related dimensions (space, time, and semantics). Below, we provide formal definitions of a SMN and an aggregate SMN, while the reader is referred to [13] for several examples that illustrate the merits of this representation:

Definition 3 (*Semantic Mobility Network—SMN*) A semantic mobility network N is a graph denoted by $N = (V, E, M)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges and $M = \{M_1, M_2, \ldots, M_n\}$ is a set of measures applicable to vertices and edges, i.e., $\forall v \in V$ and $\forall e \in E$, there is a tuple $M(v)$ of v and $M(e)$ of e, respectively, denoted as $M(v) = (M_1(v), M_2(v), \ldots, M_n(v))$ and $M(e) = (M_1(e), M_2(e), \ldots, M_n(e))$, where $M_i(v)$ and $M_i(e)$ is the value of v and e on i-th measure, $1 \leq i \leq n$. The set V of vertices corresponds to the union of all distinct LifeSteps that are of 'Stop' type, of all mobility timelines τ^{sem}, while the set E of edges corresponds to the union of the 'Move' type LifeSteps. The set M of measures is a set of scalars quantifying properties of vertices and edges.

Definition 4 (*Aggregate Semantic Mobility Network—ASMN*) Given a semantic mobility network N, a set of dimensions $D = \{D_1, D_2, \ldots, D_m\}$ with their corresponding hierarchies, an aggregation $D^a = \{D_1^a, D_2^a, \ldots, D_m^a\}$ along these hierarchies, with $D^a(v) = (D_1^a(v), D_2^a(v), \ldots, D_m^a(v))$ denoting a tuple of values $D_j^a(v)$ of v on j-th dimension, $1 \leq j \leq m$ ($D^a(e)$ is defined similarly), upon which measure M_i can be aggregated, an aggregate semantic mobility network with respect to D^a is a semantic mobility network $N^a = (V^a, E^a, M^a)$, where:

i V^a is the set of aggregate vertices $v^a \in V^a$, each of which is constructed by a unification process $U_V([v])$ upon a non-empty equivalence class $[v]$ of V, where $[v] = \{v | D_j^a(v) = D_j^a(u), v, u \in V, j = 1, \ldots, m\}$,
ii E^a is the set of aggregate edges $e^a \in E^a$, each of which is constructed by a unification process $U_E(E(v^a, u^a))$ upon a non-empty edge set (i.e., 'Move' LifeSteps), where $E(v^a, u^a) = \{(v, u) | v \in [v], u \in [u], (v, u) \in E\}$, and
iii M^a is the set of aggregate measures, each of which is computed by applying an aggregate function $A(\cdot)$ on the measure values $M_i(v), v \in [v]$ and $M_i(e), e \in E(v^a, u^a)$, respectively, $1 \leq i \leq n$.

Note that the set of measures may be different for vertices and for edges (e.g., average stop vs. move duration) depending on the application, while A(Y) aggregate function may differ from measure to measure (e.g., count(Y) or average(Y)). Of course, spatial or spatio-temporal measures may require more sophisticated aggregate functions (e.g., the 'mean' LifeStep), which are out of the scope of the current work. The key issue for the aggregate function is to avoid being holistic, because in that case, super-aggregates cannot be computed from sub-aggregates, even if we employ auxiliary measures [2]. Note also that the unification process U_V (U_E) operates on the spatio-temporal and semantic properties of the 'Stop' ('Move') LifeSteps, respectively.

In Fig. 2, we visualize a 3-days semantic mobility timeline the derived semantic mobility network and its corresponding aggregate semantic mobility network over a period of time. Notice how natural is the passage from one model to the next, thus allowing implementing analytics and extracting valuable information.

Definition 5 (*Semantic Mobility Data Cube—SMDC*) Given a semantic mobility network N = (V, E, M) and a set of dimensions $D = \{D_1, D_2, \ldots, D_m\}$ with their corresponding hierarchies, the semantic mobility cube is the lattice of the aggregate semantic mobility networks produced by all possible aggregations in D.

The above definition implies that given a SMN N, each aggregation D^a of D, called a *semantic mobility cuboid*, is itself a graph. To the best of our knowledge, this modeling approach is novel for the mobility domain, since it has been studied only for non-spatial, vertex-specific multidimensional networks of traditional datatypes [23].

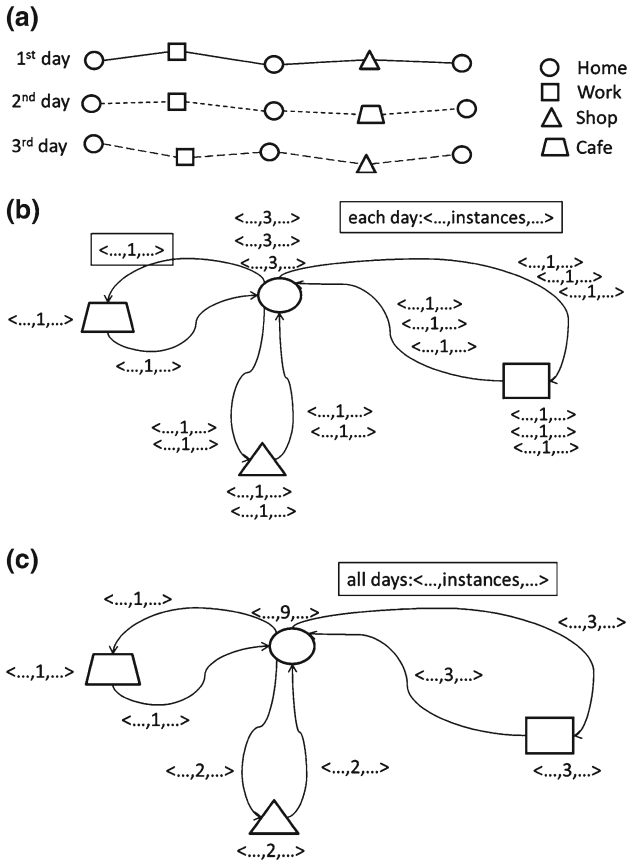

Fig. 2 **a** The abstract sequence of a 3-days semantic mobility timeline, **b** its mobility network, and **c** its aggregate network over a period of time

It is interesting to note a few example operations that can be applied to this framework for progressive analysis purposes. For instance, after we "extract the aggregate semantic mobility network A of user Bob during a period of time", we could "restrict it at a particular region of interest", by using a range query. Assuming this network is at the base cuboid level, we could then join it with "the aggregated (over a period of time) network B of a set of users (e.g., Bob's friends and co-workers according to a social network)", as such ASMN B resides at a higher level in the lattice than ASMN A. The join result (which obviously is a novel cross-SMN operation) could identify Bob's mobility network wherein he performs similar activities at similar places following similar routes with his friends.

Given the above discussion, and following the typology of query types proposed in [13], Q5- and Q6-type queries involve SMN, perhaps with the aid of crossover operations that link MOD and SMD. Clearly, both types of queries are innovative and have not been addressed in the related work on semantic trajectories [9].

Before we proceed to the discussion about $Hermes^{sem}$ framework, in Fig. 3 we present its big picture.

4 Modeling mobility timelines and semantic mobility networks

This section presents our proposal for modeling the concepts defined in Sect. 3, namely mobility timelines (Sect. 4.1) as well as SMN and SMDC (Sect. 4.2).

4.1 Modeling mobility timelines

Toward the realization of the concepts of LifeSteps and mobility timelines presented in Sect. 3, we follow the object-relational (OR) approach and extend the type system of Hermes MOD engine [12] and its associated query language. In detail, we follow the abstract data type (ADT) paradigm and define the so-called LifeStep and timeline datatypes (the former being subtype of the latter) that support Definitions 1 and 2, respectively (see Sect. 3). Upon these datatypes, we register a rich palette of object methods; some indicative examples appear in Table 1. More advanced methods include confinement of a timeline in spatial-temporal-textual cube, calculating the distance between two LifeSteps or between two timelines in each spatial-temporal-textual dimension, and more. The interested reader is referred to [13] and [12] for more details.

The resulted query language, i.e., the well-known SQL extended with methods and operators over the new datatypes, is appropriate for such complex (spatial-temporal-textual) objects, which can actually be considered as synchronized GPS traces along with diary information. Later, in Sect. 7, we present real examples of the query language in SQL syntax.

4.2 Designing efficient SMDC over SMN for OLAP purposes

Defining aggregations over mobility networks is a problem related to the graph cube problem [23]. Recalling the idea of Trajectory DW (TDW) [18], we propose a SMN to be a constellation scheme consisting of five dimensions (namely, space, time, user-profile, STOP-type-activity, MOVE-type-activity) and two fact tables (namely, STOPS-fact and MOVES-fact). Intuitively, this approach allows the support of several kinds of analysis:

- STOPS-Fact-table: who made a stop? when and where? what was the activity during the stop?
- MOVES-Fact-table: who made a movement? when and from/to where? How was the movement made and what was the activity during the movement?

Figure 4 provides a relational scheme of an effective modeling of SMN, where the measures of the fact tables

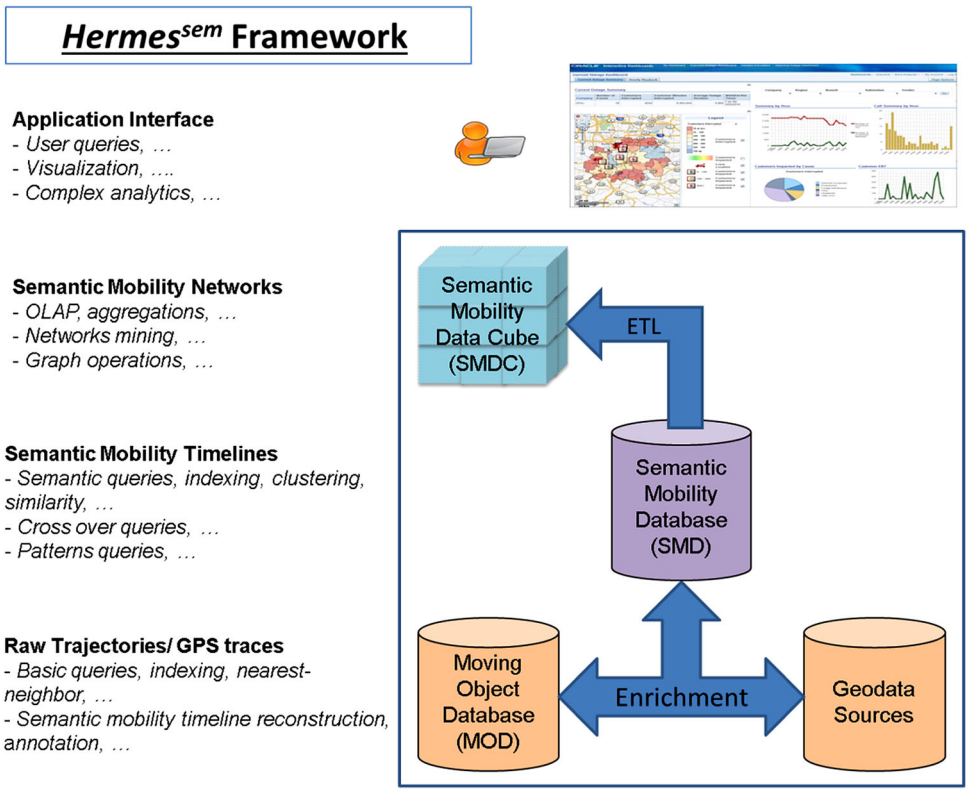

Fig. 3 An overview of $Hermes^{sem}$ framework

Table 1 Methods over LifeStep and timeline datatypes

Method name	Description
Q1-type queries	
moving_point.at_instant(timestamp) return point	Returns point geometry where moving object were at given timestamp
moving_point.at_period(time_period) return moving_point	Returns moving_point restricted at given period
moving_point.f_direction(timestamp) return number	Returns the angle of the moving object at given timestamp
moving_point.f_avg_speed return number	Returns the average speed of the moving object in its lifespan
moving_point.f_enter(geometry) return timestamp	Returns the timestamp when moving object enters a given geometry
moving_point.f_leave(geometry) return timestamp	Returns the timestamp when moving object leaves a given geometry
moving_point.f_max_speed return number	Returns the max speed of the moving object in its lifespan
moving_point.f_duration return number	Returns the duration of the moving objects' lifespan
moving_point.potential_activity_area return geometry	Returns the potential activity area of a moving object
range(geometry, time_period, root) return moving_point_array	Returns moving_points restricted in a spatiotemporal window using index
Q2-type queries	
sem_mbb.intersects(sem_mbb) return boolean	Returns TRUE if LifeStep's MBB intersects with given MBB
sem_mbb.duration return number	Returns the duration of a LifeStep
sem_mbb.area return number	Returns the area of the Lifestep's MBB
sem_episode.duration return number	Returns the lifespan of a LifeStep

Table 1 continued

Method name	Description
sem_episode.sim_episodes(sem_episode) return number	Returns the similarity (distance) of the LifeStep with given LifeStep in spatio-temporal-textual terms
sem_episode.tlink return moving_point	Returns the raw sub-trajectory of the LifeStep
sem_trajectory.getMBB return MBB	Returns the MBB of a Timeline
sem_trajectory.num_of_stops return integer	Returns the number of STOP LifeSteps contained in a Timeline
sem_trajectory.num_of_moves return integer	Returns the number of MOVE LifeSteps contained in a Timeline
sem_trajectory.episodes_with (tag varchar2) return sem_episode_tab	Given a string 'tag' that is a concatenation of tags, returns the set of LifeSteps of a Timeline that match with the content of 'tag'
sem_trajectory.confine_in(geometry, time_period, tag) return sem_trajectory	Returns the Timeline restricted in given spatio-temporal-textual window
sem_trajectory.sim_trajectories(sem_trajectory) return number	Returns the similarity (distance) of the Timeline with given Timeline in spatio-temporal-textual terms
range_episodes(sem_episode) return sem_episode_array	Returns LifeSteps that intersect with the given LifeStep using the index
range_episodes(geometry, time_period) return sem_episode_array	Returns LifeSteps of type 'STOP' that intersect with the given spatio-temporal window using the index
from-to-via(from_episode, to_episode, via_episode) return sem_episode_array	Returns LifeSteps of type 'MOVE' that began and end to given LifeSteps and obey constraints imposed by the 'via' LifeStep, using the index
Q3-type queries	
range_episodes(geometry, time_period).tlink. at_period(time_period) return sem_episode_array	Returns moving objects restricted in given time period from 'STOP' LifeSteps that intersecting a given spatio-temporal window
range_episodes(geometry, null) .tlink.f_duration return number	Returns durations of 'STOP' LifeSteps that intersecting a given spatial window
Q4-type queries	
od_matrix(SMD) return table	Returns the Origin-Destination matrix given an SMD (either on LifeSteps or on Timelines)
pattern(sem_episode_array, wildchars) return o-id	Returns objects identifiers the follow given movement pattern

correspond to the weight vectors of a SMN. These measures are similar with the ones used for TDW [18], but here they are trivially customized for 'Stops' and 'Moves'. Also note that these measures are subject to the *distinct count problem* when computing super-aggregates by sub-aggregates. This issue is tackled with a similar manner as in [18].

Adopting the relational model is in order for the proposed framework to be fully compatible with our approach of extending a real MOD engine with semantic functionality. This actually allows us to use the extended query language when feeding the SMDC during the ETL process. Deriving a SMN from a SMD is a computational challenging task. For instance, the SMN is built at a very refined spatial granularity (namely, at the level of the POIs and not at the region level, as in the case of TDW [18]). In the following section, we provide alternative approaches for feeding a SMDC.

5 Indexing and query processing over SMD and SMDC

The realization of the above-described framework raises a natural question: how would a SMD be developed to provide efficiency in storage and querying, as well as effectiveness in analytics? In this direction, we first propose access methods for indexing mobility timelines (Sect. 5.1). Then, we provide algorithms for the efficient processing of the so-called spatial-temporal-textual pattern query (Sect. 5.2) and the efficient feeding of SMDC (Sect. 5.3).

5.1 Indexing mobility timelines

As usual in ADT, the introduced query operators call for efficient index support. Motivated by solutions already proposed in the field of geographic information retrieval [21], we

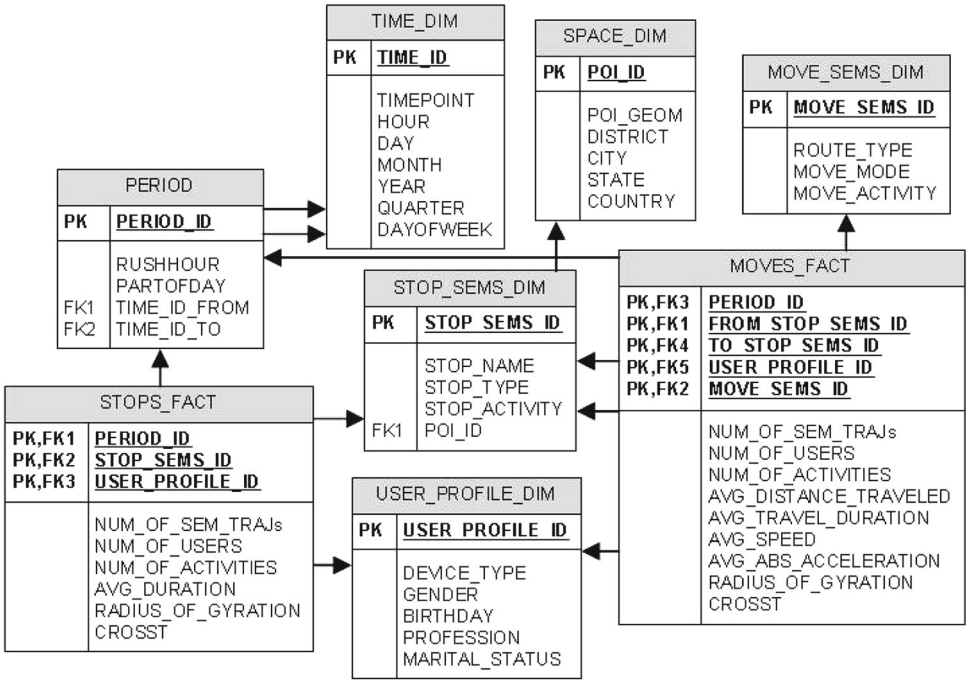

Fig. 4 A data cube for SMN; a constellation scheme consisting of two fact- and five-dimensional tables

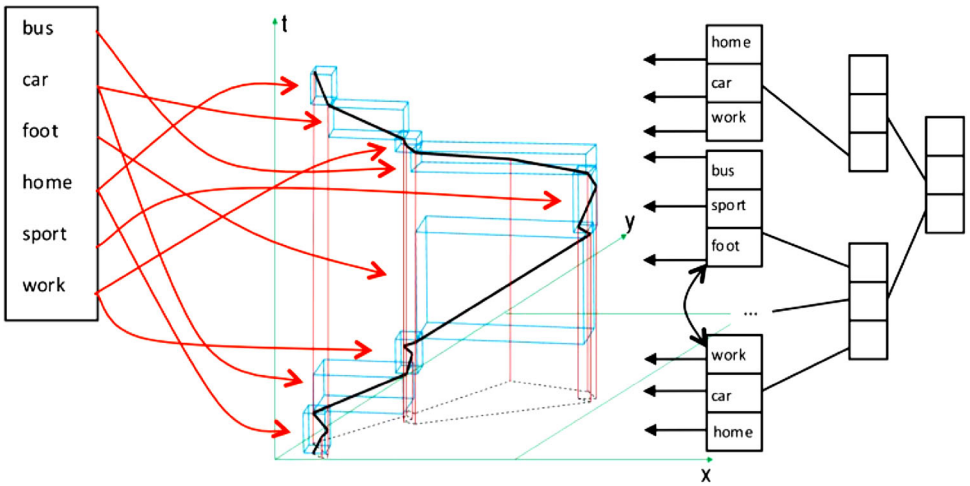

Fig. 5 Hybrid indexing of spatial-temporal-textual information combining 3-dimensional indices for (raw) trajecotries and inverted files

propose hybrid access methods that extend the well-known TB-tree [17] and 3DR-tree [19] access methods, proposed for (raw) trajectory data, by combining them with an appropriate text index (inverted file). The architecture of our hybrid indexing scheme is illustrated in Fig. 5.

In particular, Fig. 5 exhibits how the LifeSteps of a single timeline are indexed. Let us consider a timeline consisting of nine LifeSteps (the 3D MBBs of which are illustrated in the middle of Fig. 5). The right part of Fig. 5 illustrates a three-dimensional index for (raw) trajectories, could be a TB-tree or a 3DR-tree, constructed by these MBBs, enhanced at the leaf level with the textual information assigned to each LifeStep. Thus, the entries of the leaves contain the tags of the LifeSteps, as well as the MBBs of the LifeSteps' sub-trajectories (in Fig. 5 the latter are pointed by the arrows). This choice allows for queries with combined spatio-temporal and textual constraints. At the left part of the architecture, we adopt a typical inverted file organizing the tags that appear at the LifeSteps. We call the overall scheme SemTB-tree or Sem3DR-tree, depending on the respective trajectory index adopted at the right part.

Whatever is the choice, note that such a spatio-temporal index is different from the one we would have, if we decided to index the initial trajectories. This is due to the segmentation of trajectories to sub-trajectories that produces a more effective partitioning of the space-time with less dead space, as 'Stop' sub-trajectories (intuitively, they correspond to long MBBs in the time dimension with small spatial footprint) are not mixed with 'Move' sub-trajectories (intuitively, they have large spatial extent). This mixing is a source of inclusion

of dead space in the structure, and state-of-the-art indexing methods have been proposed to tackle this issue. Here, we implicitly tackle it via the prior segmentation of the initial raw trajectories with respect to their semantics.

It follows that a comparative performance assessment between TB-tree (3DR-tree) and SemTB-tree (Sem3DR-tree, respectively) is not fair or even meaningful. TB-tree indexes segments of raw trajectories, while SemTB-tree indexes approximations of sub-trajectories (which correspond to a number of segments), thus the structures of the two constructed indexes are quite different in terms of both the number of nodes and the height of the trees. Moreover, SemTB-tree incorporates semantics (i.e., textual data in the form of keywords) that the original TB-tree does not take into account. Exploiting the indexing of semantics, SemTB-tree can filter out candidate solutions faster than TB-tree. Likewise for 3DR-tree and Sem3DR-tree.

5.2 Querying mobility timelines

In a SMD, an interesting operation is *searching for mobility timelines that follow a specific sequence pattern*. Of course, the challenge that arises is that LifeSteps composing timelines impose textual as well as spatio-temporal constraints. Thus, the query "Find people who follow the home–office–*–gym pattern" implies that the user may add spatio-temporal constraints at each of the textual constraints (i.e., search for "home" in region r during temporal period p). A *Spatial-Temporal-Textual Pattern* (ST^2P) query in a SMD is essentially a (simplified) regular expression consisting of LifeStep objects. In particular, it is defined as a sequence of LifeSteps that forms a search pattern in a SMD. Formally:

$$\boldsymbol{Q}:=<\boldsymbol{p}*|\boldsymbol{p}\ \textbf{is either a LifeStep}\ \boldsymbol{ls_{\mathrm{i}}}\ \textbf{or}\ \textbf{a wildcard}\ \boldsymbol{w}\in\{>,*\}> \quad (1)$$

For ease of exposition, Q is represented as an array of LifeSteps L along with an array of wildcards W. The array of wildcards consists of either '>' or '*' (or the empty symbol $\emptyset$), which may be placed in between consecutive LifeSteps. The existence of '>' between two LifeSteps ls_{i} and $ls_{\mathrm{i+1}}$ implies that ls_{i} is immediately followed by ls_{i+1}, while '*' means that there may exist an arbitrary number of other LifeSteps in between ls_{i} and $ls_{\mathrm{i+1}}$. Thus, the array of wildcards W is consistent with the array of LifeSteps L. For instance, for $L = [ls_1, ls_2, ls_3, ls_4]$ and $W = [\emptyset, >, *, >]$, the pattern

$$\mathrm{Q}(L, W) = [ls_1 > ls_2 * ls_3 > ls_4]$$

conforms to timelines that start from ls_1, immediately followed by ls_2, then followed by an arbitrary number of LifeSteps of any type, then followed by ls_3, immediately followed by ls_4, which is the ending LifeStep of the timeline.

Algorithm ST^2P provides the pseudocode for processing ST^2P queries using the hybrid indexing scheme (either SemTB-tree or Sem3DR-tree) proposed in the previous subsection. Before providing the details of the algorithm, we should note that pattern_tags() searches the index starting from the inverted file (Fig. 5, left), while pattern_mbb() searches the index starting from the spatio-temporal index (Fig. 5, right).

The algorithm iterates through the LifeSteps of the query pattern (lines 3–9) and finds candidate solutions that satisfy textual constraints (using the pattern_tags() function), which are used to prune the spatio-temporal space (using the pattern_mbb() function). In each of the following iterations, the algorithm moves on to the next input LifeStep and retrieves a corresponding set of LifeSteps from SMD. During each iteration, candidate solutions must also constitute a 'continuation' with the solutions found in the previous step.

Algorithm ST^2P
Input: a pattern query Q as an array of LifeSteps *L* along with an array of wildcards *W*, the *root* of the index
Output: the IDs of the timelines that conform to pattern Q=(*L*,*W*)
1. **begin**
2. *curr_sol* = ∅
3. **for each** ls_i in *L*
4. *curr_sol* = *pattern_tags*(*L*(*i*), *W*(*i*), *curr_sol*, root)
5. **if** curr_sol = ∅ **then**
6. **break**
7. **end if**
8. *curr_sol* = *pattern_mbbs*(*L*(*i*), *W*(*i*), ∅, *curr_sol*, *root*)
9. **end for**
10. **return** timeline IDs from *curr_sol*
11. **end**

Algorithm pattern_tags
Input: a LifeStep *ls*, a wildcard *w*, a pointer to the *curr_sol* relation, the *root* of the index
Output: a subset of the *curr_sol* relation that satisfy *ls.tags*
1. **begin**
2. **if** *w* = ∅ **then**
3. *solutions* = *get_LifeSteps*(*ls.tags*, *root*)
4. **else**
5. *new_sol* = *get_LifeSteps*(*ls.tags*, *root*)
6. *solutions* = *combine*(*curr_sol*, *new_sol*, *w*, *root*)
7. **end if**
8. **return** *solutions*
9. **end**

Algorithm pattern_mbb
Input: a LifeStep *ls*, a wildcard *w*, a pointer to the *prev_sol* relation, a pointer to the *curr_sol* relation, the *root* of the index
Output: a subset of the candidate solutions that satisfy *ls.mbb*
1. **begin**
2. **if** *ls*.mbb = ∅ **then**
3. solutions= *curr_sol*
4. **else**
5. solutions= *get_LifeSteps*(*ls.mbb*, *curr_sol*, *root*)
6. **end if**
7. **if** *w* = ∅ **or** prev_sol =∅ **then**
8. **return** solutions
9. **else**
10. **return** *combine*(*prev_sol*, solutions, *w*, *root*)
11. **end if**
12. **end**

In detail, Algorithm pattern_tags examines the wildcard w and either retrieves solutions (i.e., LifeSteps) from the index satisfying the textual constraints of the input LifeStep or combines this result with current solutions found from a previous step. The combine function ensures the continuation of previous step's solutions with the currently found from the index, as we described earlier.

More specifically, given two sets of solutions and a wildcard (i.e., '>' or '*'), the *combine* function joins the two sets on *o_id*, *timeline_id* fields and then based on the wildcard, it imposes the continuation by using the *node_id*, *entry_id* and *numOfEntries* fields. A LifeStep of the second set is the continuation of a LifeStep of the first set if, in case having a '>' wildcard, both belong to the same leaf and their entries differing by one, or they belong to different leaves (neighboring leaves of the same timeline) and the first LifeStep is the last entry in its leaf, while the second LifeStep is the first entry in its leaf. In case having a '*' wildcard, both should belong to the same leaf and their entries differing by at least one or they should belong to different leaves (of the same timeline). Obviously, the resulting solutions come from the second set, so the algorithm is progressing one step further.

In this case, 'continuation' means that LifeSteps retrieved by searching the textual and the spatio-temporal space must belong to the same timeline and that the second LifeStep comes after the first with respect to time and according to the wildchar between them. This process guarantees that at the end *curr_sol* holds LifeSteps of timelines following the whole pattern (line 10).

Candidate solutions found to satisfy textual constraints help in pruning the search space in pattern_mbb function. The pattern_mbb algorithm in line 2 checks the existence of input LifeStep's mbb. If not, then solutions found by pattern_tags (i.e., *curr_sol*) are the only candidate solutions so far. Otherwise, solutions are found by retrieving the LifeSteps from the index by traversing it with respect to the spatio-temporal constraints of the *mbb* of the LifeStep. Note that when we reach a leaf this may be pruned, by checking its existence within the solutions found from pattern_tags, thus saving searching its entries. This is represented by the function get_LifeSteps (line 5). Solutions found are returned to main algorithm in line 8 as the combine for current iteration is already executed in pattern_tags (that is the reason why the main algorithm passes an empty set for parameter *prev_sol*).

5.3 Feeding semantic mobility networks

Feeding data cubes from databases is not a straightforward approach due to the dimensionality and the eventual complexity of the measures in the data cube as well as the size of the database. Therefore, there have been proposed appropriate Extract-Transform-Load (ETL) operations for this task. In the case of SMDC proposed in Sect. 4, while loading data into the dimension tables is straightforward, feeding the fact tables is not so. In this subsection, we provide three alternative approaches:

(i) The so-called *LifeStep-based* approach scans the SMD sequentially without taking advantage of the index; this decision is based on the rationale that using access methods for sequential data may not be efficient, as it was experimentally shown for the case of TDW [18].

(ii) The so-called cell-based approach does make use of the index by following a cell-oriented methodology: for each (spatial-temporal-textual) cell of the data cube, it searches for the LifeSteps that partially match the cell by applying spatial-temporal-textual queries then calculates measures over these LifeSteps.

(iii) The so-called text-based approach is based on the intuition that the semantics (i.e., the textual domain) is usually more selective than the spatio-temporal domain, so again an index-based search is followed but, this time, the index is propagated according to the textual constraints and then the result is refined according to the spatio-temporal constraints.

Algorithm LifeStepStopsLoad
Input: a SMD
Output: updated Stops_Fact table
1. **begin**
2. **for each** 'Stop' ls_k in SMD
3. periods=*get_Periods*(ls_k)
4. **for each** p_j in periods
5. stop_sems=*get_Stop_sems*(ls_k)
6. **for each** s_s_i in stop_sems
7. *CalcMeasures*(ls_k, s_s_i, p_j)
8. **end for**
9. **end for**
10. **end for**
11. **end**

Algorithm CellStopsLoad
Input: the root of the index
Output: updated Stops_Fact table
1. **begin**
2. **for each** s_s_i in Stop_Sems_Dim
3. **for each** p_j in Period_Dim
4. LifeSteps = *get_Stop_LifeSteps(*root, s_s_i, p_j)
5. **for each** ls_k in LifeSteps
6. *CalcMeasures*(ls_k, s_s_i, p_j)
7. **end for**
8. **end for**
9. **end for**
10. **end**

Algorithm TextStopsLoad
Input: the *root* of the index
Output: updated Stops_Fact table
1. **begin**
2. **for each** s_s_i in Stop_Sems_Dim
3. *curr_sol* =*pattern_tags*(*LifeStep(*s_s_i), ∅, ∅, root)
4. **for each** p_j in Period_Dim
5. LifeSteps = *get_Stop_LifeSteps(*root, s_s_i, p_j, *curr_sol*)
6. **for each** ls_k in LifeSteps
7. *CalcMeasures*(ls_k, s_s_i, p_j)
8. **end for**
9. **end for**
10. **end for**
11. **end**

In order to materialize the above approaches for feeding the Stops-Fact table in SMDC, we propose the respective algorithms, called LifeStepStopsLoad, CellStopsLoad, and TextStopsLoad, respectively.

Rather than calculating measures for each cell of the SMDC, Algorithm LifeStepStopsLoad first finds valid cells for each LifeStep of the SMD and then calculates measures for these cells only. Valid cells are those SMDC cells that spatially-temporally-textually match a LifeStep. In detail, for each LifeStep in the SMD (line 2), valid periods are found by checking the matching of the LifeStep's lifespan with the Period_Dim (line 4). Then, for each period found (line 4), stop_sems set is found by spatio-textually matching the LifeStep's MBR and tags with the Stop_Sems_Dim. For these cells, measures in Stops-Fact table are calculated (lines 6–7).

On the other hand, Algorithm CellStopsLoad and Algorithm TextStopsLoad take advantage of the SMD indexing scheme. Algorithm CellStopsLoad applies a spatio-temporal range query (line 4) that returns LifeSteps falling inside cells (i.e., query ranges), constructed by combining the Period_Dim and Stop_Sems_Dim dimensions (lines 2–4). Textual constraints are imposed at the leaf level of the index. Again, for each LifeStep, measures in Stops-Fact table are calculated (lines 5–6). In turn, Algorithm TextStopsLoad utilizes pattern_tags (i.e., inverted file) to find candidate solutions based on the textual constraints of each value in Stop_Sems_Dim (line 2). Then, these candidate solutions are passed to a spatio-temporal range query (line 5) to prune search space. The remaining steps are the same as in the CellStopsLoad algorithm.

In the same fashion, in order to materialize the above approaches for feeding the Moves-Fact table, we propose the respective algorithms, called LifeStepMovesLoad, CellMovesLoad, and TextMovesLoad, respectively.

Algorithm LifeStepMovesLoad scans the SMD for each 'Move' LifeStep and finds its tags (line 3) and time periods (line 4). Then, for each time period, it tries to find matching cells in the Stop_Sems_Dim dimension with its previous 'Stop' LifeStep (line 6). Similarly, it tries to find matching cells with its next 'Stop' LifeStep (line 8). Finally, for each returned cell, measures in Moves-Fact table are calculated (lines 9–10).

Algorithm LifeStepMovesLoad
Input: a SMD
Output: updated Moves_Fact table
1. **begin**
2. **for each** 'Move' ls_k in SMD
3. m_sem=*get_Move_sems*(ls_k)
4. periods=*get_Periods*(ls_k)
5. **for each** p_j in periods
6. from_stop_sems=*get_Stop_sems*(ls_{k-1})
7. **for each** from_s_s_f in from_stop_sems
8. to_stop_sems=*get_Stop_sems*(ls_{k+1})
9. **for each** to_s_s_t in to_stop_sems
10. *CalcMeasures*(ls_k, from_s_s_f, to_s_s_t, m_sem, p_j))
11. **end for**
12. **end for**
13. **end for**
14. **end for**
15. **end**

Algorithm CellMovesLoad
Input: the *root* of the index
Output: updated Moves_Fact table
1. **begin**
2. **for each** p_j in Period_Dim
3. **for each** from_s_s_f in Stop_Sems_Dim
4. from_ls=LifeStep('Stop', from_s_s_f, p_j)
5. **for each** to_s_s_t in Stop_Sems_Dim
6. to_ls=LifeStep('Stop', to_s_s_t, p_j)
7. **for each** m_s_m in Move_Sems_Dim
8. via_ls=LifeStep('Move', m_s_m)
9. L=[from_ls, via_ls, to_ls]; W=[∅, >, >]; Q=(L, W)
10. LifeSteps = ST^2P (Q)
11. **for each** ls_k in LifeSteps
12. *CalcMeasures*(ls_k, from_s_s_f, to_s_s_t, m_s_m, p_j))
13. **end for**
14. **end for**
15. **end for**
16. **end for**
17. **end for**
18. **end**

```
Algorithm TextMovesLoad
Input: the root of the index
Output: updated Moves_Fact table
1.  begin
2.    for each from_s_sf in Stop_Sems_Dim
3.      from_sol= pattern_tags(LifeStep(from_s_sf), ∅, ∅, root)
4.      for each m_sm in Move_Sems_Dim
5.        move_sol= pattern_tags(LifeStep(m_sm), '>',
                    from_sol, root)
6.        for each to_s_st in Stop_Sems_Dim
7.          to_sol=
              pattern_tags(LifeStep(to_s_st), '>', move_sol, root)
8.          candidate_moves=get_Moves(move_sol, to_sol)
9.          for each pj in Period_Dim
10.           from_sol=
                pattern_mbb(LifeStep(from_s_sf, pj), ∅, ∅,
                  from_sol, root)
11.           to_sol=
                pattern_mbb(LifeStep(to_s_st, pj), '>',
                  candidate_moves, to_sol, root)
12.           result_moves=get_Previous_Moves(to_sol)
13.           for each mg in result_moves
14.             CalcMeasures(mg, from_s_sf, to_s_st, m_sm, pj))
15.           end for
16.         end for
17.       end for
18.     end for
19.   end for
20. end
```

In turn, Algorithm CellMovesLoad in its core calls an ST^{2}P query, restricting the result to the 'Move' LifeSteps that follow query pattern Q := [from_ls > via_ls > to_ls] consisting of three LifeSteps (lines 9–10). These LifeSteps are constructed by all triplet combinations of values from the dimensions, which textually match 'Move' LifeSteps created form the Move_Sems_Dim (lines 7–8) and also match spatially-temporally-textually their previous and next 'Stop' LifeSteps, created from the Stop_Sems_Dim and Period_Dim dimensions (lines 2–6). Thus, Q is a Stop-Move-Stop- like pattern. For each 'Move' LifeStep found, measures in Moves-Fact table are calculated (line 11–12).

Finally, Algorithm TextMovesLoad tries to find in a step-by-step fashion 'Move' LifeSteps that conform to values of the Move_Sems_Dim (note that this dimension includes only textual values), also having the respective previous and next 'Stop' LifeSteps textually conforming to values in Stop_Sems_Dim (namely, we ignore the spatial part of this spatio-textual dimension). Thus, the algorithm first uses the inverted file to deal with textual constraints (lines 2–8).

Note the successive application of pattern_tags algorithm, where at each step we combine with the results found before the step to prune the search space. Then, the algorithm proceeds to apply spatio-temporal constraints to the (already found) previous and next 'Stop' LifeSteps, with respect to the candidate 'Move' LifeSteps found by the textual filtering (lines 9–12). For each qualifying 'Move' LifeStep measures in Moves-Fact table are calculated (lines 13–14).

6 Evaluation

We have developed the proposed $Hermes^{sem}$ framework extending Hermes MOD engine [12]. In this section, we present the results of an evaluation methodology that we have applied to $Hermes^{sem}$. $Hermes^{sem}$ can be evaluated by following two directions; a quantitative approach and a more qualitative one. More specifically, in Sect. 6.1, we study the performance of the framework in terms of processing time. (Note that $Hermes^{sem}$ main purpose is to efficiently store and manage semantic trajectories and as such its functionality is deterministic; thus, measuring accuracy is not applicable.) Then, in Sect. 6.2, we present real SQL queries corresponding to a real-world case study, so that the reader can have a solid understanding from these experiments of the analytical power and usefulness of the framework.

All experiments were performed in a PC of i7 CPU with 4 cores at 1.73 GHz and of 8Gb RAM. Input SMDs were simulations of mobility scenarios by the Hermoupolis semantic trajectory generator [15]. We simulated two mobility scenarios: The first scenario is a 7-days movement in the city of Athens for four different profiles (movement behaviors); the second scenario is a 1-day movement in the city of Athens, again for four different profiles. For both scenarios we created 4 SMDs consisting of 50, 100, 150, and 200 timelines. The average number of LifeSteps for each scenario is 58 and 10, respectively. We used the first scenario for ST^{2}P query experiments (i.e., in order to evaluate the performance of the algorithm proposed in Sect. 5.2) and the second for SMDC feeding experiments (i.e., in order to evaluate the performance of the algorithms proposed in Sect. 5.3).

6.1 Performance study

We start our performance study by calculating the time required to build the two alternative indexing structures. Figure 6 illustrates the results for either one-by-one insertion or bulk loading of the SMD content into the index. The significant gain when performing bulk loading on Sem3DR-tree is due to the fact that the loading task can be parallelized by partitioning the SMD timelines. (In our implementation that is found behind the numbers in Fig. 6, we used 4 parallel processes.) On the other hand, we provide a single series for Sem3DR-tree since it exploits on the DBMS built-in R-tree index. To measure processing time on ST^{2}P queries, we set up queries of 2, 4, 6, 8, and 10 input LifeSteps with various wildcards in between them. Figure 7 depicts the performance of our algorithm in all datasets using both types of indices, i.e., Sem3DR-tree and SemTB-tree. We note that increased query complexity does not increase significantly processing time, while in all cases the Sem3DR-tree performs better than SemTB-tree. This is rather expected, as the ST^{2}P query can be considered as a sequence of coordinate-based (i.e., range)

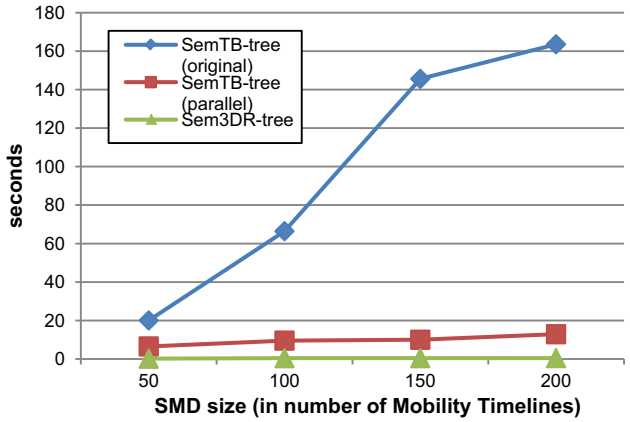

Fig. 6 Sem-TB-tree versus Sem-3DR-tree construction time

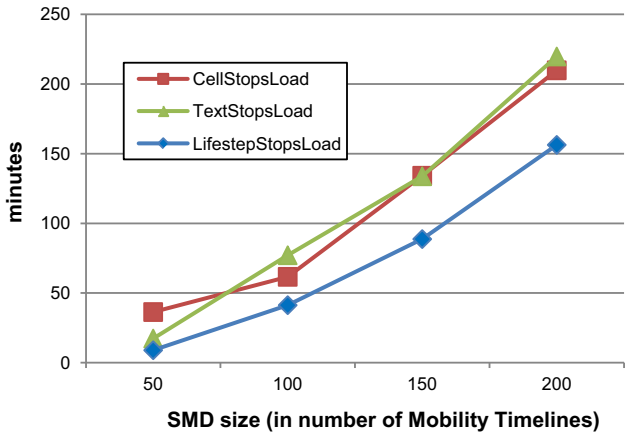

Fig. 8 SMDC feeding processing time (Stops-Fact table)

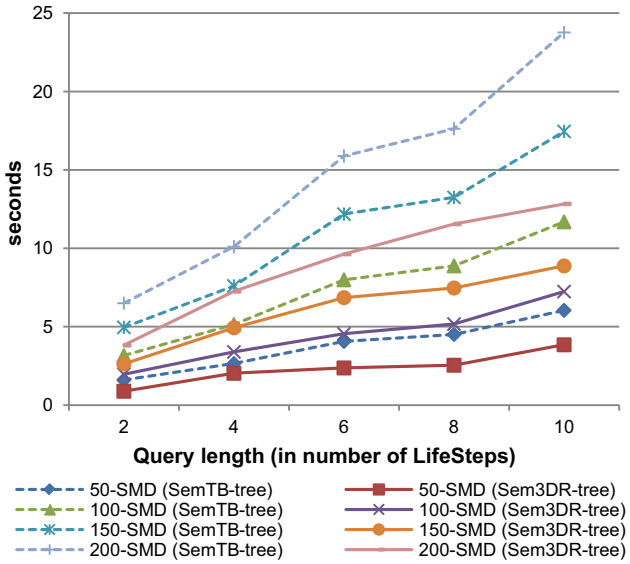

Fig. 7 ST^{2}P query processing time

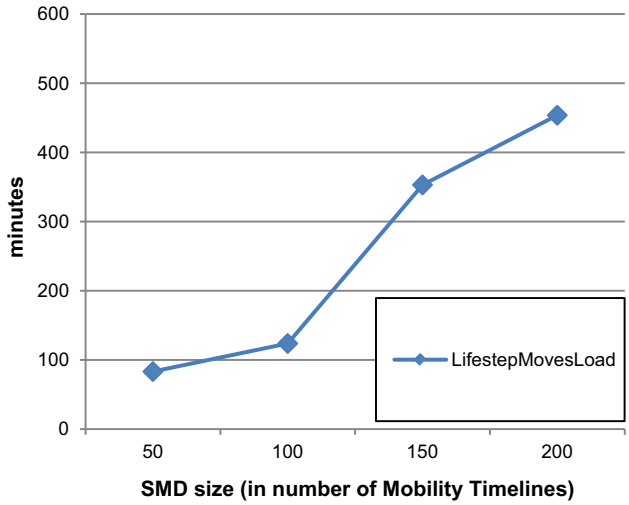

Fig. 9 SMDC feeding processing time (Moves-Fact table)

queries, which have been shown in the literature to be supported more efficiently using 3DR-trees rather than TB-trees [17].

Next, we provide the results for feeding the SMDC Stops-Fact table and Moves-Fact table, in Figs. 8 and 9, respectively. Regarding Stops (Fig. 8), the three approaches do not appear to have significantly different performance (linear with respect to the number of timelines). On the other hand, Moves (Fig. 9) are much more expensive to process; in this case, the LifeStep-based approach appears to be the only one that performs in acceptable time (therefore, the other two are omitted from the chart).

6.2 Hermes^{sem} in action

In this section, we present a qualitative evaluation of the $Hermes^{sem}$ framework providing several queries using the extended SQL query language in one of the application areas that we highlighted in the introduction. More specifically, we provide a list of SQL queries (presented in Table 2) that consists of a set of representative Q2-, Q3-, and Q4-type queries applied to the $Hermes^{sem}$ framework that a LBSN analyst could apply to gain insight in such datasets.

In detail, Query Q2.1 returns the number of STOP LifeSteps, number of MOVE LifeSteps and the total number of LifeSteps of all timelines in the SMD. Q2.2 first selects all STOP LifeSteps of all timelines and then groups them by their activity tag while measuring their duration per activity. Q2.3 applies the 'confine_in' method on a specific timeline and restricts it inside given spatio-temporal window also keeping only LifeSteps according to given tags.

Q3.1 is a crossover query in that it uses the semantic along with the raw mobility layer. It first selects LifeSteps that obey to given spatio-temporal-textual constraint in a filter-like step using the index and then it retrieves through the link the corresponding raw sub-trajectory part in which a temporal

Table 2 Benchmarking the $Hermes^{sem}$ framework

Q2-type queries	
Q2.1	**Select** o_id, t.num_of_stops(), t.num_of_moves, (t.num_of_stops() + t.num_of_moves()) as num_of_episodes **from** SMD t **order by** 4 **desc**;
Q2.2	**Select** activity_tag, sum(s.duration()) duration **from** table (**select** t.episodes_with('STOP') **from** SMD t) s **group by** activity_tag;
Q2.3	**Select** t.confined_in(sdo_geometry(2003,2100,null, sdo_elem_info_array(1,1003,3), sdo_ordinate_array(469115,4200000, 472377,4206200)), timeperiod(timestamp(2013,5,8,8,0,0), timestamp(2013,5,8,23,59,0)), 'stop+working').episodes **from** SMD t **where** t.o_id = 3125 **and** t.semtraj_id = 3125;
Q3-type queries	
Q3.1	**Select** o_id, deref(tlink).sub_mpoint.at_period(timeperiod(timestamp(2013, 5, 8, 7, 50, 00), timestamp(2013, 5, 8, 14, 10, 00))) restricted **from** std.range_episodes(sem_episode('MOVE', 'CAR', null, sem_mbb(sem_st_point(470000, 4200000,timestamp(2013, 5, 8, 7, 50, 0)), sem_st_point(474000, 4205000, timestamp(2013,5,8,14,10, 0))), tlink),SemTB-tree);
Q3.2	**Select** activity_tag, count(o_id) **from** std. from_to_via(sem_episode('STOP', 'HOME', null, null, null), sem_episode('STOP', null, null, null, null), sem_episode('MOVE', null, null, sem_mbb(sem_st_point(470000, 4200000, timestamp(2013,5,8,7,50,0)), sem_st_point(474000, 4205000, timestamp(2013,5,8,9,50,0))), null), SemTB-tree) **group by** activity_tag;
Q3.3	**Select** episode **from** SMD **where** SDO_ANYINTERACT(t.geom, sdo_geometry(3008, 2100, null, sdo_elem_info_array(1, 1007, 3), sdo_ordinate_array(450000, 4210000, timestamp(2013,11,10,7,0,0).toSpatial, 480000, 4230000, timestamp(2013,11,10,17,0,0).toSpatial))) = 'TRUE' **and** defining_tag = 'STOP' **and** episode_tag = 'UNIVERSITY' **and** activity_tag = 'STUDYING';
Q3.4	**Select** std.range_episodes(sem_episode('STOP', 'UNIVERSITY', 'STUDYING', sem_mbb(sdo_geometry(2003,2100,null,sdo_elem_info_array(1,1003,3), sdo_ordinate_array(450000, 4210000,480000,4230000)), timeperiod(timestamp(2013,11,10,7,23,18), timestamp(2013,11,10,9,24,12))), null), SemTB-tree);
Q4-type queries	
Q4.1	**Select** std.patterns(sem_episode_tab(sem_episode('STOP', 'HOME', 'RELAXING', sem_mbb(sem_st_point(473600, 4200700,timestamp(2013,11,10,3,0,0)), sem_st_point(473700,4200800, timestamp(2013,11,10,6,0,0))), null), sem_episode('STOP', 'GYM', 'SPORTING',null, null)), varchar_ntab(null, '*'), 'SMD');
Q4.2	**Select** tab1.TL1.sim_trajectories(tab2.TL2) **from** (**select** value(b) TL1 **from** SMD b **where** b.o_id=5238) tab1, (**select** value(b) TL2 **from** SMD b **where** b.o_id=9021) tab2;
Q4.3	**declare** eps = 0.05; minPts = 15; λ =0.5;, wd=[1/3, 1/3, 1/3] **begin** SemT-OPTICS(SMD, eps, minPts, $DM_T(\lambda, w_d)$); **end;**

restriction is applied. Q3.2 finds MOVE LifeSteps that are alive inside a spatio-temporal window and start from a STOP-HOME LifeStep and ends up in another STOP LifeStep. It uses a special structure build inside the index, which allows combining results from descending the tree and moving along its leaves. Finally, it computes the distribution of objects per transportation mode. Q3.3 returns all LifeSteps of the SMD that match (at all spatial, temporal and textual dimensions) a given LifeStep, namely a Stop at a university for studying, located in MBR (450,000, 4,210,000, 480,000, 4,230,000),

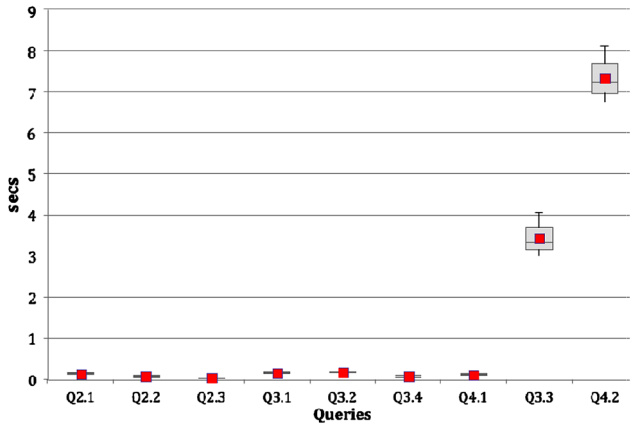

Fig. 10 Queries' execution times

between 7a.m. and 5p.m. on Nov. 10th, 2013. This query exploits on the built-in R-tree (hence, it uses Sem3DR-tree). The same result, this time using the SemTB-tree index, is achieved by delivering Q3.4.

Q4.1 implements the ST^{2}P query; in particular, it searches for timelines in SMD including LifeSteps that follow a given pattern (starting from home/ relaxing and ending at gym/sporting, where the starting LifeStep is spatially and temporally constrained within an MBR and period, respectively). A very basic function for mining operations is the implementation of a similarity function on timelines and LifeSteps objects. As described in Table 1, the similarity between timelines or between LifeSteps objects consider all spatio-temporal-textual domains [15]; Q4.2 is an implementation of the similarity function on timelines. The immediate usage of the above similarity function is in clustering. In $Hermes^{sem}$ framework SemT-OPTICS [15] algorithm uses this similarity function to cluster timelines; Q4.3 clusters the timelines of the given SMD.

Concluding, in Fig. 10, we present the boxplots of the average execution times and the corresponding variability of the above queries. In detail, we execute each query ten times, every time selecting randomly 200 timelines from the 1-day SMD consisting of 1000 timelines. This experiment provides an overview of the performance of the proposed query language and it clearly shows that the user of can simply use the extended SQL of $Hermes^{sem}$ with the same or better performance than the legacy SQL exhibits. Moreover, it turns out that SemTB-tree takes advantage of its textual part and thus filters out candidate results faster than the built-in R-tree index.

7 Summary and future challenges

Motivated by related work on semantic trajectories [9] and an envisioned framework for their realization [13], in this paper we presented the $Hermes^{sem}$ framework, an integrated MOD / SMD / SMDC engine. In this line, we proposed efficient access methods for the hybrid indexing of the spatial-temporal-textual component of this type of data.

We also devised efficient query processing algorithms to support a very interesting query type, called spatial-temporal-textual Pattern (ST^{2}P) query that receives as input a sequence of LifeSteps (to be considered as a simplified regular expression) and outputs the timelines that obey to the constraints of this sequence. To the best of our knowledge, it is the first time that such pattern queries are discussed in the spatial-temporal-textual domain. Moreover, we presented a data constellation schema for modeling SMDC and devised algorithms for the efficient feeding of its fact tables.

We should note that the proposed development approach is only a first SMD and SMDC implementation and it could only be used as a baseline in future works. For instance, the storage layer could be physically organized according to any data management paradigm (centralized, distributed, map-reduce, etc.), the hybrid indexing of the spatial-temporal-textual information as well as the ST^{2}P query processing and optimization challenges could find more attractive solutions, and so on. Our goal in this work was to design a unified framework that solves many (though not all) raised issues better than legacy approaches can. Thus, we believe that each of the tackled issues could be a research challenge per se. This makes a research plan for the near future.

Finally, recent advances in data management and analysis aim toward effective and efficient storage and application of innovative analytics techniques that would make data scientists able to extract valuable information from huge and complex datasets. Already much of this kind of data reside in cloud-based infrastructure and this trend will grow even more in near future. Researchers can adopt the Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) models to implement their solutions in the cloud. Although we do not deal with cloud-based solutions in this paper, it is evident that a cloud-based approach would be the next step of our approach.

Acknowledgements This work was partially supported by EU Horizon 2020 project datACRON (Grant Agreement No. 687591), and EU Horizon 2020 project DART (Grant Agreement No. 699299).

References

1. Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F., Renso, C., Rinzivillo, S., Trasarti, R.: Unveiling the complexity of human mobility by querying and mining massive trajectory data. VLDB J. **20**(5), 695–719 (2011)
2. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data cube: a relational aggregation operator generalizing groub-by, cross-tab and sub-totals. Data Min. Knowl. Discov. **1**(1), 29–54 (1997)

3. Güting, R.H., Behr, T., Düntgen, C.: SECONDO: a platform for moving objects database research and for publishing and integrating research implementation. IEEE Data Eng. Bull. **33**(2), 56–63 (2010)
4. Han, J., Stefanovic, N., Koperski, K.: Selective materialization: an efficient method for spatial data cube construction. In: Proceedings of PAKDD (1998)
5. H2020 datACRON project. URL: http://datacron-project.eu
6. H2020 DART project. URL: http://dart-sesar.eu
7. Leonardi, L., Orlando, S., Raffaetà, A., Roncato, A., Silvestri, C., Andrienko, G., Andrienko, N.: A general framework for trajectory data warehousing and visual OLAP. GeoInformatica **18**(2), 273–312 (2014)
8. Orlando, S., Orsini, R., Raffaetà, A., Roncato, A., Silvestri, C.: Spatio-temporal aggregations in trajectory data warehouses. In: Proceedings of DaWaK (2007)
9. Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M.L., Gkoulalas-Divanis, A., Macedo, J.A., Pelekis, N., Theodoridis, Y., Yan, Z.: Semantic trajectories modeling and analysis. ACM Comput. Surv. **45**(4), article no. 42 (2013)
10. Patroumpas, K., Alevizos, E., Artikis, A., Vodas, M., Pelekis, N., Theodoridis, Y.: Online event recognition from moving vessel trajectories. GeoInformatica (2016, online first)
11. Patroumpas, K., Artikis, A., Katzouris, N., Vodas, M., Theodoridis, Y., Pelekis, N.: Event recognition for maritime surveilance. In: Proceedings of EDBT (2015)
12. Pelekis, N., Frentzos, E., Giatrakos, N., Theodoridis, Y.: HERMES: a trajectory DB engine for mobility-centric applications. Int. J. Knowl. Based Organ. **5**(2), 19–41 (2015)
13. Pelekis, N., Theodoridis, Y., Janssens, D.: On the management and analysis of our LifeSteps. SIGKDD Explor. **15**(1), 23–32 (2013)
14. Pelekis, N., Sideridis, S., Theodoridis, Y.: Hermes^{sem}: a semantic-aware framework for the management and analysis of our LifeSteps. In: Proceedings of DSAA (2015)
15. Pelekis, N., Sideridis, S., Tampakis, P., Theodoridis, Y.: Simulating our LifeSteps by example. ACM Trans. Spat. Algorithms Syst. **2**(3), article no. 11 (2016)
16. Pelekis, N., Theodoridis, Y.: Mobility Data Management and Exploration. Springer, New York (2014)
17. Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel approaches to the indexing of moving object trajectories. In: Proceedings of VLDB (2000)
18. Raffaetà, A., Leonardi, L., Marketos, G., Andrienko, G., Andrienko, N., Frentzos, E., Giatrakos, N., Orlando, S., Pelekis, N., Roncato, A., Silvestri, C.: Visual mobility analysis using T-warehouse. Int. J. Data Warehous. Min. **7**(1), 1–23 (2011)
19. Theodoridis, Y., Vazirgiannis, M., Sellis, T.: Spatio-temporal indexing for large multimedia applications. In: Proceedings of ICMCS (1996)
20. Wagner, R., Raffaetà, A., Roncato, A., de Macedo, J.A., Trasarti, R., Renso, C.: Mob-warehouse: a semantic approach for mobility analysis with a trajectory data warehouse. In: Proceedings of SECOGIS (2013)
21. Wu, D., Cong, G., Jensen, C.S.: A framework for efficient spatial web object retrieval. VLDB J. **21**(6), 797–822 (2012)
22. Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., Aberer, K.: SeMiTri: a framework for semantic annotation of heterogeneous trajectories. In: Proceedings of EDBT (2011)
23. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and OLAP multidimensional networks. In: Proceedings of SIGMOD (2011)