A Method to Detect and Classify Inconsistencies of Moving Objects' Stops with Requested and Reported Tasks

Felipe Pinto da Silva, Renato Fileto

Department of Informatics and Statistics (INE/CTC/UFSC), Federal University of Santa Catarina, Brazil
felipe.pinto@posgrad.ufsc.br, r.fileto@ufsc.br

Abstract. Moving object trajectories automatically collected by using sensors enable tracking individual or teams doing activities in the geographic space. However, methods for analyzing trajectories rarely compare them to task execution plans and/or travel diaries. This article proposes a method to detect and classify inconsistencies of trajectories with both planned tasks and reported tasks, by checking their spatial and temporal incompatibilities. The classified inconsistencies returned help to investigate possible problems on tasks planning or reporting, and even misbehaviors of moving objects. This method has been implemented in a prototype on top of PostGIS, and helped to detect and investigate a variety of inconsistencies in real data provided by a water supply company.

Categories and Subject Descriptors: H.3 [**Information Storage and Retrieval**]: Miscellaneous; H.4 [**Information Systems Applications**]: Miscellaneous

Keywords: Behavior analysis, Moving objects trajectories, Spatiotemporal data

1. INTRODUCTION

Devices such as cell phones equipped with geographic positioning technologies (e.g., GPS, GSM) allow the automatic gathering of raw trajectories, i.e., temporally ordered sequences of spatiotemporal positions of moving objects. These trajectories can be collected without any impact for the moving objects, and they usually carry more precise and detailed spatiotemporal information than tasks schedules or travel diaries. Thus, trajectories are suitable inputs for methods that investigate moving object behaviors while these objects (are supposed to) perform tasks in the geographical space. Such methods can have many applications, ranging from the management of itinerant teams that provide services *in situ* (e.g., public works, maintenance in place, domiciliary inspection, in-home caregiving) to delivery services (e.g., conventional mail, pizza delivery). However, the analysis of trajectories along with data from other sources, particularly tasks planning and travel diaries, has not been sufficiently exploited in the related literature [Raj et al. 2008; Clark and Doherty 2008; Huang et al. 2010; Furletti et al. 2013; Chen et al. 2012; Yuan et al. 2013] to allow the extraction of further relevant information about the tasks (e.g., type, goal), and situations that may influence tasks execution performance (e.g., stops not related to planned tasks, inaccurate information in travel diaries).

Given this scenario, we propose a computational method to automatically detect and classify spatiotemporal inconsistencies of moving objects trajectories with tasks that have been planned or reported to have been performed in specific spatiotemporal points. The inputs of this method are: (i) raw trajectories of moving objects; (ii) the street address for executing and the expected duration of each planned task; and (iii) reports made by the moving objects themselves, with (possibly inaccurate) indications of the ending time, duration, and geographic coordinates of each executed task. The

Work supported by CNPq (grant 478634/2011-0), and FEESC. Thanks to the colleagues Andre S. Furtado, Juarez A. P. Sacenti, and Ricardo G. B. Nabo for their valuable help on spatio-temporal data management techniques and tools.

method confronts the data from these three sources, and classifies each detected inconsistency. The returned spatiotemporal inconsistencies among the data of distinct nature and sources support the investigation of operational problems, and even possible misbehaviors of moving objects. The proposed method has been implemented in a prototype that uses state-of-the-art methods for managing spatiotemporal data in PostGIS.

This article extends [da Silva and Fileto 2014] in the following directions: (i) inclusion of a new preprocessing step to remove inaccurate planned locations and planned tasks without reports, which were causing a high number of detected inconsistencies, many of them not relevant in practice; (ii) several fixes and improvements in the method kernel implementation; (iii) inclusion of a new post-processing step that links detected idle stops to Places of Interest (PoIs), which can be taken from geographic databases or Linked Open Data (LOD) collections, to help explain moving object behaviors; (iv) further experiments run with better tuned parameters, greater amounts of real data about the activities of maintenance teams of a water supply company, and PoIs from Wikimapia that were linked to the detected idle stops, generating new results to assess the impact of the advancements made in the proposed method and its implementation; (v) several corrections and improvements in the text, along with amended or completely new figures and tables. The new experimental results show more unequivocally than the previous ones that the proposed method can detect and help to investigate a wide variety of inconsistencies. The new preprocessing have proved helpful to filter input data, and avoid oversensitiveness of the inconsistencies detection method. The improved method generates a much lower number of detected inconsistencies, and a higher percentage of them have been considered relevant by the users. In addition, the post-processing step introduced in this version of our method allowed a better appreciation of the reasons for the idle stops by analyzing the types of PoIs in their surroundings, and other properties of these PoIs.

The rest of this article is structured as follows. Section 2 provides the foundations necessary to understand the proposal. Section 3 describes the proposed method in detail. Section 4 reports experimental results. Section 5 reports some related work, and Section 6 concludes the article, by enumerating contributions and themes for future work.

2. FOUNDATIONS

The inputs of the proposed method (namely trajectories, planned tasks, and reported tasks) are formally described in the following. These concepts are crucial for understanding the proposed method.

2.1 Trajectories

Definition 1 formalizes the concept of raw trajectory as sequential spatiotemporal positions (points) occupied by a moving object. Such positions can be collected by using some sensor (e.g., GPS, GSM).

DEFINITION 1. *A **raw trajectory** $\tau = (\mathbf{idMO}, \mathbf{idTraj}, (\mathbf{x_1}, \mathbf{y_1}, \mathbf{t_1}), ..., (\mathbf{x_n}, \mathbf{y_n}, \mathbf{t_n}))$ is a temporally ordered sequence of spatiotemporal positions of a moving object where* **idMO** *is the identifier of that object;* **idTraj** *is the unique identifier of the trajectory τ; each $(\mathbf{x_i}, \mathbf{y_i}, \mathbf{t_i})$ (with $1 \leq i \leq n$) is a spatiotemporal position that indicates that the object was at spatial coordinates (x_i, y_i) in the instant t_i; and $\mathbf{n > 0}$ is the number of spatiotemporal positions of τ.*

Raw trajectories must be preprocessed to eliminate inaccuracies caused by limitations of the sensing devices and problems of the trajectory gathering process [Yan et al. 2013]. Then, their filtered and possibly adjusted spatiotemporal positions can be structured in episodes [Mountain and Raper 2001], which are formally described by Definition 2.

DEFINITION 2. *Given a trajectory $\tau = (idMO, idTraj, (x_1, y_1, t_1), ..., (x_n, y_n, t_n))$, an **episode** is a maximal sub-sequence $\mathbf{E} = (\mathbf{idTraj}, \mathbf{idE}, (\mathbf{x_{start}}, \mathbf{y_{start}}, \mathbf{t_{start}}) ... (\mathbf{x_{end}}, \mathbf{y_{end}}, \mathbf{t_{end}}))$ of τ, with $1 \leq$*

start $\leq$ *end* $\leq n$, *that satisfies a predicate, and such that:* $\mathbf{idTraj}$ *is the trajectory identifier;* $\mathbf{idE}$ *is the episode identifier;* $\mathbf{t_{start}}$ *is the instant of the first spatiotemporal point of the episode; and* $\mathbf{t_{end}}$ *is the instant of the last spatiotemporal point of the episode.*

Episodes can be moves or stops [Spaccapietra et al. 2008], for example. The positions constituting an episode such as a stop can be determined by distinct predicates (e.g., being inside a place, having speed below a given threshold). Definition 3 formalizes structured trajectories as sequences of episodes.

DEFINITION 3. *A* ***structured trajectory*** $\mathbf{T_s} = \mathbf{E_1}, \mathbf{E_2}, ...\mathbf{E_m}$ *is a temporally ordered sequence of temporally disjoint episodes with* $m > 0$.

Comprehensive reviews about concepts and techniques related with trajectory data processing (e.g., detecting relevant episodes) can be found in [Parent et al. 2013] and [Pelekis and Theodoridis 2014]. A method for building structured trajectories by extracting stops on a given set of places is proposed by Alvares et al. [2007]. This method is called IB-SMoT (Intersection-Based Stops and Moves of Trajectories). CB-SMoT (Cluster-Based SMoT) is an alternative method for structuring trajectories based on speed [Palma et al. 2008]. It is able to identify stops by clustering adjacent positions in which the moving object is stationary or moves slowly.

In this article, we consider that a stop is necessary for a moving object to perform a task. This assumption is also used by Huang et al. [2010], Furletti et al. [2013] and Clark and Doherty [2008]. We use CB-SMoT to detect stops anywhere, even locations where there is no reported or planned task, and detect inconsistencies such as idle stops.

2.2 Tasks

According to Huang et al. [2010], a task has a location, an initial instant, a duration, and one or more goals. However, the proposed method does not impose a rigid schedule, i.e., it does not matter when a task should start, but its minimum and maximum expected duration do.

DEFINITION 4. *A* ***planned task*** *is a tuple* $\varphi = (\mathbf{idMO}, \mathbf{idTask}, (\mathbf{x}, \mathbf{y}), \mathbf{minExpDuration}, \mathbf{maxExpDuration}, \mathbf{requestDate}, \mathbf{maxCompletionDate})$ *where* $\mathbf{idMO}$ *is the moving object identifier;* $\mathbf{idTask}$ *is the planned task identifier;* $(\mathbf{x}, \mathbf{y})$ *are the spatial coordinates where the planned task should be performed;* $\mathbf{minExpDuration} \leq \mathbf{maxExpDuration}$ *are respectively the minimum and the maximum expected duration of the planned task;* $\mathbf{requestDate} \leq \mathbf{maxCompletionDate}$ *are respectively the request date and the maximum allowed completion date of the planned task execution.*

Clark and Doherty [2008] employed users' feedback to confirm the instants when they start and end each task execution. Our method, on the other hand, employs reports made by the moving objects themselves using mobile devices. Reported tasks are defined next.

DEFINITION 5. *A* ***reported task*** *is a tuple* $\rho = (\mathbf{idMO}, \mathbf{idTask}, \mathbf{startTime}, \mathbf{endTime}, (\mathbf{x}, \mathbf{y}))$ *where* $\mathbf{idMO}$ *is the moving object identifier;* $\mathbf{idTask}$ *is the reported task identifier;* $\mathbf{startTime}$ *is the starting time of the task execution;* $\mathbf{endTime}$ *is the end time of the task execution;* $(\mathbf{x}, \mathbf{y})$ *are the spatial coordinates from which the report was sent, and supposedly the location of the task execution.*

3. ANALYSIS OF SPATIOTEMPORAL INCONSISTENCIES

This section describes the proposed method to detect and classify spatiotemporal inconsistencies of trajectories with planned and reported tasks. Subsection 3.1 presents an overview of the current method, which extends the one introduced in [da Silva and Fileto 2014] with a new preprocessing task (step 2) for cleansing data, and a post-processing task (task 5) to link detected idle stops to PoIs in

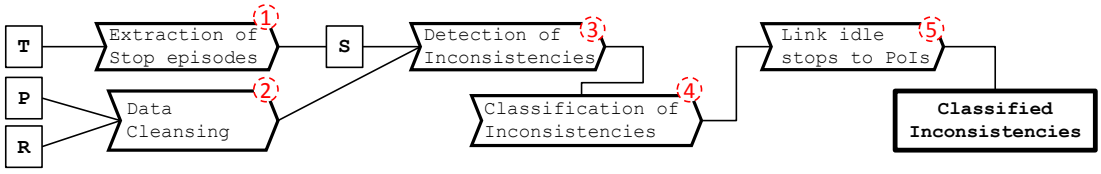

Fig. 1. Data processing steps of the proposed method

their vicinity. Subsections 3.2 and 3.3 detail the steps for inconsistency detection and inconsistency classification, respectively.

3.1 Overview

Figure 1 summarizes the proposed method. Its inputs are T, P, and R. T is a set of raw trajectories (Definition 1). P is a set of planned tasks (Definition 4). R is a set of reported tasks (Definition 5). All these datasets refer to the same set of moving objects during a certain time period (e.g., a month).

Step 1 extracts stop episodes (Definition 2) from each trajectory in T by using CB-SMoT [Palma et al. 2008]. The extracted stops from all the trajectories are inserted in the S that is used as one of the inputs for step 3. For simplicity and efficiency of the next step, the location of each stop is approximated by a single point, such as its centroid. Step 2 removes planned tasks with inaccurate geo-locations and reports that do not refer to an id of planned task scheduled for the period of time containing the report time. This cleansing prevents bad quality data causing an explosion in the number of inconsistencies detected. Then, step 3 uses spatiotemporal SQL queries to detect inconsistencies of its inputs, and step 4 classifies the inconsistencies found to help explain them. The outputs of the step 4 are classified spatiotemporal inconsistencies. Finally, the newly proposed step 5 enriches the idle stops found in step 4 with PoIs taken from existing geographic databases (e.g. Wikimapia[1]) and LOD collections (e.g., DBpedia[2]), based on their proximity to the idle stops. It helps investigate users behaviors during these idle stops, by examining the semantics (e.g., types and other properties) of the surrounding PoIs. Steps 3 and 4 are described in further detail in subsections 3.2 and 3.3, respectively.

3.2 Detection of Inconsistencies

The detection of inconsistencies is done by exploiting the results of the query presented in Figure 2, which defines a database view. It applies successive outer joins to the tables: planned tasks (P), reported tasks (R), and stop episodes of the structured trajectories (S). P, R, and S are as stated in Definitions 4, 5, and 2, respectively. The stop coordinates $(S.x_c, S.y_c)$ refer to the centroid of each stop episode. These coordinates are used instead of the whole subsequence of spatiotemporal points of each stop to represent it, for simplicity and efficiency. Different kinds of inconsistencies are detected in the final result (Res) by distinguishing the attribute values (and $null$ values) resulting from each outer join. Notice that R and S take part in more than one join in the query, with distinct join conditions. Thus, we use R' and S', which are just the projections of the attributes of R and S, respectively, necessary for the respective join conditions. In other words:

$$R' = \Pi_{idMO,startTime,endTime,x,y}(R) \qquad S' = \Pi_{idMO,startTime,endTime,x_c,y_c}(S)$$

The first operation of the query presented in Figure 2 (a) is a full outer natural join between P and R on the composite key attributes $idMO$ and $idTask$. It enables the identification of planned tasks

[1]http://wikimapia.org
[2]http://wiki.dbpedia.org

that do not match any reported task (i.e., planned tasks whose execution was not reported), and vice versa (i.e., reported tasks that do not correspond to any planned task).

The other outer joins in the query of Figure 2 (b, c, and d) have as join conditions the conjunction of the proximity of the respective spatial data (e.g., $distance(P, R') \leq SThreshold\_PR$). The spatial thresholds $SThreshold\_PR$, $SThreshold\_PT$ and $SThreshold\_RT$ can be adjusted according to the dataset properties and application goals. In other words, two spatial points are considered close to each other if the distance between them is smaller than the respective spatial threshold.

The natural and spatial join conditions of the query presented in Figure 2 avoid the combinatorial explosion of Cartesian product, and along with the conjunctive temporal conditions of the query, enable the detection of a variety of inconsistencies. The proposed method detects inconsistencies of distinct classes by posing other queries on the view *Res* (results of Figure 2), that vary according to the class of the inconsistencies being detected. These queries select tuples of *Res* according to conjunctive conditions that exploit the (absence of, i.e. *null*) values of some attributes of P, R, R', S, and S'. For example, if a task report has been done but its position is too far away from its corresponding planned task position, then $P.idTask$ and $R.idTask$ are not *null* and have the same value in *Res*. However, $R'.idReport$ may be *null*, as R' has been joined with P by proximity. Temporal inconsistencies, on the other hand, are detected by checking the temporal attributes in *Res*. For example, if a moving object reports a task lasting longer than its maximum expected duration, then:

$$(R.endTime - R.startTime) > P.maxExpDuration$$

Figure 3 (a) shows a query on *Res* that selects each planned tasks that matches a reported task and a stop, but whose matching reported task duration is shorter than the minimum expected time to perform the planned task. Analogous queries can extract other kinds of inconsistencies by changing the restrictions on the attributes of *Res* used to select tuples. Figure 3 (b) presents a query to retrieve each planned task whose matching reported location (by natural join) is far from the planned location, that is not matched to a stop, and whose reported duration exceeds the maximum expected duration of the planned task.

Figure 4 illustrates an inconsistency retrieved by the query presented in Figure 3 (b). Let $\varphi \in P$ be a planned task, $\varphi.minExpDuration = 20$, and $\varphi.maxExpDuration = 30$, both in minutes. This task should be performed within the blue circle around φ. The red circles represent two stops ($s_1, s_2 \in S$) of the moving object assigned to execute φ, with 20 and 30 minutes. The green star represent the location of report ρ, where $\rho \in R$ and $\rho.idTask = \varphi.idTask$. The report was done away from φ,

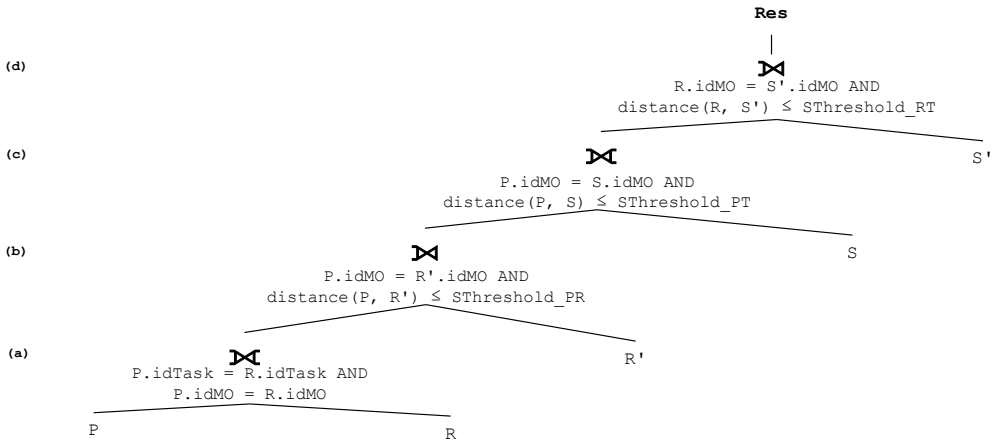

Fig. 2. General query (view) used for inconsistencies detection

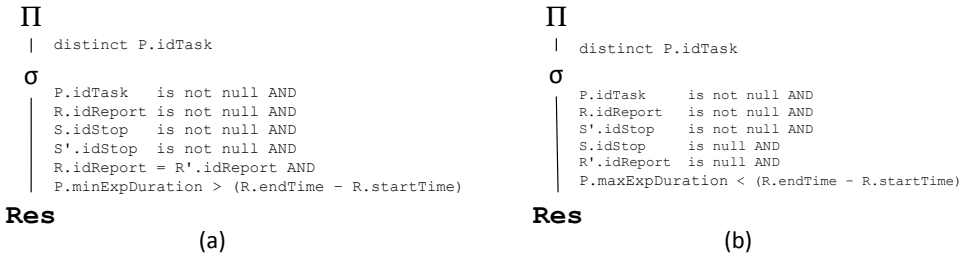

Fig. 3. Retrieving inconsistencies involving planned task duration and reported task duration

but near to s_2. Then the stop s_2 is associated with the report ρ (and indirectly associated with the planned task φ). In addition, there is a spatial contradiction between φ and ρ, and a duration contradiction between ρ and s_2.

3.3 Classification of Inconsistencies

The inconsistencies detected by the previous task are classified based on conjunctions of conditions that can hold or not on tuples of the view *Res*. These conditions can be disposed in a truth table for systematic assessment. The first five conditions are expressed by possible *null* values on the attributes of planned tasks, reported tasks or trajectory episodes (φ, ρ, ρ', s, s'), which indicate lack of matching components for natural or similarity joins of the relational algebra expression that defines *Res* (Figure 2). The following five additional conditions can also be considered in the truth table:

$$c_1 : \rho.duration \leq \varphi.maxExpDuration \qquad c_2 : \rho.duration \geq \varphi.minExpDuration$$
$$c_3 : s.duration \leq \varphi.maxExpDuration \qquad c_4 : s.duration \geq \varphi.minExpDuration$$
$$c_5 : P.idTask = R'.idTask$$

in which the duration of a reported task ρ or a stop s can be determined as follows:

$$\rho.duration = (\rho.endTime - \rho.startTime) \qquad s.duration = (s.endTime - s.startTime)$$

Thus, the truth table used to classify inconsistencies has 2^{10} combinations of Boolean values, which indicate if the attributes of P, R, R', S, and S' exist (are not *null*), and if the temporal conditions c_1 to c_5 are satisfied. The number of lines of the fact table can be reduced sharply by eliminating many combinations of Boolean values that do not make sense (e.g., if attributes of P or R are *null* then the conditions c_1 to c_5 cannot be evaluated), and by considering as inconsistencies only combinations of conditions that express relevant classes (e.g., for a particular company).

Table I exemplifies classes of inconsistencies considered relevant by the water supply company of our case study to investigate possible misbehaviors of its maintenance teams. The symbols $\exists$ and $\nexists$ indicate if the attributes of P, R, R', S, and S' are present or *null*, respectively, in the view *Res*. The columns c_1 to c_5 are checked (✓) if the respective temporal condition is satisfied, unchecked (✗) if unsatisfied, and marked as not possible to assess (-) otherwise. For instance, class 1 in Table I refers to no inconsistency, i.e., the reported task and the stop positions are sufficiently close to that

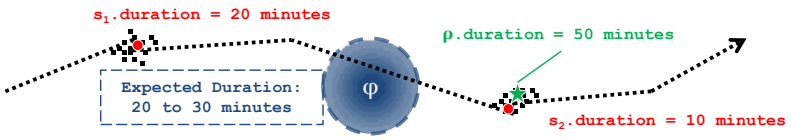

Fig. 4. Detection of inconsistency in Figure 3 (b)

of the planned task, and their durations are within the expected time limits of the planned task (conditions c_1 to c_5 are satisfied). The remaining classes of Table I refer to inconsistencies whose possible explanations appear in the last column. The inconsistency class 5 is illustrated in Figure 4.

4. EXPERIMENTS

This section reports recently performed experiments that confirm the viability of the proposal and the benefits of the new proposed steps to avoid returning excessive too many inconsistencies that in fact refer to bad data quality, and help explain user behaviors in detected idle stops.

The proposed method has been implemented as a prototype in Java, on top of a database managed with PostgreSQL 9.1.4 (64-bit) and PostGIS 2.0.3 r11132. The queries were performed in a server with an i7-2670QM 2.20GHz processor, 8GB RAM, and 750GB HD 7200 RPM.

We evaluated the prototype on data of a water supply company that manages maintenance teams, each one responsible to serve requests in a geographic region within given deadlines for each request. Each team carries a mobile device running an application that collects its geographic position at every minute. Moreover, this application allows the team to report the execution of each task, supposedly from its execution location at the moments when it begins and ends. Each maintenance team is considered a moving object, and each service request is considered a planned task. The spatiotemporal coordinates of raw trajectories and reported tasks were collected using the GPS sensors of the mobile devices which ensure 10 meters of accuracy. The planned tasks were extracted from a company's database, which have only the addresses where the tasks should be performed. Thus, we have used geocoding to infer the spatial coordinates of the planned tasks.

The set of raw trajectories used in the experiments have over 900,000 spatiotemporal positions, of 11 maintenance teams, collected between January 2008 and November 2013. The CB-SMoT method [Palma et al. 2008] implemented in Weka-SMTP [Bogorny et al. 2011] was used to extract stops from these trajectories with the following parameters: 300 *seconds* as minimum duration to consider a stop, and 0.3 m/s as the maximum allowed speed in a stop. It has generated 8,779 stops. However, the stops up to to 50$meters$ of the company's headquarters were discarded, as these positions are obviously wrong (reports made when the teams were back to their base of operations). Then, 6,080 stops remained.

There were 8,505 tasks planned to be executed in the same period. The Google Geocoding API was used to infer the planned tasks coordinates from the street addresses of the service requests. The planned tasks with inaccurate coordinates were deleted, remaining 6,167. The teams sent 6,468 reported tasks, but reported tasks of deleted planned tasks were ignored, remaining 4,731 reported tasks. The values of the spatial thresholds specified in subsection 3.2 were all the same: $STheshold\_PR = STheshold\_PT = STheshold\_RT = 50\ meters$.

Table I. Some relevant classes of inconsistencies, their definitions, and possible explanations

Class	P	R	R'	S	S'	c_1	c_2	c_3	c_4	c_5	Possible Explanation
1	∃	∃	∃	∃	∃	✓	✓	✓	✓	✓	No inconsistency
2	∃	∃	∃	∃	∃	✓	✗	✓	✓	✓	Short reported task
3	∃	∃	∃	∃	∃	✗	✓	✓	✓	✓	Prolonged report
4	∃	∃	∄	∃	∃	✗	✓	✓	✗	-	Prolonged report in wrong location and short stop
5	∃	∃	∄	∄	∄	✗	✓	-	-	-	Prolonged report and planned location unvisited
6	∃	∄	∄	∃	∄	-	-	✓	✗	-	Unreported task and planned location shortly visited
7	∃	∃	∄	∃	∄	✗	✓	✓	✓	-	Prolonged report with planned location visited
8	∃	∄	∄	∄	∄	-	-	-	-	-	Task not performed
9	∄	∃	∄	∄	∄	-	-	-	-	-	Report of unknown task
10	∄	∄	∄	∃	∄	-	-	-	-	-	Idle stop

Table II. Summary of the experimental results

Class	Count	%	%P	%R	%S
1	58	0.69	0.94	1.22	0.95
2	173	2.06	2.80	3.65	2.84
3	206	2.45	3.34	4.35	3.38
4	92	1.09	1.49	1.94	1.51
5	259	3.08	4.19	5.47	-
6	133	1.58	2.15	-	2.18
7	60	0.71	0.97	1.26	-
8	872	10.39	14.13	-	-
9	0	0	-	0	-
10	2218	26.45	-	-	36.48
Others -	4314	51.50	-	-	-

Table III. Types of PoIs near idle stops

Wikimapia PoI Type	#IdleStops
Bank	2
Church	6
Company	3
Gas station	23
Private condominium	2
Public building	12
Restaurants and bars	11
School	4
Services and stores	44

The proposed method detected 8,385 inconsistencies. Table II summarizes the results for some classes of inconsistencies described in Section 3.3 that were considered by the water supply company. The column **Count** shows the number of inconsistencies classified in each class. The 4 remaining columns in the right side of Table II show the percentages of inconsistencies of each class relative to: the total number of inconsistencies found (**%**); the total number of planned tasks (**%P**); the total number of reported tasks (**%R**); and total number of stops (**%S**). The proposed method detected that 14% of the tasks were not performed (class 8), more than one-fourth the inconsistencies found were idle stops (class 10), and only 58 planned tasks (0.69%) showed no inconsistency (class 1). It is important to note that these experiments considered all the possible classes of inconsistencies. Nevertheless, the relevant classes for a particular enterprise can be just a small subset of them. In practice, inconsistencies of several of the theoretical classes could be accounted as irrelevant inconsistencies.

Table III presents the types of Wikimapia PoIs close to idle stops detected in the experiments, along with the number of idle stops that occurred close to each type of PoI. The coordinates of each stop centroid were used to measure the distance to the PoIs obtained from the Wikimapia API. Only PoIs up to 30 meters away of such a centroid were considered close enough for linking to the respective stop. Notice that Table III shows a considerable number of idle stops in PoIs of the types *Services and stores* (44 idle stops), *Gas station* (23 idle stops), *Public buildings* (12 idle stops), and *Restaurants and bars* (11 idle stops). The types of PoIs near idle stops help to contextualize those stops. For example, a stop in a *Gas Station* may not be a misbehavior, while stops in *bars* or *stores* when the moving object was supposed to be working are clearly suspicious.

Finally, Figure 5 presents the geographical distribution of the inconsistencies of the classes presented in Table II found in a trajectory dataset collected between 1st and 30th January 2013. The cause of so many inconsistencies is currently under investigation yet. These results illustrate how the proposed method allows the detection of a wider variety of inconsistencies than related work, by considering matching/non-matching trajectories, planned tasks, and reported tasks. In addition, our classification of the detected inconsistencies provides hints for explaining them.

5. RELATED WORK

The work of Raj et al. [2008] fuses data generated by GPS and other sensors to recognize activities performed by moving objects. The used sensors include a 3-axis accelerometer, microphones for recording speech and ambient sound, photo-transistors for measuring light conditions, temperature sensors, and barometric pressure sensors. However, the use of diverse sensors requires lots of configurations. In addition, the processing of multimodal data can be quite difficult and costly.

Trajectories are compared with a schedule of planned tasks by Clark and Doherty [2008]. Their goal is to find rescheduled tasks by analyzing their execution locations (GPS coordinates), initial and final instants, planned tasks, and users' feedback. The findings are confirmed by the tasks executors

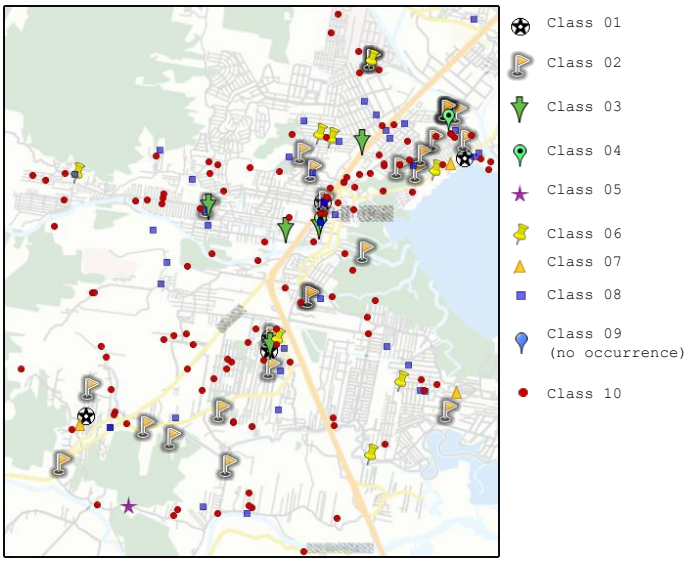

Fig. 5. Geographical distribution of considered classes of inconsistencies

via interviews. Their method does not use reported tasks, detects only temporal inconsistencies, and does not classify them.

The method for describing moving objects tasks and generating diaries introduced by Huang et al. [2010] assumes that tasks along a trajectory occur at Points of Interest (PoIs), where the moving objects remain stationary for a while. According to them a task has a location (e.g., geographic coordinates), a starting time, a duration, and a goal. They describe tasks according to categories of PoIs previously registered, and disregard that a PoI can play different roles for different moving objects (e.g., a restaurant can be a lunch place for a moving object, and a job for another one). The quantity and the quality of the PoIs set used are decisive for their method. The proposal of Furletti et al. [2013] is similar to the one of Huang et al. [2010], but instead of using PoIs previously registered, it uses PoIs from the Google Place API and OpenStreetMap. The type of a task is determined according to the type of location where the moving object stops to perform that task. However, neither Furletti et al. [2013] nor Huang et al. [2010] use data of planned tasks or reported tasks.

Another novel method presented by Chen et al. [2012] generates a travel diary based on connections to Wi-Fi access points, and signal strength of mobile phones. Finally, the method proposed by Yuan et al. [2013] constructs individual movement histories (similar to users diaries) from a smart card transactions dataset. These works do not consider planned tasks. They rely just on the use of Wi-Fi access points and smart card transactions, respectively, to produce the travel diary information.

For the best of our knowledge, our method is the only one to compare trajectories with planned and reported tasks to detect and classify inconsistencies. The version of our method described in this article is more advanced than the one presented in [da Silva and Fileto 2014] mainly because of the extra preprocessing step that removes planned tasks with inaccurate geolocations, and consequently avoid returning a high number of inconsistencies caused by problems in data quality. Moreover,it links idle stops to PoI's of existing geographic databases and LOD collections, to help explain these stops based on the semantics associated with properties of these PoIs, such as their types.

6. CONCLUSIONS AND FUTURE WORK

This article introduced a computational method to detect and classify spatiotemporal inconsistencies of trajectories with planned and reported tasks. The main advantages of the proposed method are:

(i) detection of inconsistencies among data of 3 distinct kinds (trajectories of moving objects, along with requests and reports of their tasks); (ii) classification of a wider variety of spatiotemporal inconsistencies than state-of-the-art methods; (iii) higher flexibility for input data than related methods, by not using rigid schedules (with specific times for executing planned tasks); (iv) a few parameters to tune the method according to properties of the analyzed datasets; and (v) possibility of linking some inconsistencies found such as idle stops to PoIs, which can help understand what is going on by analyzing the semantics associated to PoI properties such as type. It supports the investigation possible problems and misbehaviors of the moving objects (e.g., work evasion, fraudulent reports).

Future work includes: (i) further validation of the proposed method, with distinct datasets and some ground true to asses results quality measures, such as precision and recall; (ii) evaluation of the performance and the scalability of the proposed method with bigger datasets; (iii) finding automated systematic ways to tune the proposed method; (iv) analysis of trajectory annotations and social media posts made by the moving objects to obtain additional information for explaining their behaviors (e.g., places and events visited, actions, and intentions).

REFERENCES

Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., and Vaisman, A. A Model for Enriching Trajectories with Semantic Geographical Information. In *Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*. ACM, Seattle, USA, pp. 22:1–22:8, 2007.

Bogorny, V., Avancini, H., de Paula, B. C., Kuplich, C. R., and Alvares, L. O. Weka-STPM: a Software architecture and prototype for semantic trajectory data mining. *Transactions in GIS* 15 (2): 227–248, 2011.

Chen, Z., Wang, S., Chen, Y., Zhao, Z., and Lin, M. InferLoc: calibration free based location inference for temporal and spatial fine-granularity magnitude. In *Proceedings of International Conference on Computational Science and Engineering*. Paphos, Cyprus, pp. 453–460, 2012.

Clark, A. F. and Doherty, S. T. Use of GPS to Automatically Track Activity Rescheduling Decisions. In *Proceedings of the International Conference on Survey Methods in Transport*. Annecy, France, pp. 25–31, 2008.

da Silva, F. P. and Fileto, R. Analysis of Spatiotemporal Inconsistencies of Trajectories with Planned and Reported Tasks. In *Proceedings of the Brazilian Symposium on Geoinformatics*. Campos do Jordão, São Paulo, Brazil, pp. 1–12, 2014.

Furletti, B., Cintia, P., Renso, C., and Spinsanti, L. Inferring Human Activities from GPS Tracks. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*. Chicago, USA, pp. 1–5, 2013.

Huang, L., Li, Q., and Yue, Y. Activity Identification from GPS Trajectories Using Spatial Temporal POIs' Attractiveness. In *Proceedings of the ACM SIGSPATIAL Workshop on Location Based Social Networks*. ACM, San Jose, USA, pp. 27–30, 2010.

Mountain, D. and Raper, J. Modelling Human Spatio-temporal Behaviour: a challenge for location based services. In *Proceedings of the International Conference on GeoComputation*. University of Queensland, Brisbane, Australia, pp. 24–26, 2001.

Palma, A. T., Bogorny, V., Kuijpers, B., and Alvares, L. O. A Clustering-based Approach for Discovering Interesting Places in Trajectories. In *Proceedings of ACM Symposium on Applied Computing*. ACM, New York, USA, pp. 863–868, 2008.

Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M. L., Gkoulalas-Divanis, A., Macedo, J., Pelekis, N., Theodoridis, Y., and Yan, Z. Semantic Trajectories Modeling and Analysis. *ACM Computing Surveys* 45 (4): 42:1–42:32, august, 2013.

Pelekis, N. and Theodoridis, Y. *Mobility Data Management and Exploration*. Springer-Verlag New York, New York, USA, 2014.

Raj, A., Subramanya, A., Fox, D., and Bilmes, J. Rao-Blackwellized Particle Filters for Recognizing Activities and Spatial Context from Wearable Sensors. In *Proceedings of the International Symposium on Experimental Robotics*. Springer Berlin Heidelberg, Rio de Janeiro, Brazil, pp. 211–221, 2008.

Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., and Vangenot, C. A Conceptual View on Trajectories. *Data and Knowledge Engineering* 65 (1): 126–146, 2008.

Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., and Aberer, K. Semantic Trajectories: mobility data computation and annotation. *ACM Transactions on Intelligent Systems and Technology* 4 (3): 49:1–49:38, 2013.

Yuan, N. J., Wang, Y., Zhang, F., Xie, X., and Sun, G. Reconstructing Individual Mobility from Smart Card Transactions: a space alignment approach. In *Proceedings of International Conference on Data Mining*. Dallas, USA, pp. 877–886, 2013.