Minimal Spatio-Temporal Database Repairs

Tobias Emrich⋆, Hans-Peter Kriegel⋆, Markus Mauder⋆, Matthias Renz⋆,
Goce Trajcevski⋄[*], Andreas Züfle⋆
⋆Institute for Informatics, Ludwig-Maximilians-Universität München
⋄Northwestern University, EECS Department
{emrich,kriegel,mauder,renz,zuefle}@dbs.ifi.lmu.de
goce@eecs.northwestern.edu

ABSTRACT

This work tackles the management of novel types of inconsistencies in Spatio-Temporal Databases, different from traditional database settings where integrity constraints pertain to the explicitly stored (or, defined via views and aggregates) values. We observe that spatio-temporal data has its specific types of ßemanticconstraints and we aim at minimization of the changes needed for repairing their violations.

Categories and Subject Descriptors

H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval—*Query Processing*

General Terms

Algorithms, Performance

Keywords

Spatio-Temporal Data, Trajectory Data, Consistency, Repair

1. INTRODUCTION

Advances in networking and communications, along with miniaturization of computing and sensing devices and GPS and RFID technologies, have provided a foundation for generating extremely large volumes of location-in-time data [5]. Managing *(location,time)* data is crucial in many applications domains: from navigation and efficient traffic management, through emergency/disaster rescue management, environmental monitoring, fly-through visualization, and various military applications (e.g., radar data, troops tracking) [7]. Efficient techniques for storage, retrieval and query processing of spatio-temporal data are main research topics in the field of Moving Objects Databases (MOD) [4].

[*]Research partly supported by NSF grants CNS 0910952 and III 1213038.

SIGSPATIAL'13, Nov 05-08 2013, Orlando, FL, USA
ACM 978-1-4503-2521-9/13/11.
http://dx.doi.org/10.1145/2525314.2525468.

Physical factors (imprecision of sensing devices, communication links) often cause the location data to be inaccurate and noisy. However, even with perfect sampling accuracy – the data intended to capture a continuous motion can be represented in MOD only for discrete time-instants. A complementary observation is that data sources may be various heterogeneous devices: roadside-sensors, weather stations, satellite imagery, (mobile) weather radar, citizen supplied (crowd-sourced) observations, ground and aerial LIDAR, etc.

Uncertainty in moving objects locations due to sources-imprecision has been tackled from the perspectives of modelling, capturing it in syntactic expressions and query processing [11]. However, we identify other types of (semantic) inconsistencies that have not been addressed so far. As an example, a user posing a continuous k-Nearest Neighbor (k-NN) query, may be presented with an answer containing two (or more) vehicles that "have collided". However, this violates the semantic constraint: "two objects cannot be at the same place at the same time". In addition to imprecise location-samples – such violations may also arise due to the use of *interpolation* (linear, Bezier, etc.) [4] in-between actual samples.

The main objective of this work is to provide techniques for detection and "fixing" of such inconsistencies, which will also have some desirable properties (repair-constraints) such as minimizing the distance between the original and the repaired trajectories/databases. The main contribution of this work can be summarized as follows:

• We identify and formalize the problem of semantic inconsistencies in spatio-temporal data. This formalization identifies a wide class of problems that have been largely neglected in moving object and trajectory database literature.
• We present certain desirable properties that any approach for fixing inconsistency in a given spatio-temporal dataset should satisfy.
• We propose an approximate solutions to this problem, a greedy algorithm.

After a brief overview of some representative related works in Section 2, we present our main results in Section 3: formalization of the MOD inconsistencies problem, identifying the desirable properties of the repair-approaches, and proposing an algorithmic solution. Section 4 concludes the paper and outlines directions for future work.

2. RELATED WORK

Several bodies of research are related to the problems addressed in this paper – however, although each of them has generated interesting results, none has addressed the specific problems tackled by our work.

Traditional database approaches [1, 2, 12] *repair* the identified inconsistencies by removing objects or by changing attribute values – however, this is not directly applicable to spatio-temporal data. Arbitrarily changing a (location, time) pair is likely to yield other/new inconsistencies, as the changed trajectory may reach an unreachable state, or may have an unrealistically high speed in the repaired version of the database. Our main challenge is in incorporating rules for semantically meaningful repairs.

MOD researchers have often used linear interpolation for modelling the motion of spatio-temporal objects [6, 8, 10]. However, other approaches have been proposed – e.g., [9] introduced a framework that allows the future motion of objects to be described in a more complex manner than linear interpolation. Interestingly, the problem of *dead reckoning* can be viewed as a special case of the problem defined in this paper: Inconsistencies are given by objects not having a (location, time) in the time intervals defined by their discrete observations, and they are fixed by incurring a minimal deviation from the expected position of a moving objects. Many of the consistency-violations in trajectory databases are a consequence of the interpolation model, as any applications requires some form of interpolation between discrete measurements. The unique challenge we address is to go a step further and repair inconsistencies incurred by an imperfect interpolation model. Recent approaches model the motion of a spatio-temporal object by a stochastic process, where each possible world is associated with a probability [3]. However, more complex constraints, e.g., prohibiting objects from being at the same state at the same time, can not be straightforwardly incorporated.

3. INCONSISTENCIES AND REPAIRS

We now present the novel types of inconsistencies and desirable properties of (methodologies for) enforcing the semantic constraints in a given MOD.

A trajectory database $\mathcal{D}$ stores triples (*oid*, *location*, *time*), where *oid* $\in \{o_1, ..., o_{|\mathcal{D}|}\}$ is a unique object identifier, *location* $\in \mathcal{S}$ is a spatial position in space and *time* $\in \mathcal{T}$ is a point in time. Semantically, each such triple corresponds to the location of object o_i at some time. In $\mathcal{D}$, an object can be described by a function $tr_{o_i} : \mathcal{T} \rightarrow \mathcal{S}$ that maps each point in time to a location in space[1] $\mathcal{S}$; this function is called *trajectory*.

Knowing the location of an object o_i at *any* time requires some "balance" with the reality, since it can only be measured/determined at discrete time-instants. The frequency of location-samplings is also bounded by physical constraints, such as the availability of a GPS signal. Between discrete observations, the position of a moving object has to be estimated, which yields another type imprecision and introduce particular inconsistencies.

We assume a discrete and finite space of possible states $\mathcal{S} = \{s_1, ..., s_{|\mathcal{S}|}\}$ and a discrete and finite space of points of time $\mathcal{T} = \{0, ..., s_{|\mathcal{T}|-1}\}$. This assumption, common in MOD literature, allows a finite representation of arbitrary trajectories. Hence, we have inconsistencies in trajectory data that are consequences of the model based on discrete approximation of continuous phenomena (motion).

Generally, a *spatio-temporal constraint* can be thought of as any information describing some (semantic) constraint related to the trajectories in a given database $\mathcal{D}$. An example of such semantic constraints is *"Exit 233 from road number 421 can hold a maximum flow of ten vehicles every 30 seconds"*. The focus of this work is on the constraints pertaining to (co)locations of objects like, e.g., *two objects must be within certain distance from each other* or *two objects can not be at the same location at the same time*. $\mathcal{D}$ is considered to be *inconsistent* with respect to a constraint c, if c is violated by (some trajectories in) $\mathcal{D}$. The predicate $c(\mathcal{D})$ yields true if and only if $\mathcal{D}$ satisfies c.

Since we are considering historical data, there is no option of improving the information, e.g., by requesting the objects to give a more accurate position update. Thus, the only viable approach is to repair [1, 2, 12] the trajectories in order to mitigate the symptoms of this lack of information. More specifically we are interested in a repair which yields the *minimal difference* from the initial database $\mathcal{D}$.

DEFINITION 1 ((MINIMAL) DATABASE REPAIR). *Let $\mathcal{D}$ be a trajectory database inconsistent with respect to a set of semantic constraints C. Let $\mathcal{D}^R$ be a trajectory database derived from $\mathcal{D}$ which satisfies C, i.e., such that $\mathcal{D} \models C$ holds. Database $\mathcal{D}^R$ is called a* database repair *of $\mathcal{D}$.*

Let $dist(\mathcal{D}, \mathcal{D}^R)$ be a dissimilarity function between databases. A minimal repair $\mathcal{D}^R_{min}$ is defined as

$$\mathcal{D}^R_{min} = \underset{\mathcal{D}^R \in \overline{\mathcal{D}^{\mathcal{R}}}, \mathcal{D} \models C}{argMin} \quad dist(\mathcal{D}, \mathcal{D}^R),$$

where $\overline{\mathcal{D}^{\mathcal{R}}}$ represents the set of all possible repairs of $\mathcal{D}$.

The goal of this work is to efficiently compute, for a given trajectory database $\mathcal{D}$ and a set of semantic constraints C, a minimal repair $\mathcal{D}^R_{min}$ of $\mathcal{D}$.

The above statement are broad and they intend to provide a global categorization of our objectives. In order to provide a more grounded version and a sound problem formulation, we need to formalize a few modules.

3.1 Database Repairs

We now introduce two categories of repairs that are focal to the current work: *time-distorting* repairs and *space-distorting repairs*.

3.1.1 Time Distorting Repairs

A time distorting repair allows a trajectory to avoid inconsistencies by increasing or decreasing its velocity when traversing its sequence of states (i.e., arriving earlier or later in a given location). A waiting-based database repair allows a trajectory to avoid inconsistency by repeating, thus semantically waiting at, any state.

DEFINITION 2 (WAITING-BASED TRAJECTORY, DB REPAIR). *Let $T = [s_1, ..., s_{|T|}], T \in \mathcal{D}$ be a trajectory. A* waiting-based *repair of T is a trajectory $T^R \in \overline{\mathcal{T}}^{wait} := [s_1^+, ...s_{|T|}^+]$. Here, the notation s_i^+ corresponds to a sequence of $k \in \mathbb{N}$ repeats of state s_i. The set $\overline{\mathcal{T}}^{wait}$ denotes the infinite set of possible wait-based repairs of T.*

Let $\mathcal{D}$ be a trajectory database inconsistent with respect to a semantic constraint C and let $dist(\mathcal{D}, \mathcal{D}^R)$ be a dissimilarity function between databases. A minimal repair $\mathcal{D}^{wait}_{min}$ is defined as:

$$\mathcal{D}^{wait}_{min} =$$
$$\underset{\mathcal{D}^{wait} = \{T_1 \in \overline{\mathcal{T}}^{wait}_1, ..., T_N \in \overline{\mathcal{T}}^{wait}_N\}, \mathcal{D}^{wait} \models C}{argMin} \quad dist(\mathcal{D}, \mathcal{D}^{wait}).$$

3.1.2 Space-Distorting repairs

A space distorting repair allows a trajectory to avoid inconsistencies by replacing transitions by any spatial detour leading to the same state. A detour between two to states s_i and s_j is a path starting at s_i and leading to s_j. A trajectory repair that uses detour-based repairs only is defined as follows:

[1]Most often the Euclidian 2D space – although extensions to higher dimensions and road-network constraints have been considered.

DEFINITION 3 (DETOUR-BASED TRAJECTORY, DB REPAIR). *Let $T = [s_1, ..., s_{|T|}] \in \mathcal{D}$ denote a trajectory. A* detour-based *repair of T is a trajectory $T^R \in \overline{\mathcal{T}}^{dtr} := [s_1, D(s_1, s_2), ..., D(s_{|T|-1}, s_{|T|})]$. The notation $D(s_i, s_j)$ corresponds to a detour between state s_i and state s_j. The set $\overline{\mathcal{T}}^{dtr}$ denotes the infinite set of detour-based repairs of T.*

Let $\mathcal{D}$ be a trajectory database inconsistent with respect to a semantic constraint C. Let $dist(\mathcal{D}, \mathcal{D}^R)$ be a dissimilarity function between databases. A minimal repair $\mathcal{D}^{dtr}_{min}$ is defined as

$$\mathcal{D}^{dtr}_{min} = \underset{\mathcal{D}^{dtr}=\{T_1\in\overline{\mathcal{T}}_1^{dtr},...,T_N\in\overline{\mathcal{T}}_N^{dtr}\},\mathcal{D}^{dtr}\models C}{argMin} dist(\mathcal{D}, \mathcal{D}^{dtr}).$$

Clearly, the quality of a repair T^R of a trajectory T depends on the quality of the selected detour. To begin with, a detour should be feasible (e.g., possible to follow in a given underlying road network) and should have similar "cost" in time and space. The assessment of the quality of a detour, has to be performed by the dissimilarity function $dist(\mathcal{D}, \mathcal{D}^R)$.

Additional restrictions on the database repairs can be defined in similar fashion, depending on the given application. In particular, a database repair can be both time- and space-distorting. In the above definitions, the set of semantic constraints C and the dissimilarity function $dist(\mathcal{D}, \mathcal{D}^R)$ have been used in an abstract manner. The following two section will formalize these concepts.

3.2 Spatio-Temporal Constraints

Each violation of a constraint $c \in C$ corresponds to an (instance of some) inconsistency. A database repair $\mathcal{D}^R$ of $\mathcal{D}$ is required to satisfy all constraints. Formally, we have:

DEFINITION 4 ((INTER-)OBJECT CONSTRAINTS). *Let $\mathcal{D}$ be a trajectory database and let $T \in \mathcal{D}$.*

The set of object-specific constraints C_T is defined as:

$$T^R \in \mathcal{D}^R \Rightarrow T^R \models C_T$$

The set $C_{\mathcal{T}}$ of inter-object constraints on trajectories in $\mathcal{T}$ is defined such that

$$\mathcal{T} \subseteq \mathcal{D}^R \Rightarrow \mathcal{D}^R \models C_{\mathcal{T}}$$

The first category of constraints that we consider pertains to individual trajectories only, without any consideration of the other trajectories in $\mathcal{D}$. Examples of such constraints are rules like: "An object must not enter a specified area on Sunday 2am and 5am." which can be used to specify road blocks and "An object must travel some minimum (maximum) distance in a given time interval" which can be used to model knowledge about minimum (maximum) speed of objects. Further rules may include "*An object may not visit a state it has previously visited*", or "*a certain state must be visited in any repair*". Formally, this type of *object-specific constraints* is defined as follows.

An important property of object-specific constraints C_T of a trajectory T is the independence between C_T and other trajectories $T' \in \mathcal{D}$. This, in turn, allows to optimize each trajectory individually, without considering whether it may incur new inconsistencies in other trajectories.

The second, and computationally more complex class of constraints are *inter-object* dependencies. For example, a constraint may require two objects to be within some maximum spatial vicinity. Such objects may correspond to two vehicles of a group of friends continuously travelling together. Analogously, two objects T_1 and T_2 may be constrained to spatially co-locate at a given time t, corresponding to the information that "T_1 has seen T_2 at time t". Furthermore, an important type of inter-object constraints results from knowledge such as "two objects must not be at the same state at the same time". Formally, inter-object constraints are defined as follows.

In practice, inter-object constraints lead to hard optimization problems, as a single repair of one trajectory may have a large number of consequences for constraints involving other objects.

3.3 Quality of a Repair

To measure the quality of a repair, a dissimilarity function $dist(\mathcal{D}, \mathcal{D}^R)$ is needed which, in accordance with Definition 1, will be minimized. Thus, this function defines semantic of a "good" database repair. Semantically, a good database repair should consider the following three properties:
(1) The number of changes between $\mathcal{D}$ and $\mathcal{D}^R$ should be minimized, to minimize the distortion of the initial database $\mathcal{D}$.
(2) All trajectories should be treated equally-fair, in the sense that the number of changes should be divided evenly among all the trajectories.
(3) The changes within a trajectory should be divided fairly, such in order to avoid particular unnatural motion, such as extremely long waits and far-distance teleportations.

The choice of a proper dissimilarity function is highly application dependent. In trajectory databases, dissimilarity may be measured by some kind of edit-distance between two trajectories. The time-warping allowed by edit-distance however will be inappropriate for spatio-temporal data resulting from GPS measurements, where a single change such as the remove of a state of a trajectory will distort the whole trajectory at the time of the removal and thereafter. This will yield trajectory having a large distance from the observed (location, time) pairs. In the later case, Manhattan or Euclidean distance may be more appropriate. Clearly, mixtures of dissimilarity functions may also be appropriate in some applications.

In addition to the choice of a global dissimilarity function, many application require some fairness properties to be satisfied for a database repair to be considered "good". This can be achieved by adding additional dissimilarity functions such as

$$dist(T, T^R) = \sum_t f(dist(T(t), T^R(t))),$$

$dist(T(t), T^R(t))$ denotes a function describing the discrepancy between both trajectories T and T^R at time t and $f(x)$ is a function to weight this distance. For instance, $f(x)$ can be chosen as $f(x) = x^2$. In this case, the error at each individual point of time is given quadratic weight, such that three discrepancies of one will yield a total dissimilarity of three, while a single discrepancy of two will yield a larger total dissimilarity of four. Thus, the weighting function $f(x)$ ensures that larger discrepancies are given a larger impact, thus ensuring a degree of intra-object-fairness of changes within a trajectory.

Given a dissimilarity function $dist(T, T^R)$ defined on pairs of trajectories, the total dissimilarity $dist(\mathcal{D}, \mathcal{D}^R)$ is defined as follows

$$dist(\mathcal{D}, \mathcal{D}^R) = \sum_{T\in\mathcal{D}} g(dist(T, T^R)),$$

where $g(x)$ is a function to weight the distances of individual trajectories. Again, a meaningful value of this function is $g(x) = x^2$. In this case, each individual trajectory is given quadratic weight, such that a fair distribution of distortions among trajectories is encouraged. For example, having an distance of two in a single trajectory yields a total dissimilarity twice as large as having two trajectories with a dissimilarity of one each. The weighting function

$g(x)$ allows to specify a degree of inter-object fairness of changes between trajectories.

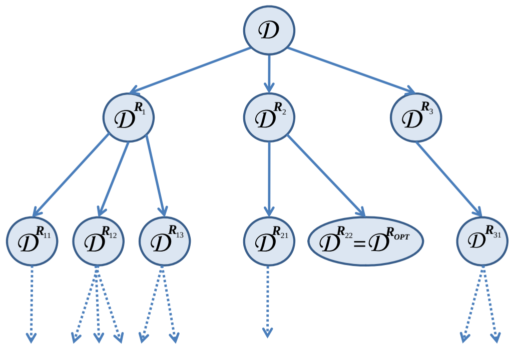

Figure 1: Search space to repair a database $\mathcal{D}$.

Figure 1 illustrates the main ideas of this work – possible repairs of a trajectory database $\mathcal{D}$. Each non-root node of the depicted tree corresponds to a possible repair $\mathcal{D}^R$, and the root node corresponds to the initial database $\mathcal{D}$. Edges between nodes indicate that a possible repair rule can be used to transform the instance of the trajectory database represented by the parent node into the trajectory database represented by the child node. Each edge is labelled with the cost incurred to perform this repair. Dotted edges, which are omitted for brevity, are assumed to have an excessively large cost. The total cost of each database repair is given by the cost of the path leading to it.

3.4 Computing Minimal Repairs

It can be shown that the problem of finding the optimal repair of a trajectory database $\mathcal{D}$ is generally NP-hard in the number of database objects N. Therefore we propose a greedy algorithm which, essentially, probes all repairs possible at a given stage (time), and iteratively chooses the best one. Clearly, this approach may "converge" into some local minima.

Algorithm 1 Greedy($\mathcal{D}$, A, C, dist(.,.))

```
 1: $\mathcal{D}^R = \mathcal{D}$
 2: while $\mathcal{D}^R \not\models C$ do
 3:     $a_{opt} = null$
 4:     $min\_cost = \infty$
 5:     for all $a \in PR(\mathcal{D}^R_H, A)$ do
 6:         if $dist(\mathcal{D}, a) < min\_cost$ then
 7:             $a_opt = a$
 8:             $min\_cost = dist(\mathcal{D}, a)$
 9:         end if
10:     end for
11:     $\mathcal{D}^R = a$
12: end while
13: RETURN $\mathcal{D}^R$
```

The main idea of the greedy approach is to make a locally optimal decision at each node of the decision tree corresponding to the problem of choosing the correct repair rule. The local optimality is based on the number of inconsistencies at a an intermediate repair. Starting at the root node, this algorithm iteratively probes all possible repair rules, and chooses to use the rule which yields minimum number of inconsistencies. The pseudo-code for this approach is presented in Algorithm 1.

4. CONCLUSIONS

In this work, we have formalized a category of problems that has been largely neglected in the MOD literature – repairing inconsistencies in historical trajectory databases. This is an important problem since such databases are inherently uncertain for a number of reasons and, in addition, attempt to capture continuous phenomena via discrete values.

We presented two types of distorting-based repairs (in spatial and temporal dimensions), and we identified quality criteria that a particular repair-approach should exhibit. This was formalized by introducing a corresponding distance function. We also presented a an algorithmic solution (albeit, not necessarily optimal) as a first step towards computing repairs of spatio-temporal database.

We note that it was not the objective of this work to tackle any efficiency/optimality aspects – we aimed at identifying and formalizing a novel problem and presenting a generic technique for addressing it. It is likely that much better efficiency can be achieved by exploiting specific (semantic) properties of a particular application, as well as specific repair rules and constraints. Moreover, it is also possible that some computationally efficient heuristics can be generated under special set of constraints, which are likely to produce solution within acceptable bounds from the optima one. However, these topics are subject of our future works.

5. REFERENCES

[1] M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. PODS*, pages 68–79, 1999.

[2] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *Proc. SIGMOD*, pages 143–154, 2005.

[3] T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Züfle. Indexing uncertain spatio-temporal data. In *Proc. CIKM*, pages 395–404, 2012.

[4] R. H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[5] M. K. G. Institute. Big data: The next frontier for innovation, competition, and productivity, 2011.

[6] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches to the indexing of moving object trajectories. In *Proc. VLDB*, pages 396–406, 2000.

[7] E. Pitoura and G. Samaras. Locating objects in mobile computing. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(4), 2001.

[8] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *Proc. SIGMOD*, pages 331–342, 2000.

[9] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *Proc. SIGMOD*, pages 611–622, 2004.

[10] Y. Tao, D. Papadias, and J. Sun. The tpr*tree: An optimized spatio-temporal access method for predictive queries. In *Proc. VLDB*, pages 790–801, 2003.

[11] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.*, 29(3):463–507, 2004.

[12] J. Wijsen. Database repairing using updates. *ACM Trans. Database Syst.*, 30(3), 2005.