IMO: A Toolbox for Simulating and Querying "Infected" Moving Objects

Jianqiu Xu[1] Hua Lu[2] Zhifeng Bao[3]
[1]Nanjing University of Aeronautics and Astronautics, China
[2]Roskilde University, Denmark
[3]RMIT University, Australia
jianqiu@nuaa.edu.cn luhua@ruc.dk zhifeng.bao@rmit.edu.au

ABSTRACT

Due to the widespread use of GPS-enabled devices such as smartphones, the research field of moving objects databases has been quite active in the past decade. Human movements are recorded, managed and analyzed for a plethora of applications. In this demo, we introduce a toolbox named IMO to simulate and query infected moving objects. This is primarily motivated by COVID-19 virus pandemic recently. We model the *spreading* behavior to demonstrate when and where people are infected. The protection policy is simulated such that one can see the isolation and self protection effects such as human movement restriction and the wearing of masks. Optimization techniques are developed to enhance the performance, including data storage, data partition and index structures. This is not a standalone software but a toolbox embedded in SECONDO, an open source and extensible database system. To the best of our knowledge, simulating and querying infected objects are not supported in existing moving objects prototype systems. Demo attendees can conveniently pose their queries and adjust parameters in the interface and the system will visualize the result after only a short delay.

PVLDB Reference Format:
Jianqiu Xu, Hua Lu, Zhifeng Bao. IMO: A Toolbox for Simulating and Querying "Infected" Moving Objects. *PVLDB*, 13(12): 2825-2828, 2020.
DOI: https://doi.org/10.14778/3415478.3415485.

1. INTRODUCTION

Moving objects databases [4] (MODs) are used to manage spatial objects that continuously change their locations over time. MODs have a wide range of applications, such as vehicle management and human behavior analysis. In the literature, a great deal of efforts have been made in this area to support a range of query processing and data analytics on interesting issues such as nearest neighbor query, similarity search and movement pattern analysis. An important feature of such data is that objects are moving around such that the relationship is modeled by a time-dependent function. The COVID-19 virus has spread over many countries and people will likely to be infected if they are ever close to each other. This motivates us to develop a toolbox IMO (infected moving objects) to: (i) efficiently look for infected objects from their movement data and (ii) simulate the spreading event with and without immediate protection policy. IMO can be used by authorities to look for people being infected or likely being infected. The simulation helps people understand how fast the virus is spreading.

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415485.

Consider the example in Figure 1. o_f is an infected object and is close to o_1 for a while, leading to o_1 being infected. Although o_2 is never close to o_f, it is infected by o_1. Later, $\{o_f, o_1, o_2\}$ will infect more objects. o_3 is close to o_1 for a while but o_1 is not infected at that time. Therefore, o_3 should not be considered as an infected object.

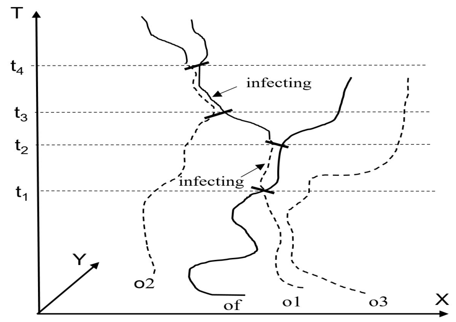


Figure 1: An example of infected moving objects

The infecting event occurs when two moving objects are close to each other for a while. Although time-dependent closeness can be determined by spatio-temporal operators such as nearest neighbor and range queries, the result is not complete because when an object is infected, it will move around and keep infecting others. One needs to repeat the searching procedure for each infected object, and the evaluation time is different for each case. This is the characteristic of the *spreading behavior*. An alternative solution is to perform the *self-join* operation to report all pairs of objects which have been close to each other for a while. However, this will result in a large number of candidates, most of which may not contribute to the result. A pair should not be reported if none of them is infected, although they are close to each other. The time order relationship has to be carefully handled because the infecting event only occurs when an object is infected. Before that, objects are safe even though they are close.

In this demo, we model the infecting event and perform different queries in a database system to look for infected objects with or without protection policy (e.g., isolation and people wearing masks). This is a new task in the domain of moving objects databases. Results are animated in the interface such that users can clearly see when and where the infecting event occurs. Our approach has two novelties: (i) moving objects can be stored by a number of fixed-length records. They are configured in different sizes such that an optimal setting is made for the current workload. This reduces I/O cost in comparison with accessing variable-length records. (ii) An optimal partition method is provided to split the movement into pieces for data locality. A range of index structures (global and local) are maintained to enhance the efficiency. The demonstration allows the audience to send infecting queries, experience the effect of protection policy and observe the animation of the spreading behavior.

Recently, a number of prototype systems have been built to manage spatio-temporal trajectories such as ST-Hadoop [2] and DITA [7]. Ratel [5] provides interactive analytics and trip planning for large scale trajectories in which range, join and sampling operators are supported. A tool for indoor mobility data is developed in [6]. However, the infecting event and spreading behavior among moving objects are novel issues and have not been investigated before.

In addition to processing infected moving targets, IMO can benefit several real-world applications such as the analysis of public transportation scheduling and indoor movement, and discovering distance-related spatio-temporal relationships (e.g., tracking). The method of providing different storage models (a number of fixed-length records with different sizes) can be utilized in other systems as well.

In this demo, we make the following contributions: (i) We formalize the problem of detecting infected moving objects and model the protection policy. Both outdoor and indoor movements are supported. (ii) We develop the toolbox inside a database system in order to effectively simulate and visualize moving objects, and efficiently process and perform analytics over large datasets. (iii) We demonstrate how an infected object can infect a large number of objects and how protection policies can effectively control the infected area and reduce the number of infected objects. The proposed temporal function can also be used to process continuous range and distance join queries over moving objects.

The rest of the paper is organized as follows: We formalize the problem in Section 2 and present our approach in Section 3. The demonstration is described in Section 4.

2. PROBLEM FORMALIZATION

Each moving object is represented by a moving point, denoted by *mo* [4]. Essentially, this is a sequence of so-called temporal units, each of which consists of a time interval and a description of a linear movement during this time interval. The movement is defined by recording the positions at start and end time points, respectively. Given a moving object, time intervals of distinct units are disjoint.

DEFINITION 2.1. ***Moving objects***
$\mathcal{O} = \{o | o = \langle u_1, \ldots, u_n \rangle\}$ *such that* :
(i) $u_i = (t_1, t_2, p_1, p_2)$, t_1, $t_2 \in$ `Instant` *and* p_1, $p_2 \in \mathcal{R}^2$
(ii) $u_i.t_1 < u_{i+1}.t_1$
(iii) $\forall u_i, u_j$: $[u_i.t_1, u_i.t_2]$ *is disjoint with* $[u_j.t_1, u_j.t_2]$.

The infecting occurs when two objects are close to each other. That is, the time-dependent distance is smaller than a threshold for a while. Let $dist(o_1, o_2, T)$ be a time-dependent function returning the distance between two objects o_1 and o_2 (o_1, $o_2 \in \mathcal{O}$) during a time interval T. Two moving objects are mapped into pieces at the same time interval, each of which follows a linear motion model. The distance changes over time, which is represented by the square root of a quadratic polynomial. The coefficients depend on locations and velocities.

DEFINITION 2.2. ***Infecting event***
An object $o' \in \mathcal{O}$ *is infected by another object* o *(infected already) if* $\exists T > \Delta T \cdot m$, $\exists \Delta d$ *such that* $\forall t \in T$: $dist(o_1, o_2, T) < \Delta d$. *The parameter* m ($m \in \underline{int} \wedge m \geq 1$) *is a variable that we use to model the effect of wearing a mask.*

Figure 2 exemplifies the condition of the infecting event. ΔT and Δd are two thresholds that trigger the infecting behavior, e.g., ΔT = 10 minutes and Δd = 1 meter. By default, we set $m = 1$, meaning that the two objects are not protected by a mask. One can increase the value if one or both objects wear masks for protection.

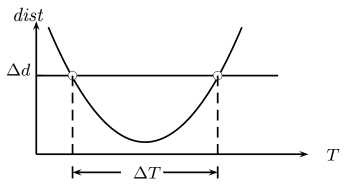


Figure 2: Distance curve

DEFINITION 2.3. ***Infected queries***
Given an infected object o, the infected query will report all infected objects $\mathcal{O}' \subseteq \mathcal{O}$ *such that* $\forall o' \in \mathcal{O}'$: $\exists o'' \in \mathcal{O}'$, o' *is infected by* o''.

The query will report all infected objects, which may be infected by different objects and later in turn infect others. This is the *spreading* behavior.

3. THE FRAMEWORK

We provide an overview in Section 3.1, present key parameters for simulation in Section 3.2 and introduce the query procedure in Section 3.3.

3.1 Overview

IMO is embedded into a database system such that we can perform query processing and data analytics over large datasets in addition to moving objects animation. Figure 3 depicts the framework consisting of six layers.

Storage layer. The current storage provides variable-length records for moving objects. The system maintains two files. One is for pointers/offsets and the other is for values. In order to reduce the I/O cost of accessing the underlying data, we propose using a range of fixed-length records with different sizes to replace variable-length records, as illustrated in Figure 4(a). The system automatically selects an optimal setting based on a cost model for the current workload. In addition to the data type for moving objects, a range of data types are defined to represent spatial objects such as roads, bus routes, walking areas, rooms and doors.

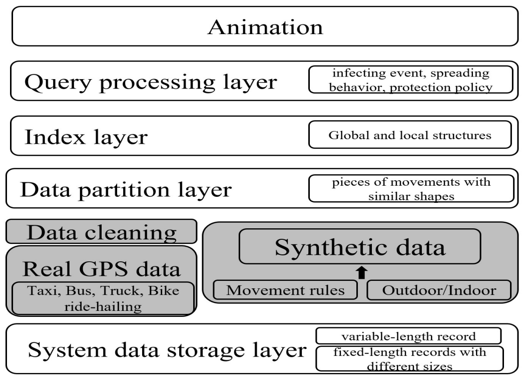

Animation
Query processing layer
infecting event, spreading behavior, protection policy
Index layer
Global and local structures
Data partition layer
pieces of movements with similar shapes
Data cleaning
Real GPS data
Taxi, Bus, Truck, Bike ride-hailing
Synthetic data
Movement rules
Outdoor/Indoor
System data storage layer
variable-length record
fixed-length records with different sizes

Figure 3: IMO architecture

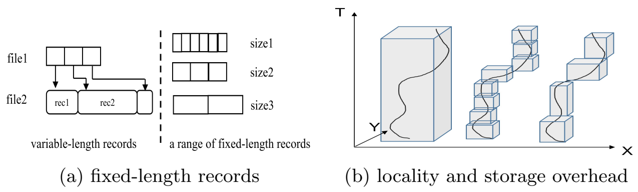


Figure 4: Data storage and partition

Data retrieval. If real datasets are processed, a data cleaning procedure is involved. There are two steps: *error detection* and *data repairing*. This is because mobile sensors usually contain a number of errors which significantly inhibit the query processing and data analytics. Synthetic data is produced based on trip planning in which both outdoor and indoor scenarios are supported.

Data partition. We combine space partitioning and data partitioning to minimize the I/O cost during query processing. There are several options to perform the partition, as demonstrated in Figure 4(b). One can use a big box or a number of small boxes to approximately represent the movement. The former does not preserve the data locality while the latter increases the index overhead. The system provides a method to balance the locality and the storage cost by splitting moving objects into pieces with similar shapes.

Index and query processing. A list of index structures are maintained to improve the query performance. We have a global structure which is a combination of grid-index and 3D R-tree, created on partitioned objects [8]. Both historical data and online updating are supported. The query processing evaluates the infecting event and reports infected objects with or without protection policy. Auxiliary structures are created on the target and candidate objects such as bounding box array and candidate tree. The bounding box array is built on the target and is able to report the trajectory approximation for the query time at constant time, speeding up the distance computation.

User interface. This part performs the animation such that users can clearly see when and where objects are infected. Operators are defined to transform 3D indoor structures (e.g., rooms and doors) to 2D space.

3.2 Key Parameters

Movement rules. A number of movement rules are defined to generate human trips such as visiting POI, visiting nearest neighbors, and commuting. For an outdoor trip, a person starts from one place (e.g., residential/work area), uses the public transportation system (bus or metro) and arrives at another place with some walking distances. Within a building, the trip starts from one location (e.g., an office room), follows the time or distance shortest path to the destination (e.g., a meeting room) during which elevators or staircases may be used. We use public floor plans to construct the building.

Protection policy. One effective solution is to perform the isolation by restricting the area where objects move. We partition the space into a set of disjoint areas and each object only moves in one area. The partition rule is arbitrary, e.g., a set of equal-size rectangles, Voronoi diagram, triangulation. The *mask* effect is modeled by a time interval m representing the time span during which two objects stay. If both objects wear masks, m^2 is the result.

Incubation period. If a safe object meets an infected object, there will be no symptom for a few days before illness. Such a stage is called *incubation period.* We model the value by a time interval and configure different settings as humans may have long or short incubation period.

3.3 Query Processing

We outline the query procedure of looking for infected objects in Figure 5. There are three steps. Step 1 takes the first infected object and performs the data partition to define an approximate spatio-temporal search region during which objects will be infected. Step 2 accesses the spatio-temporal index structure to do the filtering. That is, objects that cannot be infected will be pruned. Step 3 processes each candidate through the infecting evaluation to report infected objects. Such objects will be put into the result set and also be considered as targets infecting others later. The spread evaluation performs two tasks: (i) restrict the time interval of the infected object because its start time is not the original time but after the infecting event; (ii) if the object has been considered as the target infecting others, we will not process it again. Otherwise, the searching procedure continues as the objects infect each other. A priority queue is maintained for targets which are increasingly ordered by time. For each object from the queue, we repeat steps 1-3. The procedure terminates when there is no infected object in the queue. For each infected object, when and where the infecting occurs can also be reported.

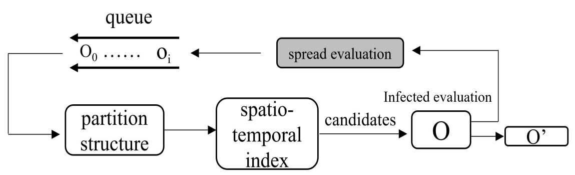


Figure 5: Query procedure

When protection rules are triggered, we follow the same query procedure by changing the condition of determining the infecting event. The method is straightforward for mask as one can adjust the parameter m. If the isolation occurs, we only report the result when two evaluated objects are located in the same partition area. Suppose that we have a relation $MO(Id\colon int, Trip\colon mpoint)$ that embeds $\underline{mpoint}$ as a relational attribute. The infected query is executed by the following two statements in the system syntax. The first statement defines the first infected object. Operators are used to do the filtering and extract a particular object from a relation. The second statement performs the query in which users can vary parameters to simulate different scenarios.

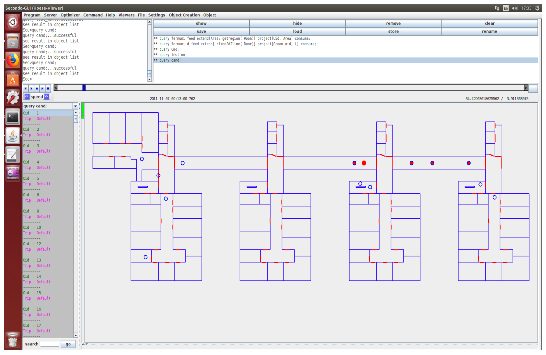
```
Let Qmo = MO feed filter[.Id = 12] extract[Trip];
query infected(Qmo, MO, Index,ΔT,Δd,m) consume;
```

4. DEMONSTRATION

We implement the toolbox in an open source and extensible database system SECONDO [3] using C/C++. A desktop PC (Intel(R) Xeon(R) E5-2620 v4 , 2.1GHz × 14, 64GB memory, 2TB hard disk) running Ubuntu 14.04 (62 bits, kernel version 4.4.0) is used for the demonstration.

The system supports processing both real datasets (e.g., taxi and ride-hailing trajectories [1]) and synthetic datasets. Raw GPS data are uploaded into the system and then transformed into moving objects after data cleaning. One can set the number of objects to simulate the effect of population density. Intuitively, people inside a building have a higher probability being infected than people in an open area. We have a range of open floor plans to simulate indoor scenarios including office buildings, universities, hotels, and train stations. Indoor trips contain movement involving elevators and staircases in which people are likely to be infected. After preparing the datasets, we perform the data partition and build index structures. These tasks are executed by calling a list of operators in a running script file. Users do not have to type the commands one by one.

Scenario 1: Infecting and spreading behavior. We choose one object from the dataset as the first infected target. This is selected randomly or intentionally with long life time and long moving distance. Users can configure ΔT and Δd to demonstrate time and distance effects. The query procedure is executed to progressively report all infected objects. The terminal returns a list of moving objects and the graphical interface displays their movement such that one can clearly see how the event occurs. In particular, when and where an object is infected. We demonstrate the indoor scenario in Figure 6 by using an university floor plan (the 2D area is [0, 80] meter × [0, 35] meter). For visualizing purpose, we extend the life time of an infected object to the ending time among all objects. Otherwise, the object will disappear when its life time is over.

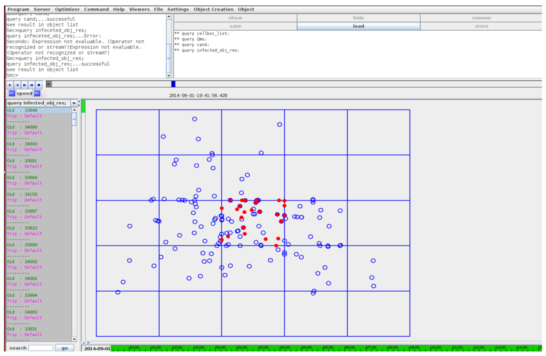


```
see result in object list
Sec>query cand;
query cand;...successful
see result in object list
Sec>query cand;
query cand;...successful
see result in object list
Sec>query cand;
query cand;...successful
see result in object list
Sec>
```

```
** query fernuni feed extend[Area: getregion(.Room)] project[Oid, Area] consume;
** query fernuni_d feed extend[L:line3d2line(.Door)] project[Groom_oid, L] consume;
** query Qmo;
** query test_mo;
** query cand;
```

Figure 6: Infected moving objects inside a building

Scenario 2: Protection policy. If the protection policy is executed, the number of infected objects decreases. Furthermore, the range of infected objects is limited. We demonstrate the isolation effect that restricts the movement in a certain area, as shown in Figure 7. Among the infected areas, we can report top-k regions with the largest number of infected objects. The mask effect is demonstrated such that objects stay close to each other for a longer time to be infected. Table 1 reports the simulation statistics including the overall time costs of determining all infected objects with and without protection policy. Since the protection policy restricts the movement area, the processing time is reduced.


Program Server Optimizer Command Help Viewers File Settings Object Creation Object
show hide remove
save load store
speed
2014-09-01-19:41:56.428
query infected_obj_res;

```
query cand;...successful
see result in object list
Sec>query infected_obj_res;
query infeceted_obj_res;...Error:
Secondo: Expression not evaluable. (Operator not
recognized or stream?)Expression not evaluable.
(Operator not recognized or stream?)
Sec>query infected_obj_res;
query infected_obj_res;...successful
see result in object list
Sec>
```

```
** query cellbox_list;
** query Qmo;
** query cand;
** query infected_obj_res;
```

```
Oid  : 33948
Trip : Default
---------
Oid  : 34090
Trip : Default
---------
Oid  : 34043
Trip : Default
---------
Oid  : 33991
Trip : Default
---------
Oid  : 33964
Trip : Default
---------
Oid  : 34150
```

Figure 7: The effect of isolation

Table 1: Statistics

X range	Y range	# objects	without protection	protection
[0, 100000]	[0, 100000]	32000	8.2s	3.1s

Scenario 3: Time-dependent distance. The infecting event is based on processing time-dependent distances among moving objects. The function also can be used to answer *continuous range queries* and *distance join queries* which report objects keeping within a certain distance to the target for some time and pairs of nearby objects, respectively. Continuous range queries can be used to find out the *tracking* behavior and distance join queries can be used to find out travelers with similar routes.

Acknowledgment. This work is supported by NSFC under grants 61972198, Natural Science Foundation of Jiangsu Province of China under grants BK20191273. Hua Lu is partly supported by the Independent Research Fund Denmark (No. 8022-00366B). Zhifeng Bao is supported in part by ARC DP200102611 and Google.

5. REFERENCES

[1] https://www.datatang.ai/ (2020).

[2] L. Alarabi and M. F. Mokbel. A demonstration of st-hadoop: A mapreduce framework for big spatio-temporal data. *PVLDB*, 10(12):1961–1964, 2017.

[3] R. H. Güting, T. Behr, and C. Düntgen. Secondo: A platform for moving objects database research and for publishing and integrating research implementations. *IEEE Data Eng. Bull.*, 33(2):56–63, 2010.

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

[5] H. Li, G. Li, J. Liu, H. Yuan, and H. Wang. Ratel: Interactive analytics for large scale trajectories. In *SIGMOD*, pages 1949–1952, 2019.

[6] H. Li, H. Lu, F. Shi, G. Chen, K. Chen, and L. Shou. TRIPS: A system for translating raw indoor positioning data into visual mobility semantics. *PVLDB*, 11(12):1918–1921, 2018.

[7] Z. Shang, G. Li, and Z. Bao. DITA: A distributed in-memory trajectory analytics system. In *SIGMOD*, pages 1681–1684, 2018.

[8] J. Xu, Z. Bao, and H. Lu. Continuous range queries over multi-attribute trajectories. In *IEEE ICDE*, pages 1610–1613, 2019.