Querying Multi-Source Contact Events Using Trajectory Data

Mingyue Zhang

ABSTRACT

Employing the movement paths of objects and executing contact event inquiries during the spread of diseases serve as effective prevention and control strategies. Current algorithms for contact query processing are limited to single-source (one-to-one) events, failing to capture multi-source (n-to-one) contact scenarios. This paper introduces efficient approaches for processing queries of multi-source contact events. Initially, we establish a clear definition for these multi-source events. We then describe a fundamental algorithm for their query processing that incorporates a sliding window for sequential scanning. Enhancements to this method include the introduction of a 2-dimensional bitmap filter and scanning based on anchor time points to boost efficiency. Thorough testing with real-world data validates that our proposed algorithms effectively identify more potential contact scenarios and perform efficiently in reducing query time costs. We conducted experiments using three spatial data repositories, and the experimental results validate the efficiency of our proposed spatial dataset search scheme.

KEYWORDS

Spatial Trajectory Dataset, Contact Event, Contact Event

1 INTRODUCTION

As wireless communication, satellite positioning technology, and mobile devices continue to advance rapidly, an immense volume of spatio-temporal trajectory data from moving objects is being collected. Utilizing this data to analyze the behavioral patterns of these objects offers considerable benefits for practical applications. A wide array of behavioral patterns in moving objects has been rigorously explored, covering phenomena like convoys[4, 5], traveling companions[6, 7], co-movement[3], gatherings[11], and trajectory clustering[8, 9]. These patterns find extensive applications across various fields including travel recommendations, traffic management, urban computing, and location prediction. Consequently, a thorough investigation of these behavioral patterns is crucial for maximizing their practical benefits.

Within the scope of trajectory behavior patterns, the exploration of contact events involving trajectories represents a notable research gap. Contact events are defined as instances when two or more moving objects are in close proximity for a specific period, facilitating the exchange or transmission of information, such as diseases, social interactions, or other pertinent data. A practical example of this is monitoring contacts during outbreaks of infectious diseases. Traditional methods like Bluetooth connections between devices, while useful, pose implementation challenges in this setting due to their need for significant user participation and their inability to detect contacts across varying distances because of hardware constraints. In contrast, trajectory data, which is readily accessible, computationally efficient, and broadly available, proves to be a more effective tool for identifying contact events under these circumstances.

Various methods have been developed to identify contact events. Alarabi and colleagues [1]introduced a real-time method called TraceAll, designed to monitor all moving objects that come into proximity with a known contact source. Xu and colleagues [10] developed a toolbox named IMO, which employs a rigorous model for identifying contact events and is utilized for simulating and discovering such events. Chao and colleagues[2] devised a generalized approach called the trajectory contact search (TCS) query, which tackles the contact tracing issue along with other trajectory-based challenges. However, these existing approaches primarily address one-to-one contact events and overlook scenarios in which a moving object may come into contact with multiple sources independently.

In this paper, we propose an efficient multi-source contact event query for moving objects. The model definition of multi-source contact event is first presented, which describes the specific determination of multi-source contact events. A baseline query processing algorithm based on sliding window (MCEQ) is presented to discover the contact events. To improve the efficiency of query processing, we propose two optimization strategies. First, a 2-dimensional bitmap filter is designed to filter the moving objects where no contact events have occurred. Second, the anchor time points are used to filter the time points when no contact event has occurred. By using these strategies, an optimized query processing algorithm (MCEQ+) is proposed. Experimental results on a real dataset show that the proposed algorithms can discover more potential contact events, and in addition, the query time cost of MCEQ+ is significantly reduced compared to MCEQ. Overall, the contributions of this paper are summarized as follows:

- We introduce a concept for multi-source contact events along with a foundational algorithm for querying such events, utilizing a sliding window-based scanning technique.
- We have developed two optimization strategies: anchor time points and a 2-dimensional bitmap filter, both integrated into our refined query algorithm.
- Our experimental results on a real spatial data repositories evaluate the performance of our schemes.

2 PROBLEM FORMULATION

The problem formulation of this paper is given as follows.

Definition 1 (Trajectory). *A trajectory of a moving object oi, can be represented as a polyline consisting of a finite sequence of positions at different time points. The trajectory of O_i is define as $Tra_i = \{(x_1^i, y_1^i), (x_2^i, y_2^i), \ldots, (x_m^i, y_m^i)\}$, where (x_j^i, y_j^i) is the 2D space coordinates of o_i at time point t_j . The set of time points for the trajectory is denoted as $T = \{t_1, t_2, ..., t_m\}$.*

Definition 2 (Sliding Window). *Given a window width threshold τ, a sliding window W_i is composed of τ consecutive time points starting from the time point t_i, $W_i =< t_i, t_{i+1}, \ldots, t_{i+\tau-1} >$. The j-th time point in W_i is denoted as $W_i[j]$.*

Definition 3 (**One-point-contact Event).** *An one-point contact event to the object o at time point t indicates that o is contacted with a set of contact sources, which is denoted as a triple (S_t, o, t). S_t is called one-point-contact sources, the calculation of which is shown in Eq.(1),*

$$S_t = \{s | s \in S \land dist(o, s, t) \leq d\} \tag{1}$$

where S is the set of contact sources, $dist(o, s, t)$ denotes the Euclidean distance between the moving object o and the contact source s at time t.

Definition 4 (**Multi-source Contact Event).** *For a moving object oi, given a distance threshold d and a sliding window width threshold τ , if there are a sliding window W_j and a sequence of one-point-contact sources $Seq_i =< S_{W_j[1]}, S_{W_j[2]}, \dots, S_{W_j[\tau]}>$, satisfying the following conditions at the same time, we said that a multi-source contact event occurs, which is denoted as $e = (Seq_i, o_i, W_j[\tau])$.*

- $\forall W_j[k] \in W_j \left(S_{W_j[k]} = \emptyset \land \forall s \in S_{W_j[k]} \left(dist(o_i, s, W_j[k]) \leq d\right)\right)$
- $\forall W_l < W_j \ (\exists W_l[k] \in W_l \ (\forall s \in S \ (\neg dist(o_i, s, W_l[k]) \leq d)))$

In Definition 4, for a multi-source contact event $e = (Seq_i, o, W_j[\tau])$, the first condition requires that one-point contact occurs at each time point in the sliding window W_j , while the second condition means that W_j is the first sliding window satisfying the first condition.

Definition 5 (**Multi-source Contact Event Query).** *Given a moving object set O, a contact source set S, a distance threshold d and a sliding window width threshold τ , a multisource contact event query is to get a result set R which contains all contact events caused by S.*

The goal of this paper is to propose contact event query processing algorithms based on the trajectory data of moving objects, which can discover all contact events efficiently and accurately.

3 THE BASELINE MULTI-SOURCE CONTACT EVENT QUERY PROCESSING ALGORITHM

In this section, we propose a baseline algorithm for the multi-source contact event query processing. According to the definition of multi-source contact event (Definition 4), for an innocent object o_i, if a contact event $(Seq_i, o_i, W_j[\tau])$ occurs, then one-point contacts has occurred at every time points within the sliding window W_j . The end time points of W_j is $W_j[\tau]$ and $W_j[\tau]$ is the time points when $(Seq_i, o_i, W_j[\tau])$ occurs. Based on this, we present the idea of the baseline algorithm as follows. Every time points in $T = t_1, t_2, \dots, t_m$ are sequentially scanned to determine whether one-point-contact events occur in innocent objects. For an innocent object, once there are τ consecutive time points in a sliding window satisfying that a one-point-contact event has occurred at each time point, a contact event to the innocent object is discovered and the innocent object becomes a new contact source. The details of the baseline algorithm MCEQ are presented in Algorithm 1.

In Algorithm 1, lines 5-7 are to obtain the one-point contact sources for each innocent object; lines 8-17 are to discover contact events within sliding windows for each innocent object. We assume that the average numbers of innocent objects and contact sources are α and β, respectively, and the time complexity of the baseline algorithm MCEQ is $O(m\alpha(\beta + \tau))$ according to the procedures of the algorithm.

```
Algorithm 1: MCEQ($O$, $S$, $d$, $\tau$)
   Input: $O$: the set of moving object, $S$: the initial set of
          contact sources, $d$: the distance threshold, $\tau$: the
          sliding window width threshold
   Output: $R$: The set of multi-source contact events
 1 $R \leftarrow$ Initialize a result set
 2 foreach $t_j \in < t_1, t_2, \dots, t_m >$ do
 3     $O' = O - S$
 4     foreach $o_i \in O'$ do
 5         foreach $s_u \in S$ do
 6             if $Dist(o_i, s_u, t_j) \leq d$ then
 7                 $S_{t_j} = S_{t_j} + \{S_u\}$
 8         if $j \geq \tau$ then
 9             $flag = true$
10             $Seq_i = \varnothing$
11             foreach $k = 0, 1, \dots, \tau - 1$ do
12                 $Seq_i = Seq_i + \{S_{t_{j-k}}\}$
13                 if $S_{t_{j-k}} = \varnothing$ then
14                     $flag = false$
15                     break
16             if $flag = true$ then
17                 $R = R + \{(Seq_i, o_i, t_j)\}$
18     $O' = O' - \{o_i\}$
19     $S = S + \{o_i\}$
20 return $R$
```

4 THE OPTIMIZED MULTI-SOURCE CONTACT EVENT QUERY PROCESSING ALGORITHM

The baseline algorithm MCEQ requires checking whether contact events occur for all innocent objects at each time point. For the innocent objects that have no chance to be contacted with contact sources at some time points, if they could be identified and filtered out in time, the time cost of query processing will be saved. Meanwhile, if the time points where no contact event has occurred could be determined and ignored, the time cost of query processing will be further saved. Inspired by the idea, we present two optimization strategies: the 2-dimensional bitmap filter-based optimization and the anchor time point-based optimization.

4.1 2-Dimension Bitmap Filter-based Optimization

According to the definition of contact event (Definition 4), for each contact source, the contact range is fixed and only the innocent object in the contact range of a contact source could incur contact event. Thus, we design a novel structure called a 2-dimensional bitmap filter to record all innocent objects within the contact range of the contact sources, which can be used to accelerate the query processing.

Definition 6 (**2-Dimensional Bitmap Filter (BF)).** *Given a grayscale image I_a of D_a, The 2- dimension bitmap filter at the time point t_k is denoted as BF_{t_k} which is a two-dimensional bit matrix. Given a set of contact sources S, the value setting of BF_{t_k} follows Eq.(2), where $o_i \in O$ - $S \wedge s_j \in S$.*

$$BF_{t_k}[i,j] = \begin{cases} 1 & dist(o_i, s_j, t_k) \le d \\ 0 & dist(o_i, s_j, t_k) > d \end{cases} \tag{2}$$

In Definition 6, $BF_{t_k}[i,j] = 1$ represents that the distance between o_i and s_j at t_k is less than d, while $BF_{t_k}[i,j] = 0$ represents that the distance between o_i and s_j at t_k is greater than d.

BF can be constructed by measuring the distance between every innocent object and contact source. To speed up the construction of BF, indexing all moving objects at each time point is a good approach. There are a few commonly used spatial indexes, such as R-tree [19], etc.

According to Definition 3, BF can be used to quickly find out the innocent objects in which one-point-contact events occur. We call such innocent objects as the candidate objects. In the query processing, only the candidate objects need to be checked to discover contact events. How to obtain the candidate objects is demonstrated in Algorithm 2.

4.2 Anchor Time Point-based Optimization

In this section, we propose an anchor time point checking strategy to filter out the time points when no contact event has occurred, which further improves query efficiency. The anchor time point is defined as follows.

Definition 7 (**Anchor Time Point (ATP).).** *An anchor time point is a special time point in $T = t_1, t_2, \ldots, t_m$. Given a sliding window width threshold τ , the set of anchor time points denoted as A is calculated according to Eq.(3).*

$$A = \begin{cases} \{a_k | a_k = t_i, i = \frac{\tau(k-1)}{2} + \lfloor \frac{\tau}{2} \rfloor\} & \tau\%2 = 0 \\ \{a_k | a_k = t_i, i = \frac{\tau k}{2}\} & \tau\%2 = 1 \end{cases} \tag{3}$$

Definition 8 (**Atomic Scanning Window (ASW)).** *Given two adjacent anchor time points a_{k-1} and a_k, the atomic scanning window of (a_{k-1}, a_k), denoted as H_k, is a set of time points between a_{k-1} and a_k.*

We can get the lemma that a sliding window must cover two adjacent anchor time points, which indicates that if a contact event occurs, there must be two one-point-contact events at two adjacent anchor time points in a sliding window. Othersize, we can also obtain the following lemma:

Given two adjacent anchor time points a_{k-1} and a_k, if there is a sliding window W covering a_{k-1} and a_k, then we have

$$W \subset < H_{k-1}, a_{k-1}, H_k, a_k, H_{k+1} >, \tag{4}$$

where $< H_{k-1}, a_{k-1}, H_k, a_k, H_{k+1} >$ is a consecutive time point sequence composed by $H_{k-1}, a_{k-1}, H_k, a_k$ and H_{k+1}.

According to Lemma 2, for two adjacent ATPs a_{k-1} and ak when one-point-contact events have both occurred, the time point sequence$< H_{k-1}, a_{k-1}, H_k, a_k, H_{k+1} >$ must cover some sliding windows, and the contact events may occur in those sliding windows. Thus, we denoted $< H_{k-1}, a_{k-1}, H_k, a_k, H_{k+1} >$ as the candidate window. To discover contact events, we have to scan all time points in the candidate window.

Following the idea above, we can check the ATPs to filter out the time points when no contact events have occurred. Meanwhile, by looking for two adjacent ATPs when one-point-contact events have both occurred, we can find out the candidate windows and further discover contact events.

```
Algorithm 2: GetCandObj($BF_{t_k}$, $O$, $S$)
Input: $O$: a set of moving objects, $d$: a distance threshold, $\tau$:
       a sliding window width threshold, $S$: an initial set of
       contact sources
Output: $C$: A set of contact events
1 foreach $o_i \in O - S$ do
2     $flag = false$
3     foreach $s_j \in S$ do
4         if $BF_{t_k}[i,j] = 1$ then
5             $flag = true$
6         if $flag = true$ then
7             C = C + { $o_i$ }
8 return $C$
```

```
Algorithm 3: MCEQ+($O$, $S$, $d$, $\tau$)
Input: $O$: a set of objects, $BF_{t_k}$: a 2-dimensional bitmap
       filter at the time point $t_k$, $S$: an initial set of contact
       sources
Output: $R$: A set of candidate objects
1  $BF, \mathcal{A}, AS$ ← Construct the sets of BFs, ATPs and ASWs
     according to Definitions 6, 7 and 8, respectively
2  $count[1, 2, \ldots, n] = 0$ ← Initialize a counter array
3  foreach $a_k \in \mathcal{A}$ do
4      $C = GetCandObj(BF_{a_k}, O - S, S)$
5      foreach $o_i \in C$ do
6          $count[i] = count[i] + 1$
7          if $count[i] = 2$ then
8              Construct a candidate window
9              $Win =< H_{k-1}, a_{k-1}, H_k, a_k, H_{k+1} >$
10             foreach $t_j \in Win$ do
11                 Perform the steps of line 5-17 in Algorithm 1
                     to determine whether a contact event $e$
                     exists
12                 if $e$ exists then
13                     $R = R + \{e\}$
14                     $S = S + \{o_i\}$
15                     break
16             $count[i] = count[i] - 1$
17 return $R$
```

Table 1: Parameter settings

Parameter	Meanings	Default values
n	the number of moving objects	5000
m	the trajectory length of moving objects	1500
d	the distance threshold	10
τ	the sliding window width threshold	12

4.3 The Optimized Query Processing Algorithm

In this section, we propose the optimized multi-source contact event query algorithm by using the 2-dimensional bitmap filter and the anchor time points scanning. The optimized algorithm is denoted as MCEQ+. The sketch of the algorithm are as follows: All ATPs are sequentially checked to find any two adjacent ATPs when one-point-contact events have both occurred, and all candidate windows are obtained. During each check on the ATPs, the 2-dimensional bitmap filter is used to filter out the innocent objects that have no one-point-contact event occurred. Then, each candidate window is scanned to discover contact events. The details of the optimized algorithm are presented in Algorithm 3.

In Algorithm 3 (MCEQ+), for the innocent object o_i, $count[i]$ records the number of consecutive ATPs when one-point-contact events have occurred. Since MCEQ+ can filter out both time points that do not need to be scanned and innocent objects that are outside the contact range of contact sources, it further accelerates the contact event query processing. We will comprehensively evaluate its performance in the subsequent experiments. The time complexity of MCEQ+ is $O(\gamma(c\beta + c\tau + \alpha\beta))$, where γ is the number of ATPs and c is the average number of candidate objects.

5 PERFORMANCE EVALUATION

In this section, we conduct a comprehensive evaluation of the proposed MCEQ and MCEQ+ algorithms using a real moving trajectory dataset, Taxi. The evaluation is based on two metrics, the number of contact events in results and the query time cost. The proposed algorithms of this paper focus on multi-source contact event discovery while the existing work Iterative-DCS focuses on single-source contact event discovery. They are different in the definition of contact events. Thus, we just compare the number of contact events obtained by these algorithms. In the experiments, we use R-tree to generate a 2-dimension bitmap filter. The parameter settings related to the experiments are shown in Table 1.

5.1 Query Result Evaluation

Experimental results for query outcome assessment omitted.

5.2 Query Time Cost Evaluation

Experimental results for query time evaluation omitted.

6 CONCLUSION

Using the trajectory of moving objects is an effective way to discover contact events. Existing methods only focused on single-source contact events. In this paper, We propose efficient multi-source contact event query processing methods. A baseline algorithm is first presented, which adopts the idea of sliding window scanning. To improve the query efficiency, the 2-dimensional bitmap filter and the anchor time point scanning are designed and used in the optimized algorithm. Experimental results demonstrate that the proposed algorithms can discover more contact events, and the optimized algorithm significantly improves the query efficiency.

REFERENCES

[1] Louai Alarabi, Saleh Basalamah, Abdeltawab Hendawi, and Mohammed Abdalla. 2021. Traceall: A real-time processing for contact tracing using indoor trajectories. *Information* 12, 5 (2021), 202.

[2] Pingfu Chao, Dan He, Lei Li, Mengxuan Zhang, and Xiaofang Zhou. 2021. Efficient trajectory contact query processing. In *Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part I 26*. Springer, 658–666.

[3] Qi Fan, Dongxiang Zhang, Huayu Wu, and Kian-Lee Tan. 2016. A general and parallel platform for mining co-movement patterns over large-scale trajectories. *Proceedings of the VLDB Endowment* 10, 4 (2016), 313–324.

[4] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. 2010. Discovery of convoys in trajectory databases. *arXiv preprint arXiv:1002.0963* (2010).

[5] Faisal Moeen Orakzai, Toon Calders, and Torben Bach Pedersen. 2019. k/2-hop: fast mining of convoy patterns with effective pruning. *Proceedings of the VLDB Endowment* 12, 9 (2019), 948–960.

[6] Sutheera Puntheeranurak, Thi Thi Shein, and Makoto Imamura. 2018. Efficient discovery of traveling companion from evolving trajectory data stream. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1. IEEE, 448–453.

[7] Yao Ruihong, Wang Fei, and Chen Shuhui. 2019. Tcod: A traveling companion discovery method based on clustering and association analysis. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–7.

[8] Haotian Wang, Jie Gao, and Min-ge Xie. 2022. Clustering of Trajectories using Non-Parametric Conformal DBSCAN Algorithm. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 451–462.

[9] Sheng Wang, Zhifeng Bao, J Shane Culpepper, Timos Sellis, and Xiaolin Qin. 2019. Fast large-scale trajectory clustering. *Proceedings of the VLDB Endowment* 13, 1 (2019), 29–42.

[10] Jianqiu Xu, Hua Lu, and Zhifeng Bao. 2020. IMO: A toolbox for simulating and querying" infected" moving objects. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2825–2828.

[11] Kai Zheng, Yu Zheng, Nicholas Jing Yuan, and Shuo Shang. 2013. On discovery of gathering patterns from trajectories. In *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, 242–253.