CoMing: A Real-time Co-Movement Mining System for Streaming Trajectories

Ziquan Fang[†], Yunjun Gao[†♯], Lu Pan[†],Lu Chen[§],Xiaoye Miao[‡], Christian S. Jensen[§]
[†]College of Computer Science, Zhejiang University, Hangzhou, China
[§]Department of Computer Science, Aalborg University, Denmark
[‡]Center for Data Science, Zhejiang University, Hangzhou, China
[♯]Alibaba–Zhejiang University Joint Institute of Frontier Technologies, Hangzhou, China
[†‡]{zqfang, gaoyj, panlu96, miaoxy}@zju.edu.cn [§]{luchen, csj}@cs.aau.dk

ABSTRACT

The aim of real-time co-movement pattern mining for streaming trajectories is to discover co-moving objects that satisfy specific spatio-temporal constraints in real time. This functionality serves a range of real-world applications, such as traffic monitoring and management. However, little work targets the visualization and interaction with such co-movement detection on streaming trajectories. To this end, we develop **CoMing**, a real-time co-movement pattern mining system, to handle streaming trajectories. CoMing leverages **ICPE**, a real-time distributed co-movement pattern detection framework, and thus, it has its capacity of good performance. This demonstration offers hands-on experience with CoMing's visual and user-friendly interface. Moreover, several applications in the traffic domain, including object monitoring and traffic statistics visualization, are also provided to users.

KEYWORDS

Co-movement Pattern; Trajectory; Visualization; System

ACM Reference Format:
Ziquan Fang[†], Yunjun Gao[†♯], Lu Pan[†],Lu Chen[§],Xiaoye Miao[‡], Christian S. Jensen[§]. 2020. CoMing: A Real-time Co-Movement Mining System for Streaming Trajectories. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3318464.3384703

1 INTRODUCTION

With the proliferation of GPS-equipped devices, massive and increasing volumes of trajectory data that capture the movements of humans and vehicles are being collected. A deep insight into this data is useful in a wide range of application areas [2, 4]. One important type of trajectory analysis is the co-movement pattern mining, which aims to discover the co-moving objects that satisfy specific spatio-temporal constraints. Recently, we have proposed a real-time distributed co-movement pattern mining framework called ICPE [1], to overcome the limitations that previous work only targets offline pattern detection on historical trajectories. However, little work puts attention on the visualization and interaction of co-movement pattern detection for streaming trajectories, making it hard for users to observe and understand the mining process. In addition, the visualization of real-time co-movement pattern mining is important in traffic scenarios. For example, vehicles (e.g., taxis and buses) send their current locations to the backend periodically, thus enabling traffic congestion monitoring and real-time traffic scheduling. Assume that an unexpected event such as an accident occurs, the real-time pattern detection is able to detect new patterns immediately, which is useful in traffic management.

We demonstrate CoMing, a real-time co-movement pattern mining system for streaming trajectories, which can be employed in a variety of real-world applications as below.

Real-time Co-movement Mining. CoMing detects the co-movement patterns in four steps. First, it formats and cuts the input trajectory streams into snapshot streams using window operations. Second, it matches trajectory points to the road network and builds a two-layered GR-index in the Flink system, in order to accelerate the subsequent Range-Join and clustering. Third, it performs clustering for each snapshot based on the Range-Join and the DBSCAN method [3] to obtain cluster snapshot streams. Finally, it utilizes pattern enumeration algorithms to obtain all valid co-movement patterns when each cluster snapshot arrives. Moreover, CoMing can also visualize the results on a map interactively.

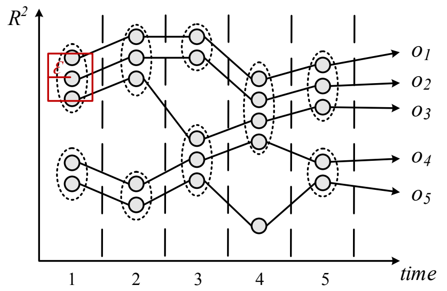


Figure 1: Example of Co-movement Pattern

Object Monitoring. Given a set of moving objects (e.g., taxies), CoMing can track their latest spatial locations dynamically on a map in real time, which is useful in transportation monitoring. In addition, offline trajectory visualization is also provided for users to enable traffic analytics.

Traffic Statistics. Traffic statistics are helpful in traffic management scenarios. CoMing can display hot areas in real time based on the clustering results obtained as part of its mining process. Besides, CoMing supports periodic traffic statistics on trajectories, which is useful in urban planning.

2 BACKGROUND

In this section, we proceed to detail the real-time co-movement detection problem with formal definitions and examples.

Definition 2.1. **(Time Sequence & Segment)**. Let $\mathbb{T} = \{1, 2, ..., \mathbb{N}\}$ be a discretized time dimension. A time sequence T is defined as a sequence of elements from $\mathbb{T}$. A sequence T is called a segment if $\forall 1 \leq i < |T|$, $(T[i+1] = T[i] + 1)$.

For example, $T = \langle 1, 2, 4, 5\rangle$ is a time sequence but not a segment, because time 3 is missing in T.

Definition 2.2. **(L-consecutive)**. A time sequence T is L-consecutive, if T is a segment and $|T| \geq L$.

Definition 2.3. **(G-connected)**. A time sequence T is G-connected if the gap between any neighboring times is no greater than G, i.e., $\forall 1 \leq i \leq |T| - 1$, $(T[i+1] - T[i] \leq G)$, where $T[i]$ denotes the i-th element in T.

For instance, $T = \langle 1, 2, 4, 5\rangle$ is 2-consecutive and 2-connected. Specifically, there are two segments $T_1 = \langle 1, 2\rangle$ and $T_2 = \langle 4, 5\rangle$ in T, and the length of each segment is no smaller than 2. Thus, T is 2-consecutive. Since $\forall 1 \leq i \leq 3$, $(T[i+1] - T[i] \leq 2)$, T is 2-connected according to Definition 2.3.

Definition 2.4. **(Co-movement Pattern)**. Given a set ST of trajectories, a subset O of ST is a co-movement pattern $CP(M, K, L, G)$ if a time sequence T exists that satisfies the following constraints: (i) **closeness:** the locations of trajectories in O belong to the same cluster at every time of T; (ii) **significance:** $|O| \geq M$; (iii) **duration:** $|T| \geq K$; (iv) T is L-**consecutive**; and (v) T is G-**connected**.

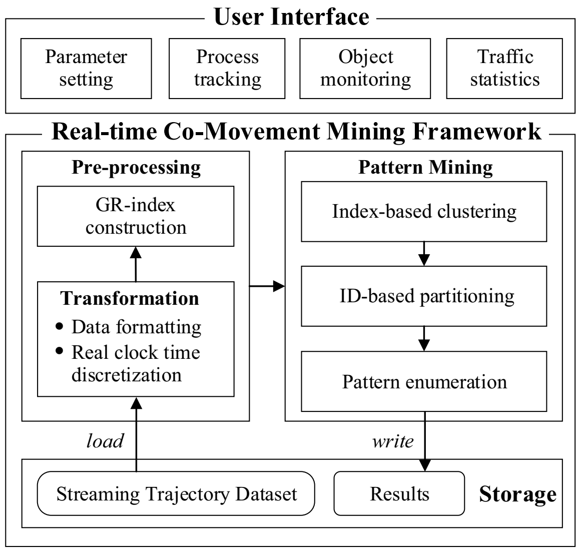


Figure 2: CoMing Architecture

We use the popular clustering method DBSCAN for the first constraint (i.e., closeness), and more details can refer to [1]. Considering the example in Fig. 1, a dotted circle denotes a cluster. If $M = 3, K = 4, L = 2$, and $G = 2$, $O = \{o_1, o_2, o_3\}$ is a co-movement pattern. Specifically, for $T = \langle 1, 2, 4, 5\rangle$, the following holds: (i) o_1, o_2, and o_3 belong to the same cluster at times 1, 2, 4, and 5; (ii) $|O| = 3$; (iii) $|T| = 4$; (iv) T is 2-consecutive; and (v) T is 2-connected.

Definition 2.5. **(Streaming Trajectory)**. A streaming trajectory is an unbounded ordered sequence of GPS location records generated by a moving object o.

Definition 2.6. **(Snapshot)**. A snapshot S_t contains all the locations of all moving objects at time t.

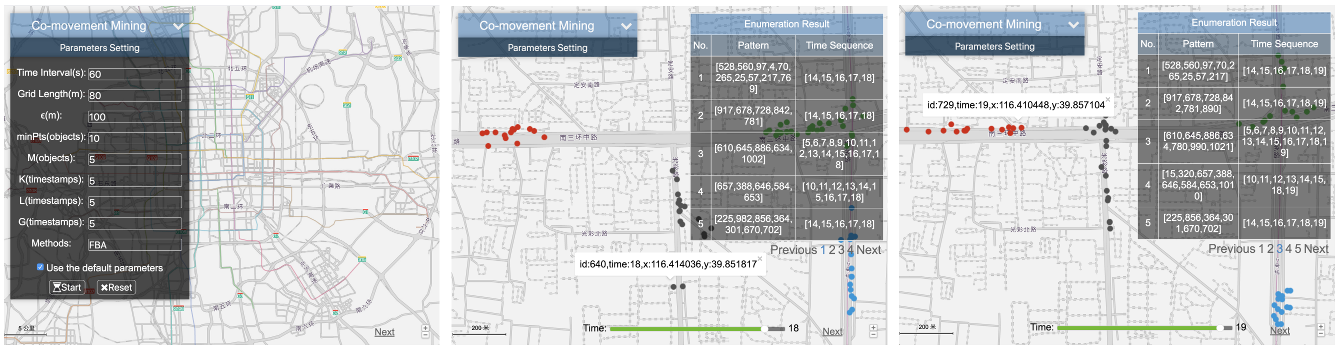
In Fig. 1, o_1 to o_5 are streaming trajectories. For simplicity, we use $\{o_1, o_2, ..., o_n\}$ to represent their spatial locations, and there are five snapshots $(i, \{o_1, o_2, ..., o_n\})$ $(1 \leq i \leq 5)$.

Definition 2.7. **(Real-time Co-movement Pattern Mining)**. Given parameters M, K, L, and G that define a general co-movement pattern, real-time co-movement pattern mining finds all co-movement patterns in the snapshot set $S = \{S_1, S_2, ..., S_t\}$, where t is the current time.

As an example, if the current time is 4, $\{o_1, o_2\}$ is a CP(2, 4, 2, 2) pattern, where $T = \langle 1, 2, 3, 4\rangle$. However, no CP(3, 4, 2, 2) pattern exists until time 5, i.e., $\{o_1, o_2, o_3\}$ becomes a co-movement pattern with $T = \langle 1, 2, 4, 5\rangle$.

3 SYSTEM OVERVIEW

The CoMing architecture is shown in Fig. 2. It contains two parts: (i) a real-time mining framework and (ii) a user-friendly interactive interface, that makes the real-time co-movement pattern detection transparent to users.



Time Interval(s): 60
Grid Length(m): 80
ϵ(m): 100
minPts(objects): 10
M(objects): 5
K(timestamps): 5
L(timestamps): 5
G(timestamps): 5
Methods: FBA
☑ Use the default parameters

Enumeration Result

No.	Pattern	Time Sequence
1	[528,560,97,4,70,265,25,57,217,769]	[14,15,16,17,18]
2	[917,678,728,842,781]	[14,15,16,17,18]
3	[610,645,886,634,1002]	[5,6,7,8,9,10,11,12,13,14,15,16,17,18]
4	[657,388,646,584,653]	[10,11,12,13,14,15,16,17,18]
5	[225,982,856,364,301,670,702]	[14,15,16,17,18]

Enumeration Result

No.	Pattern	Time Sequence
1	[528,560,97,70,265,25,57,217]	[14,15,16,17,18,19]
2	[917,678,728,842,781,890]	[14,15,16,17,18,19]
3	[610,645,886,634,780,990,1021]	[5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
4	[15,320,657,388,646,584,653,1010]	[10,11,12,13,14,15,18,19]
5	[225,856,364,301,670,702]	[14,15,16,17,18,19]

(a) Parameter setting (b) Detected result at timestamp 18 (13:18:00) (c) Detected result at timestamp 19 (13:19:00)

Figure 3: Pattern Detection Visualization

First, we give an overview of the real-time pattern mining framework using the example in Fig 1. The framework features a pipelined modular architecture consisting of two phases, i.e., *pre-processing* and *pattern mining*. The *pre-processing* contains transformation and GR-index construction modules, and the *pattern mining* contains index-based clustering, ID-based partioning, and pattern enumeration modules. A detailed framework description can be found in [1].

Transformation discretizes the real clock times of trajectory points to timestamps, which are the indices of the time intervals during which they occurred. For instance, assume that an interval duration of 5s is used for discretization and that the start time is 13:00:20 UTC. Then, the time series ⟨13:00:21 UTC, 13:00:24 UTC, 13:00:28 UTC, 13:00:32 UTC, 13:00:42 UTC⟩ is mapped to $\langle 0, 0, 1, 2, 4\rangle$. Moving objects with the same timestamp are windowed into the same snapshot. So far, the raw trajectory streams have been transformed into the snapshot streams. As depicted in Fig. 1, i.e., a snapshot at time 1, a snapshot at time 2, and so on.

GR-index construction builds a GR-index for each snapshot to accelerate the Rang-Join used in DBSCAN clustering. It uses a grid as a global index, and builds an R-tree as a local index for each grid cell. The detailed design of this two-layered index can refer to [1] due to space limitation.

Index-based clustering first computes the Range-Join using the GR-index of each snapshot to obtain the neighbor streams, where several lemmas are developed to avoid unnecessary query costs. Then, DBSCANs are performed in parallel on neighbor streams to get the clustering snapshots. For example, as shown in Fig. 1, given a fixed ϵ, $minPts = 2$ and current time 2, we get clustering snapshot at 1 (including cluster 0: $\{o_1, o_2, o_3\}$, cluster 1: $\{o_4, o_5\}$) and clustering snapshot at 2 (including cluster 0: $\{o_1, o_2, o_3\}$, cluster 1: $\{o_4, o_5\}$). These clustering snapshots form the cluster streams.

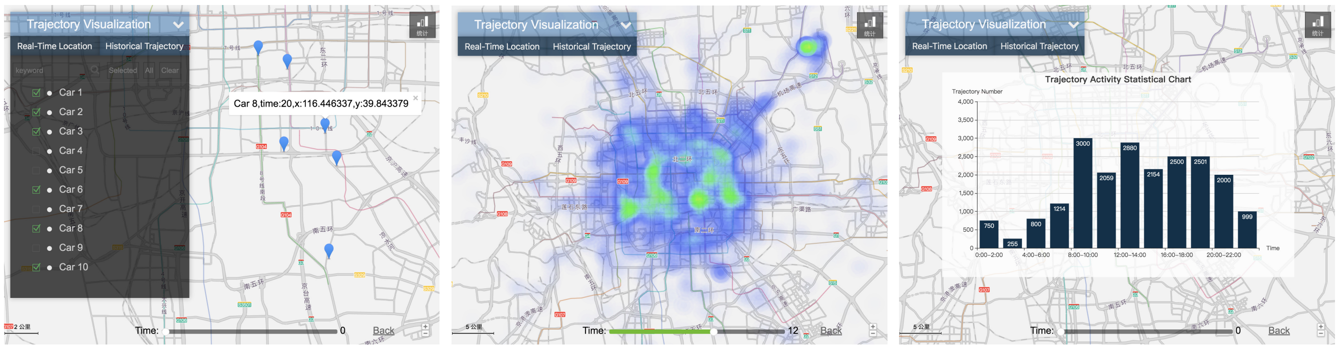
ID-based partitioning creates a Flink subtask for each trajectory o. We use partition $P_t(o)$ to denote the set of trajectories distributed to subtask $o.id$ at time t. In order to avoid duplications, $P_t(o)$ contains trajectories (except for o) in the same cluster with their ids larger than $o.id$. Note that, at different times t_i, partitions $P_{t_i}(o)$ will be sent to same subtask $o.id$ for processing. For example in Fig. 1, system creates 5 subtasks for 5 trajectories. At time 1, partitions include $P_1(o_1) = \{o_2, o_3\}$ for subtask 1, $P_1(o_2) = \{o_3\}$ for subtask 2, $P_1(o_3) = \emptyset$ for subtask 3, $P_1(o_4) = \{o_5\}$ for subtask 4, and $P_1(o_5) = \emptyset$ for subtask 5. At time 2, $P_2(o_1) = \{o_2, o_3\}$ will be sent to subtask 1 to process with $P_1(o_1) = \{o_2, o_3\}$.

Pattern enumeration enumerates all possible combinations of trajectories for each partition $P_t(o)$ at time t. Considering the example in Fig. 1, given a partition $P_1(o_1) = \{o_2, o_3\}$ and $M = 2$, the possible patterns include $\{o_2\}$, $\{o_3\}$, and $\{o_2, o_3\}$, where o_1 is a common element, and is omitted in the patterns. Next, we determine whether each pattern O enumerated in $P_t(o)$ is valid. Specifically, for a pattern O, T is first initialized as $\{t\}$. If O also exists in $P_{t'}(o)$ at next time t', then $T = T \cup \{t'\}$. If T satisfies the K, L, and G constraints in Definition 2.4, then O is valid and output. Moreover, two more efficient methods (i.e., FBA and VBA) based on bit-compression and candidate-based enumeration are presented. The former offers better latency, the later provides better throughput, thus providing more options for users.

In the next, we cover each module in the user interface.

Parameter setting requires users to input the time intervals for streaming trajectory transforming. In addition, the clustering parameters (i.e., grid length, ϵ, and *minPts*) and enumeration parameters (M, K, L, and G) are also needed to be set for the real-time co-movement pattern detection.

Process tracking can help users to track the mining process by means of three components, i.e., a progress map, a progress window, and a time bar. The time bar indicates the current timestamp, the progress map shows the clustering results, and the progress window outputs the valid co-movement patterns at current timestamp. In addition, the co-movement pattern results will be written to the storage layer during the process for the future analysis.



(a) Object Monitoring (b) Hotspot Detection (c) Period Statistics

Figure 4: Streaming Trajectory Data Analysis

Object monitoring enables users to monitor the latest positions of selected moving objects in real time. In addition, users can retrieve the complete trajectories by inputting their IDs and a time period, which is useful in many applications such as transportation monitoring.

Traffic statistics provides two kinds of statistics helping users to analyze streaming trajectories visually in urban computing. The first one displays hot areas dynamically based on the clustering results in the form of heat maps. The second provides moving information (i.e., the number of moving taxis) in different time periods by histograms.

4 DEMONSTRATION

The CoMing system is built as a cross-platform web-based application. The basic interface is shown in Figure 3. We demonstrate CoMing using a real trajectory dataset generated by 10,357 taxis in Beijing, China, containing 15 million trajectory points, with the following interactive scenarios.

Parameter Setting. First, users are required to input the time interval, which is used to transform the trajectory streams into snapshots. As depicted in Fig. 3(a), if we set the interval to 60 seconds, taxis will send their location to the backend server every minute. Next, users can set the clustering parameters (i.e., the grid length (meters), ϵ (meters), and $minPts$ (objects)) and enumeration parameters (i.e., M (objects), K (time), L (time), and G (time)) used for the pattern detection. In Fig. 3(a), grid length is set to 80, ϵ is set to 100, $minPts$ is set to 10, while M, K, L and G are set to 5.

Process Tracking. Having completed the parameter setting, users can choose a particular detection method (i.e., FBA or VBA), and then press the "Start" button to start the pattern detection. The detection is dynamic and automatic, which includes clustering and enumeration. The real-time clustering results are displayed on the map, and the enumeration results are listed in the right window, as shown in Fig. 3(b) and Fig. 3(c), respectively. Also, there is a time bar at the bottom showing the current timestamp that is proceeded, and the detected patterns are written to the storage layer.

Object Monitoring. CoMing provides two kinds of object monitoring operations, as illustrated in Fig. 4(a), including the real-time and the offline monitoring. For the real-time monitoring, users select the cars in the mutable options on the left, and CoMing shows their latest locations on the map in real time. For the offline monitoring, users can input a time period (i.e., a start time and an end time), and then, CoMing shows the complete trajectories of selected taxis on the map during the time period. In addition, if a user is interested in a particular trajectory, the user can click on the location point or the trajectory, which is highlighted with the detailed information depicted in a panel. Fig. 4(a) only shows the real-time object monitoring due to space limitation.

Traffic Statistics. Users can drag the time bar at the bottom to detect hot spots at different times in the form of heat maps, as depicted in Fig. 4(b). In addition, CoMing also offers period traffic statistics function to show the number of taxis in different time periods (e.g., different hours in one day) for trajectory data analysis, as illustrated in Fig 4(c).

Acknowledgments. This work was supported in part by the National Key R&D Program of China under Grant No. 2018YFB1004003, the NSFC under Grant No. 61972338, the NSFC-Zhejiang Joint Fund under Grant No. U1609217, and the ZJU-Hikvision Joint Project. Yunjun Gao is the corresponding author of the work.

REFERENCES

[1] Lu Chen, Yunjun Gao, Ziquan Fang, Xiaoye Miao, Christian S. Jensen, and Chenjuan Guo. 2019. Real-time Distributed Co-Movement Pattern Detection on Streaming Trajectories. *PVLDB* 12, 10 (2019), 1208–1220.

[2] Lei Duan, Tinghai Pang, Jyrki Nummenmaa, Jie Zuo, Peng Zhang, and Changjie Tang. 2018. Bus-OLAP: A Data Management Model for Non-on-Time Events Query Over Bus Journey Data. *Data Science and Engineering* 3, 1 (2018), 52–67.

[3] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. 2008. Discovery of Convoys in Trajectory Databases. *PVLDB* 1, 1 (2008), 1068–1080.

[4] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: Distributed In-Memory Trajectory Analytics. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD).* 725–740.