Querying Continuous Recurrent Convoys of Interest

Munkh-Erdene Yadamjav
RMIT University
Melbourne, Australia
munkh-erdene.yadamjav@rmit.edu.au

Zhifeng Bao
RMIT University
Melbourne, Australia
zhifeng.bao@rmit.edu.au

Farhana M. Choudhury
The University of Melbourne
Melbourne, Australia
farhana.choudhury@unimelb.edu.au

Hanan Samet
University of Maryland
College Park, Maryland 20742
hjs@cs.umd.edu

Baihua Zheng
Singapore Management University
Singapore
bhzheng@smu.edu.sg

ABSTRACT

Moving objects equipped with location-positioning devices continuously generate a large amount of spatio-temporal trajectory data. An interesting finding over a trajectory stream is a group of objects that are travelling together for a certain period of time. Existing studies on mining co-moving objects do not consider an important correlation between co-moving objects, which is the reoccurrence of the movement pattern. In this study, we define a problem of finding recurrent pattern of co-moving objects from streaming trajectories and propose an efficient solution that enables us to discover recent co-moving object patterns repeated within a given time period. Experimental results on a real-life trajectory database show the efficiency of our method.

CCS CONCEPTS

• **Theory of computation** → *Data structures and algorithms for data management*; • **Information systems** → *Data stream mining*.

KEYWORDS

recurrent convoy query, co-moving pattern, spatio-temporal index

ACM Reference Format:
Munkh-Erdene Yadamjav, Zhifeng Bao, Farhana M. Choudhury, Hanan Samet, and Baihua Zheng. 2019. Querying Continuous Recurrent Convoys of Interest. In *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '19), November 5–8, 2019, Chicago, IL, USA*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3347146.3359083

1 INTRODUCTION

With the prevalence of location-positioning devices, a vast amount of spatio-temporal data of moving objects is being generated. Systematically analysing trajectory data of moving objects enables us to extract a variety of interesting patterns and knowledge that can lead to many real-life applications such as transportation analysis and urban computing.

SIGSPATIAL '19, November 5–8, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6909-1/19/11.
https://doi.org/10.1145/3347146.3359083

One interesting finding in trajectory databases is the exploration of convoys [5] that occur recurrently. Informally, a *convoy* refers to a group of spatially close-by objects moving together for a specific period of time. In essence, a convoy of interest is defined by the number of objects (τ) and the time duration of moving together (k), where τ and k are user-specified parameters.

A number of variations of the convoy have been proposed in the literature which consider both offline [2, 3, 6, 9, 13] and online [1, 7, 11, 14] trajectory data processing. The definitions and the techniques to mine patterns of co-moving objects vary depending on the parameters and scenarios of consideration. However, the existing literature on mining patterns of co-moving objects treats each mined pattern independently by ignoring the possible correlations between patterns. Moreover, in many real-life applications such as military surveillance, urban emergency response systems, and transport mangement systems, an immediate analysis is required for incoming trajectory data [14]. For example, a real-time military surveillance system can detect a co-moving pattern of suspicious group with frequent occurrences. Transportation application can detect abnormal traffic congestions caused by accidents by distinguishing them from recurring traffic delays.

Motivated by the aforementioned observation, we study a new problem of querying a *Continuous Recurrent Convoy of Interest*. Informally, a sequence of similar convoys forms a recurrent convoy. The recurrent convoy of interest varies depending on the recurrence parameter ρ, which is defined by the time interval between two successive convoy occurrences. Mining recurrent convoys in a streaming case helps us to distinguish the convoys emerging recently from the ones that occur recurrently.

The significance of a recurrent convoy varies w.r.t. the parameters, such as the number of objects that form the convoy ***(prominence)***, the duration of the convoy ***(timespan)***, and the time interval of the convoy repetition ***(recurrence)***. Values of the parameters that define the interestingness of the convoy may vary over time as we accumulate more data. Thus, the exploration of recurrent convoys of interest is an iterative process as the distribution of the parameters' values is not uniform across the whole search space. Setting appropriate values as a query input gives us a new insightful explanation about the dataset. Thus, it is crucial to facilitate the mining effort as the mining task that has a one-off parameter setting might not achieve the goal of extracting all interesting convoys.

In this paper, given a sliding time window and thresholds for the prominence, timespan, and recurrence, we study the problem

of finding recurrent convoys in a sliding window that satisfy the given thresholds. Our main focus is to propose a general approach to compute the similarity between convoys and store them in a structure that facilitates the mining effort.

The main challenges in identifying recurrent convoys in a trajectory database are three-fold. **First**, The use of common objects to find the co-moving patterns in most of the related studies is not suitable to find the correlation between convoys. **Second**, The number of occurences of each resulting convoy in a sliding windows varies w.r.t. the recurrence threshold. Thus, an effective filtering structure that only retrieves potential candidate convoys is required to speed up the mining task. **Third**, Finding convoys of interest is an iterative process of exploration rather than a task of one-off parameter settings.

In order to address the above three challenges, we make the following contributions: (i) we define a problem which considers timespan, prominence, and recurrence of a convoy all together for the first time to find recurrent convoys of interest over a trajectory database; (ii) we propose an indexing structure that organizes clusters in an effective and efficient way to mine recurrent convoys; and (iii) we conduct an experiment on a real-life dataset to evaluate the efficiency of our method.

Related work. Many studies [2, 3, 5, 8, 9, 13] on co-moving pattern mining have been proposed in the literature. Two main parameters that define a co-moving pattern are: (i) the timespan of a pattern; and (ii) the number of objects that constitute a pattern. Existing studies introduced additional constraints on top of those two parameters, such as different spatial clustering techniques, local temporal consecutiveness, and temporal gap. Objects at each timestamp of a co-moving pattern are contained in a circle with a pre-defined radius in [3] while the density-based clustering is used in other work. Timestamps in swarm [9] are not necessarily consecutive. A local temporal consecutiveness threshold is introduced to allow a temporal gap in the co-moving pattern [2, 8]. A moving cluster [6] does not often contain the same objects during its timespan. Thereby, a threshold for the common objects of two consecutive clusters is used to mine moving clusters. Zheng et al. [13] proposed a problem of finding gathering patterns where a set of objects called dedicated members travels for at least a certain period of time.

The streaming case of finding co-moving patterns for a trajectory database has been considered in [1, 7, 11, 14]. However, the definitions of the co-moving pattern in these work all differ. Thus, the proposed algorithm only mines particular co-moving patterns. Moreover, objects are required to record their locations at every timestamp during the whole database timespan in [11, 14] while we consider an object to record locations at every timestamp during its time interval.

None of existing work can be used directly to find recurrent convoys of interest which is a combination of online and historic convoy generations.

2 PROBLEM FORMULATION

In this section, we first present the necessary preliminaries and then give the formal problem definitions.

Given a set of moving objects $O = \{o_1, o_2, \ldots, o_n\}$ in a trajectory database with the time domain $\mathcal{T} = \{t_1, t_2, \ldots, t_\infty\}$, a trajectory of moving object $o \in O$ is represented as a finite sequence of location samples within the time interval $[t_i, t_j]$, i.e., $o = loc_i, loc_{i+1}, \ldots, loc_j$, where loc_a is a recorded position of o in a two-dimensional space at timestamp t_a. We assume the trajectory of the object o is recorded at every timestamp during its lifetime $[t_i, t_j]$. Trajectories may have varying lengths.

Let $C = \{C^{t_1}, C^{t_2}, \ldots, C^{t_m}\}$ be the overall set of clusters generated by applying a chosen clustering algorithm over the trajectory database. Here, $C^t = \{c_1^t, c_2^t, \ldots, c_l^t\}$ represents the set of clusters obtained at timestamp t, where c^t is a cluster of a non-empty subset of objects in O that satisfies the clustering conditions. Since we do not require recording the location of each object at every timestamp throughout the trajectory database timespan, there can be no cluster associated with some timestamps. The time-based query sliding window W_I of length I shifts at a time.

The goal of our approach is to find recurrent convoys that satisfy the thresholds given by the user. Towards this goal, we first give the definitions of a *recurrent convoy* by adopting the definition of the *convoy* [5].

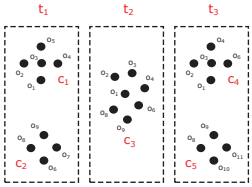
Definition 1. ***(Convoy).*** Given a set of clusters C and thresholds for prominence (τ) and timespan (k), a convoy $g = \{c^{t_i}, c^{t_{i+1}}, \ldots, c^{t_j}\}$ is defined as a sequence of clusters at consecutive timestamps that satisfies the following constraints: (i) $\forall c^{t_a} \in g, \exists C^{t_a} \in C$ such that $c^{t_a} \in C^{t_a}$; (ii) the number of common objects shared by all clusters, denoted as $g.\tau$, is no less than τ, i.e., $g.\tau = |c^{t_i} \cap c^{t_{i+1}} \cap c^{t_j}| \geq \tau$; and (iii) the time duration of the convoy, denoted as $g.k$, is no less than k, e.g., $g.k = |t_j - t_i + 1| \geq k$, where $k > 1$.

Next, we define the similar convoys w.r.t. the thresholds of interest in Definition 2. Function $\textsc{similar}(c_1, c_2)$ is to measure the similarity between two object clusters c_1 and c_2. There are multiple similarity metrics available to quantify the similarity between two object clusters and we assume that the selection of the similarity metric is application dependent. For illustration purpose, we adopt the Hausdorff distance [10] to compute the similarity between clusters in convoys, similar to the trajectory pattern mining work by Zheng et al. [13]. Nonetheless, the problem definition and our approach could be easily adjusted to other similarity metric.

Definition 2. ***(Similar convoys).*** *Given the thresholds τ, δ and k and two convoys $g_a = \{c^{t_i}, c^{t_{i+1}}, \ldots, c^{t_{i+u-1}}\}$ and $g_b = \{c^{t_j}, c^{t_{j+1}}, \ldots, c^{t_{j+v-1}}\}$, convoy g_a is similar to convoy g_b w.r.t. τ and k iff (i) $MIN(g_a.\tau, g_b.\tau) \geq \tau$; (ii) $MIN(g_a.k, g_b.k) \geq k$; and (iii) $\exists g'_a = \{c^{t_a}, c^{t_a+1}, \ldots, c^{t_a+k-1}\} \subseteq g_a$, $\exists g'_b = \{c^{t_b}, c^{t_b+1}, \ldots, c^{t_b+k-1}\} \subseteq g_b$ such that $\forall l \in [0, k-1]$, $\textsc{similar}(g'_a.c^{t_a+l}, g'_b.c^{t_b+l}) \leq \delta$.*

Now we are ready to introduce recurrent convoys in Definition 3, and present the *Continuous recurrent convoy* query in Definition 4 that takes varying parameters as an input and only considers convoys in the sliding window (note, we skip parameter δ in Definition 4).

Definition 3. ***(ρ-Recurrent Convoy).*** Given a sequence of convoys $p_{i,j} = \{g_i, \ldots, g_j\}$ and a recurrence threshold ρ, $p_{i,j}$ is a *ρ-recurrent convoy* iff $\forall a \in [i, j-1]$, two successive convoys g_a and g_{a+1} are similar and the difference between their starting timestamps is less than ρ, i.e., $|g_a.t_s - g_{a+1}.t_s| \leq \rho$ where $g.t_s$ denotes the starting timestamp of convoy g.



(a) Running example

Cluster	Timestamp	# of objects	Previous cluster information
c_3	t_2	4	c_1:{4}{o_1,o_2,o_3,o_4} c_2:{3}{o_6,o_8,o_9}
c_4	t_3	5	c_3:{5}{o_1,o_2,o_3,o_4,o_6}

(b) Intersection index ($\tau_{min} = 3$)

Convoy	Cluster	Timestamp	# of objects	Previous cluster information
g_1	c_3	t_2	4	c_1:{4}{o_1:2,o_2:2,o_3,:2,o_4:2} c_2:{3}{o_6,:2,o_8:1,o_9:1}
g_1	c_4	t_3	5	c_3:{5}{o_1,o_2,o_3,o_4,o_6}

(c) Convoy Index ($\tau_{min} = 3$)

Figure 1: Index structure

Definition 4. ***(Continuous Recurrent Convoy (CRC) query).*** Given a trajectory database that is continuously updated and a current sliding window of length I, the $CRC\langle k, \tau, \rho\rangle$ query finds a set of recurrent convoys $\mathcal{P}$, where $\forall p_i = \{g_a, \ldots, g_b\} \in \mathcal{P}$ satisfies the recurrence threshold ρ, $\forall g_b$ is within the sliding window, and $\forall g_j \in p_i$ is a valid convoy w.r.t. k and τ.

3 METHODOLOGY

In this section, we first outline the baseline steps to answer the *recurrent convoy* query and we then present two enhancements to further improve the search performance.

Baseline. A general approach to answer the *recurrent convoy* query consists of three phases: (i) *timestamp clusters generation* that generates clusters using object locations at the current timestamp; (ii) *convoy discovery* that either expands existing convoy candidates in a sliding window based on newly generated clusters or generates new convoy candidates from newly generated clusters; and (iii) *recurrent convoy discovery* that searches for the historical occurrences of each convoy in a sliding window that satisfies the given thresholds.

Timestamp Clusters Generation. Once we receive a set of objects whose locations are recorded at the current timestamp, we apply the chosen clustering algorithm on the object set to generate the object clusters. Clusters that are obtained after the clustering step may have different sizes. Since τ is user-specified and unknown a priori, we define a parameter $\tau_{min} \leq \tau$ which serves as a lower bound of τ to enable the query processing. Clusters with at least τ_{min} objects are indexed using the traditional R-tree [4] structure.

Convoy Discovery. As the clusters in a convoy [5] are consecutive in time, the incoming clusters at timestamp t are only compared to the candidate convoys that have the clusters at timestamp $t-1$. We are only interested in a convoy that contains at least τ number of common objects throughout its timespan. Thus, the incoming clusters are filtered by the number of objects and only the clusters that satisfy τ threshold are checked against the candidate convoys. However, all clusters are passed to the indexing step regardless of the number of objects inside.

Recurrent Convoy Discovery. Once we find a set of convoys of interest that fall inside the sliding window, we search for the previous occurrences of each convoy by checking historical clusters stored in the index w.r.t. the given thresholds. As we search for recurrence of each convoy within ρ time period, we load clusters within the time interval of length ρ at each iteration. Convoys are created using the same procedure that we described previously based on the clusters w.r.t. the thresholds. The similarity between historical convoys and convoys in a sliding window is computed using the given thresholds. Convoys in a sliding window that are extended by historic convoys retrieved within the time interval of length ρ are sent to next iteration to check for another occurrence.

Discussion. The computational overhead of the baseline to generate convoys is substantially high as the clusters at the current timestamp need to be checked against all candidate convoys that end at the previous timestamp. We propose two enhancements with corresponding index structures over the baseline to accelerate the *recurrent convoy* query processing.

Recurrent Convoy over Intersection index (RCI). A cluster with at least τ objects is not guaranteed to share at least τ objects with any cluster at the next timestamp. To avoid checking each cluster against all candidate convoys, we propose an *Intersection index* that considers the overlap in terms of objects between clusters at consecutive timestamps. A cluster node in the idxi stores a set of clusters at the previous timestamp that share at least τ_{min} common objects with the cluster (shown in Figure 1b where $\tau_{min} = 3$) as data embedded in the node. This index has two advantages over the baseline index that stores clusters independently. First, it eliminates the extra check of two clusters at consecutive timestamps in the stage of convoy discovery if they do not share τ common objects. Second, each retrieved cluster is only checked against the convoys that end with the cluster in an embedded set of the previous timestamp.

Recurrent Convoy over Convoy index (RCC). The two common scenarios of convoys are *converge* and *diverge*. As shown in Figure 1a, objects in clusters c_1 and c_2 converge into cluster c_3 from t_1 to t_2 while objects in cluster c_3 diverge into two clusters from t_2 to t_3. Based on this observation, we propose a *Convoy index* which groups incoming clusters into a set of distinctive convoys with each cluster assigned to only one convoy. We compute the number of consecutive appearances of each object in a cluster for a sequence of clusters that belongs to the same convoy. The index convoys are generated w.r.t. the parameter τ_{min}. During the historical convoy search, the *Convoy index* retrieves clusters that satisfy the time interval and the number of common objects. A sequence of retrieved clusters with the same convoy identifier is merged without performing actual intersections between clusters.

4 EXPERIMENTS

In this section, we evaluate the performance of our proposed algorithms via experiments using a real dataset.

4.1 Experimental Settings

All algorithms are implemented in JAVA. Experiments were ran on a 24 core Intel Xeon E5-2630 2.3 GHz using 256GB RAM, and 1TB 6G SAS 7.2K rpm SFF (2.5-inch) SC Midline disk drives running Red Hat Enterprise Linux Server release 7.5. We test the following methods to answer Continuous recurrent convoy queries on a real-life

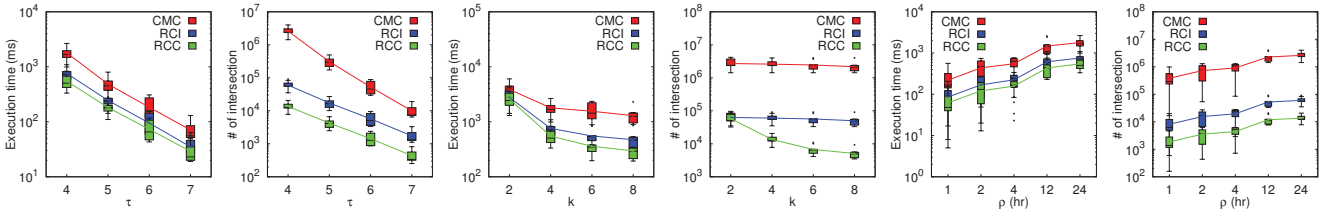


Figure 2: Effect of Varying Parameters on T-drive Dataset

dataset: (1) **CMC:** Coherent Moving Cluster algorithm [5] on top of the R-tree index, (2) **RCI:** recurrent convoy algorithm based on the *Intersection index*, and (3) **RCC:** recurrent convoy algorithm based on the *Convoy index*. Note CMC refers to the baseline approach and RCI and RCC refer to the two enhancements proposed in Section 3.

Datasets. Experiments were conducted using the T-drive dataset[12]. The T-drive dataset contains the raw trajectory information of 10,357 taxis in Beijing, China collected for a week in Feb 2008. Each trajectory in the dataset is a sequence of GPS locations with the corresponding timestamps. We obtained 455,891 clusters with 2,048,088 points by running DBSCAN with the parameter settings of $m = 4, \epsilon = 100m$ over the dataset. τ_{min} was set to 4.

Parameter	Description	Values
τ	# of objects	**4**, 5, 6, 7
k	Timespan	2 **4**, 6, 8
ρ	Recurrence(hr)	1, 2, 4, 12 , **24**

Table 1: Experimental parameters

Evaluation and Parameterization. We compare the performance of the baseline and two enhancements by varying the query input parameters as shown in Table 1, where the values in bold represent the default values.

4.2 Efficiency Study

We compare the impacts of each parameter by running 100 queries and report the average query execution time and the average number of intersections between convoy candidates.

Effect of τ. The effect of the parameter that controls the number of objects in a convoy on the query performance is presented in Figure 2a. As we search for larger convoys by increasing the threshold τ, the overall query execution time decreases due to the distribution of objects in the clusters. Both RCI and RCC perform up to three times faster than the baseline. The performance gap between RCI and RCC shows a small margin as shown in Figure 2b. RCC computes up to one order of magnitude fewer intersections than RCI. However, there is no substantial difference in the query performance which is likely due to the distribution of the convoy sizes and lengths.

Effect of k. The effect of the parameter that controls the duration of the convoys on the query performance is presented in Figure 2c. As we increase the threshold k for the convoy, the number of historic convoys that satisfy the threshold declines, resulting in a smaller number of clusters to be retrieved from the index. RCI and RCC perform up to five times faster than the baseline when varying timespan thresholds. The margin between RCI and RCC increases with the increase of k leading to fewer intersections as shown in Figure 2d. Longer convoys have a higher chance of using information in the *Convoy index* instead of performing intersections. This confirms that the convoy index works *better for searching convoys that last for longer timespans.*

Effect of ρ. The effect of the parameter that controls the recurrence of the convoys on the query performance is presented in Figure 2e. As we increase the threshold ρ, the number of clusters that fall in the time interval of search raises, leading to longer times to generate convoys based on retrieved clusters. The number of historical convoys that satisfy the thresholds also increases leading to longer query execution time. RCC performs up to four times faster than the baseline for varying settings of ρ threshold. The increase in the number of historical clusters leads to more intersections to be performed for historic convoy generation as shown in Figure 2f.

Acknowledgement. This research was supported in part by ARC DP170102726, DP180102050, NSFC 61728204, 91646204, the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative, and the National Science Foundation of the US under grant IIS-1816889.

REFERENCES

[1] Alka Bhushan, Umesh Bellur, Kuldeep Sharma, Srijay Deshpande, and Nandlal L Sarda. 2017. Mining swarm patterns in sliding windows over moving object data streams. In *SIGSPATIAL*. ACM, 60.
[2] 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. *VLDB* 10, 4 (2016), 313–324.
[3] Joachim Gudmundsson and Marc van Kreveld. 2006. Computing longest duration flocks in trajectory data. In *SIGSPATIAL*. ACM, 35–42.
[4] Antonin Guttman. 1984. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*. ACM, 47–57.
[5] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. 2008. Discovery of convoys in trajectory databases. *VLDB* 1, 1 (2008), 1068–1080.
[6] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. 2005. On discovering moving clusters in spatio-temporal data. In *SSTD*. Springer, 364–381.
[7] Xiaohui Li, Vaida Ceikute, Christian S Jensen, and Kian-Lee Tan. 2013. Effective online group discovery in trajectory databases. *IEEE TKDE* 12 (2013), 2752–2766.
[8] Yuxuan Li, James Bailey, and Lars Kulik. 2015. Efficient mining of platoon patterns in trajectory databases. *Data & Knowledge Engineering* 100 (2015), 167–187.
[9] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. 2010. Swarm: Mining relaxed temporal moving object clusters. *VLDB* 3, 1-2 (2010), 723–734.
[10] Sarana Nutanong, Edwin H Jacox, and Hanan Samet. 2011. An incremental Hausdorff distance calculation algorithm. *VLDB* 4, 8 (2011), 506–517.
[11] Lu-An Tang, Yu Zheng, Jing Yuan, Jiawei Han, Alice Leung, Chih-Chieh Hung, and Wen-Chih Peng. 2012. On discovery of traveling companions from streaming trajectories. In *ICDE*. IEEE, 186–197.
[12] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *SIGSPATIAL*. ACM, 99–108.
[13] Kai Zheng, Yu Zheng, Nicholas Jing Yuan, and Shuo Shang. 2013. On discovery of gathering patterns from trajectories. In *ICDE*. IEEE, 242–253.
[14] Kai Zheng, Yu Zheng, Nicholas J Yuan, Shuo Shang, and Xiaofang Zhou. 2014. Online discovery of gathering patterns over trajectories. *IEEE TKDE* 26, 8 (2014), 1974–1988.