MR-Swarm: Mining Swarms from Big Spatio-Temporal Trajectories Using MapReduce

Yanwei Yu[1(✉)], Jianpeng Qi[1], Yunhui Lu[2], Yonggang Zhang[2], and Zhaowei Liu[1]

[1] School of Computer and Control Engineering, Yantai University, Yantai, China
yuyanwei@ytu.edu.cn
[2] SCKE Key Laboratory of Ministry of Education, Jilin University, Changchun, China

Abstract. The increasing pervasiveness of object tracking technologies has enabled collection of huge amount of spatio-temporal trajectories. Discovering the useful movement patterns from such big data is gaining in importance and challenging. In this paper we propose an distributed mining framework on Hadoop for efficiently discovering swarm patterns from big spatio-temporal trajectories in parallel. We first define the notion of maximal objectset that captures swarms by recombining clusters in timeset domain. Second, we propose a parallel model based on timeset independent property of swarm pattern to parallel the mining process. Furthermore we propose a distributed algorithm using MapReduce chain architecture based on the proposed parallel model, which features two optimization pruning strategies designed to minimize the computation costs. Our empirical study on the real Taxi dataset demonstrates its effectiveness in finding object-closed swarms. Extensive experiments on 5 network-connected workstations also validate that our proposed algorithm nearly achieves 5-fold speedups against the serial solution.

1 Introduction

The increasing availability of location-acquisition technologies including all kinds of GPS, RFID, WLAN networks, mobile phones, and the emerging location-based APPs have enabled tracking almost any kind of moving objects, which results in huge volumes of spatio-temporal trajectory data that records a variety of action features, including location, time, and velocity. Such big data provides the huge opportunity of discovering usable knowledge about movement behaviour, which fosters novel many applications and services ranging from intelligent traffic management, urban computing to location-based services [1,2].

Patrick Laube et al. [4] first propose *flock*, *leadership*, and *aggregation* patterns in geospatial lifelines. Jeung et al. [5] define the notion of *convoy* pattern, in which a set of objects that move together in a density-based cluster for at least k continuous time points, instead of the strick size and shape of group by specifying the disk radius in *flock*. A recent study by Zhenhui Li et al. [6] proposes the *swarm* pattern, which also employs the density-connected clusters

H. Yin et al. (Eds.): IDEAL 2016, LNCS 9937, pp. 568–575, 2016.
DOI: 10.1007/978-3-319-46257-8_61

and relax the requirement that objects must form group for consecutive time points. In contrast to above mentioned patterns, swarm permit patterns where moving objects travel together for a number of nonconsecutive time points. In other words, moving objects could leave from the group transitorily, and then come back in swarm. Therefore, swarm is a more general and relaxed pattern that does not require k consecutive time points, also is more inline with the practical situations.

In our previous works [7,8], we propose online solutions of discovery of swarms and trajectory clusters to improve both efficiency and scalability. However the works are just suitable for trajectory data mining under streaming environment with normal rate. For the huge amounts of spatio-temporal trajectory data, the traditional serial solution is difficult to satisfy requirements of big trajectory pattern mining, which is a key issue addressed in this work. This paper focuses on efficient discovery of swarm pattern, one of useful group pattern, from high-volume moving object trajectories. We propose a distributed swarm pattern mining algorithm employing Hadoop platform, which incorporates three principles. First, we propose a notion of maximal objectset, and optimize serial method using minimal time support optimization. Second, we propose a parallel model based on timeset independent of swarm pattern, which parallelizes clustering and local swarm discovery in sub-time domain. Third, we implement an efficient distributed solution using MapReduce chain architecture on Hadoop platform. Finally, We conduct an extensive empirical study on real trajectory data to evaluate the proposed distributed framework. Our results offer insight into the effectiveness and efficiency of the proposed framework.

2 Preliminary Definition

We denote the set of trajectories TD. Let $O_{TD} = \{o_1, o_2, \ldots, o_m\}$ be the set of all moving objects and $T_{TD} = \{t_1, t_2, \ldots, t_n\}$ be the set of all time points in TD. We denote a subset of O_{TD} objectset O and a subset of T_{TD} timeset T. The size, $|O|$ and $|T|$, is the number of objects and time points in O and T respectively. A set of clusters, obtained by DBSCAN at each time point t_i, is denoted $C_{t_i} = \{C^1_{t_i}, C^2_{t_i}, \ldots, C^k_{t_i}\}$, where $C^j_{t_i} (1 \leq j \leq k)$ is a cluster at t_i.

Definition 1 (Swarm). *Given minimal thresholds min_o and min_t, a set pair $< O, T >$ is called a swarm pattern if $|O| \geq min_o$, $|T| \geq min_t$, and $\forall t \in T$, there exists a cluster $C \in C_t$ such that $O \subseteq C$.*

By Definition 1, all pairs $< O, T >$ that satisfy the minimal thresholds and all objects of O belong to a cluster at any time point in T are swarms. To avoid mining redundant swarms, Li et al. [6] further give the notion of closed swarm.

Definition 2 (Maximal Objectset or MO). *Given minimal threshold min_o and a timeset T, the objectset O is called a maximal objectset with respect to T iff $O = \bigcap_{t_i \in T} C^{k_i}_{t_i} (C^{k_i}_{t_i} \in C_{t_i})$ and $|O| \geq min_o$.*

By Definition 2, the maximal objectset can be identified by clusters of each time point in T.

Lemma 1. *Given a set pair $< O, T >$, if objectset O is a maximal objectset w.r.t. timeset T and $|T| \geq min_t$, then $< O, T >$ is an object-closed swarm (OSm).*

Lemma 1 is intuitive and easily proved by Definitions 1 and 2.

Definition 3 (Minimal Time Support). *Given minimal threshold min_t and a set pair $< O, T >$, if the objectset O is a maximal objectset w.r.t. T and $|T| = min_t$, then T is a minimal time support of $< O, T >$ to be a swarm.*

The minimal time support provides us a minimal amount of timeset for finding a swarm. Therefore, this concept of minimal time support guides us to propose the minimal time support optimization to reduce time points examination and lookup costs related to intersection operation of clusters of moving objects.

Definition 4 (Potential Swarm or PSm**).** *Given a set pair $< O, T >$, if the objectset O is a maximal objectset w.r.t. T and $|T| < min_t$, then we call $< O, T >$ a potential swarm.*

3 Optimization Principles

Minimal Time Support Optimization. By Definition 3, if timeset T corresponding to the maximal objectset O satisfies the condition of $|T| \geq min_t$, the we confirm that $< O, T >$ is a OSm. Next, we optimize CLUR [7] algorithm by using minimal time support instead of completed timeset (closed timeset), named *CLIP* for distinguishing from CLUR.

We also employ DBSCAN to get clusters C_t at each time point t. CLIP maintains two lists L and L_{sm}, which store potential swarms $PSms$ and object-closed swarms $OSms$ with minimal time support, respectively. PSm or OSm are stored in L and L_{sm} in form of *key-value* pairs. *key* correspond to objectset O of patterns, and *value* to timeset T.

Parallel Model. We observe the combination property of timeset corresponding to MO by Definition 2, as depicted in following Property.

Property 1. Given a maximal objectset O and its corresponding timeset T, for $\forall T_1, T_2$ $(T_1 \cap T_2 = \phi, T_1 \cup T_2 = T)$, there must exist O_1 (O_1 is a MO w.r.t. T_1) and O_2 (O_2 is a MO w.r.t. T_2) such that $O = O_1 \cap O_2$.

From Property 1, we observe that it is independent of the order of time points when examine maximal objectset by using cluster recombinant method. Thus we can adjust the order of time points or re-group time points of timeset arbitrarily. Therefore, we further conclude the parallel model based on timeset-independent for mining swarm pattern.

4 Distributed Algorithm

4.1 Framework of Distributed Mining Algorithm

In this section, we introduce our distributed algorithm called *MR-Swarm*, capable of mining swarm using MapReduce chain architecture based on proposed CLIP as a plug-in parallel method. The overall framework of our MR-Swarm is depicted in Fig. 1. The framework can be dived into four stages:

Stage 1: Preprocessing in Map1 phase. This phase reads trajectory data in parallel, and then partition data according to the attribute of time point, which provides Stage 2 load-balanced data partitions.

Stage 2: Distributed mining phase in Reducer1 phase. Stage 2 performs Reduce1 process on each reducer for mining local swarm patterns in independent timeset T_i.

Stage 3: Parallel merge phase in Map2 phase. The stage merges the local swarms from stage 2 in parallel, and generate intermediate patterns.

Stage 4: Final merge phase in Reduce2 phase. Stage 4 is executed on single machine to merge all intermediate patterns into expected swarm patterns.

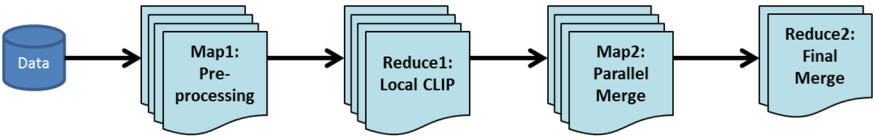


Fig. 1. Overall architecture of MR-Swarm framework

4.2 Preprocessing and Local Swarm Mining

The main task of preprocessing is load trajectory data of massive moving objects, map location points of same timestamp to same *key*, and output data partition to reducers. Therefore, the main challenges for an efficient partition are load balancing, minimized communication or shuffling cost.

To minimize traffic and shuffling cost, we employ combiner in preprocessing. In Mappers, trajectory data are mapped to *key-value* pairs according the information of time point, and then the *value*s of same *key* are merged in combiner, namely, trajectory points at same time point are merged into a pair of *key-value*. This would significantly reduce the number of shuffle in Shuffle phase. Finally, we mod *key* by the number of reducers r, and output the data partition into specified reducers grouped by time point.

As shown in Fig. 1, Stage 2 performs $CLIP$ to mine local swarm candidates in subset of T_{TD} on each reducer. Each reducer receives the trajectory points of all moving objects at specified time points, So all reducers can work in parallel by the proposed parallel model. For reducer R_i, it first performs DBSCAN clustering to get density-connected clusters at each time point, and then executes $CLIP$ in any order of time points to find local swarm candidates in local timeset.

4.3 Parallel Merge Phase

Stage 3 that runs on second Map phase aims to merge the local candidates on two or more timesets. Taking two timesets as an example, pseudocode of merge process is shown in Algorithm 1. min_o, min_t and T_{TD} are global parameters.

Stage 3 first reads distributed intermediate files from reducers of Stage 2, and distributes evenly them to Mappers. So each Mapper should first merge the assigned $OSms$ stored in L_{sm} as line 1. Then, examining whether the $PSms$ in T_1 or T_2 construct OSm in $T_1 \cup T_2$. As shown on lines 2–14, the examination also employs the strategy of minimal time support optimization to identify OSm in advance. Furthermore, we also utilize predicted timeset pruning rule to prune the $PSms$ that could not become swarms in future, given as following lemma.

Lemma 2. *Given the set of all time points T_{TD} and threshold min_t, for a potential swarm $< O, T >$ in $T_1 (T_1 \subset T_{TD})$, if $|T| < min_t + |T_1| - |T_{TD}|$, then $< O, T \cup (T_{TD} - T_1) >$ is certainly not a (closed) swarm.*

Algorithm 1. *Merge*: Merge Local Swarms on Two Timesets

Require: L_1, L_{sm_1}, T_1, L_2, L_{sm_2}, T_2
Ensure: L_1 and L_{sm_1}
1: $L_{sm_1} \leftarrow L_{sm_1} \cup L_{sm_2}$;
2: **for** each candidate $v_2 \in L_2$ **do**
3: **for** each candidate $v_1 \in L_1$ **do**
4: $O \leftarrow v_1.O \cap v_2.O$; $T \leftarrow v_1.T \cup v_2.T$;
5: **if** $|O| \geq min_o$ **then**
6: **if** $!MTS(O, T, L_{sm_1})$ **then**
7: **if** $|T| \geq |T_1| + |T_2| + min_t - |T_{TD}|$ **then**
8: $L_1.put(O, T)$;
9: **end if**
10: **else**
11: **if** $O == v_1.O$ **then**
12: $L_1.remove(v_1)$;
13: **end if**
14: **if** $O == v_2.O$ **then**
15: $v_2.exist \leftarrow$ TRUE;
16: **end if**
17: **end if**
18: **end if**
19: **end for**
20: **if** $(!v_2.exist) \&\& |v_2.T| \geq |T_1| + |T_2| + min_t - |T_{TD}|$ **then**
21: $L_1.put(v_2.O, v_2.T)$;
22: **end if**
23: **end for**
24: **for** each candidate $v_1 \in L_1$ **do**
25: **if** $|v_1.T| < |T_1| + |T_2| + min_t - |T_{TD}|$ **then**
26: $L_1.remove(v_1)$;
27: **end if**
28: **end for**
29: $T_1 \leftarrow T_1 \cup T_2$;

k mappers merge the local swarms on multiple subsets of time domain in parallel, however, we still need to perform State 4 on one machine to merge all intermediate swarms and output final swarms in final reduce phase.

5 Experiments

5.1 Experimental Setup

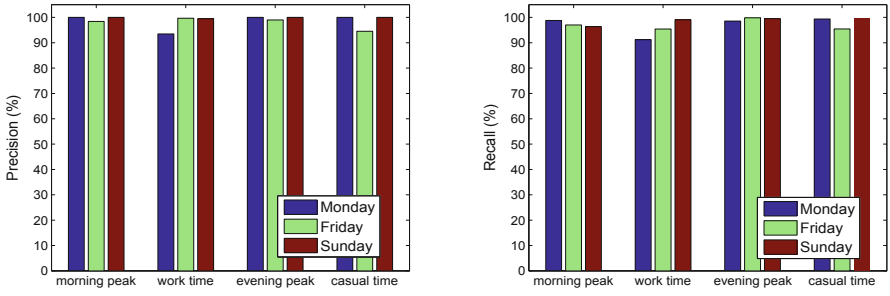
A comprehensive performance study has been conducted on 5 network connected workstations that deployed Hadoop platform of version 2.4.0 to evaluate the effectiveness and efficiency of the proposed algorithm. Each node is equipped with an Intel Core 2 Duo i5-3470 processor with 4 GB memory and runs a Centos release 6.6 operating system.

Real Dataset. We use a real spatio-temporal trajectory data *Taxi* in our experiments. The dataset is from T-drive project [3,9] developed by Microsoft Research Asia. We divide a day into four time periods, morning and evening peak time (7 am to 10 am and 5 pm to 8 pm), work time (10 am to 4 pm) and casual time (8 pm to 2 am). We interpolate the time domain into the granularity of minute, and get 7,560 time points in T_{TD}.

Metrics. We measure the correctness of MR-Swarm by Precision and Recall as follows: $Precision = (R \cap D)/R$, $Recall = (R \cap D)/D$, where D denotes swarms obtained by the serial CLUR [7] and R is discovered swarms of MR-Swarm algorithm.

5.2 Effectiveness Evaluation

First, we evaluate the correctness of MR-Swarm algorithm by measuring the *Precision* and *Recall* on the real *Taxi* dataset. To demonstrate generality, we use the data of Monday, Friday and Sunday, three representative days. From Fig. 2(a), We can see that the *Precision* of most time is nearly 100 % except Monday work time and Friday casual time, and *Precisions* of Monday work time and Friday casual time also reach at 93.5 %, 94.5 % respectively. Figure 2(b) shows the *Recall* of MR-Swarm is also good and robust. The average *Recall* of all time intervals reaches at 97.5 %.



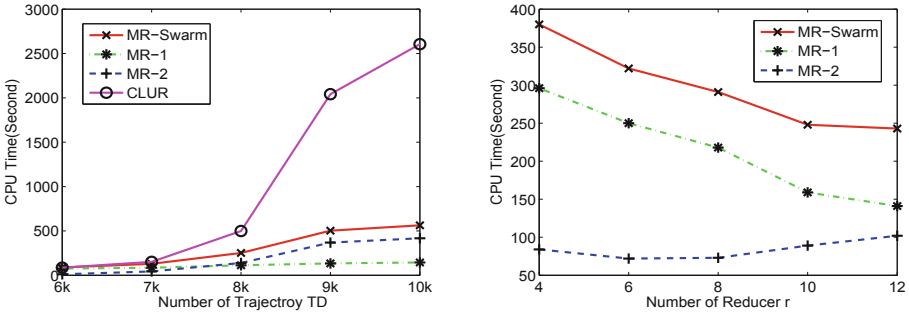
(a) Precision (b) Recall

Fig. 2. Effectiveness evaluation on *Taxi*

5.3 Efficiency Evaluation

We also measure the CPU time of two MapReduce phase in MR-Swarm referred as MR-1 and MR-2 respectively. The default setting follows: $Eps = 0.002, MinPts = 10$, thresholds $min_o = 30, min_t = 10$ and Taxi data on Sunday.

We first evaluate the scalability of MR-Swarm algorithm in terms of the volume of trajectories. In this experiment we randomly extract four subsets from the Taxi data from 6k to 9k trajectories. Figure 3(a) shows the total running time of MR-SWARM, MR-1, MR-2 and CLUR on the five datasets. MR-Swarm exhibits much better scalability than CLUR in terms of CPU time. From the CPU time utilized by MR-1 and MR-2, we observe that the number of trajectories less affects parallel clustering and local swarm mining on subset of time domain but greatly affects the phase of merging local $PSms$. In particular, MR-Swarm nearly achieves 5-fold speedup compared against CLUR when $TD = 10\,k$.



(a) Varying TD (b) Varying r

Fig. 3. Efficiency evaluation on $Taxi$

Next, we evaluate the effect of varying the number of reducers r from 4 to 12 in MR-1 phase, when the number of $Mapper$ in MR-2 phase is fixed to the number of nodes. As shown in Fig. 3(b), the CPU cost of MR-Swarm algorithm decreases as the number of reducers r increases. This is expected, MR-1 performs parallel clustering and local swarm discovery in sub timeset, thus the more reducers, the less time points assigned to each reducer, hence the average time consumption is also less. However since the number of node in our platform is limited, too many reducers does not further reducer the time consumption.

6 Conclusion

In this paper, we study the problem of discovering object-closed swarm patterns from big spatio-temporal trajectories. We propose distributed mining framework

using MapReduce chain architecture scalable to big trajectories to efficiently discover swarms. At last we demonstrate the effectiveness and efficiency of proposed distributed solution by conducting comprehensive evaluations on a real large scale taxi trajectory data.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China (nos. 61403328 and 61572419), the open project program of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University (no. 93K172014K13), the Key Research & Development Project of Shandong Province (no. 2015GSF115009), and the Shandong Provincial Natural Science Foundation (nos. ZR2013FQ023 and ZR2013FM011).

References

1. Zheng, Y.: Trajectory data mining: an overview. ACM Trans. Intell. Syst. Technol. **6**, 1–41 (2015)
2. Yu, Y., Cao, L., Rundensteiner, E.A., et al.: Detecting moving object outliers in massive-scale trajectory streams. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 422–431. ACM (2014)
3. Yuan, J., Zheng, Y., Xie, X., et al.: T-Drive: Enhancing driving directions with taxi drivers' intelligence. IEEE Trans. Knowl. Data Eng. **25**(1), 220–232 (2013)
4. Laube, P., Imfeld, S.: Analyzing relative motion within groups oftrackable moving point objects. In: Egenhofer, M.J., Mark, D.M. (eds.) GIScience 2002. LNCS, vol. 2478, pp. 132–144. Springer, Heidelberg (2002). doi:10.1007/3-540-45799-2_10
5. Jeung, H., Yiu, M.L., Zhou, X., et al.: Discovery of convoys in trajectory databases. In: Proceedings of The 34th Very Large Databases Conference, Auckland, New Zealand, 23–28 August, pp. 1068–1080 (2008)
6. Li, Z., Ding, B., Han, J., et al.: Swarm: mining relaxed temporal moving object clusters. In: Proceedings of the 36th Very Large Databases Conference, Singapore, pp. 13–17 (2010)
7. Qi, Y., Yu, Y., Kuang, J., et al.: Efficient algorithm for real time mining swarm patterns. J. Univ. Sci. Technol. Beijing **34**(1), 32–37 (2012)
8. Yu, Y., Wang, Q., Wang, X., Wang, H.: On-line clustering for trajectory data stream of moving objects. Comput. Sci. Inf. Syst. **10**(3), 1319–1342 (2013)
9. Yuan, J., Zheng, Y., C. Zhang, et al.: T-drive: driving directions based on taxi trajectories. In: ACM SigSpatial Geographic Information Science, pp. 99–108. ACM (2010)