Algorithm 5: Variable Length Bit Compression based Algorithm (VBA)

Input: Partition $P_t(o) = \{o_i | 1 \leq i \leq |P_t(o)|\}$, a global hashmap H, a global candidate list C, (M, K, L, G) required for co-movement pattern

```
1  a local candidate list C_l ← ∅
2  for each object o_i ∈ H do
3      if o_i ∈ P_t(o) then
4          append '1' to the end of H[o_i].B
5          remove o_i from P_t(o)
6      else
7          append '0' to the end of H[o_i].B
8          tag ← isVaild(H[o_i].B)
9          if tag = 1 then
10             C_l.insert(o_i)
11         else if tag = −1 then
12             H.delete(o_i)
13 for each object o_i ∈ P_t(o) do
14     H.insert(o_i, ⟨t, '1'⟩)
15 for each object o_i ∈ C_l do
16     L ← ∅
17     for each object o_j ∈ C (o_i ≠ o_j) do
18         if min{et_i, et_j} − max{st_i, st_j} ≥ K then
19             L.insert(o_j)
20     find and output valid patterns in L ∪ {o_i} as in lines 10–18
       of Algorithm 4 by applying Lemma 8
21 C ← C ∪ C_l
```

(line 8). If $tag = 1$ (i.e., $H[o_i].B$ is a maximal pattern time sequence), VBA inserts o_i into C_l (lines 9–10). If $tag = -1$ (i.e., $H[o_i].B$ is an invalid pattern), it deletes o_i from H (lines 11–12). Subsequently, VBA processes trajectories that do not exist in the global H. For each trajectory o_i left in $P_t(o)$, VBA inserts a new entry $(o_i, \langle t, \text{'1'}\rangle)$ into H, where t is the start time and '1' is the current bit string of o_i (lines 13–14). Thereafter, for each trajectory o_i in local candidate list C_l, VBA first filters the global candidate list C using Lemma 8 to get a candidate list L (lines 16–19). Then, it finds the valid patterns in $L \cup \{o_i\}$ as in lines 9–17 of Algorithm 4 by applying Lemma 8. Finally, the global candidate list C is updated to $C \cup C_l$ (line 21).

Time Complexity and Storage Cost. According to Theorem 1, variable length bit compression reduces the storage cost from $O(n\eta)$ to $O(n\frac{G+L}{L})$. Here, n denotes the total number of occurrences for trajectories in one subtask, and $\frac{G+L}{L}$ is much smaller than η. Next, since the pattern enumeration methods of VBA and FBA are similar, the two have similar time complexity. The only difference is that, due to the use of maximal pattern time sequences, the candidate size of VBA is much smaller. However, the use of maximum pattern time sequences comes with the cost that, the response time of answering real-time co-movement pattern detection increases. Thus, VBA trades latency for throughput.

7. EXPERIMENTAL EVALUATION

In this section, we evaluate the efficiency and scalability of our proposed framework and methods, and also include comparisons with alternative methods. All experiments were conducted on a cluster consisting of 11 nodes, where one node serves as the master node, and the remaining nodes serve as slave nodes. Each node is equipped with two 12-core processors (Intel Xeon E5-2620 v3 2.40GHz), 64GB RAM, and a Gigabit Ethernet. Each cluster node runs Ubuntu 14.04.3 LTS and Flink 1.3.2. All system algorithms were implemented in Java.

Table 2: Datasets Used in our Experiments

Attributes	GeoLife	Taxi	Brinkhoff
# trajectories	18,670	20,151	10,000
# locations	24,876,978	189,419,934	23,906,131
# snapshots	92,645	502,559	97,241
Storage Size	1.5G	14G	1.7G

Table 3: Parameter Ranges and Default Values

Parameter	Range
grid cell width l_g	0.2%, 0.4%, **0.8%**, 1.6%, 3.2%, 6.4%
distance threshold ϵ	0.02%, **0.04%**, 0.06%, 0.08%, 0.10%, 0.12%
min objects M	5, 10, **15**, 20, 25
min duration K	120, 150, **180**, 210, 240
min local duration L	10, 20, **30**, 40, 50
max gap G	10, 20, **30**, 40, 50
ratio of objects O_r	10%, 20%, 40%, 60%, 80%, **100%**
machine number N	1, 2, 4, 6, 8, **10**

Datasets: We use two real-life datasets and one synthetic dataset to model streaming trajectories, as summarized in Table 2.

- GeoLife[7]: This dataset records the travel records of users during a period of more than three years. The GPS records are collected periodically, and 91% of the trajectories are sampled every 1 to 5 seconds.
- Taxi[8]: This is a real trajectory dataset generated by taxis in Hangzhou. Trajectories are segmented into trips, and each resulting trajectory represents the trace of a taxi during a month. The trajectories are sampled every 5 seconds.
- Brinkhoff[9]: This dataset is generated via the Brinkhoff generator [5]. The trajectories are generated on the real road network of Las Vegas. An object position is generated every second while an object moves through the road network with random but reasonable direction and speed.

Comparison Methods: As this is the first study of real-time distributed co-movement pattern detection over streaming trajectories, no competitors are available. Instead, we adapt the state-of-the-art distributed co-movement pattern detection method [10] over historical trajectories, so that it can serve as a baseline method. In addition, we include comparisons between our clustering method RJC (discussed in Section 5) and two existing clustering methods.

- SRJ [36] is the state-of-the-art distributed range join method for streaming trajectories. We extend it to support DBSCAN clustering similar as our method RJC.
- GDC [14] is a grid-based DBSCAN clustering approach for a centralized environment. We extend it to work with Flink.

Parameters: In the experiments, we study the effect on performance of several factors, as summarized in Table 3, where the default values are shown in bold. In Table 3, l_g and ϵ are set to a percentage of the maximal distance of the whole dataset. In each set of experiments, we vary one parameter while fixing the others at their default values. Recall that, both ϵ and $minPts$ are used to control the density-based clustering, we vary only ϵ because similar performance is observed for different values of $minPts$. We fix $minPts$ at 10.

Performance Metrics: We study both latency and throughput. According to the definition of real-time co-movement pattern detection, we need to find all the current co-movement patterns in the snapshot set $\{S_1, S_2, \cdots, S_t\}$ at each timestamp t. Hence,

[7]https://research.microsoft.com/en-us/projects

[8]This is a proprietary dataset.

[9]https://iapg.jade-hs.de/personen/brinkhoff/generator/