Evolutionary Clustering of Moving Objects

Tianyi Li[1], Lu Chen[2]✉, Christian S. Jensen[1], Torben Bach Pedersen[1], Yunjun Gao[2], Jilin Hu[1]
[1]Department of Computer Science, Aalborg University, Denmark
[2]College of Computer Science, Zhejiang University, Hangzhou, China
{tianyi, csj, tbp, hujilin}@cs.aau.dk, {luchen,gaoyj}@zju.edu.cn

***Abstract*—The widespread deployment of smartphones, networked in-vehicle devices with geo-positioning capabilities, and vessel tracking technologies renders it feasible to collect the evolving geo-locations of populations of land- and sea-based moving objects. The continuous clustering of such data can enable a variety of real-time services, such as road traffic management and vessel collision risk assessment. However, little attention has so far been given to the quality of moving-object clusters—for example, it is beneficial to smooth short-term fluctuations in clusters to achieve robustness to exceptional data and to improve existing applications.**

We propose the notion of evolutionary clustering of moving objects, abbreviated ECM, that enhances the quality of moving object clustering by means of temporal smoothing that prevents abrupt changes in clusters across successive timestamps. Employing the notions of snapshot and historical costs, we formalize ECM and formulate ECM as an optimization problem. We prove that ECM can be performed approximately in linear time, thus eliminating iterative processes employed in previous studies. Further, we propose a minimal-group structure and a seed-point shifting strategy to facilitate temporal smoothing. Finally, we present all algorithms underlying ECM along with a set of optimization techniques. Extensive experiments with three real-life datasets offer insights into ECM and show that it outperforms state-of-the-art solutions in terms of both clustering quality and clustering efficiency.

***Index Terms*—evolutionary clustering, moving objects, temporal smoothness**

I. Introduction

It is increasingly possible to equip moving objects, including people, vehicles, and vessels, with devices that are capable of transmitting the objects' positions to a central location in real time. This scenario provides opportunities for the real-time discovery of hidden mobility patterns. Being a typical mobility pattern discovery approach, clustering aims to group a set of moving objects according to their spatio-temporal similarity. The ability to cluster moving objects continuously benefits a wide range of services and applications such as vessel collision risk assessment [1], path modeling [2], [3], real-time road traffic management [4], and animal migration analyses [5].

Existing real-time clustering methods focus primarily on running time efficiency and generally disregard clustering quality [6]–[20]. As pointed out in previous studies [9], [21], [22], clusterings should evolve smoothly over time because object locations generally change continuously and gradually. Fluctuations in clusterings are not only undesirable but may also have adverse effects on applications and services. An example illustrates this.

Example 1. *In vessel tracking systems, clustering can be employed as a fundamental step to identify vessels positioned spatially close to each other to enable real-time vessel collision risk assessment [1]. Fig. 1 shows the locations of 12 vessels at three timestamps, $k = 1, 2, 3$. Traditional clustering algorithms return clusters $c_1 = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ and $c_2 = \{o_7, o_8, o_9, o_{10}, o_{11}, o_{12}\}$ at the first timestamp, clusters $c_1 = \{o_1, o_2, o_3, o_4, o_5\}$, $c_2 = \{o_7, o_8, o_9, o_{11}\}$, and $c_3 = \{o_6, o_{10}, o_{12}\}$ at the second timestamp, and again the two clusters from the first timestamp at the third timestamp. Given the clusterings at $k = 1$ and $k = 3$, the occurrence of c_3 at $k = 2$ is undesirable because both vessel movement and the evolution of the clusterings should be smooth during a short time period. In this case, at $k = 2$, o_6 is only warned w.r.t. o_{10} and o_{12} because these three are the only vessels in c_3 and because collision risks only occur among vessels in the same cluster [1]. Thus, the occurrence of c_3 at $k = 2$ degrades the downstream collision risk assessment quality, which increases the risk of collisions.*

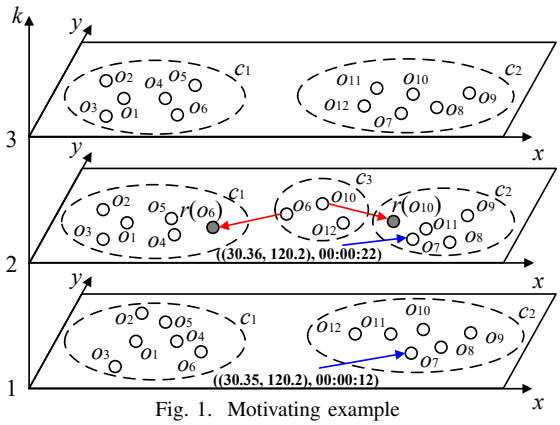


Fig. 1. Motivating example

Further, fluctuations in clusterings in the context of road traffic management may incur unnecessary and frequent vehicles re-routings and adjustments of traffic signals [23]. Thus, a high-quality clustering should be robust to short-term fluctuations in the underlying spatio-temporal data. A potential approach to avoid fluctuations in clusterings is to perform cleaning before performing the clustering. However, studies on three real-life datasets (cf. Section VI-A) show that among the moving object positions that cause the fluctuations, 75.9%, 97.7%, and 88.8% of the objects comply with the speed constraint [24], while 96.1%, 87.3%, and 82.8% of the objects are categorized as inliers [25]. Moreover, in real-time applications, it is impractical to correct previous clusterings retroactively. Hence, it is difficult for existing cleaning techniques to enable smoothly changing clusterings [24]–[26].

Existing real-time moving object clustering methods cannot eliminate fluctuations in clusterings. The key reason is that they perform clustering at the current timestamp without taking historical clusterings into account (cf. Example 1). This problem can be solved by applying the notion of evolutionary clustering [27]–[38], from the domain of dynamic networks. By utilizing temporal smoothness, which yields robustness to short-term variability, evolutionary clustering is able to achieve better clustering than traditional clustering techniques [33]. In Example 1, two stable clusters, $c_1 = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ and $c_2 = \{o_7, o_8, o_9, o_{10}, o_{11}, o_{12}\}$ (by adjusting o_6 to $r(o_6)$ and o_{10} to $r(o_{10})$) can be obtained for all three timestamps if applying evolutionary clustering. This is a meaningful and better-quality result because it conforms to the smoothness of object movement [21]. With these clusterings and continuing Example 1, o_6 is warned w.r.t. $o_1, o_2, \cdots, o_5$, which supports collision avoidance better [1]. Motivated by this, we study how to enable efficient evolutionary clustering of moving objects.

Existing evolutionary clustering studies target dynamic networks and are not suitable for moving objects [27]–[38], mainly for three reasons. First, the solutions are designed specifically for dynamic networks, which differ substantially from spatio-temporal data. Second, the movements of moving objects are generally much faster than the evolution of dynamic networks, which renders the temporal smoothness used in existing studies too "strict" for moving objects. Third, existing studies often optimize the clustering quality iteratively at each timestamp [29]–[37], which is computationally costly and infeasible for large collections of moving objects.

We propose an efficient and effective method for evolutionary clustering of moving objects, called ECM. First, we adopt the idea of neighbor-based smoothing [27] and develop a *minimal group* structure that summarizes groups by *seed points* in order to facilitate smoothing. Second, following existing studies [30], [31], [33]–[37], [39], we formulate ECM as an optimization problem that employs the new notions of snapshot cost and historical cost designed specifically for moving objects. The snapshot cost evaluates the spatial distances between clusterings in evolutionary and non-evolutionary settings. The historical cost evaluates the temporal distances between the current and recent historical clusterings. Next, we decompose the proposed optimization function and prove that each component can be solved approximately in constant time. The effectiveness of smoothing is further facilitated by a *seed point shifting* strategy. Finally, we introduce a grid index and present algorithms for each component of evolutionary clustering along with a set of performance optimization techniques. The paper's main contributions are summarized as follows.

- We formalize the ECM problem. To the best of our knowledge, this is the first proposal for moving object clustering that takes into account temporal smoothness.
- Based on the new notions of snapshot cost and historical cost, we formulate ECM as an optimization problem. We prove that the problem can be solved approximately in linear time.
- We propose a *minimal group* structure and a *seed point shifting* strategy to facilitate temporal smoothing; and we present all algorithms needed to enable evolutionary clustering, along with a set of optimization techniques.
- Extensive experiments on three real-life datasets show that ECM is capable of advancing the state-of-the-arts in terms of both clustering quality and efficiency.

The rest of paper is organized as follows. We present preliminaries in Section II. We formulate the problem in Section III and derive its solution in Section IV. Section V presents the algorithms and optimization techniques. Section VI covers the experimental study. Section VII reviews related work, and Section VIII concludes and offers directions for future work.

II. Preliminaries

A. Data Model

Definition 1. *A **GPS record** of a **moving object** o is a pair (l, t), where t is a **timestamp** and $l = (x, y)$ is a **location**, with x being a longitude and y being a latitude.*

The GPS records of a moving object may be transmitted to a central location in an unsynchronized manner. To avoid this affecting the subsequent processing, we adopt an existing approach [40] and discretize time into short intervals that are indexed by integers. We then map the timestamp of each GPS record to the index of the interval that the timestamp belongs to. Assuming the start time is 00:00:00 and that we partition time into intervals of duration $\Delta t = 10s$ then the time series ⟨00:00:01, 00:00:12, 00:00:20, 00:00:31⟩ is mapped $\langle 0, 1, 2, 3\rangle$. We call such a sequence a discretized time sequence and call each discretized timestamp a ***time step*** dt.

Definition 2. *A moving object o is **active** at time step $dt = [t_1, t_2]$ if a GPS record (l, t) exists for o such that $t \in [t_1, t_2]$.*

Definition 3. *A **snapshot** $\mathcal{O}_k$ is the set of moving objects that are active at time step dt_k.*

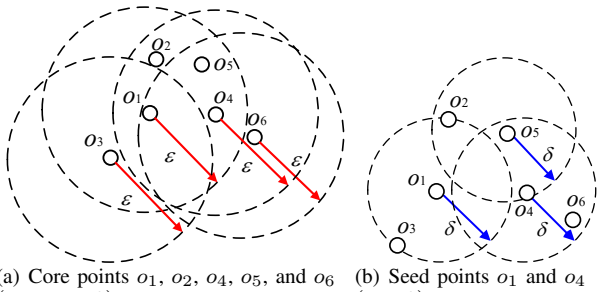
A GPS record of o at dt_k, if any, is denoted as $(o.l_k, o.t_k)$. If o is active at both dt_{k-1} and dt_k and the current time step is dt_k, $o.l_k$ and $o.t_k$ are simplified as $o.l$ and $o.t$, and $o.l_{k-1}$ and $o.t_{k-1}$ are simplified as $o.\tilde{l}$ and $o.\tilde{t}$. At time step dt_2 ($k = 2$) in Fig. 1, $o_7.\tilde{l} = o_7.l_1 = (30.35, 120.2)$, $o_7.\tilde{t} = o_7.t_1 =$ 00:00:12, $o_7.l = o_7.l_2 = (30.36, 120.2)$, and $o_7.t = o_7.t_2 =$ 00:00:22. For simplicity, we use o in figures to denote the location of o.

Definition 4. *The **θ-neighbor set** $\mathcal{N}_\theta(o)$ of a moving object $o\,(\in \mathcal{O}_k)$ at time step dt_k is defined as $\{o' | o' \in \mathcal{O}_k \wedge d(o.l, o'.l) \leq \theta\}$, where $d(\cdot)$ is Euclidean distance and θ is a distance threshold. $|\mathcal{N}_\theta(o)|$ is called the **local density** of o w.r.t. θ at dt_k.*

B. DBSCAN

We adopt the well-known density-based clustering approach, DBSCAN [41], for clustering. DBSCAN relies on two parameters to characterize density or sparsity, i.e., real ε and integer *minPts*.

Definition 5. *A moving object $o \in \mathcal{O}_k$ is a **core point** w.r.t. ε and minPts, if $\mathcal{N}_\varepsilon(o) \geq$ minPts.*



(a) Core points o_1, o_2, o_4, o_5, and o_6 (*minPts* $= 3$)

(b) Seed points o_1 and o_4 ($\rho = 3$)

Fig. 2. $o_i\,(1 \leq i \leq 6)$ at dt_1 in Fig. 1

Definition 6. *A moving object $o \in \mathcal{O}_k$ is **density reachable** from another moving object $o' \in \mathcal{O}_k$, if a sequence of moving objects $o_1, o_2, \cdots o_n$ $(n \geq 2)$ exists such that (i) $o_1 = o'$ and $o_n = o$; (ii) o_w $(1 \leq w < n)$ are core points; and (iii) $d(o_w, o_{w+1}) \leq \varepsilon$ $(1 \leq w < n)$.*

A cluster is formed by a set of core points and their density reachable points [41]. We omit the detailed algorithm for brevity. Given ε and *minPts*, $o \in \mathcal{O}_k$ is an ***outlier***, if it is not in any cluster; $o \in \mathcal{O}_k$ is a ***border point***, if $\mathcal{N}_\varepsilon(o) <$ *minPts* and $d(o, o') \leq \epsilon$, where o' is a core point.

Definition 7. *A **clustering result** $\mathcal{C}_k = \{c_1, c_2, \cdots, c_n\}$ is a set of clusters obtained from snapshot $\mathcal{O}_k$.*

Example 2. *In Fig. 1, $\mathcal{C}_1$ has clusters $c_1 = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ and $c_2 = \{o_7, o_8, o_9, o_{10}, o_{11}, o_{12}\}$. Further, $o_i\,(i = 1, 2, 4, 5, 6)$ in Fig. 2a are core points.*

C. Evolutionary Clustering

In evolutionary clustering, a sequence of clusterings is created from streaming data such that a clustering is created for the snapshot at each time step. Evolutionary clustering takes into account the smoothness characteristics of streaming data to obtain high-quality clusterings [30]. Specifically, two quality aspects, ***historical quality*** and ***snapshot quality***, are considered and are integrated into a cost function $\mathcal{F}_k$.

$$\mathcal{F}_k = \mathcal{TC}_k(\mathcal{C}_{k-1}, \mathcal{C}_k) + \alpha \cdot \mathcal{SC}_k(\mathcal{C}_o, \mathcal{C}_k) \tag{1}$$

Function $\mathcal{F}_k$ is a weighted sum of two terms: a snapshot cost ($\mathcal{SC}_k$) and a historical cost ($\mathcal{TC}_k$). C_k is the clustering after smoothing, and C_o is the one before smoothing. The snapshot cost $\mathcal{SC}_k$ captures the similarity between $\mathcal{C}_k$ and $\mathcal{C}_o$. The smaller $\mathcal{SC}_k$ is, the better the snapshot quality. The historical cost $\mathcal{TC}_k$ measures how similar clustering $\mathcal{C}_k$ and the previous clustering $\mathcal{C}_{k-1}$ are. The smaller $\mathcal{TC}_k$ is, the better the historical quality. Weight $\alpha\,(> 0)$ enables controlling the trade-off between the snapshot cost and the historical cost. Evolutionary clustering tries to find clusterings that minimize Formula 1 at each time step.

III. Problem Statement

A. Uncertainty of "Border" Points

We observe that moving objects having fewer neighbors (i.e., are generally located at the border of clusters) are more likely to leave their current cluster at the next time step than those having more neighbors (i.e., are generally located near centers of clusters). This is validated by statistics from three real-life datasets (cf. Section VI-A). Specifically, we denote the average number of neighbors of the objects that shift to other clusters or becoming outliers during the next time steps as n_1 and denote those remaining in the same cluster during the next time steps as n_2. $o'(\neq o)$ is a neighbor of o if $d(o, o') \leq \varepsilon$. For the three real-life datasets, $\frac{n_1}{n_2}$ is 0.16, 0.59 and 0.25 when ε is set to 1.2km, 35km, and 1.2km, respectively, and *minPts* is set to 3, 3, and 5, respectively.

B. Problem Definition

*a) **Cost embedding**:* Existing evolutionary clustering studies generally perform temporal smoothing on the clustering result [30], [31], [33]–[39]. Specifically, they adjust $\mathcal{C}_k$ iteratively so as to minimize Formula 1, which incurs very high costs. We adopt cost embedding [27], which pushes the cost formula down from the clustering result level to the data level, thus enabling flexible and efficient temporal smoothing. However, the existing cost embedding technique [27] targets dynamic networks only. To apply cost embedding to moving objects, we propose a minimal-group structure and snapshot and historical cost functions.

*b) **Snapshot cost**:* We perform smoothing by computing adjustments of the locations of moving objects, defined as follows.

Definition 8. *An **adjustment** $r_k(o)$ is a location of a moving object o obtained through smoothing at dt_k. The set consisting of the adjustments at dt_k is defined as $\mathcal{R}_k = \{r_k(o) | o \in \mathcal{O}_k\}$.*

We simplify $r_k(o)$ to $r(o)$ if the context is clear. In Fig. 1, $r(o_6)$ is an adjustment of o_6 at dt_2. According to Formula 1, C_k is the clustering after smoothing, and C_o is the one before smoothing. The snapshot cost measures how similar $\mathcal{C}_k$ is to $\mathcal{C}_o$. Moreover, we adopt cost embedding that smooths moving objects at the data level. Based on these considerations, we formulate the snapshot cost of o w.r.t. its adjustment $r(o)$ at dt_k (denoted as $\mathcal{SC}_k(r(o))$) as the deviation between o and $r(o)$ at dt_k:

$$\mathcal{SC}_k(r(o)) = d(r(o), o.l)^2 \quad s.t. \quad d(r(o), o.\tilde{l}) \leq \mu \cdot (o.t - o.\tilde{t}), \tag{2}$$

where μ is a speed constraint. Formula 2 requires that any adjustment $r(o)$ must follow that constraint. Obviously, the larger the distance between $o.l$ and its adjustment $r(o)$, the higher the snapshot cost.

*c) **Historical cost**:* According to one study [22], gradual location changes lead to stable co-movement relationships among moving objects during short periods of time. Thus, similar to neighbor-based smoothing in dynamic communities [27], it is sensible to smooth the location of each moving object in the current time step using its neighbours at the previous time step. One previous study [27] smooths the distance between each pair of neighboring nodes. However, simply applying this to moving objects may degrade the performance of smoothing if a "border" point is involved. Recall the observation from Section III-A and assume that $o_1.l$ is smoothed according to $o_3.l$ at dt_2 in Figs. 1 and 2 . As o_3 is a border point at dt_1 with a higher probability of leaving the cluster c_1 at dt_2, using o_3 to smooth o_1 may result in o_1

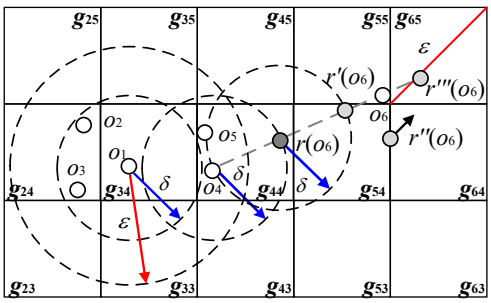


Fig. 3. $o_i\,(i=1,\ldots,6)$ at dt_2 in Fig. 1 ($\rho=3$)

also leaving c_1 or being located at the border of c_1 at dt_2. The first case may incur an abrupt change to the clustering while the second case may increase the intra-distance of $c_1\,(\in \mathcal{C}_2)$ and decrease the inter-distance of clusters in $\mathcal{C}_2$. To tackle this problem, we model neighboring moving objects as minimal groups summarized by seed points.

Definition 9. *A **seed point** $s\,(s \in \mathcal{S}_k)$ summarizes a **minimal group** $\mathcal{M}_k(s) = \{o \in \mathcal{O}_k | \, d(o,s) \leq \delta \wedge \forall s' \in \mathcal{S}_k\,(s' \neq s \Rightarrow d(o,s) \leq d(o,s'))\}$ at dt_k, where $\delta\,(0 < \delta \leq \varepsilon)$ is a given parameter and $\mathcal{S}_k\,(\mathcal{S}_k \subset \mathcal{O}_k)$ is a seed point set at dt_k. Further, we require that $|\mathcal{M}_k(s)|$, exceeds a parameter ρ. Any moving object o in $\mathcal{M}_k(s)$ that is different from s is called a **non-seed point**. Note that $\mathcal{M}_k(s) \cap \mathcal{M}_k(s') = \emptyset$ if $s \neq s'$.*

With dt_k being the current time step, we use s to denote the seed point of o at dt_k (i.e., $o \in \mathcal{M}_k(s)$), and we use $\tilde{s}$ to denote the seed point of o at dt_{k-1} (i.e., $o \in \mathcal{M}_{k-1}(\tilde{s})$).

Example 3. *Fig. 2b contains two minimal groups, i.e., $\mathcal{M}_1(o_1) = \{o_1, o_2, o_3\}$ and $\mathcal{M}_1(o_4) = \{o_4, o_5, o_6\}$. Next, Fig. 3, has only one minimal group before smoothing, i.e., $\mathcal{M}_2(o_1) = \{o_1, o_2, o_3\}$. Further, with $k=2$ being the current time step, both s and $\tilde{s}$ of o_2 is o_1, and $\tilde{s}.\tilde{l} = o_1.l_1$.*

We propose to use the location of a seed point s to smooth the location of a non-seed point $o\,(o \in \mathcal{M}_k(\tilde{s}))$ at dt_{k+1}. In order to guarantee the effectiveness of smoothing, Def. 9 introduces two constraints when generating minimal groups: (i) $d(o,s) < \delta\,(\delta \leq \varepsilon)$ and (ii) $|\mathcal{M}_k(s)| \geq \rho$. Setting δ to a small value, the first constraint ensures that $o \in \mathcal{M}_k(s)$ are close neighbors at dt_k. Specifically, we require $\delta \leq \varepsilon$ because this makes it very likely that moving objects in the same minimal group are in the same cluster. The second constraint ensures large neighbor sets $\mathcal{N}_\delta(s)$. This is to avoid that using a "border" point as a "pivot" to smooth the movement of other moving objects, which may lead to an abrupt change between clusterings or a low-quality clustering (according to the quality metrics of traditional clustering [42]). We present the algorithm for generating minimal groups in Section V-B.

Based on the above analysis, we formalize the historical cost of o w.r.t. its adjustment $r(o)$ at dt_k, denoted as $\mathcal{TC}_k(r(o))$, as follows.

$$\mathcal{TC}_k(r(o)) = \left(\left\lceil \frac{d(r(o),\tilde{s}.l)}{\delta} \right\rceil - 1\right)^2 \quad s.t.\, d(r(o),o.\tilde{l}) \leq \mu \cdot (o.t - o.\tilde{t}), \tag{3}$$

where $o \in \mathcal{M}_{k-1}(\tilde{s}) \backslash \{\tilde{s}\}$. Given the threshold δ, the larger the distance between $r(o)$ and $\tilde{s}.l$, the higher the historical cost. Here, we use the degree of closeness (i.e., $\left\lceil \frac{d(r(o),\tilde{s}.l)}{\delta} \right\rceil - 1$) instead of $d(r(o),\tilde{s}.l)$ to evaluate the historical cost, due to two reasons. First, constraining the exact relative distance between any two moving objects' locations during a time interval may be too restrictive, as it varies over time in most cases. Second, using the degree of closeness to constrain the historical cost is sufficient to obtain a smooth evolution of clusterings.

*d) **Total cost**:* Formulas 2 and 3 give the snapshot and historical costs for each o w.r.t. its adjustment $r(o)$. However, the former is distance based, while the latter is proximity based. Thus, we normalize them to a specific range $[0,1)$:

$$\mathcal{SC}_k(r(o)) = \left(\frac{d(r(o),o.l)}{4\mu \cdot \Delta t + \delta}\right)^2 s.t.\, d(r(o),o.\tilde{l}) \leq \mu \cdot (o.t - o.\tilde{t}) \tag{4}$$

$$\mathcal{TC}_k(r(o)) = \left(\frac{\left\lceil \frac{d(r(o),\tilde{s}.l)}{\delta} \right\rceil - 1}{\frac{4\mu\cdot\Delta t+\delta}{\delta}}\right)^2 \quad s.t.\, d(r(o),o.\tilde{l}) \leq \mu \cdot (o.t - o.\tilde{t}), \tag{5}$$

where $o \in \mathcal{M}_{k-1}(\tilde{s}) \backslash \{\tilde{s}\}$ and Δt is the duration of a time step. Clearly, $\mathcal{SC}_k(r(o)) \geq 0$ and $\mathcal{TC}_k(r(o)) \geq 0$. Thus, we only need to prove $\mathcal{SC}_k(r(o)) < 1$ and $\mathcal{TC}_k(r(o)) < 1$.

Lemma 1. *If $d(o.l,o.\tilde{l}) \leq (o.t - o.\tilde{t}) \cdot \mu$ then $d(r(o),o.l) \leq 4\mu \cdot \Delta t$.*

Proof. According to our mapping of original timestamps (Section II-A), $o.t - o.\tilde{t} \leq 2\Delta t$. Next, the speed constraint is $d(r(o),o.\tilde{l}) \leq (o.t - o.\tilde{t}) \cdot \mu$. Further, $d(r(o),o.l) \leq d(o.l,o.\tilde{l}) + d(r(o),o.\tilde{l})$ due to the triangle inequality. Thus, we get $d(r(o),o.l) \leq d(o.l,o.\tilde{l}) + 2\mu \cdot \Delta t$. Since $d(o.l,o.\tilde{l}) \leq (o.t - o.\tilde{t}) \cdot \mu$, $d(r(o),o.l) \leq 4\mu \cdot \Delta t$. □

It follows from Lemma 1 that $\mathcal{SC}_k(r(o)) < 1$ if $d(o.l,o.\tilde{l}) \leq (o.t - o.\tilde{t}) \cdot \mu$. However, $d(o.l,o.\tilde{l}) \leq (o.t - o.\tilde{t}) \cdot \mu$ does not necessarily hold. To address this problem, we pre-process $o.l$ according to $o.\tilde{l}$ so that it follows the speed constraint before conducting evolutionary clustering. The details are given in Section IV-C.

Lemma 2. *Given $o \in \mathcal{M}_{k-1}(\tilde{s}) \backslash \{\tilde{s}\}$, if $d(\tilde{s}.l,\tilde{s}.\tilde{l}) \leq (\tilde{s}.t - \tilde{s}.\tilde{t}) \cdot \mu$ then $d(r(o),\tilde{s}.l) \leq 4\mu \cdot \Delta t + \delta$.*

Proof. $d(r(o),o.\tilde{l}) = (o.t - o.\tilde{t}) \cdot \mu \leq 2\mu \cdot \Delta t$ due to $o.t - o.\tilde{t} \leq 2\Delta t$ (cf. Lemma 1). Further, $d(o.\tilde{l},\tilde{s}.\tilde{l}) \leq \delta$ according to Def. 9. Moreover, $d(r(o),\tilde{s}.l) \leq d(r(o),o.\tilde{l}) + d(o.\tilde{l},\tilde{s}.l)$ and $d(o.\tilde{l},\tilde{s}.l) \leq d(o.\tilde{l},\tilde{s}.\tilde{l}) + d(\tilde{s}.l,\tilde{s}.\tilde{l})$ according to the triangle inequality. Since $d(\tilde{s}.l,\tilde{s}.\tilde{l}) \leq (\tilde{s}.t - \tilde{s}.\tilde{t}) \cdot \mu \leq 2\mu \cdot \Delta t$, we derive that $d(r(o),\tilde{s}.l) \leq 2\mu \cdot \Delta t + d(o.\tilde{l},\tilde{s}.l) \leq 2\mu \cdot \Delta t + d(o.\tilde{l},\tilde{s}.\tilde{l}) + d(\tilde{s}.l,\tilde{s}.\tilde{l}) \leq 2\mu \cdot \Delta t + \delta + 2\mu \cdot \Delta t = 4\mu \cdot \Delta t + \delta$. □

According to Lemma 2, we can derive $\lceil \frac{4\mu\cdot\Delta t+\delta}{\delta} \rceil - 1 < \frac{4\mu\cdot\Delta t+\delta}{\delta}$, and thus $\mathcal{TC}_k(r(o)) < 1$. Following existing studies [30], [31], [38] and letting $4\mu \cdot \Delta t + \delta = \pi$, the total cost $\mathcal{F}_k$ is:

$$\mathcal{F}_k = \sum_{o,\tilde{s} \in \Theta_k \wedge o \neq \tilde{s}} \frac{1}{\pi^2} \left(\mathcal{TC}'_k(r(o)) + \alpha \cdot \mathcal{SC}'_k(r(o))\right) \quad s.t.\, \forall o \in \Theta_k\, (d(r(o),o.\tilde{l}) \leq \mu \cdot (o.t - o.\tilde{t})), \tag{6}$$

where $\Theta_k = \mathcal{O}_k \cap (\bigcup_{\tilde{s}\in\mathcal{S}_{k-1}} \mathcal{M}_{k-1}(\tilde{s}))$, $\mathcal{SC}'_k(r(o)) = d(r(o), o.l)^2$, and $\mathcal{TC}'_k(r(o)) = \left(\delta \cdot \left(\left\lceil \frac{d(r(o),\tilde{s}.l)}{\delta} \right\rceil - 1\right)\right)^2$. Formula 6 indicates that we do not smooth the location of o at dt_k if o is not summarized in any minimal group at dt_{k-1}. This is in accordance with the basic idea that we conduct smoothing by exploring the neighboring moving objects. We can now formulate our problem.

Definition 10. *Given a snapshot $\mathcal{O}_k$, a set of previous minimal groups $\bigcup_{\tilde{s}\in\mathcal{S}_{k-1}} \mathcal{M}_{k-1}(\tilde{s})$, a time duration Δt, a speed constraint μ, and parameters α, ρ, δ, ε, and minPts, **evolutionary clustering of moving objects (ECM)**:*

- *finds a set of adjustments $\mathcal{R}_{k_{opt}}$, such that $\mathcal{R}_{k_{opt}} = \arg\min_{\mathcal{R}_k} \mathcal{F}_k$;*
- *computes a set of clusters $\mathcal{C}_k$ over $\mathcal{R}_{k_{opt}}$.*

Specifically, $r_{opt}(o) \in \mathcal{R}_{k_{opt}}$ denotes the optimal adjustment of $o.l$ at dt_k, and this location is used as the previous location of o (i.e. $o.\tilde{l}$) when performing evolutionary clustering at dt_{k+1}. Clearly, the objective function in Formula 6 is neither continuous nor differentiable. Thus, computing the optimal adjustments using existing solvers involves iterative processes [43] that are too expensive for online scenarios. We thus prove that Formula 6 can be solved approximately in linear time in Section IV.

IV. Computation of Adjustments

Given the current time step dt_k, we start by reformulating the total cost $\mathcal{F}_k$ (cf. Formula 6) as follows.

$$\begin{aligned}\mathcal{F}_k &= \sum_{\tilde{s}\in\mathcal{S}_{k-1}} f_k(\tilde{s}) \\ &= \sum_{\tilde{s}\in\mathcal{S}_{k-1}} \sum_{o\in\Omega} \left(\mathcal{TC}'_k(r(o)) + \alpha \cdot \mathcal{SC}'_k(r(o))\right) \qquad (7) \\ &s.t. \forall o \in \Theta_k \left(d(r(o), o.\tilde{l}) \le \mu \cdot (o.t - o.\tilde{t})\right),\end{aligned}$$

where $\Theta_k = \mathcal{O}_k \cap (\bigcup_{\tilde{s}\in\mathcal{S}_{k-1}} \mathcal{M}_{k-1}(\tilde{s}))$, $\Omega = \mathcal{M}_{k-1}(\tilde{s})\backslash\{\tilde{s}\}$, $r(o)$ is the adjustment of $o.l$ at dt_k, $\tilde{s}$ is the seed point of o at dt_{k-1}, $\tilde{s}.l$ is the location of $\tilde{s}$ at dt_k, and $f_k(\tilde{s})$ is the sum of the total costs of $o \in \Omega$ at dt_k. This way, $\mathcal{F}_k$ is reformulated as the sum of the total costs of moving objects in each minimal groups. We omit the multiplier $\frac{1}{\pi^2}$ from Formula 6 because Δt, μ, and δ are constants and do not affect the results.

A. Linear Time Solution

We show that Formula 7 can be solved approximately in linear time. However, Formula 7 uses each previous seed point $\tilde{s}$ for smoothing, and such points may also exhibit unusual behaviors from dt_{k-1} to dt_k. Moreover, $\tilde{s}$ may not be in $\mathcal{O}_k$. We address these problems in Section IV-B by proposing a seed point shifting strategy, and we assume here that $\tilde{s} \in \mathcal{O}_k$ has already been smoothed, i.e., $r(\tilde{s}) = \tilde{s}.l$.

Lemma 3. *$\mathcal{F}_k$ achieves the minimum value if each $f_k(\tilde{s})$ ($\tilde{s} \in \mathcal{S}_{k-1}$) achieve its minimum value.*

Proof. We have $f_k(\tilde{s}) = \sum_{o\in\Omega} \left(\mathcal{TC}'_k(r(o)) + \alpha \cdot \mathcal{SC}'_k(r(o))\right)$ (cf. Formula 7), where $\Omega = \mathcal{M}_{k-1}(\tilde{s})\backslash\{\tilde{s}\}$. Thus, given the speed constraint μ and the parameter δ for generating minimal groups, $f_k(\tilde{s})$ is only determined by locations of moving objects $o \in \mathcal{M}_{k-1}(\tilde{s})$. Since we require $\mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{M}_{k-1}(\tilde{s}') = \emptyset$ in Def. 9, $f_k(\tilde{s})$ and $f_k(\tilde{s}')$ ($\tilde{s} \neq \tilde{s}'$) do not affect each other. As $\mathcal{F}_k$ is the sum of $f_k(\tilde{s})$ ($\tilde{s} \in \mathcal{S}_{k-1}$), $\mathcal{F}_k$ achieves its minimum value if each $f_k(\tilde{s})$ ($\tilde{s} \in \mathcal{S}_{k-1}$) achieves its minimum value. □

Following Example 3, $\mathcal{F}_2$ achieves its minimum value if $f_2(o_1)$ and $f_2(o_4)$ achieve their minimum values. Lemma 3 implies that Formula 7 can be solved by minimizing each $f_k(\tilde{s})$. Next, we push down the cost shown in Formula 7 to each pair of o ($o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k\backslash\{\tilde{s}\}$) and $\tilde{s}$.

$$\begin{aligned} f_k(r(o), \tilde{s}) &= \mathcal{TC}'_k(r(o)) + \alpha \cdot \mathcal{SC}'_k(r(o)) \\ s.t.\, d(r(o), o.\tilde{l}) &\le \mu \cdot (o.t - o.\tilde{t}) \end{aligned} \qquad (8)$$

Lemma 4. *$f_k(\tilde{s})$ achieves its minimum value if each $f_k(r(o), \tilde{s})$ ($o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k\backslash\{\tilde{s}\}$) achieves its minimum value.*

Proof. Formula 8 indicates that given the speed constraint μ and the parameter δ for generating minimal group, $f_k(r(o), \tilde{s})$ is only determined by $r(o)$, $o.l$, and $\tilde{s}.l$, where $\tilde{s}.l$ is assumed to have been smoothed and fixed. Thus, $f_k(r(o), \tilde{s})$ is independent of $f_k(r(o'), \tilde{s})$ ($o \neq o'$). As $f_k(\tilde{s})$ is the sum of $f_k(r(o), \tilde{s})$ ($o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k\backslash\{\tilde{s}\}$), $f_k(\tilde{s})$ achieves its minimum value if each $f_k(r(o), \tilde{s})$ ($o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k\backslash\{\tilde{s}\}$) achieves its minimum value. □

In Example 3, $f_2(o_1)$ achieves its minimum values if $f_2(r(o_2), o_1)$ and $f_2(r(o_3), o_1)$ achieve their minimum values. According to Lemma 4, the problem is simplified to computing $r_{opt}(o) = \arg\min_{r(o)} f_k(r(o), \tilde{s})$ ($o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k\backslash\{\tilde{s}\}$). However, it is still intractable as the objective function of Formula 8 is not continuous. We thus aim to transform it into a continuous function. Before doing so, we cover the case where the computation of $r_{opt}(o)$ w.r.t o can be skipped.

Lemma 5. *$o.l = \arg\min_{r(o)} f_k(r(o), \tilde{s})$ if $d(o.l, \tilde{s}.l) \le \delta$.*

Proof. Let $r(o)$ ($r(o) \neq o.l$) be an adjustment of $o.l$. Given $d(o.l, \tilde{s}.l) \le \delta$, $\mathcal{TC}'_k(o.l) = 0 \le \mathcal{TC}'_k(r(o))$. On the other hand, as $d(r(o), o.l) > d(o.l, o.l) = 0$, the snapshot cost $\mathcal{SC}'_k(o.l) = 0 < \mathcal{SC}'_k(r(o))$. Thus, $r_{opt}(o) = o.l$ if $d(o.l, \tilde{s}.l) \le \delta$. □

A previous study [27] smooths the distance between each pair of neighboring nodes regardless of their relative distances. In contrast, Lemma 5 suggests that if a non-seed point remains close to its previous seed point at the current time step, smoothing can be ignored. This avoids over-smoothing. Following Example 3, $o_2.l = \arg\min_{r(o_2)} f_2(r(o_2), o_1)$.

Next, we let $\mathcal{Q}(e, x)$ denote a ***circle*** with center e and radius x. Further, the ***line segment*** that connects locations l and l' is denoted as $se(l.l')$. Finally, the ***intersection*** of circle $\mathcal{Q}(e, x)$ and line segment $se(l, l')$ is denoted as $se(l, l') \oplus \mathcal{Q}(e, x)$. Fig. 3 shows a circle $\mathcal{Q}(o_1.l, \delta)$ that contains $o_1.l$, $o_2.l$, and $o_3.l$. Further, $r(o_6) = se(o_6.l, o_4.l) \oplus \mathcal{Q}(o_4.l, \delta)$.

Lemma 6. *$se(o.l, \tilde{s}.l) \cap \mathcal{Q}(o.\tilde{l}, \mu \cdot (o.t - o.\tilde{t})) \neq \emptyset$.*

Proof. In Section III-B, we constrain $d(o.l, o.\tilde{l}) \le \mu \cdot (o.t - o.\tilde{t})$ before smoothing, which implies that $o.l \in \mathcal{Q}(o.\tilde{l}, \mu \cdot (o.t - o.\tilde{t}))$. Hence, $se(o.l, \tilde{s}.l) \cap \mathcal{Q}(o.\tilde{l}, \mu \cdot (o.t - o.\tilde{t})) \neq \emptyset$. □

In Fig. 4, it follows from Example 3 and $o_6.t - o_6.\tilde{t} = 3$ that $o_6.l \in se(o_6.l, o_4.l) \cap \mathcal{Q}(o_6.\tilde{l}, 3\mu)$. Next, we show that without

utilizing the speed constraint, an optimal adjustment $r_{opt'}(o)$ of $o.l$ that minimizes $f(r(o), \tilde{s})$ can be derived in constant time. Based on this, we explain how to compute $r_{opt}(o)$ based on $r_{opt'}(o)$.

*a) **Omitting the speed constraint**:* We start by giving Lemma 7.

Lemma 7. *Given $r'(o) \notin se(o.l, \tilde{s}.l)$, $r(o)$ must exist such that $r(o) \in se(o.l, \tilde{s}.l) \wedge f_k(r(o), \tilde{s}) \leq f_k(r'(o), \tilde{s})$.*

Proof. Let $d(o.l, \tilde{s}.l) = y$. First, we prove that $\forall r'(o) \notin \mathcal{Q}(\tilde{s}.l, y)(\exists r(o) \in se(o.l, \tilde{s}.l)(f_k(r'(o), \tilde{s}) \geq f_k(r(o), \tilde{s})))$. Two cases are considered: (i) $d(r'(o), o.l) \leq y$ and (ii) $d(r'(o), o.l) > y$. In the first case, we can always find an adjustment $r(o) \in se(o.l, \tilde{s}.l)$ such that $d(r'(o), o.l) = d(r(o), o.l)$. Hence, $\mathcal{SC}'_k(r'(o)) = \mathcal{SC}'_k(r(o))$. However, we have $\mathcal{TC}'_k(r'(o)) \geq \mathcal{TC}'_k(r(o))$ due to $d(r'(o), \tilde{s}.l) > d(r(o), \tilde{s}.l)$. Thus, $f_k(r'(o), \tilde{s}) \geq f_k(r(o), \tilde{s})$. In Example 3, for $r'''(o_6) \notin \mathcal{Q}(o_4.l, d(o_6.l, o_4.l))$ (see Fig. 3), we can find $r'(o_6) \in se(o_6.l, o_4.l)$ such that $d(o_6, r'(o_6)) = d(o_6, r'''(o_6))$. This way, $\mathcal{SC}'_2(r'''(o_6)) = \mathcal{SC}'_2(r'(o_6))$ and $\mathcal{TC}'_2(r'''(o_6)) \geq \mathcal{TC}'_2(r'(o_6))$, and thus $f_2(r'''(o_6), o_4) \geq f_2(r'(o_6), o_4)$. In the second case, it is clear that $f_k(r'(o), \tilde{s}) > f_k(r(o), \tilde{s})$, because of $\forall r(o) \in se(o.l, \tilde{s}.l)(\mathcal{SC}'_k(r'(o)) > \mathcal{SC}'_k(r(o)) \wedge \mathcal{TC}'_k(r'(o)) \geq \mathcal{TC}'_k(r(o)))$.

Second, we prove that $\forall r'(o) \in \mathcal{Q}(\tilde{s}.l, y) \backslash se(o.l, \tilde{s}.l)(\exists r(o) \in se(o.l, \tilde{s}.l)(f_k(r'(o), \tilde{s}) \geq f_k(r(o), \tilde{s})))$. We can always find $r(o) \in se(o.l, \tilde{s}.l)$ such that $d(r'(o), \tilde{s}.l) = d(r(o), \tilde{s}.l)$. Hence, $\mathcal{TC}'_k(r'(o)) = \mathcal{TC}'_k(r(o))$. However, in this case, $\mathcal{SC}'_k(r'(o)) > \mathcal{SC}'_k(r(o))$ due to $r(o) \in se(o.l, \tilde{s}.l) \wedge r'(o) \notin se(o.l, \tilde{s}.l)$. Thus, we have $f_k(r'(o), \tilde{s}) > f_k(r(o), \tilde{s})$. □

Continuing Example 3, $f(r(o_6), o_4.l) \leq f(r''(o_6), o_4.l)$ and $f(r'(o_6), o_4.l) \leq f(r''(o_6), o_4.l)$ due to $r''(o_6) \notin se(o_6.l, o_4.l)$ (see Fig. 3). Lemma 7 indicates that if we ignore the speed constraint in Formula 8, we can search $r_{opt'}(o)$ just on $se(o.l, \tilde{s}.l)$ without missing any result. Next, we let $d(r(o), \tilde{s}.l) = b \cdot \delta$ and $d(r_{opt'}(o), \tilde{s}.l) = b_{opt'} \cdot \delta$ and further narrow the range of $r_{opt'}(o) \in se(o.l, \tilde{s}.l)$.

Lemma 8. *If $d(o.l, \tilde{s}.l) > \delta$ then $b_{opt'} \in \{\mathbf{N}^+ \cap [1, \lfloor\lambda\rfloor]\} \cup \lambda$, where $\lambda = \frac{d(o.l, \tilde{s}.l)}{\delta}$ and $\mathbf{N}^+$ is the positive numbers.*

Proof. We start by proving $b_{opt'} \in [1, \lambda]$. We have $r_{opt'}(o) \in se(o.l, \tilde{s}.l)$ according to Lemma 7. Thus, $b_{opt'} \cdot \delta \leq d(o.l, \tilde{s}.l)$, i.e., $b_{opt'} \leq \lambda$. Moreover, given $r(o), r'(o) \in se(o.l, \tilde{s}.l)$ such that $d(r'(o), \tilde{s}.l) = b' \cdot \delta \wedge d(r(o), \tilde{s}.l) = b \cdot \delta \wedge 0 < b' < 1 \wedge b = 1$, $\mathcal{TC}'(r'(o)) = \mathcal{TC}'(r(o))$ due to $\lceil d(r(o), \tilde{s}.l)/\delta \rceil = \lceil d(r'(o), \tilde{s}.l)/\delta \rceil$. Further, if follows from $r(o), r'(o) \in se(o.l, \tilde{s}.l)$ that $d(o.l, r'(o)) = d(o.l, \tilde{s}.l) - b' \cdot \delta$ and $d(o.l, r(o)) = d(o.l, \tilde{s}.l) - b \cdot \delta$. Since $b' \cdot \delta < b \cdot \delta$, we have $\mathcal{SC}'(r(o)) < \mathcal{SC}'(r'(o))$. Thus, $f(r(o), \tilde{s}.l) < f(r'(o), \tilde{s}.l)$ and $b_{opt'} \in [1, \lambda]$.

Next, we prove $b_{opt'} \in \{\mathbf{N}^+ \cap [1, \lfloor\lambda\rfloor]\}$ if $1 \leq b \leq \lfloor\lambda\rfloor$. Given $r'(o), r(o) \in se(o.l, \tilde{s}.l)$ such that $(b-1) \cdot \delta < d(r'(o), \tilde{s}.l) < b \cdot \delta \wedge d(r(o), \tilde{s}.l) = b \cdot \delta \wedge 1 \leq b \leq \lfloor\lambda\rfloor \wedge b \in \mathbf{N}^+$, $\mathcal{TC}'(r'(o)) = \mathcal{TC}'(r(o))$ and $\mathcal{SC}'(r'(o)) > \mathcal{SC}'(r(o))$ due to $\lceil d(r(o), \tilde{s}.l)/\delta \rceil = \lceil d(r'(o), \tilde{s}.l)/\delta \rceil$ and $d(r'(o), o.l) > d(r(o), o.l)$. Thus, $f(r(o), \tilde{s}.l) < f(r'(o), \tilde{s}.l)$ and $b_{opt'} \in \{\mathbf{N}^+ \cap [1, \lfloor\lambda\rfloor]\}$ if $1 \leq b \leq \lfloor\lambda\rfloor$.

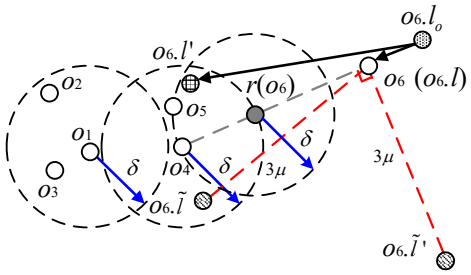


Fig. 4. An example of speed-based pre-processing, $o_6.l_o \rightarrow o_6$, in Fig. 3

Finally, we consider the case where $\lfloor\lambda\rfloor < \lambda$ and prove $b_{opt'} = \lambda$ if $\lfloor\lambda\rfloor < \lambda \wedge \lfloor\lambda\rfloor < b \leq \lambda$. Given $r'(o) \in se(o.l, \tilde{s}.l)$ such that $b \cdot \delta = d(r'(o), \tilde{s}.l) \wedge \lfloor\lambda\rfloor < b \leq \lambda$, similar to the above proof, $\mathcal{TC}'(r'(o)) = \mathcal{TC}'(o.l) \wedge \mathcal{SC}'(r'(o)) > \mathcal{SC}'(o.l)$. Thus, $f(o.l, \tilde{s}.l) < f(r'(o), \tilde{s}.l)$. Since $d(o.l, \tilde{s}.l) = \lambda \cdot \delta$, $b_{opt'} = \lambda$ if $\lfloor\lambda\rfloor < \lambda \wedge \lfloor\lambda\rfloor < b \leq \lambda$. □

Continuing Example 3 and given $d(o_4, r(o_6)) = \delta$ and $d(o_4, r'(o_6)) = 2\delta$ (see Fig. 3), we have $r_{opt'}(o_6) \in \{o_6.l, r(o_6), r'(o_6)\}$. Based on Lemmas 5–8, we let $d(r(o), \tilde{s}.l) = b \cdot \delta\, (b \in \{\mathbf{N}^+ \cap [1, \lfloor\lambda\rfloor]\} \cup \lambda)$ and simplify Formula 8 to the following function:

$$f_k(r(o), \tilde{s}) = \begin{cases} (\delta \cdot (b-1))^2 + \alpha \cdot (d - b \cdot \delta)^2, & b \in \Gamma \\ (\delta \cdot (\lceil b \rceil - 1))^2 + \alpha \cdot (d - b \cdot \delta)^2, & b = \lambda, \end{cases} \tag{9}$$

where $d = d(o.l, \tilde{s}.l)\, (> \delta)$, $\Gamma = \{\mathbf{N}^+ \cap [1, \lfloor\lambda\rfloor]\}$, and $\lambda = \frac{d(o.l, \tilde{s}.l)}{\delta}$. The objective function in Formula 9 is quadratic if $b \in \Gamma$; otherwise, its value is unique. Thus, the $b_{opt'}\, (\in \{\mathbf{N}^+ \cap [1, \lfloor\lambda\rfloor]\} \cup \lambda)$ that minimizes the function can be obtained in constant time without sacrificing accuracy.

Example 4. *Continuing Example 3 and given $d(o_6.l, o_4.l) = 25$, $\alpha = 0.3$, and $\delta = 10$, $b_{opt'} = 1$ and $r_{opt'}(o_6) = r(o_6)$.*

*b) **Introducing the speed constraint**:* Recall that $r_{opt'}(o)$ is the optimal adjustment of $o.l$ when disregarding the speed constraint in Formula 8, while $r_{opt}(o)$ takes the constraint into account. We have narrowed the range of $r_{opt'}$ to a set of discrete locations on $se(o.l, \tilde{s}.l)$ without sacrificing accuracy. Further, if $r_{opt'} \in \mathcal{Q}(o.\tilde{l}, \mu \cdot \Delta t)$ then $r_{opt}(o) = r_{opt'}(o)$. However, if $r_{opt'} \notin \mathcal{Q}(o.\tilde{l}, \mu \cdot \Delta t)$, $r_{opt'}(o)$ is an invalid adjustment. In this case, letting $d(r_{opt}(o), \tilde{s}.l) = b_{opt} \cdot \delta$, we propose to approximate r_{opt} by searching only in the narrowed range of $r_{opt'}$, i.e., we propose to compute b_{opt} approximately as follows.

$$\begin{aligned} & b_{opt} = \arg \min_{b \in \{\mathbf{N}^+ \cap [1, \lambda]\} \cup \lambda} |b - b_{opt'}| \\ & s.t.\ \mathcal{Q}(o.\tilde{l}, \mu \cdot (o.t - o.\tilde{t})) \cap \mathcal{Q}(\tilde{s}.l, b \cdot \delta) \neq \emptyset, \end{aligned} \tag{10}$$

where $\mathcal{Q}(o.\tilde{l}, \mu \cdot (o.t - o.\tilde{t})) \cap \mathcal{Q}(\tilde{s}.l, b \cdot \delta) \neq \emptyset$ indicates that $r_{opt}(o) \in \mathcal{Q}(o.\tilde{l}, \mu \cdot (o.t - o.\tilde{t}))$ must hold due to $d(r_{opt}(o), \tilde{s}.l) = b_{opt} \cdot \delta$. After getting $b_{opt'}$, b_{opt} can be located according to $se(o.l, \tilde{s}.l) \oplus \mathcal{Q}(o.\tilde{l}, \mu \cdot (o.t - o.\tilde{t}))$ in constant time. Following Example 4 and given $o_6.t - o_6.\tilde{t} = 3$, $r_{opt}(o_6) = r(o_6)$ if $o_6.l_1 = o_6.\tilde{l}$, while $r_{opt}(o_6) = o_6.l$ if $o_6.l_1 = o_6.\tilde{l}'$ (shown in Fig. 4). Specifically, in the latter case, $o_6.l$ is the only feasible solution of $r_{opt}(o_6)$ according to Formula 10, as $se(o_6.l, o_4.l) \oplus \mathcal{Q}(o_6.\tilde{l}', 3\mu) = o_6.l$. So far,

the efficiency of computing $r_{opt}(o)$ using Formula 8 has been improved to $O(1)$ time complexity.

B. Shifting of Seed Points

Section IV-A assumes that the previous seed point $\tilde{s}.l$ evolves gradually when smoothing $o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k \backslash \{\tilde{s}\}$ at dt_k, which is not always true. Thus, $\tilde{s}.l$ may also need to be smoothed. We first select a "pivot" for smoothing $\tilde{s}.l$. An existing method [43] maps a noisy point to the accurate point that is closest to it in a batch mode. Inspired by this, we smooth $\tilde{s}.l$ using $o.l\,(o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k)$, which is a set of discrete locations. This is based on the observation that the travel companions of moving objects evolve gradually. Next, we determine which moving object o should be selected as a "pivot" to smooth $\tilde{s}.l$.

Evolutionary clustering assigns a low cost (cf. Formula 1) if the clusterings change smoothly during a short time period. Since we use cost embedding, we consider the location of o as evolving smoothly if the distance between o and o' ($o' \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k \backslash \{o\}$) varies only little between adjacent time steps. This is essentially evaluated by $f_k(o)$ (cf. Formula 7), which measures the cost of smoothing o' ($o' \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k \backslash \{o\}$) according to o. Hence, we select the "pivot" for smoothing $\tilde{s}.l$ using the following formula:

$$\tilde{s}_{new} = \arg \min_{o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k} f_k(o) \tag{11}$$

After obtaining a "pivot" $\tilde{s}_{new}$, instead of first smoothing $\tilde{s}.l$ according to $\tilde{s}_{new}.l$ and then smoothing $o.l\,(o \in \mathcal{M}_{k-1}(\tilde{s}) \backslash \{\tilde{s}\})$ by $\tilde{s}.l$, we shift the seed point of $o \in \mathcal{M}_{k-1}(\tilde{s}) \backslash \{\tilde{s}\}$ from $\tilde{s}$ to $\tilde{s}_{new}$ and use $\tilde{s}_{new}$ to smooth the locations of other moving objects $o\,(o \neq \tilde{s}_{new})$. The reason is twofold: (i) by Formula 11, $\tilde{s}_{new}$ is the moving object with the smoothest movement from dt_{k-1} to dt_k among moving objects in $\mathcal{M}_{k-1}(\tilde{s})$, and thus it is less important to smooth it; (ii) we can save $|\mathcal{M}_{k-1}(\tilde{s})|$ computations in Formula 7. Formula 11 suggests that the seed point may not be shifted, i.e., $\tilde{s}_{new}$ may be $\tilde{s}$. Intuitively, when computing $\tilde{s}_{new}$, the locations of all moving objects in their corresponding minimal groups are smoothed; and with the seed point shifting strategy, smoothing does not require that the previous seed point is active at the current time step. The time complexity of smoothing a minimal group $\mathcal{M}_{k-1}(\tilde{s})$ is $O(|\mathcal{M}_{k-1}(\tilde{s})|^2)$.

C. Speed-based Pre-processing

We present the pre-processing that forces each to-be-smoothed moving object $o \in \mathcal{O}_k \cap (\bigcup_{\tilde{s} \in \mathcal{S}_{k-1}} \mathcal{M}_{k-1}(\tilde{s}))$ to observe the speed constraint. The pre-processing guarantees the correctness of the normalization of the snapshot and historical costs. We denote the location of o before pre-processing as $o.l_o$ and the possible location after as $o.l_p$.

A naive pre-processing strategy is to map $o.l_o$ to a random location on or inside $\mathcal{Q}(o.\tilde{l}, \mu \cdot (o.t - o.\tilde{t}))$. However, this random strategy may make the smoothing less reasonable.

Example 5. *Continuing Example 4, Fig. 4 shows two possible locations $o_6.l$ and $o_6.l'$ of $o_6.l_o$, both of which are chosen at random while observing the speed constraint, i.e., they are located on and inside $\mathcal{Q}(o_6.\tilde{l}, 3\mu)$, respectively. Since $d(o_6.l', o_4.l) < \delta < d(o_6.l, o_4.l)$, the adjustment of $o_6.l'$ is $o_6.l'$ itself while that of $o_6.l$ is $r(o_6)$.*

In this example, $o_6.l'$ is less reasonable than $r(o_6)$. Specifically, according to the minimum change principle [43], the changes to the data distribution made by the speed-based pre-processing and the neighbor-based smoothing should be as small as possible. However, considering $d(o_6.l_o, o_4.l)$, $o_6.l'$ is too close to $o_4.l$ compared with $o_6.l$ and $r(o_6)$. Given a pre-processed location $o.l$, its change due to smoothing has already been minimized through Formula 8. Thus, to satisfy the minimum change principle, we just need to make the impact of speed-based pre-processing on neighbor-based smoothing as small as possible. Hence, we find the pre-processed $o.l$ via the speed constraint as follows.

$$\begin{aligned} o.l &= \arg \min_{o.l_p} |d(o.l_p, \tilde{s}.l) - d(o.l_o, \tilde{s}.l)| \\ s.t.\, &d(o.l_p - o.\tilde{l}) \leq \mu \cdot (o.t - o.\tilde{t}) \end{aligned} \tag{12}$$

This suggests that the difference between $d(o.l, \tilde{s}.l)$ and $d(o.l_o, \tilde{s}.l))$ is expected to be as small as possible, in order to mitigate the effect of speed-based pre-processing on computing $r(o)$. Before applying Formula 12, we pre-process $\tilde{s}.l$ so that it also follows the speed constraint:

$$\begin{aligned} \tilde{s}.l &= \arg \min_{\tilde{s}.l_p} d(\tilde{s}.l_o, \tilde{s}.l_p) \\ s.t.\, &d(\tilde{s}.l_p - \tilde{s}.\tilde{l}) \leq \mu \cdot (\tilde{s}.t - \tilde{s}.\tilde{t}) \end{aligned} \tag{13}$$

As $\tilde{s}$ is not smoothed by any moving objects in $\mathcal{M}_{k-1}(\tilde{s})$ (cf. Section IV-A), Formula 13 lets the closest location to $\tilde{s}.l_p$ satisfying the speed constraint be $\tilde{s}.l$. This is also in accordance with the minimum change principle [43]. According to the seed point shifting strategy, we examine each $o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k$ to identify the most smoothly moving object as $\tilde{s}_{new}$. Thus, before this process, we have to force each $o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k \backslash \tilde{s}$ to follow the speed constraint w.r.t. the current to-be-examined seed point $\tilde{s}$, i.e., computing $o.l$ w.r.t. $\tilde{s}.l$ according to Formula 12. Obviously, speed-based pre-processing is only needed when $d(o.l_o - o.\tilde{l}) > \mu \cdot (o.t - o.\tilde{t})$; otherwise, $o.l = o.l_o$. Clearly, Formulas 12 and 13 can be computed in constant time.

V. Algorithms

We first introduce a grid index and then present all algorithms needed to enable evolutionary clustering, together with a set of optimization techniques.

A. Grid Index

We use a grid index [44], [45] to accelerate the algorithms we propose. Fig. 3 shows an example index. The diagonal of each grid cell has length ε, which is the parameter used in DBSCAN [41]. This accelerates the process of finding core points [44]. Given the location of a moving object and the grid cell width $\frac{\varepsilon}{\sqrt{2}}$, it is straightforward to compute the grid cell that it belongs to [45]. If a moving object is located on a border between grid cells, we assign it to the top-right cell. For example, in Fig. 3, $r''(o_6)$ is assigned in g_{64}. The number of moving objects that fall into grid cell g is denoted as $|g|$. Next, we introduce the concept of h-close.

Algorithm 1: Generating minimal groups

Input: a snapshot $\mathcal{O}_k$, a threshold δ
Output: a set of seed point $\mathcal{S}_k$ and minimal groups $\bigcup_{s\in\mathcal{S}_k}\mathcal{M}_k(s)$

1 **for** *each* $o \in \mathcal{O}_k$ **do**
2 | $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup o$ if $\forall s \in \mathcal{S}_k(d(o.l, s.l) > \delta)$
3 **end**
4 **for** *each* $o \in \mathcal{O}_k \backslash \mathcal{S}_k$ **do**
5 | $\mathcal{M}_k(s) \leftarrow \mathcal{M}_k(s) \cup o$ if $s \leftarrow \arg\min_{\{o\in g \wedge s\in\mathcal{I}_\delta(g)\wedge d(o.l,s.l)\le\delta\}} d(o.l, s.l)$
6 **end**
7 **return** $\mathcal{S}_k$ *and* $\bigcup_{s\in\mathcal{S}_k}\mathcal{M}_k(s)$

Definition 11. *Two grid cells g and g' are **h-close** if $\exists o \in g \wedge \exists o' \in g'\,(d(o, o') \le h)$. The set of h-close grid cells of g is denoted $\mathcal{I}_h(g)$.*

We utilize two distance parameters, i.e., ε for clustering (cf. Def. 5) and δ for finding minimal groups (cf. Def. 9). Thus, we only need to consider $\mathcal{I}_h(g)$, where $h = \varepsilon, \delta$. Following an existing work [45], we define $\mathcal{I}_\varepsilon(g_{ij}) = \Lambda_{ij}\backslash(g_{i_1j_1} \cup g_{i_1j_2} \cup g_{i_2j_1} \cup g_{i_2j_2})$, where $\Lambda_{ij} = \{g_{i'j'} | i_1 \le i' \le i_2 \wedge j_1 \le j' \le j_2 \wedge i_1 = i-2 \wedge i_2 = i+2 \wedge j_1 = j-2 \wedge j_2 = j+2\}$. Since we set $\delta \le \varepsilon$, next we only need to compute $\mathcal{I}_h(g)$, such that $0 < h < \varepsilon$.

Lemma 9. *$\mathcal{I}_h(g_{ij}) = \mathcal{I}_\varepsilon(g_{ij})$ if $\frac{\varepsilon}{\sqrt{2}} < h < \varepsilon$; otherwise, $\mathcal{I}_h(g_{ij}) = \{g_{i'j'} | i-1 \le i' \le i+1 \wedge j-1 \le j' \le j+1\}$.*

Proof. Since we set $0 < h < \varepsilon$, we have $\mathcal{I}_h(g_{ij}) \subset \mathcal{I}_\varepsilon(g_{ij}) = \Lambda_{ij}\backslash(g_{i_1j_1} \cup g_{i_1j_2} \cup g_{i_2j_1} \cup g_{i_2j_2})$, where $\Lambda_{ij} = \{g_{i'j'} | i_1 \le i' \le i_2 \wedge j_1 \le j' \le j_2 \wedge i_1 = i-2 \wedge i_2 = i+2 \wedge j_1 = j-2 \wedge j_2 = j+2\}$. For example, in Fig. 3, $\mathcal{I}_\varepsilon(g_{44}) = \Lambda_{44}\backslash(g_{22} \cup g_{26} \cup g_{62} \cup g_{66})$, where $\Lambda_{44} = \{g_{ij} | 2 \le i, j \le 6\}$. Then, we partition $\mathcal{I}_\varepsilon(g_{ij})$ into Λ'_{ij} and Λ''_{ij}, where $\Lambda'_{ij} = \{g_{i'j'} | i-1 \le i' \le i+1 \wedge j-1 \le j' \le j+1\}$ and $\Lambda''_{ij} = \mathcal{I}_\varepsilon(g_{ij})\backslash\Lambda'_{ij}$. For example, in Fig. 3, $\Lambda'_{44} = \{g_{i'j'} | 3 \le i', j' \le 5\}$. Intuitively, $\forall g \in \Lambda''_{ij} \wedge \forall o' \in g \wedge \forall o \in g_{ij}\ (d(o, o') > \frac{\varepsilon}{\sqrt{2}})$; while $\forall g' \in \Lambda'_{ij} \wedge \exists o'' \in g' \wedge \exists o \in g_{ij}\ (d(o, o') \le \frac{\varepsilon}{\sqrt{2}})$. Thus, according to Def. 11, we have $\mathcal{I}_h(g_{ij}) = \mathcal{I}_\varepsilon(g_{ij})$ if $\frac{\varepsilon}{\sqrt{2}} < h < \varepsilon$ and $\mathcal{I}_h(g_{ij}) = \{g_{i'j'} | i-1 \le i' \le i+1 \wedge j-1 \le j' \le j+1\}$ if $\frac{\varepsilon}{\sqrt{2}} \ge h$. □

In Fig. 3, given $\delta \le \frac{\varepsilon}{\sqrt{2}}$, $\mathcal{I}_\delta(g_{44}) = \{g_{ij} | 3 \le i, j \le 5\}$.

B. Generating Minimal Groups

Sections III and IV indicate that o is smoothed at dt_k if $\exists\mathcal{M}_{k-1}(\tilde{s})(o \in \mathcal{M}_{k-1}(\tilde{s}))$; otherwise, o is considered as an "outlier," to which neighbor-based smoothing cannot be applied. In order to smooth as many moving objects as possible, we thus aim to include as many moving objects as possible in the minimal groups. According to Def. 9, a set of minimal groups should satisfy (i) $\forall o \in \mathcal{M}_k(s)\ (d(o, s) \le \delta)$ and (ii) $|\mathcal{M}_k(s)| \ge \rho$. The former ensures the closeness of moving objects within a minimal group, while the latter guarantees that there is at least one moving object o in a minimal group that is not located on the border of a cluster. Considering the above analysis and two constraints, the optimal set of seed points $\mathcal{S}_{k_{opt}} = \arg\max_{\{\mathcal{S}_k\subseteq\mathcal{O}_k\}} \left|\bigcup_{s\in\mathcal{S}_k}\mathcal{M}_k(s)\right|$. Clearly, computing $\mathcal{S}_{k_{opt}}$ requires us to enumerate all the possible combinations, which is infeasible.

Algorithm 2: Smoothing

Input: a minimal group $\mathcal{M}_{k-1}(\tilde{s})$
Output: a set of adjustments $\mathcal{R}_k(\mathcal{M}_{k-1}(\tilde{s}))$

1 *sum* $\leftarrow 0$, $\mathcal{R}_k(\mathcal{M}_{k-1}(\tilde{s})) \leftarrow \emptyset$, *glp* $\leftarrow \infty$, $\tilde{s}_{new} \leftarrow$ *null*, $\mathcal{A} \leftarrow \emptyset$
2 **for** *each* $o' \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k$ **do**
3 | compute $o'.l$ according to Formula 13, $\mathcal{A} \leftarrow o'.l$, *sum* $\leftarrow 0$
4 | **for** *each* $o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k \backslash \{o'\}$ **do**
5 | | compute $o.l$ according to Formula 12
6 | | compute $r_{opt}(o)$ according to Lemma 5 and Formulas 9–10
7 | | *sum* $\leftarrow$ *sum* $+ f_k(r_{opt}(o), o')$
8 | | **if** *sum* $\ge$ *glp* **then**
9 | | | break /* o' must not be $\tilde{s}_{new}$ */
10 | | **end**
11 | | $\mathcal{A} \leftarrow \mathcal{A} \cup r_{opt}(o)$
12 | **end**
13 | **if** $|\mathcal{A}| = |\mathcal{M}_{k-1}(\tilde{s})|$ **then**
14 | | *glp* $\leftarrow$ *sum*, $\tilde{s}_{new} \leftarrow o'$, $\mathcal{R}_k(\mathcal{M}_{k-1}(\tilde{s})) \leftarrow \mathcal{A}$
15 | **end**
16 **end**
17 **return** $\mathcal{R}_k(\mathcal{M}_{k-1}(\tilde{s}))$

Therefore, we propose a greedy algorithm, shown in Algorithm 1, for computing a set of minimal groups at dt_k. Each o is mapped to a grid cell g before generating minimal groups. If we generate a minimal group summarized by s once assigning s as a seed point, a non-seed point o attached to a minimal group $\mathcal{M}_k(s)$ at this time may turn out to be closer to another newly obtained seed point s'. This incurs repeated processes for finding a seed point for o. Thus, we first greedily determine $\mathcal{S}_k$ and then generate minimal groups according to $\mathcal{S}_k$. This way, we compute the seed point s for each $o \in \mathcal{O}_k$ exactly once. According to Def. 11, we will not miss any possible seed point for $o \in g$ by searching $\mathcal{I}_\delta(g)$ rather than $\mathcal{S}_k$ (Line 5). Specifically, $\mathcal{I}_\delta(g)$ is obtained directly according to Lemma 9. Note that Algorithm 1 generates minimal groups $\mathcal{M}_k(s)$ such that $|\mathcal{M}_k(s)| < \rho$. We simply ignore these during smoothing.

C. Evolutionary Clustering

*a) **Smoothing**:* Algorithm 2 is the smoothing algorithm. We maintain *glp* to record the current minimal $f_k(o') = \sum_{o\in\mathcal{M}_{k-1}(\tilde{s})\cap\mathcal{O}_k\backslash\{o'\}} f_k(r(o), o')\ (o' \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k)$ and maintain $\mathcal{A}$ to record adjustments w.r.t. $o'.l$ (Line 1). The computation of $f_k(o')$ is terminated early if its current value exceeds *glp* (Lines 8–9). As can be seen, if $o'\ (o' \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k)$ is identified as the moving object with the smoothest movement in its minimal group, the set of adjustments $r_{opt}(o)\ (o \in \mathcal{M}_{k-1}(\tilde{s}) \cap \mathcal{O}_k\backslash\{o'\})$ w.r.t $o'.l$ is returned, i.e., $\mathcal{R}_k(\mathcal{M}_{k-1}(\tilde{s}))$.

*b) **Optimizing modularity**:* Modularity is a well-known quality measure for clustering [27], [34], which is computed as follows.

$$QS = \sum_{c\in\mathcal{C}_k}\left(\frac{IS(c)}{TS} - \left(\frac{DS(c)}{TS}\right)^2\right) \tag{14}$$

Algorithm 3: Evolutionary clustering (ECM)

Input: a snapshot $\mathcal{O}_k$, a clustering result $\mathcal{C}_{k-1}$, a set of minimal groups $\bigcup_{s \in \mathcal{S}_{k-1}} \mathcal{M}_{k-1}(\tilde{s})$, a speed constraint μ, thresholds δ, ρ, α, ε_{k-1}, $\Delta\varepsilon$, *minPts*

Output: a clustering result C_k

1 smooth $\mathcal{O}_k$ and get the set of adjustments $\mathcal{R}_{k_{opt}}$
2 build a grid index according to ε_{k-1} and $\mathcal{R}_{k_{opt}}$
3 generate minimal groups based on $\mathcal{R}_{k_{opt}}$
4 cluster $\mathcal{R}_{k_{opt}}$ to get $\mathcal{C}_k$ /* DBSCAN */
5 update ε_{k-1} to ε_k according to $\mathcal{C}_k$
6 map $c \in \mathcal{C}_k$ to $c' \in \mathcal{C}_{k-1}$ /* reference [27] */
7 **return** $\mathcal{C}_k$

Here, *TS* is the sum of the closeness of all pairs of moving objects' locations in $\mathcal{C}_k$, *IS*(c) is the sum of the closeness of all pairs of moving objects' locations in cluster c, *DS*(c) is the sum of the closeness between a moving object in cluster c and any moving object in cluster c' ($c' \in \mathcal{C}_k \backslash \{c\}$). A high *QS* indicates a good clustering result. The closeness between any two moving objects o and o' is defined as $1/d(o.l, o'.l)$.

A previous study [27] iteratively adjusts ε to find the (locally) optimal *QS* as well as the clustering result at each time step. Specifically, it increases or decreases ε by $\Delta\varepsilon$ in each iteration. This iterative optimization of modularity has a relatively high time cost. We improve the cost by only updating ε at dt_k (denoted as ε_k) once to "approach" the (locally) optimal modularity of $\mathcal{C}_k$. Specifically, ε is still obtained by the iterative optimization at the first time step.

*c) **Grid index and minimal group based accelerations**:* As we set the grid cell width to $\frac{\varepsilon}{\sqrt{2}}$, each $o \in g$ is a core point if $|g| \geq$ *minPts* [44]. Similarly, if $|\mathcal{M}_k(s)| \geq$ *minPts*, s is a core point. This accelerates the search for core points as well as DBSCAN.

*d) **Evolutionary clustering of moving objects**:* All pieces are now in place to present the ECM algorithm in Algorithm 3. The sub-procedures in lines 1–5 are detailed in the previous sections. Finally, we connect clusters at adjacent time steps with each other (Line 6) as proposed in the literature [27]. This mapping aims to find the evolving, forming, and dissolving relationships between $c \in \mathcal{C}_{k-1}$ and $c' \in \mathcal{C}_k$. The details of the mapping are available elsewhere [27]. The time complexity of ECM at time step dt_k is $O(|\mathcal{O}_k|^2)$.

VI. Experiments

A. Experimental Design

*a) **Datasets**:* Three real-life datasets, Chengdu (CD)[1], Automatic Identification System data (AIS)[2], and T-Drive (TD)[3] are used. CD is collected from 13,431 taxis over one day (Aug. 30, 2014) in Chengdu, China and contains 30 million GPS records. AIS is collected by the U.S. Coast Guard and records the locations of large vessels in U.S. and international waters. We use data from March 22 to 28, 2021 covering 16,816 vessels and containing 55 million records. TD is collected from 10,357 taxis over one week (from February 2 to 8, 2008) in Beijing, China and contains 15 million GPS records [46], [47] . The sample intervals of CD, AIS, and TD are 10s, 120s, and 300s, respectively.

[1]This is a private dataset.

[2]https://marinecadastre.gov/ais/

[3]https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/

TABLE I
Parameter ranges and default values

Parameter	Range
minPts	2, **3**, 4, **5**, 6, 7, 9, 11
δ (km)	0.6, 0.7, 0.8, 0.9, **1.0**, **1.1**, 27, 29, 31, 33, **35**
α	**0.1**, 0.3, 0.5, 0.7, 0.9
ρ	2, 3, 4, **5**, 6, 7, 8, 11, 14

*b) **Performance metrics**:* We adopt *modularity QS* (cf. Formula 14) to measure the quality of clustering and *normalized mutual information NMI* [48] to measure the similarity between two clustering results obtained at consecutive time steps. We report both *QS* and *NMI* as average values over all time steps. The higher the *QS*, the better the clustering. The higher the *NMI*, the smoother the evolution of clusters. Efficiency is measured as the average processing time per record at each time step, denoted as Time.

*c) **Comparison algorithms and experimental settings**:* We compare with three existing methods:

- KH [27] is a representative density-based evolutionary clustering method for dynamic networks. It evaluates costs at the individual distance level to improve efficiency.
- OST [17] is the state-of-the-art method of trajectory clustering. It exploits a novel compact synopsis and sliding windows to facilitate clustering in streaming settings.
- MC [7] is the state-of-the-art method of moving object clustering. It incrementally maintains clustering features to predict when clusters are to be split, thus avoiding to handle large amounts of events.

We study the effect on performance of the parameters summarized in Table I. Since we propose to smooth the locations of moving objects according to the locations of their neighbors and use minimal groups to summarize neighboring moving objects, we suggest to set $\delta \in [1.8avg_1, 3.0avg_5]$, where avg_1 and avg_5 are the average distances between a moving object and its closest neighbor and its fifth closest neighbor, respectively. We do this because the seed points used to smooth moving objects are generally not the closest neighbors to the moving objects, according to our seed point shifting strategy (cf. Section IV-B) and the algorithm of minimal group generation (cf. Section V-B). As shown in Section V-B, we aim to include as many moving objects as possible when generating minimal groups, so that locations of more objects can be smoothed. Thus, after determining δ, we estimate the average local density of moving objects w.r.t. δ, denoted as ld. Here, we suggest to set $\rho \in [\lfloor 0.4ld \rfloor, \lfloor 0.8ld \rfloor]$, which is below the average local density of the objects. However, ρ should not be set too small because we want to exclude border points when smoothing. Parameters avg_1, avg_5, and *ld* can be estimated by sampling historical data.

Next, parameter α controls the trade-off between snapshot and historical costs. A smaller α generates a smoother result,

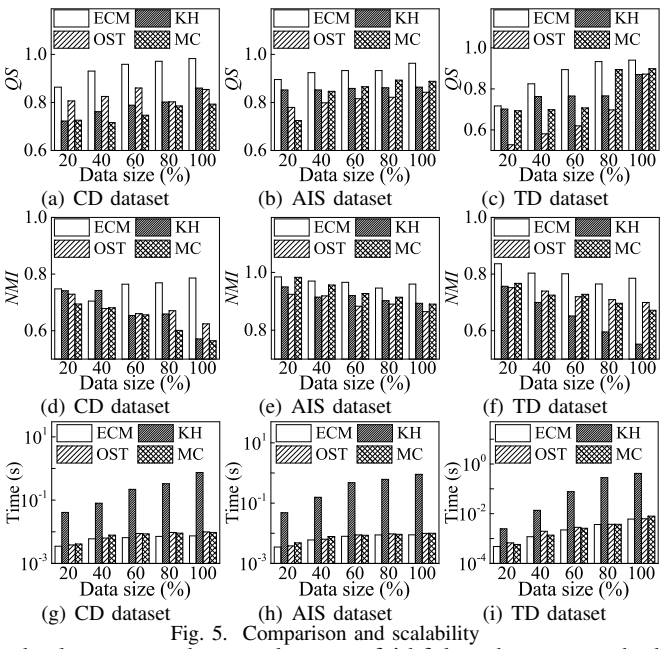


Fig. 5. Comparison and scalability

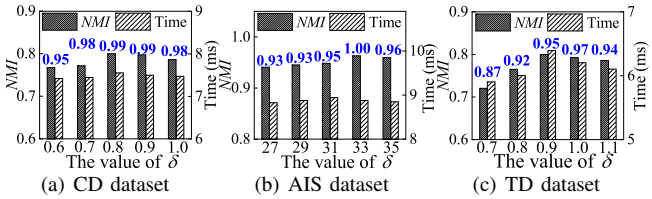


Fig. 6. Effects of varying δ

and a larger α makes results more faithful to the unsmoothed data. Thus, α should be set to a small value, e.g., 0.1, if users aim for very smooth results; and it should be set to a large value, e.g., 0.9, if users prefer results that are close to the original data.

Finally, $\Delta\varepsilon$ is set to 0.05km on both CD and AIS, and is set to 0.01km on TD. The initial value of ε is set to 1.2km, 35km, and 1.2km on CD, AIS, and TD, respectively. As ε is adjusted adaptively to optimize the clustering quality over time, we do not study the effect of the initial value of ε on the performance. Instead, we study the case where ECM does not optimize modularity over time (to be covered in Section VI-D). Note that *minPts* and ε are defined by DBSCAN [41] and that their settings have been studied widely [49], [50]. Thus, we do not provide additional guidance of setting *minPts* and the initial value of ε. All algorithms are implemented in C++, and the experiments are run on a computer with an Intel Core i9-9880H CPU (2.30 GHz) and 32 GB memory.

B. Comparison and Scalability

Fig. 5 shows results when the data size is varied from 20% to 100% of the total data size. ECM outperforms the baselines in terms of all performance metrics mainly due to three reasons: (i) ECM has no iterative processes (except for the initialization) and is accelerated by grid indexing and the proposed optimizing techniques; (ii) ECM takes into account temporal smoothness, which is designed specifically for moving objects; (iii) Objects with locations with the potential to incur mutations of a clustering have their locations adjusted to be closer to neighbors that evolve smoothly, generally decreasing intra-distances and increasing inter-distances. On the other hand, KH computes neighbors of each moving object and iteratively optimizes the clustering at each time step, which degrades efficiency. Moreover, it measures the distance (similarity) between two moving objects (nodes) according to their common neighbors and smooths the distance if it is non-zero. These strategies work very well for dynamic networks, but are not suitable for moving objects. Next, both OST and MC do not consider temporal smoothing when performing clustering. Further, OST needs to continuously update its synopsis structure, while the splitting of large clusters is costly in MC. Thus, the two are slightly less efficient than ECM.

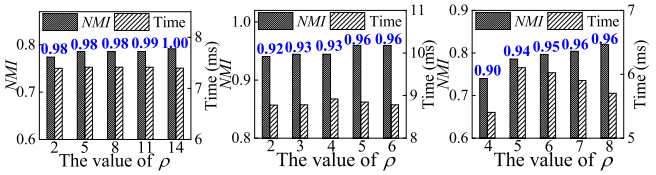
Second, as the data size increases, the *QS* of the four methods generally improves, and the *NMI* of ECM first drops and then grows while that of all baselines generally drops continuously. The findings suggest that as the data size increases, the data distribution becomes denser. Then, both intra-distances and inter-distances of clusters become small, but the former decreases more significantly than the later. Thus, *QS* grows with the data size. Further, small inter-distances of clusters yield "ambiguous" partitions of the clusters, generally resulting in small *NMI*. However, ECM is equipped with a neighbor-based clustering smoothing technique specifically deisgned for moving objects, which proves more effective for dense datasets. Hence, its *NMI* first drops and then grows as the data size increases. Finally, Time of all methods increases with the data size as their efficiency depends on the volume of data arriving at each time step.

C. Parameter Study

Next, we study the effect of δ, ρ, α, and *minPts* on ECM's performance. The blue numbers in Figs. 6–9 are *QS* values.

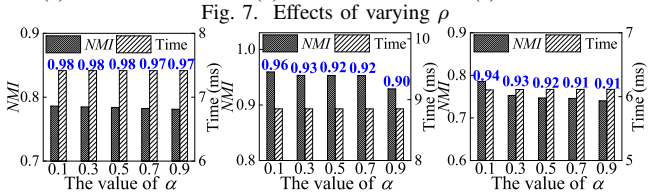
a) ***Effects of varying δ:*** Fig. 6 reports on the effects of varying δ, which is the maximum distance between a non-seed point o and a seed point s in a minimal group. As δ increases, *QS*, *NMI*, and Time first increase and then drop. On the one hand, a too small δ leads to a small number of moving objects forming minimal groups and being smoothed; on the other hand, a too large δ also leads to few smoothing operations, as more pairs of moving objects o and o' ($o, o' \in \mathcal{M}_{k-1}(\tilde{s})$) satisfy $d(o.l, o'.l) \leq \delta$ at dt_k (cf. Section IV-A). Through smoothing, locations with the potential to incur mutation of a clustering are adjusted to be closer to their neighbors that evolve smoothly, generally increasing the inter-distance and decreasing the intra-distance of clustering and thus improving *QS*. Next, a small δ generally leads to small minimal groups, while a large δ generally results in large minimal groups. The former case yields fast smoothing, and the minimal group based acceleration performs better in the latter case.

b) ***Effects of varying ρ:*** Fig. 7 shows the effects of varying ρ, which is the minimum size of a minimal group. As ρ increases, *QS* and *NMI* also increase. This is because moving objects with high local density are generally more stable, i.e., more likely to remain in the same cluster during consecutive time steps, and thus smoothing according to such



(a) CD dataset (b) AIS dataset (c) TD dataset

Fig. 7. Effects of varying ρ



(a) CD dataset (b) AIS dataset (c) TD dataset

Fig. 8. Effects of varying α

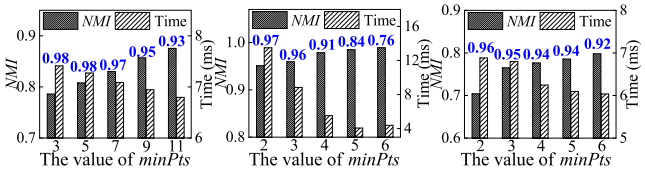
moving objects achieves higher performance. Next, Time first grows and then drops as ρ increases. The reason is similar to that when varying δ, i.e., the sizes of minimal groups generally increase with ρ, which affects the performance of smoothing and the minimal group based acceleration.

*c) **Effects of varying** α:* Fig. 8 shows the effects of varying α, which balances the snapshot cost and the historical cost of Formula 8. First, both *QS* and *NMI* decrease as α increases. On the one hand, a larger α generally leads to a larger distance between two moving objects in the same cluster; on the other hand, the historical cost is given higher weight when α becomes smaller, which renders the evolution of clusters smoother. Second, Time remains almost unchanged when varying α. This is because the efficiency of temporal smoothing is mainly determined by the sizes of minimal groups, which are unaffected by variations in α.

*d) **Effects of varying minPts**:* Fig. 9 shows the effects of *minPts*, which specifies the minimum local density of core points w.r.t. ε in DBSCAN [41]. As can be seen, *QS* drops as *minPts* increases. The findings indicate that the average distance between moving objects in different clusters decreases with *minPts*. Assume that o is a core point when *minPts* is small and that o' is a border point that is density reachable from o. As *minPts* increases, o may no longer be a core point. In this case, o' and o may be density reachable from different core points and may thus be in different clusters, even if $d(o.l, o'.l) \leq \varepsilon$. As a result, the distances between moving objects in different clusters decrease. In addition, *NMI* increases with *minPts*. The findings suggest that with a smaller *minPts*, the moving objects at the border of a cluster are more likely to shift between being core points and being non-core points over time. In this case, clusters fluctuate more between consecutive time steps. Finally, Time generally decreases as *minPts* increases. This is because the number of core points decreases, meaning that fewer computations of density-connected sets are performed [41].

D. Ablation Study

Table II shows the results of an abaltion study using TD and AIS. Specifically, EWG denotes ECM without grid index when generating minimal groups, while EWO denotes ECM without optimizing modularity when clustering. As can be seen, EWG's Time (ms) is higher than ECM's, which shows that the grid indexing is able to improve efficiency. Both *QS* and *NMI* of EWG are the same as those of ECM, which means that the grid indexing does not change the data distribution. Next, both *QS* and *NMI* of EWO are lower than those of ECM, which validates the effectiveness of optimizing modularity. Specifically, since a higher-quality clustering generally has lower intra-distances and higher inter-distances, it is more stable over time than a lower-quality clustering and thus has a higher *NMI*. Further, ECM's Time is not much higher than EWO's, as we exclude the iterative optimization [27] except for the initialization.



(a) CD dataset (b) AIS dataset (c) TD dataset

Fig. 9. Effects of varying *minPts*

TABLE II
ABLATION STUDY

Metric	TD			AIS		
	ECM	EWG	EWO	ECM	EWG	EWO
QS	0.941	0.941	0.926	0.963	0.963	0.905
NMI	0.785	0.785	0.732	0.959	0.959	0.912
Time	6.100	9.533	4.680	8.851	13.192	7.123

E. Accuracy Study

We generate a synthetic dataset Syn, which contains 15 million GPS records from 10,000 moving objects. The ground truth clustering of Syn at each time step is known. Next, we randomly select 5% and 25% of all objects that arrive at each time step, respectively, and add noise on their locations (longitude, latitude). We denote the datasets with noise as Syn-5% and Syn-25%, respectively. The noise follows a normal distribution $\mathcal{N}(0, 0.2^2)$. With this setting, we obtain both large and small noise, where the former changes the membership of an object in a cluster and the latter just slightly changes the location of an object. We study the accuracy of clustering by comparing the clustering obtained from Syn-5% and Syn-25% with the ground truth clustering, respectively. Specifically, the accuracy is evaluated using the Adjusted Rand Index (*ARI*) [51] metric. The higher the *ARI*, the higher the accuracy.

Table III reports the results: all methods achieve higher *ARI*, *QS*, and *NMI* on Syn-5% than on Syn-25%. It is clear that the clustering quality decreases as more noise is introduced. However, ECM's *ARI*, *QS*, and *NMI* change considerably less than do those of the three baselines. This is because ECM uses temporal smoothing to enhance the clustering over low-quality data, i.e., adjusts object locations according to those of their neighbors, which mitigates the adverse effects of noise on the clustering and thus results in clustering with higher accuracy and higher quality. In contrast, each baseline either cannot or is not suitable for addressing the noise when clustering moving objects. This is also the reason why ECM outperforms the baselines in terms of *ARI*, *QS*, and *NMI* on both Syn-5% and Syn-25%. Finally, ECM's Time (ms) is lower than those of the

TABLE III
ACCURACY STUDY

Metric	Syn-5%				Syn-25%			
	ECM	KH	MC	OST	ECM	KH	MC	OST
ARI	0.76	0.67	0.57	0.66	0.72	0.53	0.33	0.55
QS	0.74	0.69	0.67	0.64	0.68	0.59	0.49	0.51
NMI	0.91	0.88	0.85	0.79	0.88	0.71	0.60	0.58
Time	6.40	409	7.96	8.32	6.71	431	7.84	8.66

three baselines, because it has no iterative processes (beyond the initialization) and is accelerated by grid indexing and the proposed optimizing techniques.

VII. Related Work

A. Spatio-temporal Clustering

Spatio-temporal clustering is the process of grouping objects based on their spatial and temporal similarity. Given the dynamic spatial locations of moving objects, spatio-temporal clustering can be classified into two categories: moving-object clustering and trajectory clustering [52].

The available information for moving object clustering [6]–[11] is only the most recent positions of moving objects, as in the context of real-time monitoring of vehicles in smart-city applications. Put differently, no traces of the past locations are kept [52]. Li et al. [8] employ micro-clustering on moving objects, which enables dynamically maintaining bounding boxes of clusters. Kalnis et al. [9] leverages the fact that clusterings evolve smoothly to decrease the number of moving objects to be processed using an approximate algorithm. The above studies either exhibit high costs of dynamically maintaining their summary data structures or trades clustering quality for efficiency. To improve the performance of moving object clustering, Jensen et al. [7] develop clustering features that can be maintained incrementally in efficient fashion and predict when clusters are to be split to avoid handling large amounts of events. Studies also exist that focus on modeling and monitoring cluster transitions [10], [11].

Trajectory clustering [12]–[19] can be applied when the history of a moving object, i.e., the sequence of spatial locations visited by the moving objects, is available [52]. Such methods generally connect the locations of a moving object by line segments and essentially perform line segment clustering over time. Such methods address the needs for memory-efficient schemes for huge stream volumes and limited system resources and for processing outdated data [17]. OCluST [17], a representative study on trajectory clustering, exploits a sliding window and a novel synopsis data structure to summarize clusters incrementally and to eliminate expired tuples safely.

To the best of our knowledge, no existing studies of moving-object or trajectory clustering exploit temporal smoothness to improve clustering quality. The experimental results validate that ECM is able to outperform state-of-the art methods in both categories [7], [17].

B. Evolutionary Clustering

Evolutionary clustering has been studied to discover evolving community structures in applications such as social [27] and financial networks [28] and recommender systems [29]. Most studies target k-means, agglomerative hierarchical, and spectural clustering [30], [31], [33], [35], [37]. Chakrabarti et al. [30] propose a generic framework for evolutionary clustering. Chi et al. [31] develop two functions for evaluating historical costs, PCQ (Preserving Cluster Quality) and PCM (Preserving Cluster Membership), to improve the stability of clustering. Xu et al. [33] estimate the optimal smoothing parameter α in evolutionary clustering. Recent studies model evolutionary clustering as a multi-objective problem [34], [36], [38], [39] and use genetic algorithms to solve it. Such algorithms are too expensive for online scenarios and are not appropriate for the clustering of spatio-temporal data.

The Kim-Han proposal [27] is the one that is closest to ECM. It uses neighbor-based smoothing and a cost embedding technique that smooths the similarity between each pair of nodes. However, ECM differs significantly from Kim-Han. First, Kim-Han's cost function is designed specifically for nodes in dynamic networks and is neither readily applicable to, or suitable for, spatio-temporal data; in contrast, ECM's cost function is shaped according to the characteristics of the movements of objects. Second, Kim-Han smooths the similarity between each pair of neighboring nodes; in contrast, ECM smooths only the locations of objects that indicate abrupt movements, and smoothing is performed only according to the most smoothly moving neighbors. Finally, Kim-Han includes iterative processes that degrade its efficiency, while ECM achieves $O(|\mathcal{O}_k|^2)$ complexity at each time step in the worst case and adopts a grid index to improve efficiency.

VIII. Conclusion and Future Work

We propose a new framework for evolutionary clustering of moving objects that targets faster and better clustering. We propose so-called snapshot and historical costs specifically for spatio-temporal data, and we formalize the problem of evolutionary clustering of moving objects, called ECM. We formulate ECM as an optimization problem and prove that it can be solved approximately in linear time, which eliminates iterative processes employed in previous proposals and improves efficiency. Further, we propose a minimal-group structure and a seed point shifting strategy that facilitate temporal smoothing. We also present the algorithms necessary to enable evolutionary clustering along with a set of optimization techniques that aim to enhance performance. Extensive experiments with three real-life datasets show that ECM outperforms existing state-of-the-art proposals in terms of clustering quality and running time efficiency.

In future research, it is of interest to deploy ECM on a distributed platform and to exploit more information for smoothing such as road conditions and driver preferences.

Acknowledgments

This work was supported by the DiCyPS and DIREC centers, both funded by Innovation Fund Denmark, and the NSFC under Grants No. 62102351, 62025206, and 61972338. Lu Chen is the corresponding author.

REFERENCES

[1] R. Zhen, M. Riveiro, and Y. Jin, "A novel analytic framework of real-time multi-vessel collision risk assessment for maritime traffic surveillance," *Ocean Eng.*, vol. 145, pp. 492–501, 2017.

[2] B. T. Morris and M. M. Trivedi, "A survey of vision-based trajectory learning and analysis for surveillance," *IEEE Trans Circuits Syst Video Technol*, vol. 18, no. 8, pp. 1114–1127, 2008.

[3] B. Liu, E. N. de Souza, S. Matwin, and M. Sydow, "Knowledge-based clustering of ship trajectories using density-based approach," in *IEEE BigData*. IEEE, 2014, pp. 603–608.

[4] H. Yuan and G. Li, "A survey of traffic prediction: from spatio-temporal data to intelligent transportation," *Data Science and Engineering*, vol. 6, no. 1, pp. 63–85, 2021.

[5] M. Ma, Q. Luo, Y. Zhou, X. Chen, and L. Li, "An improved animal migration optimization algorithm for clustering analysis," *Discrete Dyn Nat Soc*, vol. 2015, 2015.

[6] S. Har-Peled, "Clustering motion," *DCG*, vol. 31, no. 4, pp. 545–565, 2004.

[7] C. S. Jensen, D. Lin, and B. C. Ooi, "Continuous clustering of moving objects," *TKDE*, vol. 19, no. 9, pp. 1161–1174, 2007.

[8] Y. Li, J. Han, and J. Yang, "Clustering moving objects," in *SIGKDD*, 2004, pp. 617–622.

[9] P. Kalnis, N. Mamoulis, and S. Bakiras, "On discovering moving clusters in spatio-temporal data," in *SSTD*. Springer, 2005, pp. 364–381.

[10] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult, "Monic: modeling and monitoring cluster transitions," in *SIGKDD*, 2006, pp. 706–711.

[11] I. Ntoutsi, M. Spiliopoulou, and Y. Theodoridis, "Tracing cluster transitions for different cluster types." *Control. Cybern.*, vol. 38, no. 1, 2009.

[12] Z. Deng, Y. Hu, M. Zhu, X. Huang, and B. Du, "A scalable and fast optics for clustering trajectory big data," *Cluster Comput*, vol. 18, no. 2, pp. 549–562, 2015.

[13] Y. Yu, Q. Wang, X. Wang, H. Wang, and J. He, "Online clustering for trajectory data stream of moving objects," *Comput. Sci. Inf. Syst.*, vol. 10, no. 3, pp. 1293–1317, 2013.

[14] Y. Yu, Q. Wang, and X. Wang, "Continuous clustering trajectory stream of moving objects," *China Commun.*, vol. 10, no. 9, pp. 120–129, 2013.

[15] Z. Li, J.-G. Lee, X. Li, and J. Han, "Incremental clustering for trajectories," in *DASFAA*. Springer, 2010, pp. 32–46.

[16] T. L. C. Da Silva, K. Zeitouni, and J. A. de Macêdo, "Online clustering of trajectory data stream," in *MDM*, vol. 1. IEEE, 2016, pp. 112–121.

[17] J. Mao, Q. Song, C. Jin, Z. Zhang, and A. Zhou, "Online clustering of streaming trajectories," *Front. Comput. Sci.*, vol. 12, no. 2, pp. 245–263, 2018.

[18] M. Riyadh, N. Mustapha, M. Sulaiman, N. B. M. Sharef *et al.*, "CC_TRS: Continuous clustering of trajectory stream data based on micro cluster life," *Math. Probl. Eng.*, vol. 2017, 2017.

[19] J. Mao, Q. Song, C. Jin, Z. Zhang, and A. Zhou, "Tscluwin: Trajectory stream clustering over sliding window," in *DASFAA*. Springer, 2016, pp. 133–148.

[20] G. Du, L. Zhou, Y. Yang, K. Lü, and L. Wang, "Deep multiple auto-encoder-based multi-view clustering," *Data Science and Engineering*, vol. 6, no. 3, pp. 323–338, 2021.

[21] C. Bettstetter, "Smooth is better than sharp: A random mobility model for simulation of wireless networks," in *MOBICOM*, 2001, pp. 19–27.

[22] L.-A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C.-C. Hung, and W.-C. Peng, "On discovery of traveling companions from streaming trajectories," in *ICDE*. IEEE, 2012, pp. 186–197.

[23] N. S. Nafi, R. H. Khan, J. Y. Khan, and M. Gregory, "A predictive road traffic management system based on vehicular ad-hoc network," in *ATNAC*. IEEE, 2014, pp. 135–140.

[24] V. Patil, P. Singh, S. Parikh, and P. K. Atrey, "Geosclean: Secure cleaning of GPS trajectory data using anomaly detection," in *MIPR*. IEEE, 2018, pp. 166–169.

[25] A. Idrissov and M. A. Nascimento, "A trajectory cleaning framework for trajectory clustering," in *MDC workshop*, 2012, pp. 18–19.

[26] L. Li, X. Chen, Q. Liu, and Z. Bao, "A data-driven approach for GPS trajectory data cleaning," in *DASFAA*. Springer, 2020, pp. 3–19.

[27] M.-S. Kim and J. Han, "A particle-and-density based evolutionary clustering method for dynamic networks," *PVLDB*, vol. 2, no. 1, pp. 622–633, 2009.

[28] D. J. Fenn, M. A. Porter, M. McDonald, S. Williams, N. F. Johnson, and N. S. Jones, "Dynamic communities in multichannel data: An application to the foreign exchange market during the 2007–2008 credit crisis," *J Nonlinear Sci*, vol. 19, no. 3, p. 033119, 2009.

[29] J. Chen, C. Zhao, L. Chen *et al.*, "Collaborative filtering recommendation algorithm based on user correlation and evolutionary clustering," *Complex Syst.*, vol. 6, no. 1, pp. 147–156, 2020.

[30] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *SIGKDD*, 2006, pp. 554–560.

[31] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *SIGKDD*, 2007, pp. 153–162.

[32] M. Gupta, C. C. Aggarwal, J. Han, and Y. Sun, "Evolutionary clustering and analysis of bibliographic networks," in *ASONAM*. IEEE, 2011, pp. 63–70.

[33] K. S. Xu, M. Kliger, and A. O. Hero III, "Adaptive evolutionary clustering," *Data Min Knowl Discov*, vol. 28, no. 2, pp. 304–336, 2014.

[34] Y. Yin, Y. Zhao, H. Li, and X. Dong, "Multi-objective evolutionary clustering for large-scale dynamic community detection," *Inf. Sci.*, vol. 549, pp. 269–287, 2021.

[35] X. Ma and D. Dong, "Evolutionary nonnegative matrix factorization algorithms for community detection in dynamic networks," *TKDE*, vol. 29, no. 5, pp. 1045–1058, 2017.

[36] F. Liu, J. Wu, S. Xue, C. Zhou, J. Yang, and Q. Sheng, "Detecting the evolving community structure in dynamic social networks," *World Wide Web*, vol. 23, no. 2, pp. 715–733, 2020.

[37] X. Ma, D. Li, S. Tan, and Z. Huang, "Detecting evolving communities in dynamic networks using graph regularized evolutionary nonnegative matrix factorization," *Physica A*, vol. 530, p. 121279, 2019.

[38] F. Folino and C. Pizzuti, "An evolutionary multiobjective approach for community discovery in dynamic networks," *TKDE*, vol. 26, no. 8, pp. 1838–1852, 2013.

[39] F. Liu, J. Wu, C. Zhou, and J. Yang, "Evolutionary community detection in dynamic social networks," in *IJCNN*. IEEE, 2019, pp. 1–7.

[40] L. Chen, Y. Gao, Z. Fang, X. Miao, C. S. Jensen, and C. Guo, "Real-time distributed co-movement pattern detection on streaming trajectories," *PVLDB*, vol. 12, no. 10, pp. 1208–1220, 2019.

[41] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[42] S. Gong, Y. Zhang, and G. Yu, "Clustering stream data by exploring the evolution of density mountain," *PVLDB*, vol. 11, no. 4, pp. 393–405, 2017.

[43] S. Song, C. Li, and X. Zhang, "Turn waste into wealth: On simultaneous clustering and cleaning over dirty data," in *SIGKDD*, 2015, pp. 1115–1124.

[44] J. Gan and Y. Tao, "Dynamic density based clustering," in *SIGMOD*, 2017, pp. 1493–1507.

[45] A. Gunawan and M. de Berg, "A faster algorithm for DBSCAN," *Master's thesis*, 2013.

[46] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *SIGKDD*, 2011, pp. 316–324.

[47] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *SIGSPATIAL*, 2010, pp. 99–108.

[48] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *J Mach Learn Res*, vol. 3, no. Dec, pp. 583–617, 2002.

[49] A. Karami and R. Johansson, "Choosing DBSCAN parameters automatically using differential evolution," *Int. J. Comput. Appl. Technol.*, vol. 91, no. 7, pp. 1–11, 2014.

[50] A. Starczewski, P. Goetzen, and M. J. Er, "A new method for automatic determining of the DBSCAN parameters," *J. Artif. Intell. Soft Comput. Res.*, vol. 10, no. 3, pp. 209–221, 2020.

[51] J. M. Santos and M. Embrechts, "On the use of the adjusted rand index as a metric for evaluating supervised classification," in *ICANN*. Springer, 2009, pp. 175–184.

[52] S. Kisilevich, F. Mansmann, M. Nanni, and S. Rinzivillo, "Spatio-temporal clustering," in *Data Min Knowl Discov*. Springer, 2009, pp. 855–874.