Mobile r-gather: Distributed and Geographic Clustering for Location Anonymity

Mobile r-gather: Distributed and Geographic Clustering for Location Anonymity

Jiemin Zeng[1], Gaurish Telang[2], Matthew P. Johnson[3], Rik Sarkar[4], Jie Gao[1], Esther M. Arkin[2], Joseph S. B. Mitchell[2]

[1] Department of Computer Science, Stony Brook University. {jiezeng,jgao}@cs.stonybrook.edu
[2] Department of Applied Mathematics and Statistics, Stony Brook University. {gtelang,estie,jsbm}@ams.stonybrook.edu
[3] Department of Mathematics and Computer Science, Lehman College. mpjohnson@gmail.com
[4] School of Informatics, University of Edinburgh. rsarkar@inf.ed.ac.uk

ABSTRACT

We study the r-gather clustering problem in a mobile and distributed setting. In this problem, nodes must be clustered into groups of at least r nodes each, and the goal is to minimize the diameter of the clusters. This notion of clustering is motivated by protecting user anonymity in location-based services or trajectory publication. Prior works on r-gather problems are centralized and cannot be easily adapted to the mobile setting. We describe a distributed algorithm that produces compact clusters, within an approximation factor 4 of the minimum cluster diameter possible. The algorithm can run on the mobile nodes and access points at the network edge locally, and can handle node mobility, rapidly switching cluster memberships as needed. The distributed approach naturally comes with the advantage of greater resilience and stability. Additionally, we show that it achieves local optimality; i.e., from the point of view of any particular node, the solution is nearly as favorable as possible, irrespective of the global configuration. We also show how to cluster trajectories with dynamic re-groupings. Further, we improve the theoretical hardness results for the problem in the Euclidean setting.

CCS CONCEPTS

•Security and privacy → Pseudonymity, anonymity and untraceability; •Networks → Location based services;

KEYWORDS

Location, clustering, anonymity, in-network computing, edge computing

ACM Reference format:
Jiemin Zeng[1], Gaurish Telang[2], Matthew P. Johnson[3], Rik Sarkar[4], Jie Gao[1], Esther M. Arkin[2], Joseph S. B. Mitchell[2]. 2017. Mobile r-gather: Distributed and Geographic Clustering for Location Anonymity. In *Proceedings of Mobihoc '17, Chennai, India, July 10-14, 2017,* 10 pages.
DOI: 10.1145/3084041.3084056

Mobihoc '17, Chennai, India

DOI: 10.1145/3084041.3084056

1 INTRODUCTION

We study the problem of clustering mobile nodes into meaningful, highly location-dependent clusters. In order to quantify "meaningful" clusters, we establish a lower bound, r, on the size of clusters. The resulting *r-gather* clustering problem is formally stated as follows: Given a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ in Euclidean space and a value r, cluster the points into groups of at least r points each such that the largest diameter of the clusters is minimized. We consider two popular notions of "diameter" of a cluster: the usual notion of *diameter* of a point set (the maximum distance between two points of the set), and the diameter of the minimum enclosing ball (MEB) of the set. In this paper we focus on the r-gather problem in a mobile and distributed setting and propose algorithms for this problem.

1.1 Motivation

One motivation of this version of clustering arises in location privacy in wireless networking. With the ubiquitous use of GPS receivers on mobile devices, it is now common practice that the locations of these mobile devices are recorded and collected. This raises privacy concerns as location information is sensitive and can be used to identify the user of the devices [9]. This problem is more challenging when location is part of the input in location-based queries, for example, finding the coffee shop closest to the user, querying traffic situations, and security-related applications such as reporting suspicious behaviors. In these settings, the user submits a query to location-based services (LBS) through a mobile device. An adversary that compromises the LBS server can infer private information about the user. Protection of privacy is characterized into two different yet related types: *query privacy*, e.g., whether an adversary can identify the user who issued the query (i.e., associate user IDs with queries), and *location privacy*, e.g., how much an adversary can learn regarding the location of a user. Thorough discussions of this topic can be found in [14, 25].

For query privacy, one approach is to use the k-anonymity measure [26], which groups locations into clusters, each of at least k points. In previous work [18, 22] "cloaking boxes" have been used to group spatio-temporal user queries into a box with at least k queries, and then the box (instead of the query locations themselves) is submitted to the LBS server. In this way, the query sender is indistinguishable from the $k-1$ other users in the same box. It is ideal to group queries from nearby locations into the same box such that the query accuracy is maximized. Towards this goal, the objective is to minimize the diameter of the clusters, since this yields location data with the best possible accuracy, while

not hurting user privacy under the k-anonymity measure. This is precisely the r-gather problem [2].

Apart from protecting the privacy of a single snapshot, which is formulated by the static r-gather problem, it is natural to consider the dynamic setting for mobile users. When mobile phones continuously issue location-based queries, continuous spatial cloaking boxes are created [13, 27, 28]. The challenge is that the cloaking box may become huge after a long time period – if the mobile nodes move away from each other – leading to high computational cost and low query accuracy. Velocity of movement has been taken into consideration in constructing these continuous spatial cloaking boxes [23] but nothing provable has been found. Clearly there is a tradeoff between the quality of the cluster size and the stability of the clusters. Ideally we would like to keep tight clusters, almost as tight as what could be achieved for the node distribution at each snapshot, and keep the number of membership changes low.

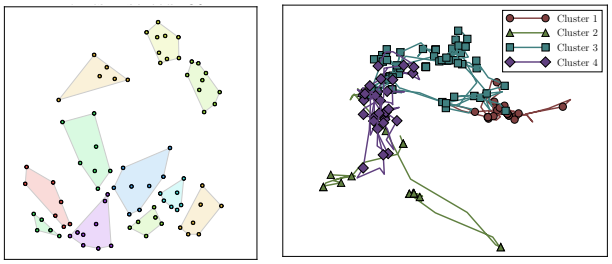
The r-gather formulation also appears in protecting privacy in trajectory publication. Here trajectories are collected and before they are published to the third party or the public, they need to be anomymized to remove sensitive information. Towards this objective, one approach is to group k or more co-localized trajectories within the same time period into a single aggregate trajectory with k-anonymity [10]. The IDs of the trajectories are removed so that one cannot associate an ID with any particular trajectory in a group of at least k. This is essentially the r-gather problem applied for grouping trajectories. It can be considered as an alternative way of clustering mobile nodes. Here the cluster memberships remain fixed and we minimize the distance between two trajectories in the same cluster, while each cluster has a lower bound on cardinality.

In this paper we also discuss an approach in between the two extremes of clustering complete trajectories and clustering location snapshots. This approach is to split trajectories into a number of segments that are then clustered. This problem can be stated as: given a parameter k specifying the maximum number of times we can re-cluster over the time period of interest, when and how should re-clustering be done so that the maximum cluster diameter is minimized, while satisfying the r-gather constraint that each cluster has at least r elements?

Besides the connection to location privacy issues, the r-gather problem is a natural and useful variant of mobile clustering in general. Many mobile applications rely on grouping the mobile nodes into clusters for management purposes; thus, clustering mobile nodes, including distributed clustering, has been studied in many prior papers, e.g., in [7, 8, 11, 12, 17, 20]. However, none of the previous work strictly enforces a lower bound on the cluster cardinality, which is a natural condition to ensure proper allocation of resources. In contrast, our method can handle mobile clustering subject to a cardinality bound and is applicable to trajectories and trajectory segments.

1.2 Related Work

The r-gather problem has been studied for instances in general metric spaces. Aggarwal et al. [2] give a 2-approximation algorithm and show that, for $r > 6$, it is NP-hard to approximate with an approximation ratio better than 2. The approximation algorithm first guesses the optimal diameter, then greedily selects clusters with twice the diameter; finally, a flow algorithm is used to assign at least r points to each cluster. This procedure is repeated until a good choice of diameter is found. Note that this solution only selects input points as cluster centers.



(a) (b)

Figure 1: Examples: (a) A 5-gather clustering on 80 points; (b) A 3-gather clustering on 20 trajectories, in which trajectories in the same group have the same color.

Armon [6] extended the result of Aggarwal et al. proving that, for $r > 2$, it is NP-hard to approximate with a ratio better than 2 for the case of general metric spaces. Armon also considers a generalization of the r-gather clustering problem, called the *r-gathering problem*, which also considers a given set of potential cluster centers (potential "facility locations"), each having a fixed set-up cost that is included in the objective function. Armon provides a 3-approximation for the min-max r-gathering problem and proves that it is NP-hard to obtain a better approximation factor. Additional results include various approximation algorithms for the min-max r-gathering problem with a *proximity requirement* that each point be assigned to its nearest cluster center.

For the case $r = 2$, both [5] and [24] provide polynomial-time exact algorithms. Shalita and Zwick's [24] algorithm runs in $O(mn)$ time, for a graph with n nodes and m edges. All of these algorithms were for a centralized setting. Not much is known in the distributed and mobile networks. Distributed clustering has been considered from a general distributed computing point of view [19, 21], however, these do not satisfy the r-gather requirement. Additionally, clustering mobile location data at the network edge requires a local approach with which general distributed algorithms are not compatible.

1.3 Our Results

In this paper we investigate the r-gather problem in the Euclidean metric for dynamic/mobile nodes, and in the decentralized setting. We obtain the following results.

In the decentralized setting we design a 4-approximation algorithm in which each node makes local decisions. The algorithm is based on a certain type of sweeping procedure. The sweep-clustering of a point depends only on local configurations; i.e., it is not influenced by outliers elsewhere in the network. This nice property ensures that the clustering is robust to noise/outliers, and the size of the cluster containing a node is determined only by the local node density. Fig. 1(a) shows an example of such clustering. This algorithm can be extended to the mobile setting, and the solution adapts naturally according to the mobility. We analyze the stability of this algorithm and show that under certain

mobility models the number of changes to the clustering membership can be bounded by $O(n^2)$ for n mobile nodes, while the optimal r-gather solution may have to change $\Omega(n^3)$ times. By relaxing the quality of clusters by a factor of 4, we achieve better stability of the clustering solution.

In the setting of clustering trajectories, we show that if we minimize the maximum distance of a pair at any time of the trajectory, then the same approximate r-gather algorithms apply for suitably defined distances between trajectories (Fig. 1(b)). When we allow k regroupings for a given parameter k and minimize the maximum diameter of the clusters, we show that one can use dynamic programming and obtain a 2-approximation.

We also show new results on hardness of approximation for the r-gather problem in the planar Euclidean metric. For minimizing the largest diameter of the clusters, we show that it is NP-hard to approximate better than a factor $\sqrt{2+\sqrt{3}} \approx 1.932$ when $r \geq 3$. Recall that the diameter of a set is the maximum distance between a pair of points in the set. For minimizing the largest radius of the minimum enclosing balls (MEB) of the clusters, we show that it is NP-hard to approximate better than a factor $\sqrt{13}/2 \approx 1.802$ when $r \geq 3$.

Finally, we show clustering results and comparisons of the various algorithms introduced here on a real mobility dataset. These results are reported in the same order in the next few sections.

2 A DISTRIBUTED ALGORITHM

In this section, we consider the r-gather problem as a distributed computation problem for n nodes placed at arbitrary locations in the plane. The distributed, or *local* perspective developed here will later be critical in building the lightweight methods for maintaining clusters in motion.

We assume that a node can detect its own location, either by the device itself, or by triangulation, utilizing locations of nearby access points. For now, we assume that the nodes themselves can carry out distributed computations; later, we will explain how computations can be generalized to be carried out by local infrastructure devices in the spirit of *edge computation.*

2.1 Clustering with distributed maximal independent neighborhoods

In this subsection, we assume that the nodes are static, and our objective is to group them into compact clusters of size at least r. We will use the r-nearest neighbor (r-NN) graph for our computations. We count a node itself as one of its r nearest neighbors.

Symbolically, we write p_i to denote the location of node i. The set P is the set of all node locations. For any point p_i, we let $p_i^{(r)}$ denote its r^{th} nearest neighbor in P, and we let $d_r(p_i) = |p_i - p_i^{(r)}|$ denote the corresponding distance. We write $N_r(p_i)$ for the set of r nodes nearest to point p_i. That is, $N_r(p_i)$ are the neighbors of i in the r-NN graph. We write $N(P)$ for the set of all such *r-neighborhoods.* If $N_r(p_i) \cap N_r(p_j) = \emptyset$, we say that the r-neighborhoods of p_i and p_j are *independent.*

Algorithm Description. Our algorithm develops and refines a set $\mathcal{G}$ of clusters. Initially, $\mathcal{G} = \emptyset$. We let c_G denote the *center* of a cluster $G \in \mathcal{G}$.

The basic algorithm executes the following steps to construct the set $\mathcal{G}$ of clusters:

M1. At each point $p_i \in P$, compute $p_i^{(r)}$, $d_r(p_i)$ and $N_r(p_i)$.
M2. Select a maximal independent subset of neighborhoods from the set $N_r(P)$, add each as a cluster in $\mathcal{G}$, and *mark* the nodes in the selected neighborhood sets as "clustered".
M3. For any unmarked node $p_i \in P$, assign p_i to the cluster $G \in \mathcal{G}$ whose center, c_G, is closest to p_i.

The nodes that belong to r-neighborhoods of cluster centers and are added to clusters in step M2 are called *inner* members of the cluster, while nodes that are added in step M3, are called the *outer* members. Note that the clustering is not unique, and a neighborhood centered at any node is a candidate for forming an inner cluster. One example of the result of the algorithm is shown in Fig. 2, for neighborhoods of size 5.

The maximal independent subset in step M2 can be computed rapidly, in time $O(\log n)$, using the randomized distributed algorithm of Alon et al. [3]. In this classical algorithm, nodes work in a parallel distributed model. In each round, a node v marks itself as a candidate to be in the maximal independent set (MIS) with probability $\frac{1}{2 \cdot \deg(v)}$. If no neighbor with higher degree is marked, then it joins the MIS; otherwise, it unmarks itself. Any MIS node and neighbors naturally withdraw from the contention. This approach leverages the parallel nature of the system to complete computations in $O(\log)$ expected number of rounds of time, where any centralized sequential algorithm would have required at least $\Omega(n)$.

In our case, this algorithm can be adapted by constructing a graph whose nodes correspond to elements of $N_r(P)$, and whose edges link two nodes if the r-neighborhoods represented by them have a nonempty intersection. The maximal independent set algorithm [3] operating on this graph then produces the independent neighborhoods.

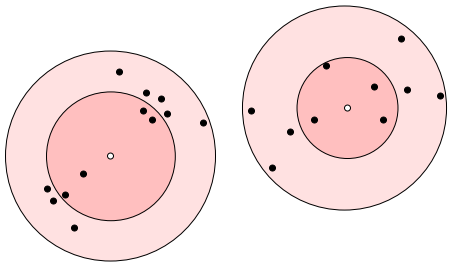


Figure 2: Clustering based on independent neighborhoods, for $r = 5$. The dark shaded disks show the inner cluster, while the nodes in the lightly shaded regions are the outer members of the clusters. The randomly selected cluster centers are colored white.

Proof of approximation. We will see later that computing the optimal r-gather clustering is NP-hard in the Euclidean setting. Here we show that the simple algorithm above approximates an optimal clustering: if D_{OPT} is the diameter of the largest cluster in an optimal clustering, then the diameter of any of the clusters from the algorithm above is at most $4 \cdot D_{OPT}$.

First, denote by $D_{OPT}(p_i)$ the diameter of the cluster that contains p_i in any optimal clustering solution. We observe a lower bound.

Observation 2.1. $d_r(p_i) \leq D_{OPT}(p_i)$.

Proof. Since any disk of radius less than $d_r(p_i)$ centered at p_i does not contain r nodes, a disk that contains p_i as well as (at least) $r-1$ other nodes must have radius at least $d_r(p_i)/2$. Thus, the diameter $D_{OPT}(p_i)$ must be at least $d_r(p_i)$. □

Let $d_r^{\max} = \max_i d_r(p_i)$ be the largest distance from a node to its r^{th} nearest neighbor in the given configuration P. We have an easy corollary.

Observation 2.2. $d_r^{\max} \leq D_{OPT}$.

Next, we see that in any cluster, the distance of a node from the center can be bounded by the sum $d_r(p_i) + d_r^{\max}$.

Lemma 2.3. *For any cluster $G \in \mathcal{G}$ and any node $p_i \in G$, $|p_i - c_G| \leq d_r(p_i) + d_r^{\max}$.*

Proof. Consider a cluster $G \in \mathcal{G}$, centered at c_G, and $p_i \in G$. If p_i was assigned to cluster G in step M2, then we know that $p_i \in N_r(c_G)$, implying that $|p_i - c_G| \leq d_r(c_G) \leq d_r(p_i) + d_r(c_G)$.

If p_i was not assigned to cluster G in step M2, but was instead assigned to G in step M3, we know, by maximality of the independent set, that the r-neighborhood $N_r(p_i)$ intersects some other r-neighborhood, say $N_r(p_j)$, that was a cluster in the maximal independent set in step M2. (It may or may not be the case that $G = N_r(p_j)$.) Thus, there is a node $p_y \in N_r(p_i) \cap N_r(p_j)$, implying that $|p_i - p_y| \leq d_r(p_i)$ and that $|p_y - p_j| \leq d_r(p_j)$. The triangle inequality implies then that $|p_i - p_j| \leq |p_i - p_y| + |p_y - p_j| \leq d_r(p_i) + d_r(p_j) \leq d_r(p_i) + d_r^{\max}$. Since p_i is closer to c_G than to the alternative center p_j, we get the claimed inequality, $|p_i - c_G| \leq |p_i - p_j| \leq d_r(p_i) + d_r^{\max}$. □

Using the properties above, it follows that the algorithm produces a 4-approximation of the diameter:

Corollary 2.4. *The diameter of any $G \in \mathcal{G}$ is at most $4D_{OPT}$.*

Proof. Consider any $p_i, p_j \in G$. By Lemma 2.3, $|p_i - c_G| \leq d_r(p_i) + d_r^{\max} \leq 2d_r^{\max}$ and $|p_j - c_G| \leq d_r(p_j) + d_r^{\max} \leq 2d_r^{\max}$. Thus, by the triangle inequality, $|p_i - p_j| \leq 2d_r^{\max} + 2d_r^{\max} = 4d_r^{\max} \leq 4D_{OPT}$. □

The algorithm described above allows arbitrary nodes to be cluster centers, and therefore can result in unnatural clusters. This effect is seen in Fig. 2, where the cluster on the left contains two relatively dense subsets that could have been clusters of size 5 or more on their own, but neither of these being in the 5-neighborhood of the selected center, they are not in the inner cluster.

Thus, the algorithm is competitive for the worst case, but for an individual node or neighborhood, it can be suboptimal where nodes in dense neighborhoods are placed near the boundary of the cluster, and far from the center. We would like results for which clusters are more compact, as shown in Fig. 4, and each node is represented by a cluster center close to itself. We describe next the algorithm for such coherent clustering.

2.2 Distributed sweep algorithm with coherence guarantee

To ensure that the dense neighborhoods form clusters, we take a greedy approach in which we first create clusters in dense regions, and then in progressively sparser regions.

At each node p_i, we consider the function $d_r(p_i)$, the distance to the r^{th} nearest neighbor of p_i. This function can be seen as the inverse of the local density of nodes at any point. Thus, following our strategy of progressing from dense to sparse regions, we can process nodes by increasing value of d_r.

In a distributed setting, instead of sorting all nodes by function values, we can proceed more locally, starting at local minima. Intuitively, the algorithm works as follows: any node that is a local minimum of d_r, becomes a cluster center, and its r nearest neighbors are assigned to its cluster. Following this, any node whose r-neighborhood intersects with existing clusters gives up the possibility of becoming a cluster center. Nodes with lowest value of d_r that have neighborhoods independent of existing clusters can now become cluster centers. This process can be seen as an upward *sweep* of the d_r function values, where a node does not make a decision whether or not to become a cluster center until all of its neighbors that have a lower value have made a decision. This decision making is locally sequential, but allows distributed parallel operation overall (see Fig. 3).

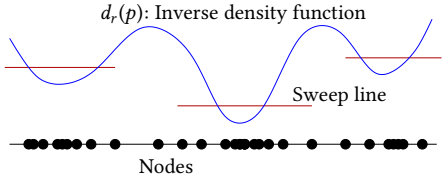


Figure 3: Processing nodes in the increasing order of d_r can be equivalently done by a distributed sweep. Instead of a single sweep line, the distributed algorithm is equivalent to multiple sweep lines starting at each minimum. Nodes become cluster centers if none of their r-neighbors are clustered or claiming to be centers.

Algorithm description. We present here a more formal description of the algorithm. We assume that the function values are distinct at all nodes; ties can be broken by node ids. Each node p_i maintains two variables:

- Its cluster center pointer, intialized to NULL. When node p_i is assigned a cluster, its cluster center pointer is assigned.
- A decision state, *decided/undecided*, to indicate whether p_i is still in contention for becoming a cluster center.

Each node p_i is initially in contention to become cluster center; we prefer nodes p_i with smaller values of $d_r(p_i)$. The algorithm operates in rounds. In each round, every undecided and unclustered node p_i requests permission from nodes in $N_r(p_i)$ to become a cluster center. After all responses from nodes in $N_r(p_i)$ arrive at p_i, p_i will make a decision as follows.

(1) **If all nodes in $N_r(p_i)$ grant permission:** then p_i becomes a cluster center, and all nodes in $N_r(p_i)$ are marked as *clustered* and *decided*. Additionally, they all set their cluster center pointer to p_i.

(2) **If one or more nodes in $N_r(p_i)$ *deny* permission for p_i:** then p_i marks itself as *decided*, implying that it will not try to become a cluster center any more.
(3) **If one or more nodes in $N_r(p_i)$ *defer* permission:** then p_i does not make a decision and tries again in the next round.

Any node p_j that receives a permission request from p_i responds as follows:

(1) **If p_j is *clustered*:** then p_j *denies* permission to p_i;
(2) **Else, if all *undecided* nodes $p_{j'} \in N_r(p_j)$ have values $d_r(p_{j'}) > d_r(p_i)$:** then node p_j gives permission to p_i;
(3) **Else:** p_j *defers* permission to p_i.

The "defer" response from p_j essentially implies that the local sweep has not yet reached the neighborhood, and thus p_j does not have the information to grant or deny permission to p_i. Any node left unclustered after all nodes have set state to *decided*, is assigned to the cluster of the nearest center, as in step M3.

Let us refer to the 2-hop neighbors of p_i as $N_r^2(i)$. Formally: $N_r^2(i) = \{k : N_r(i) \cap N_r(k) \neq \emptyset\}$. The following observation implies that the algorithm terminates:

Observation 2.5. *In each round, at least one node sets the state to* decided.

Proof. Let us suppose to the contrary that in some round no node makes a decision, but there are undecided nodes in the system. In the subset U of undecided nodes, suppose p_i is a node where $d_r(i)$ the minimum at i within $N_r^2(i) \cap U$ (note that one such minimum must exist since d_r is unique at each node).

Since p_i does not make a decision, it must be that all nodes in $N_r(i)$ have deferred permission. Therefore, for any $j \in N_r(p_i)$ we can conclude that p_j is unclustered since otherwise p_j would have denied permission. Since p_j defers permission, it must be that at least one node $p_k \in N_r(p_j) \cap U$ satisfies $d_r(p_k) < d_r(p_i)$. Since $p_k \in N_r^2(p_i)$, this contradicts the earlier conclusion that p_i is a minimum of d_r in $N_r^2(p_i)$.

□

The correctness property that any cluster produced by this algorithm has at least r nodes is easy to see. When a cluster center p_i is marked, it is assigned its r nearest neighbors. Since any of these neighbors then deny permission to their other neighbors, no node in $N_r^2(p_i)$ can become a center, and thus clusters cannot overlap. All nodes are clustered, since the final step of the algorithm is to assign cluster centers to all unclustered nodes.

The result of this algorithm is shown in Fig. 4. We see that the greedy approach yields results, and for the same point set as Fig. 2, in this case we get smaller and more compact clusters. The following theorem formalizes the fact that for every node, its assigned cluster center is a good representative.

Theorem 2.6. *If node p_i belongs to cluster G with center c_G, then $|p_i - c_G| \leq 2d_r(p_i)$. Also, for any two nodes p_i, p_j in the same cluster G, $|p_i - p_j| \leq 2d_r(p_i) + 2d_r(p_j)$.*

Proof. Regarding the first statement, if $p_i = c_G$ then the claim is trivially true. If not, then there exists a node $p_y \in N_r(p_i) \cap N_r(p_j)$ for some cluster center p_j, where p_y denied permission to p_i. Then, $d_r(p_j) \leq d_r(p_i)$, since otherwise p_y could not have denied permission to p_i while granting one to p_j. Thus $|p_i - p_j| \leq |p_i - p_y| + |p_y - p_j| \leq d_r(p_i) + d_r(p_j) \leq 2d_r(p_i)$. If $p_j = c_G$, then this concludes the proof. If $p_j \neq c_G$, then since after all decisions, each unclustered node is assigned to the nearest center, we have $|p_i - c_G| \leq |p_i - p_j| \leq 2d_r(p_i)$.

To obtain the second claim we apply the triangle inequality. □

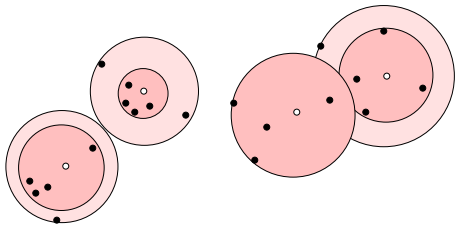


Figure 4: Coherent clustering: smaller, more compact clusters than those in Fig. 2.

This proof implies that the center assigned to any node is at distance at most twice the distance to its r^{th} nearest neighbor, irrespective of locations of the rest of the point set. Thus, nodes in dense regions are guaranteed to be assigned to a correspondingly nearby cluster center. Nodes in sparse regions may have correspondingly distant centers, but in sparse regions, OPT cannot do much better.

The results above apply to any metric space. Thus, we can perform clustering using $L_1, L_2, \ldots, L_\infty$ or any other metric as required.

Also observe that all results in this section apply to a weighted version of r-gather, in which each node has a weight, and the total weight in each cluster is required to be at least r.

3 THE r-gather CLUSTERING IN THE MOBILE SETTING

When we consider nodes that move over time, the nature of the problem changes. Depending on the application, we can consider clustering of mobile nodes in two different ways:

(1) **Maintain dynamic clustering of node locations.** As nodes move, update the clusters to be a good clustering of the current locations of mobile nodes.
(2) **Offline clustering of trajectories.** Given the recorded trajectories of nodes, cluster them into groups that stay close at all times and thus have similar trajectories.

The online version (1) can be seen as the dynamic case of maintaining instantaneous clustering of node locations we have discussed above. The offline version (2) can be seen as the data analysis version where we look at mobility patterns to find groups or communities of nodes. A variant of (2) is the case in which we allow nodes to belong to different groups at different time periods, as can be expected in location data over long periods.

3.1 Dynamic distributed clustering of node locations

In this subsection, we consider the problem of distributedly updating clusters as nodes move, so that at any instant we have a good clustering of current locations. The challenge in maintaining

the clusters in the face of mobility is to dynamically update the cluster membership while maintaining stability and coherence.

Mobility and computation. We assume that the mobile nodes are tracked by a static infrastructure, which may consist of a sensor network, communication network, access points, or any combination thereof. Localizations may be obtained either through GPS or through local triangulations. The static system acts as an edge computing infrastructure so that the location data and updates can be handled locally without the need for long communication and server bottlenecks.

For simplicity, we assume that computations are carried out using the L_∞ metric. Note that analogous results hold for other metrics; further, L_∞ distances approximate Euclidean distances within a constant factor. The advantage of the L_∞ metric is that disks in this metric take the shape of axis-aligned squares that can tile the plane. This feature is useful in the location management we use below.

Mobility and location management. We assume that a location service such as [1] is runnning on the edge system for location and computation management. This service works as follows. It divides the plane into a quadtree hierarchy, in which a square region is recursively subdivided into four square subregions. The lowest level with smallest squares (of unit length) is called level 0, the next level is level 1 etc. A square s_α at a level α has 8 neighbors at the same level, and we write this neighborhood as $N(s_\alpha)$ (see Fig. 5). In the following, these squares are our unit of measurement.

Each square at each level is assigned a location server by the infrastructure. The presence of a mobile node is noted at the server for squares containing the node at each level. To avoid excessive updates to the hierarchy, when a node m leaves a square s_α, the servers at level $\alpha+1$ are not updated immediately. Instead, the node simply leaves a pointer at s_α to the new host for m in $N(s_\alpha)$. The level $\alpha+1$ gets updated when the node has passed out of $N(s_\alpha)$. This lazy scheme guarantees a low amortized communication cost to keep the data up to date. In particular, assuming that communication cost between location servers distance d apart is bounded by $O(d)$, the amortized update cost is $O(d \log d)$ when a mobile node travels a distance d.

Clustering using location hierarchy. This location hierarchy can be used to generate approximate $r-NN$ graphs and run the static algorithms described above. The clustering is carried out by these location servers using the weighted version of r-gather, in which the weight for each server equals the number of mobile nodes in its square. In this approach, for any node m, we consider the neighborhoods of squares at different levels containing it. We write them as $N(s_\alpha(m))$. And we look for the lowest level at which $N(s_\alpha(m))$ contains at least r nodes.

Thus, when a node m makes a query for the neighborhood that contains at least r nodes, the query travels up the hierarchy through servers for squares hosting m, and eventually arrives at a s_α such that $N(s_\alpha(m))$ reports to contain at least r nodes. Thus we have found a neighborhood of m that contains at least r nodes. We can assume the corresponding distance $d'_r(m) = 2 \cdot 2^\alpha$. Also note that the previous neighborhood at level $\alpha - 1$ did not contain r nodes. Thus, we know that our estimate d' is a good approximation of the correct distance: $d'_r(m) \leq 2 \cdot d_r(m)$.

Due to the lazy update scheme, some of the mobile nodes may have moved to nearby squares, and we must accommodate this possibility. To ensure that the neighborhood contains r nodes, we should take the union of neighborhoods of all the squares in $N(s_\alpha(m))$, and set $d'_r(m) = 3.2^\alpha$. Thus we have $d'_r(m) = 3 \cdot 2^\alpha$, and $d'_r(m) \leq 3 \cdot d_r(m)$. Let us write the extended neighborhood computed as the neighbors of $N(s_\alpha(m))$ as $E(s_\alpha(m)) = \bigcup_{x \in N(s_\alpha)} N(x)$ (See Fig. 5).

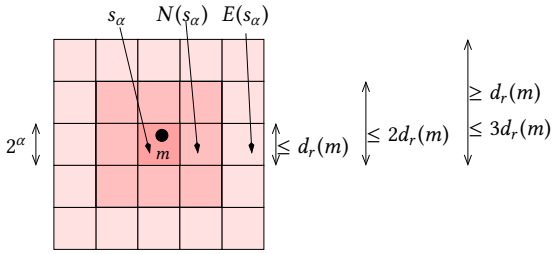


Figure 5: Neighborhoods for mobile node m. Server for $s_\alpha(m)$ detects that there are at least r nodes in $N(s_\alpha(m))$.It reports $E(s_\alpha(m))$ as the neighborhood containing r nodes.

Thus, with this method, we select the area $E(s_\alpha)$ as the neighborhood of a node. The weight in this neighborhood is the number of mobile nodes in the neighborhood. The weighted versions for algorithms from the previous section apply directly. The approximations hold with a further multiplicative factor of 3, as our measure of $d_r(m)$ is off by a factor of at most 3.

Cluster maintenance in location hierarchy. Next, we modify this protocol to adapt to mobility of nodes. In this modified version, each server $s_\alpha(m)$, stores the count of all nodes in the region $N(s_\alpha(m))$. Observe that since we take the extended neighborhood, a node moving from $N(s_\alpha(m))$ to a neighboring square does not require an immediate update to $d'_r(m)$. The update is made only when it passes out of the extended neighborhood. Thus, the number of updates caused by the mobility of a node is $O(x \log x)$ when the node has moved a distance x (see [1]).

The server $s_\alpha(i)$ simply updates its nodes count on these events and does not modify cluster, until it detects that number of nodes in its neighborhood has fallen below r, in which case it triggers a re-clustering for all clusters with centers in the neighborhood $N(s_{\alpha+2}(m))$. This guarantees that cluster sizes of r are preserved.

Complexity of computing r-NN. If the area of the mobility region is A, then the hierarchy has $O(\lg A)$ levels. At each level, a server needs a constant number of messages to check if its neighborhood has weight of r. Thus, the neighborhoods containing r nodes around each location (server in the system) are computed at $O(\lg A)$ messages, and a total cost of $O(A \lg A)$ to build the neighborhoods for all servers.

Number of Changes in Clustering Solution. Besides the clustering quality, we also hope that the clusters are stable and coherent over time. Here we argue an upper bound on the number of changes of clustering membership in our algorithm. The clustering produced by our algorithm does not change if all the points' r-neighborhoods remain the same. Thus, we can simply bound the number of changes to the r-neighborhoods. Again

as an upper bound for that, we bound the number of times that the r-th nearest neighbor of p_i changes, for a fixed i. We define $f_j(t) = |p_i(t) - p_j(t)|$, where $p_i(t)$ describes the position of p_i at time t. Thus, the r-th nearest neighbor of p_i is described by the complexity of r-th levels in the arrangement of $\{f_1(t), f_2(t), \cdots, f_{i-1}(t), f_{i+1}(t), \cdots, f_n(t)\}$. The complexity of the r-th level, when the trajectories are algebraic of constant degree (a common assumption on motion complexity,in order to compare the complexity or stability of a structure) is $O(\min\{rn, n^{4/3}\})$ [16]. Thus, the total number of changes to the clustering solution is $O(\min\{n^2 r, n^{7/3}\})$. When r is a constant, this is quadratic complexity.

Last, we show a lower bound on the number of changes needed if we maintain the *optimal* solution at all times. We show that in this setting, the optimal clustering may change as many as $\Omega(n^3)$ times. Consider this example: $n/2$ points lie on a line, with the points evenly spaced with spacing 1, and 3 points are clustered on top of each other at each end. In this example, $r = 3$. The optimal clustering of the points on the line is to have three points in a row be in one cluster with a diameter of 2. There are three different such clusterings which differ in the parity of the clusterings. In each clustering, there are $O(n)$ clusters. If another point v_i travels along the line, when it is within the boundaries of a cluster, it will just join that cluster. However, when it reaches the boundary of a cluster and exits it, the optimal clustering would be to shift the parity of the clustering (e.g., from the top configuration to the bottom configuration as shown in Fig. 6). This results in a change in all of the clusters along the line. The clustering changes every time the point travels a distance of 2. Therefore, as the point v_i travels along the line, the number of times the entire clustering changes is $\Omega(n)$, which results in a total of $\Omega(n^2)$ changes to individual clusters. We will now send $n/2$ points along the line; thus, the total number of clusters that change is $\Omega(n^3)$.

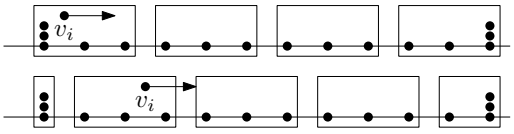


Figure 6: Lower bound of $\Omega(n^3)$ changes to the optimal mobile r-gather.

To summarize, our algorithm is comparably much more stable, incurring $O(n^2)$ changes while the optimal clustering might need to change $\Omega(n^3)$ times.

3.2 Clustering trajectories

In the second setting, we group the trajectories into clusters such that trajectories in the same cluster are 'similar' and each cluster has at least r trajectories. To make it concrete, suppose the distance function $d_t(p, q)$ is $|p(t) - q(t)|$ for two trajectories $p(t)$ and $q(t)$ over a time period T. We define the distance between p, q in a time period of $[1, T]$ to be $d(p, q) = \max_{t \in T} d_t(p, q)$. We would like to minimize the largest diameter of each cluster over the entire time period. The clustering membership does not change over time. First we show that the distance $d(p, q)$ forms a metric:

Lemma 3.1. *The function $d(p, q)$ is a metric.*

Proof. The function by definition is symmetric, follows the identity condition, and is always non-negative. To show that the metric follows the triangle equality, we first assume that there is a pair of trajectories x and z where $d(x, z) > d(x, y) + d(y, z)$ for some y. There is some time $t \in T$, where $d_t(x, z) = d(x, z)$. By the triangle inequality,

$$d_t(x, z) \leq d_t(x, y) + d_t(y, z).$$

In addition, clearly $d_t(x, y) \leq d(x, y)$, and $d_t(y, z) \leq d(y, z)$. This contradicts our assumption and concludes our proof. □

With this distance metric, we can now apply either our approximation algorithms above, or the 2-approximation from [2]. This distance measure results in clustering together trajectories of mobile agents based on who were close at all times, analogous to finding groups that travelled together. The algorithm we described works for any metric space and is possible to use it on a different metric such as the Frechet distance [4] to cluster based on similar travel patterns irrespective of time.

3.3 Clustering trajectories with regrouping.

Now, we consider the more general setting in which nodes are allowed to change groups over time. We allow K regroupings of the nodes over a given time horizon. We use the same distance metric as in Lemma 3.1 above.

Each regrouping allows all clusters to be modified or changed completely. We claim that with the assumption that the trajectories are piecewise-linear, we can optimize the choice of regrouping times, using dynamic programming, in conjunction with any α-approximation algorithm (e.g., $\alpha = 2$ for the algorithm of [2], or $\alpha = 4$ for our distributed algorithm described earlier) for the static case, achieving the same approximation factor α for the optimal regrouping problem.

We consider the time horizon to be discretized and indexed by integers $t \in [0, T]$. Each trajectory is a piecewise-linear function that only changes directions at times in $[0, T]$. Let $C_{t',t}$ denote the maximum diameter of a cluster in the α-approximation clustering computed at time t', over the time period $[t', t]$. We can compute and store in a table the values $C_{t',t}$.

A subproblem, specified by (t, k), seeks to determine the optimal value, $S(t, k)$, that is the minimum possible diameter of the points of the trajectories in a cluster, over the time period $[0, t]$, using exactly k regroupings.

Then, the main recursion in our dynamic program is given by optimizing over all choices of time $t' \in (0, t]$ when the last (kth) regrouping should be done:

$$S(t, k) = \min_{0 < t' \leq t} \max\{S(t', k-1), C_{t',t}\}, \qquad k > 1$$

with the base of the recursion given by $S(t, 1) = C_{0,t}$, the solution corresponding to a single grouping done at time 0, and active over the time horizon $[0, t]$. Our overall objective is to determine the value $S(T, K)$, corresponding to having K regroupings over the full time horizon. The dynamic program takes time $O(KT^2)$ to evaluate the values $S(t, k)$, after computation of the $O(T^2)$ table entries $C_{t',t}$.

Theorem 3.2. *We can achieve an α-approximation for the mobile r-gather problem, when K regroupings are allowed, by optimizing*

the choice of regrouping times, while utilizing, at each regrouping, an α-approximation algorithm for the static clustering problem.

Note that our lower bound proofs on the approximation factor for the static r-gather apply as well to the mobile r-gather problem. The points arranged in any of the lower bound proofs can be static points for the duration of $[0, T]$ or may move in a way that the distances between points do not increase. Then the arguments for static r-gather translate to this simple version of dynamic r-gather directly.

4 HARDNESS OF APPROXIMATION

In this section, we consider the r-gather problem in the Euclidean plane and show lower bounds on approximation in this case. Recall that, in general metric spaces, it is known that it is NP-hard to approximate better than a factor 2 (see [2]).

For minimizing the largest diameter of the clusters, we show that it is still NP-hard to approximate better than a factor $\sqrt{2+\sqrt{3}} \approx 1.932$ when $r \geq 3$ even in Euclidean setting. For minimizing the largest radius of the minimum enclosing balls (MEB) of the clusters, we show that it is NP-hard to approximate better than a factor $\sqrt{13}/2 \approx 1.802$, when $r \geq 3$.

Theorem 4.1. *For the r-gather problem for minimizing the maximum MEB, it is NP-hard to approximate better than a factor of $\sqrt{13}/2 \approx 1.802$ when $r \geq 3$.*

Proof. We reduce from the NP-hard problem called the planar circuit SAT [15]. We are given a planar directed acyclic graph. Each node in the graph is either a source (of in-degree zero), a NAND gate (of in-degree 2 and any out-degree), or a sink (out-degree zero). There is exactly one sink and its in-degree must be one. The question is whether the edges (aka wires) can be colored with the two colors TRUE and FALSE such that the following holds:

- The value of the edge going to the sink is TRUE.
- The value of any edge out of a NAND gate is the NAND of the truth values of the two edges going into the gate.
- The value of an edge out of a source can be anything.

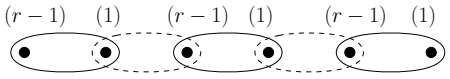


Figure 7: A wire gadget

A wire gadget consists of a line of points that alternate between a single point and a group of $r-1$ points at the same location. Fig. 7 illustrates such an example. In the figure, a point may represent multiple points in the same location, the number of which is noted in parenthesis. All distances between adjacent groups of points on a wire are distance 1 apart. The parity of the clusters chosen signify a true signal or a false signal. If the cluster has $r-1$ points first, followed by one point of distance one away, the signal of the wire is true. In this figure the solid clusters are true and the dashed clusters indicate a false signal.

It is simple to enforce the output to be a true signal by ending the output wire with a single point. The beginning of the input wires have a group of r points so that the inputs can be either true or false. Fig. 8 illustrates the NAND gadget, a universal gate. The solid clusters illustrate two true inputs into the gate and a false output. If either or both of the inputs is false, then two groups of points in the triangle (or all three) will become a cluster and the output will be true. Fig. 9 illustrates the splitter circuit which serves the role of a wire splitting into two carrying the same signal. The solid clusters indicate a true signal and the dashed clusters indicate a false signal. If the planar circuit SAT is satisfiable, the r-gather problem has a solution of cluster diameter of 1 – only the solid or dashed clusters are used. Otherwise, the r-gather solution uses clusters that are formed by three groups of points. The smallest of such clusters have two from the triangle and one adjacent to the triangle (see the shaded cluster). The diameter of such a cluster is $\sqrt{13}/2 \approx 1.802$.

Finally, note that in order to connect the wires, they must be able to turn somehow. We can bend the wire such that no three groups of points can form a cluster that has diameter smaller than $\sqrt{13}/2$. Thus concludes our proof. □

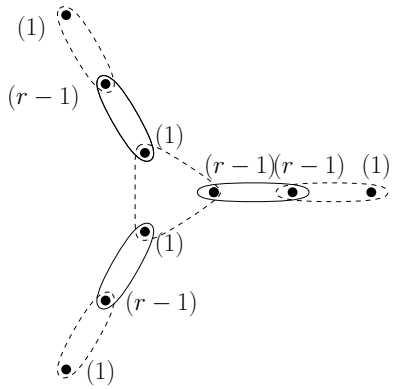


Figure 8: NAND gadget

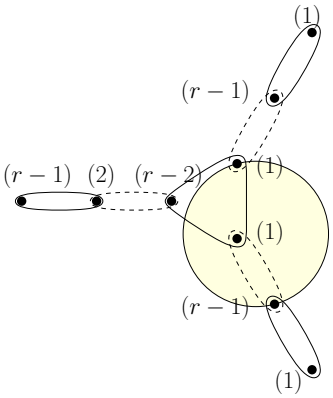


Figure 9: A splitter gadget

Theorem 4.2. *For the r-gather problem for minimizing the maximum diameter (as the distance between furthest pair), it is NP-hard to approximate better than a factor of $\sqrt{2+\sqrt{3}} \approx 1.932$ when $r \geq 3$.*

Proof. The proof is almost the same as the proof of Theorem 4.1 except that with the different definition of diameter, the triangles in

the gadgets are slightly larger. The pairs in a solid or dashed cluster are distance 1 apart. The shaded cluster has diameter $\sqrt{2+\sqrt{3}}$. □

5 EXPERIMENTAL RESULTS

We implemented our distributed approximation algorithm and tested its performance against the existing centralized method from [2] and a baseline of local density: the distance to the r^{th} nearest neighbor: $d_r^{\max}$. We used a real dataset of moving cars in Shenzhen, China. All cars had their GPS points sampled every 5 minutes in sync for 24 hours. Our main observations are that:

- Our distributed algorithm performs on par with the centralized algorithm [2] on real location data, and both produce cluster diameters close to the baseline lower bound of $d_r^{\max}$.
- In clustering trajectories, our distributed algorithm analogously performs on par with [2] and the lower bound.
- On mobility data representing trajectories of moving vehicles, the dynamic algorithm produces smaller, more compact clusters than those produced by clustering whole trajectories.

We discuss the results in more detail below.

For brevity in the figure legends, we refer to the centralized algorithm [2] as the "2-APX" algorithm, while our distributed algorithm with a 4-approximation guarantee is referred to as the "4-APX" algorithm. Fig. 10 compares the maximum cluster sizes of the two algorithms. It also plots the baseline lower bound of $d_r^{\max}$.

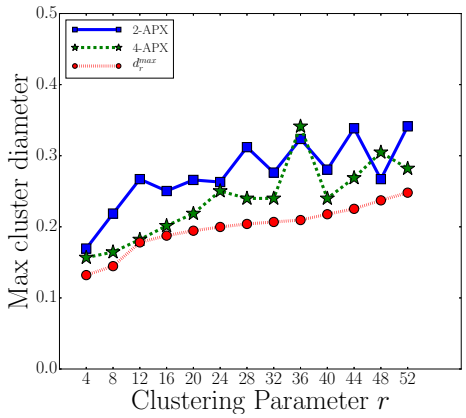


Figure 10: Cluster diameters for different cluster size lower bounds r for centralized 2-APX and distributed 4-APX algorithms on 150 stationary points from a random snapshot in the data. The distributed algorithm performs comparably and often better than the centralized, and close to the lower bound.

For the experiments, we took a random snapshot of 150 arbitrarily selected cars from the dataset and calculated the maximum cluster diameter returned by both algorithms for increasing values of r. We see that in fact both algorithms stay within a factor of 2 of the lower bound, and the distributed algorithm often performs better than the centralized one. Similar patterns hold for other snapshots (See Fig. 12).

Next, we conducted experiments on clustering whole trajectories. We took a random sample of 150 trajectories. Each trajectory contained 100 sample points. And we clustered them using the metric defined in Subsection 3.2. Fig. 11 compares the clustering quality of the two algorithms for trajectories.

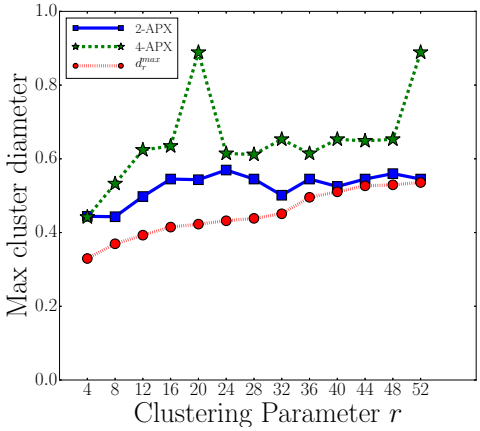


Figure 11: Cluster diameters for different cluster size lower bounds r for centralized 2-APX and distributed 4-APX algorithms on 150 trajectories, 100 GPS points per trajectory. The distributed algorithm performs comparably but slightly worse than the centralized, but still close to the lower bound.

We see that once again, both algorithms perform similarly and have cluster diameters within a factor of about 2 of the bound. In this case, however, the distributed algorithm performs slightly worse than the centralized algorithm with occasional larger spikes.

Finally, we compared changing cluster diameters in the mobile case, between the results of statically clustering whole trajectories (Subsection 3.2) and dynamic location clustering (Subsection 3.1). In the first case, the output is only one clustering of the mobile nodes and the cluster membership of a node does not change, while in the second case, in each snapshot, a fresh clustering is computed, and a node's membership may change.

Fig. 12 shows results for a set of 150 trajectories, each trajectory containing 50 sample points. We set $r = 5$ for this experiment. Fig. 12 shows that while the static offline clustering has consistent cluster membership, the diameter can grow large when the nodes move apart. The dynamic distributed clustering incurs changing cluster memberships, but consistently produces tighter, more coherent clusters and can produce results online.

6 CONCLUSION

In this paper we investigated the r-gather problem – a variant of geometric clustering – for the mobile settings, with the goal of enabling anonymity in location based services. We described distributed clustering methods with provable results, and showed that the solution can be adapted to cluster mobile devices in a distributed, online computation setting. We improved hardness results for metrics in the Euclidean setting, and proposed an algorithm for the dynamic setting when nodes move around and

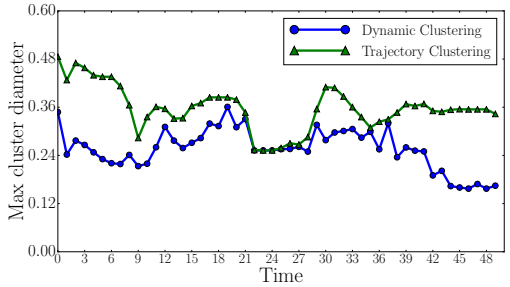


Figure 12: Comparing a static Clustering on Trajectories (blue) vs Dynamic Clustering at each time-step (green). The dynamic online clustering produces tighter, more compact clusters. The experiment was run on 150 trajectories with 50 sample points per trajectory.

regrouping is allowed. We evaluated the algorithms on a real data set and show that the distributed algorithm actually performs comparably in practice, in terms of maximum cluster diameter. We expect that the algorithms find other applications in mobile computing.

Acknowledgement: J. Zeng and J. Gao would like to acknowledge federal support through NSF DMS-1418255, CCF-1535900, CNS-1618391 and AFOSR FA9550-14-1-0193. J. Mitchell and E. Arkin acknowledge support from NSF (CCF-1018388, CCF-1526406). R. Sarkar acknowledges support from the UK National Cyber Security Center.

REFERENCES

[1] I. Abraham, D. Dolev, and D. Malkhi. LLS: a locality aware location service for mobile ad hoc networks. In *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 75–84, 2004.

[2] G. Aggarwal, S. Khuller, and T. Feder. Achieving anonymity via clustering. In *In PODS*, pages 153–162, 2006.

[3] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986.

[4] H. Alt and M. Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.

[5] E. Anshelevich and A. Karagiozova. Terminal backup, 3D matching, and covering cubic graphs. *SIAM Journal on Computing*, 40(3):678–708, 2011.

[6] A. Armon. On min–max r-gatherings. *Theoretical Computer Science*, 412(7):573–582, 2011.

[7] S. Basagni. Distributed clustering for ad hoc networks. In *Proc. 99' International Symp. on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, pages 310–315, June 1999.

[8] P. Basu, N. Khan, , and T. D. Little. A mobility based metric for clustering in mobile ad hoc networks. In *Proc. of IEEE ICDCS 2001 Workshop on Wireless Networks and Mobile Computing*, April 2001.

[9] A. J. Blumberg and P. Eckersley. On locational privacy, and how to avoid losing it forever. EFF whitepaper, https://www.eff.org/files/eff-locational-privacy.pdf, 2009.

[10] F. Bonchi, O. Abul, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. *2008 IEEE 24th International Conference on Data Engineering (ICDE '08)*, 00(undefined):376–385, 2008.

[11] M. Chatterjee, S. K. Das, and D. Turgut. WCA: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5(2):193–204, 2002.

[12] G. Chen and I. Stojmenovic. Clustering and routing in mobile wireless networks. Technical Report TR-99-05, SITE, June 1999.

[13] C.-Y. Chow and M. F. Mokbel. Enabling private continuous queries for revealed user locations. In *Proceedings of the 10th International Conference on Advances in Spatial and Temporal Databases*, SSTD'07, pages 258–273, Berlin, Heidelberg, 2007. Springer-Verlag.

[14] C.-Y. Chow and M. F. Mokbel. Trajectory privacy in location-based services and data publication. *SIGKDD Explor. Newsl.*, 13(1):19–29, Aug. 2011.

[15] E. Demaine. Lecture notes of Algorithmic Lower Bounds: Fun with Hardness Proofs.

[16] T. K. Dey. Improved bounds on planar k-sets and k-levels. In *IEEE Symposium on Foundations of Computer Science*, pages 165–161, 1997.

[17] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. *Discrete and Computational Geometry*, 30(1):45–65, 2003.

[18] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, MobiSys '03, pages 31–42, New York, NY, USA, 2003. ACM.

[19] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.

[20] C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7):1265–1275, 1997.

[21] S. Merugu and J. Ghosh. Privacy-preserving distributed clustering using generative models. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 211–218. IEEE, 2003.

[22] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 763–774. VLDB Endowment, 2006.

[23] X. Pan, X. Meng, and J. Xu. Distortion-based anonymity for continuous queries in location-based mobile services. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 256–265, 2009.

[24] A. Shalita and U. Zwick. Efficient algorithms for the 2-gathering problem. *ACM Transactions on Algorithms (TALG)*, 6(2):34, 2010.

[25] K. Shin, X. Ju, Z. Chen, and X. Hu. Privacy protection for users of location-based services. *Wireless Communications, IEEE*, 19(1):30–39, February 2012.

[26] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10:557–570, Oct. 2002.

[27] T. Xu and Y. Cai. Location anonymity in continuous location-based services. In *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, GIS '07, pages 39:1–39:8, New York, NY, USA, 2007. ACM.

[28] T. Xu and Y. Cai. Exploring historical location data for anonymity preservation in location-based services. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages –, April 2008.