Distributed r-gather: Geographically Coherent Cardinality Clustering for Mobile Networks

***Abstract*—Grouping mobile nodes into clusters can ease the management of a large number of devices and their information. Since applications and communication in mobile devices are highly location dependent, clustering by location is particularly useful in this context. In this paper, we consider the r-gather problem in which we must group nodes into clusters each having at least r nodes (so that each cluster has a meaningful population), while minimizing the maximum diameter of the clusters (so that each cluster is geographically coherent).**

We present several new results on the r-gather problem, including hardness of approximation in a variety of geometric settings and new distributed approximation algorithms for optimizing the maximum diameter of a cluster. In particular, our distributed algorithm for r-gather clustering allows computation to be pushed to the edges of the network. This method produces provably near-optimal results and can adapt to the dynamics of nodes in motion. The distributed approach naturally comes with the advantage of greater resilience. 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.

I. 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: 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.

A. 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 [8]. This problem is more challenging when location is part of the input in location-based queries, for example, finding the coffee shop nearest 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. In this setting, 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 location of a user. A nice survey on this topic can be found in [12], [19].

For query privacy, the most common approach is to use the k-anonymity measure [20], which says that the locations are grouped into clusters, each of at least k points. In previous work [14], [16] "cloaking boxes" were used to group spatio-temporal user queries into a box with at least k queries and then submit the box (instead of the queries themselves) to the LBS server. In this way, the query sender is indistinguishable from the $k-1$ other query users in the same box. It is ideal to group queries from similar locations into the same box. Towards this goal, one objective to minimize the diameter of the clusters, since this yields location data with the best possible accuracy, while not intruding on user privacy. 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 services, temporal spatial cloaking boxes are created [11], [21], [22]. The challenge is that the cloaking box may become huge after a long time period, leading to high computational cost and low query accuracy. Clearly there is a tradeoff between the quality of the cluster size and the number of re-clusterings we want to afford. Formulating this problem using the r-gather framework, we may ask the follow question: Given a parameter k specifying the maximum number of times we can recluster 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 have at least r elements?

Further, the r-gather problem has been investigated in centralized settings, when all information is gathered at a single location. In mobile applications, it is desirable to develop distributed algorithms, with local operations and computations distributed among the nodes.

Besides the natural connection to location privacy issues, the r-gather problem is a natural and useful variaent of mobile clustering in general. Many mobile applications rely on grouping the mobile nodes into clusters for management purposes. These clusters need to be geographically coherent and for a cluster to be meaningful, it is natural to require each cluster to have a minimum number of members. Clustering mobile nodes has been studied a lot in the past, for example in [9], [15], [10], [7], [6], [13]. But none of the previous work strictly enforce a lower bound on the cluster cardinality.

B. 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 diamter is found. Note that this solution only selects input points as cluster centers.

Armon [5] 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 [4] and [18] provide polynomial-time exact algorithms. Shalita and Zwick's [18] algorithm runs in $O(mn)$ time, for a graph with n nodes and m edges.

All the algorithms were for centralized setting. Not much is known in the distribtued and dynamic settings.

C. Our Results

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

- We show new results on hardness of approximation for r-gather problem in the Euclidean setting. For minimizing the largest diameter of the clusters, we show that it is NP-hard to approximate better than a factor 2 when $r \geq 5$, and that it is NP-hard to approximate better than a factor $\sqrt{2+\sqrt{3}} \approx 1.931$ when $r = 3$ or 4. Recall that the diameter of a set is the maximum distance between a pair of points in the set.
  For minimizing the largest of the minimum enclosing balls (MEB) of the clusters, we show that it is NP-hard to approximate better than a factor $\frac{\sqrt{35}+\sqrt{3}}{4} \approx 1.912$ when $r \geq 4$, and that it is NP-hard to approximate better than a factor $\sqrt{13}/2 \approx 1.802$ when $r = 3$.
- In the mobile setting, we are given a set of trajectories. We show that if we minimize the maximum distance of a pair at any time of the trajectory then the same 2-approximate r-gather algorithm for points apply. When we allow k regroupings for a given parameter k and minimize the maximum diameter of any clusters, we show that one can use dynamic programming and obtain a 2-approximtion. On the other hand, we show that in the worst case the number of cluster changes can be $\Omega(n^3)$, if we wish to maintain optimal r-gather solution at *all* time.
- We consider the decentralized setting and design a 4-approximate algorithm. Each node makes local decisions and operations. The algorithm is based on a certain type of sweeping procedure. The sweep-clustering of a point depends only on local configurations, it is not influenced by outliers elsewhere in the network. This nice property ensures that the clustering is robust to noises/outliers. This algorithm can be naturally extended to the mobile setting and the solution adapts according to the mobile node mobility.
- Finally, we presented simulation results and comparisons of the various algorithms introduced here.

These results are reported in the same order in the next few sections.

II. Hardness of Approximation

For mobile networks and location-based services, the Euclidean plane is a reasonable model for the underlying metric space. In this section, we consider the r-gather problem in the Euclidean plane and show lower bounds on approximation even in this special case. Recall that, in general metric spaces, it is known that it is NP-hard to approximate better than a factor 2.

For minimizing the largest diameter of the clusters, we show that it is NP-hard to approximate better than a factor 2 when $r \geq 5$, and that it is NP-hard to approximate better than a factor $\sqrt{2+\sqrt{3}} \approx 1.931$ when $r = 3$ or 4. (Recall that the diameter of a set is the maximum distance between a pair of points in the set.)

For minimizing the largest of the minimum enclosing balls (MEB) of the clusters, we show that it is NP-hard to approximate better than a factor $\frac{\sqrt{35}+\sqrt{3}}{4} \approx 1.912$ when $r \geq 4$, and that it is NP-hard to approximate better than a factor $\sqrt{13}/2 \approx 1.802$ when $r = 3$.

Theorem 2.1. *The r-gather problem for minimizing the maximum diameter, it is NP-hard to approximate better than a factor of 2 when $r \geq 5$.*

Proof: Our reduction is from the NP-hard problem, planar 3SAT. Given a formula in 3CNF composed of variables $x_i, i = 1, \ldots, n$ and their complements $\overline{x_i}$, we construct an instance of r-gather on the plane. Figure 1 illustrates a clause gadget of the clause $C = x_i \vee x_j \vee x_k$ and part of a variable gadget for x_i. In the figure, each point represents multiple points in the same location, the number of which is noted in parenthesis. All distances between groups of points connected by a line are distance 1 apart. Note that all clusters shown in the figure have a diameter of 1. If all clusters have a diameter of 1, then we can signify the parity of a variable by whether solid or dashed clusters are chosen. Here the solid clusters signify a positive value for x_i that satisfies the clause since the center point of the clause gadget is successfully assigned to a cluster. Note that the variable gadget in Figure 1 swaps the parity of the signal sent away from the gadget. We also include a negation gadget shown in Figure 2 that swaps the parity of the signal and can be used when connecting parts of the variable gadget together. If an optimal solution to this r-gather construction can be found, the diameter of all clusters is 1.

The center point of the clause gadget must be assigned to a cluster that contains all r points of one of the variable clusters or else a cluster of diameter 2 is forced. WLOG, let the center point be clustered with the r points of the x_i gadget. What results is the solid clusters in figure 1 are selected above the triangle splitter and the dashed clusters are selected below the splitter. The group of points at the top of the triangle splitter is unassigned to a cluster. It must merge with one of the neighboring clusters which results in a cluster of diameter 2. Therefore, it is NP-hard to approximate r-gather below a factor of 2 for $r \geq 5$. □

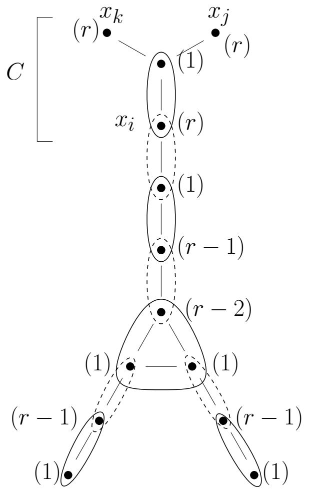

Fig. 1. clause and splitter gadget

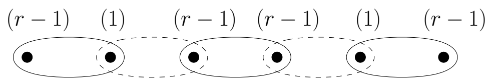

Fig. 2. signal negation gadget

Theorem 2.2. *The r-gather problem for minimizing the maximum MEB, it is NP-hard to approximate better than a factor of* $\frac{\sqrt{35}+\sqrt{3}}{4} \approx 1.912$ *when* $r \geq 4$.

Proof: The reduction is very simlar to the proof of Theorem 2.1. The only difference is the splitter which is illustrated in Figure 3. □

Theorem 2.3. *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* $r = 3$.

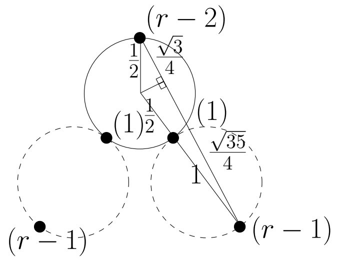

Fig. 3. close up of the splitter

Proof: We reduce from the NP-hard problem planar circuit SAT. We are given a planar boolean circuit with a single output. Similar to the previous proofs, 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. The parity of the clusters chosen signify a true signal or a false signal. When the clusters combine a group of $r-1$ points followed by a single point, the signal of the wire is true. 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. Figure 4 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. Figure 5 ilustrates the splitter circuit where the solid clusters indicate a true signal and the dashed clusters indicate a false signal. As before, if the optimal solution to the r-gather construction can be found, then cluster diameter will be 1. Otherwise, three groups will form a cluster, two from the triangle and one adjacent to the triangle. The diameter of such a cluster is $\sqrt{13}/2 \approx 1.802$ when $r = 3$. 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. □

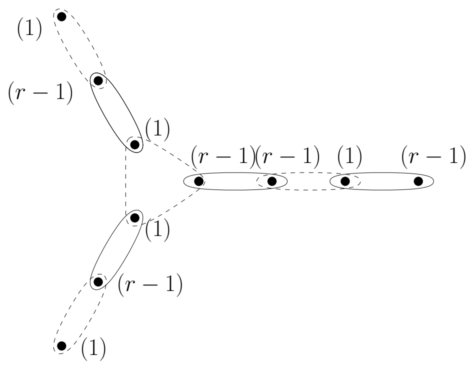

Fig. 4. NAND gadget

Theorem 2.4. *The r-gather problem for minimizing the maxi-*

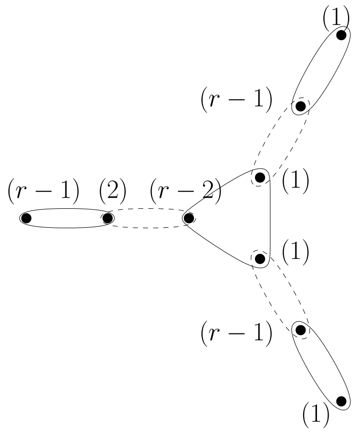

Fig. 5. splitter gadget

mum diameter, it is NP-hard to approximate better than a factor of $\sqrt{2+\sqrt{3}} \approx 1.931$ *when* $r = 3$ *or* 4.

III. The r-Gather Problem in the Mobile Setting

We continue our investigation of the r-gather problem by now considering the dynamic situation in which the points are in motion. There are various models to consider in the mobile setting; in this section, we consider several versions. In each case, we assume that the trajectories of the mobile agents are piecewise-linear.

Clustering Trajectories. In the simplest formulation of r-gather in a mobile setting, we are given a set of trajectories over a time period T and we want to cluster the trajectories such that each cluster has at least r trajectories and the largest diameter of each cluster over the entire time period is minimized. Here we designate the diameter of a cluster at a single point in time to be the distance between the furthest pair of points. Points are assigned to a single cluster for the entire length of T and do not switch clusters. We claim that the 2-approximation strategy in [2] for static r-gather in any metric space can also be applied to this problem. We use the distance metric defined by Lemma 3.1.

Lemma 3.1. *The distance function* $d_t(p, q)$ *between two trajectories* p *and* q *over a time period* T *is defined as the distance between* p *and* q *at time* $t \in T$. *Then* $d(p, q) = \max_{t \in T} d_t(p, q)$. *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. $\square$

Clustering Trajectories with Re-grouping. Now, we consider the more general setting in which the clusters are allowed to change, via regroupings, during the time period T. We let k be a parameter that specifies the maximum number of regroupings allowed during T. Each regrouping allows all clusters to be modified or changed completely. The lower bounds for the earlier version of r-gather apply here as well, for the same reasons. We claim that with the assumption that the trajectories are piecewise-linear, we can construct a 2-approximation solution using dynamic programming.

Let $|T|$ be the number of time steps in the time period T. Each trajectory is a piecewise-linear function that only changes directions at a time step in T. Let C_{ij} denote the max diameter of the 2-approximation clustering at time i over the time period $[i, j]$, $i < j$. We can create a $|T| \times |T|$ table $\mathcal{T}$ where entry $\mathcal{T}(i, j) = C_{ij}$.

We formulate a subproblem $S(t, i)$, where $0 \leq t \leq |T|$ and $i \leq k$, for our dynamic program to find the optimal clustering of the points in the time period $[0, t]$ where there are exactly i reclusterings. Let $l(t, i)$ denote the last time step a reclustering occured for the optimal solution of $S(t, i)$.

The subproblem of our dynamic program is:

$$S(t, i) = \min(\max_{j<t}(S(j, i), C_{l(t,i)t}), \max_{j<t}(S(j, i-1), C_{tt}))$$

The entry $S(t, i)$ checks $2t$ previous entries and $2t$ entries in the table $\mathcal{T}$. The entire table takes $k|T|^2$ to execute with the additional preprocessing of the table $\mathcal{T}$.

Our lower bound proofs for static r-gather apply here as well. The points arranged in any of the lower bound proofs can be static points for the duration of T or may move in a fashion where the distances between points do not increase. Then the arguments for static r-gather translate to this simple version of dynamic r-gather directly.

Theorem 3.2. *The lower bound results for static* r*-gather apply to any definition of dynamic* r*-gather. Further, we can approximate mobile* r*-gather, when* k *clusterings are allowed, within a factor of* 2.

Number of Changes in Optimal Solution. 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 many times, as much as $O(n^3)$ times. Consider this example: $n/2$ points lie on a line where the points are spaced apart by 1 and 3 points are overlapping on the ends. 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 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. This results in a change in all of the clusters along the line. The clustering change every time the point travels a distance of 2. Therefore, as the point 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 individial clusters. Since there are $O(n)$ points that travel along the line, the total number of clusters that change is $\Omega(n^3)$.

IV. Decentralized Algorithm

In this section, we consider the r-gather as a distributed computation problem. This approach is particularly relevant in the context of locations, where data is naturally spread over a spatial region, and we can use local computations at access points and local devices for anonymization and cloaking. This approach also provides better security and privacy, since it is harder for an attacker to compromise many devices spread over a large region.

A. Distributed Computation and Location Management

We consider the problem from an *edge computation* perspective, where computation is pushed away from central servers and toward the edges of the network. Computations may be carried out in mobile phones themselves, or in other local facilities, such as access points, cellular base stations or other local servers. In such setups, a mobile device may not need to perform its own computations, which may instead be performed by servers in charge of each locality.

We assume that each mobile device is capable of finding its approximate location, either from GPS or from the presence of nearby transmitters (e.g., cell towers). We also assume that the devices report their location changes to a distributed location management system such as [1]. Such location management systems can be modified easily to respond to range queries, such as how many nodes are present in a given area [17]. In the following, we assume that every node can query the location server to find nodes within any particular distance from it, and thus derive its r nearest neighbors. We follow the general distributed computation terminology of a node performing computations, but, in general, location servers may be carrying out the computations on behalf of the nodes. We give more details of location management and neighborhood queries in Subsection IV-D

B. Maximal Independent Neighborhoods

We assume there is a set of n mobile nodes $1, 2, \ldots, n$, and use p_i to denote the location of node i. The set P is the set of locations (p_is) of the nodes. 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 let $N_r(p_i)$ denote the set of r nodes nearest to point p_i, and let $N(P) = \{N(p_i) : p_i \in P\}$ denote the set of 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*.

We let $\mathcal{G}$ denote the set of clusters, at any stage of our algorithm; $\mathcal{G}$ is initialized to $\emptyset$. We let c_G denote the *center* of cluster $G \in \mathcal{G}$; the center c_G may be either a node (in P) or another location.

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 Find a maximal independent subset of neighborhoods from the set $N_r(P)$, add each as a cluster in $\mathcal{G}$, and *mark* the nodes (as "clustered") in these sets.
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 added to clusters in step M2 are called *canonical* members of the cluster, while nodes that are added in step M3, are called the *outer* members.

Next, we argue that this simple algorithm approximates an optimal clustering. Let $d_r^{\max} = \max_i d_r(p_i)$ be the largest distance from a node to its r^{th} nearest neighbor. Let D_{OPT} be the diameter of the largest cluster in an optimal clustering. We then observe

Observation 4.1. $D_{OPT} \geq d_r^{\max}$.

Proof: Suppose $p_i \in P$ is a point achieving $d_r^{\max}$: $d_r(p_i) = d_r^{\max}$. Since any disk of radius less than d_r^{max} 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 $d_r^{max}/2$. Thus the diameter D_{OPT} must be at least $d_r^{\max}$. □

Lemma 4.2. *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, then 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, $|p_i - c_G| \leq |p_i - p_j| \leq d_r(p_i) + d_r^{\max}$. □

From the above lemma it follows that the algorithm produces a 4-approximation of the diameter:

Corollary 4.3. *The diameter of any $G \in \mathcal{G}$ is at most $4D_{OPT}$.*

Proof: Consider any $p_i, p_j \in G$. By Lemma 4.2, $|p_i - c_G| \leq d_r(p_i) + d_r^{\max} \leq 2d_r^{\max}$ 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 maximal independent subsets in step M2 can be computed rapidly, in time $O(\log n)$, using the randomized parallel algorithm of Alon et al. [3], applied to compute a maximal independent set in the intersection graph of the neighborhoods

$N(P)$ (i.e., in the graph whose nodes are the r-neighborhoods $N(p_i)$ and whose edges link two r-neighborhoods that have a nonempty intersection).

The algorithm just described guarantees a 4-approximation overall; however, from the point of view of a particular node this may not be satisfactory. The bound on the diameter for all clusters is dominated by the worst case – the clusters in the sparsest neighborhood. A node in a densely populated region can justifiably expect to be assigned to a cluster center close to itself, which is not guaranteed by the algorithm above. We thus describe next another algorithm that guarantees geographic coherence of clusters, meaning that the distance of a node p_i to its cluster center is bounded by a factor of its distance, $d_r(p_i)$, to its r^{th} nearest neighbor.

C. Distributed Sweep Algorithm with Coherence Guarantee

In this strategy we create clusters in the dense regions first, and then move to sparser regions.

Finding maximal independent sets. At each node p_i, we consider the function $d_r(p_i)$, and we assume, without loss of generality, that the function values $d_r(p_i)$ are distinct (ties can be broken according to node id numbers). 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, as follows. In each round:

1) Every undecided and unclustered node p_i requests permission from nodes in $N_r(p_i)$ to become cluster center.
2) 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.
3) If one or more nodes in $N_r(p_i)$ *deny* permission for p_i to become cluster center, then p_i marks itself as *decided*, implying that it will not try to become cluster center any more.

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

1) If p_j is unclustered *and* all *undecided* nodes $p_{j'} \in N_r(p_j)$ have values of $d_r(p_{j'})$ greater than $d_r(p_i)$, then node p_j gives permission to p_i; else,
2) if p_j is already clustered, then p_j *denies* permission to p_i; else,
3) if p_j is not clustered, then p_j *defers* permission to p_i.

This approach essentially performs a sweep, starting from the densest regions of the network, working towards the less dense regions. The nodes with their r nearest neighbors the closest have a chance to become cluster centers, while other nodes have to wait until these clusters have been formed. Once nodes in dense regions have been clustered into tight clusters, or have decided that they cannot form an independent cluster, nodes in neighboring sparser regions get the chance to become cluster centers.

Any node left unclustered after the above process is assigned to the cluster of the nearest center, as in step M3.

Theorem 4.4. *If node p_i belongs to cluster G with center c_G, then $|p_i - c_G| \leq 2d_r(p_i)$.*

Proof: 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. Without loss of generality, suppose p_j is the first such center, that is, the one with smallest $d_r(p_j)$. Then, $d_r(p_j) \leq d_r(p_i)$, since otherwise p_j could not have been a center before p_i. 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 in step M3 each node is assigned to the nearest center, we have $|p_i - c_G| \leq |p_i - p_j| \leq 2d_r(p_i)$. □

This proof implies that the center assigned to any node is at most twice the distance to its r^{th} nearest neighbor, irrespective of locations of rest of the point set.

D. Dynamic Algorithm

In this subsection, we briefly outline the adaptation of the algorithm to mobility of nodes.

Mobility management. The challenge in maintaining the clusters in face of mobility is that motion on part of any of the nodes in a cluster requires a possible update on part of the cluster. We assume that a location service such as [1] is available. This system works as follows. It divides the plane into a quadtree hierarchy, where a square region is recursively subdivided into four square subregions. Each square at at each level is assigned a location server. The presence of each node is noted at the server for squares at each level containing the node. To avoid excessive updates to the hierarchy, when a node leaves a square s in level α, the servers at level $\alpha + 1$ are not updated immediately. Instead, they get updated when the node has passed out of the neighborhood of s consisting of 8 other squares in level α. This lazy scheme guarantees a low amortized cost to keep the data up to date.

Cluster maintenance in location hierarchy. We can adapt this scheme to our purposes as follows. Each server maintains a count of the nodes in its square. And for simplicity, we let the location servers perform the computations instead of mobile nodes and become cluster centers. Now, when a server i queries for its r-neighborhood, this query propagates up the server hierarchy, at each level α, checking the square $s_\alpha(i)$ containing i, and its eight neighbors, written as $N(s_\alpha(i))$ to see if they contain a total of r mobile nodes. Suppose level β is the first level where $N(s_\beta(i))$ contains r nodes. The radius is this neighborhood is within a constant factor of $d_r(i)$. The system then returns the neighborhood $N(s_{\beta+1}(i))$ of the next higher level $\beta + 1$ as the level containing at least r nodes. Thus, nodes in this set plays the role of N_r neighborhood.

And the algorithms from the previous subsections apply as usual. Observe that the radius of $\beta + 1$ is also $O(d_r(i))$.

Next, we modify this protocol to adapt to mobility of nodes. Observe that since we take $N(s_{\beta+1}(i))$ neighborhood, a node moving from $N(s_\beta(i))$ to a neighboring square does not require an immediate update to $d_r(i)$. The update is made only when it passes out of $N(s_{\beta+1}(i))$. 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 $s_\beta(i)$ 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 center in the neighborhood $N(s_{\beta+2}(i))$. This guarantees that cluster sizes of r are preserved.

It is possible to conversely trigger re-custering when a server detects a large influx of mobile nodes. Suppose $N(s_\beta(i))$ is the current cluster, and $N(s_\alpha(i)) \subset N(s_\beta(i))$ detects at least r nodes in its domain. Then, if $\beta - \alpha \geq 2$, it triggers a reclustering in $N(s_\beta(i))$.

V. Experimental Results

We have implemented this distributed algorithm along with the r-gather algorithm described in [2] to compare their clustering qualities.

First, just to have an intuitive understanding of the results in the static setting, what is shown below in Figure 6 are examples of clusters, for $r = 3, 5, 7, 9$ generated by the distributed algorithm on sample point cloud of size 50.

To test the quality of the clusterings generated for these algorithms for different r's we calculated the maximum of the clusters' diameters. The distributed algorithm can be tweaked in several ways, such as finding a good maximal independent set of the r-neighborhoods of the nodes. In the first implementation, we calculate the distance from each node to the rth nearest neighbor. Then find the node p which has the smallest such distance. Remove p and its r-neighbors, repeat. In a different implementation, we just ran greedy maximal independent set and repeat 20 times and take the best solution.

We also plotted these statistics against the lower-bound $d_r^{\max}$, which is defined as follows. We take each node and calculate the distance to the r-th nearest neighbor, and take the maximum such value for all nodes. Clearly, any r-gather solution cannot have maximum diameter lower than $d_r^{\max}$. For each clustering algorithm, we calculate the following two statistics:

- The maximum over all clusters, the diameter of a cluster.
- The 90th percentile of the diameter of a cluster, over all clusters.

Further, we use a real data set including the GPS coordinates of 9386 taxis in Shenzhen for a whole day, sampled at the interval of 5 minutes. The average velocity is 14 km/h.

Figure 7 shows the performance of our algorithm in comparison to [2] and $d_r^{\max}$ as a baseline, on a snapshot containing 60 random users from the dataset. Figure 8 shows the 90% percentile results.

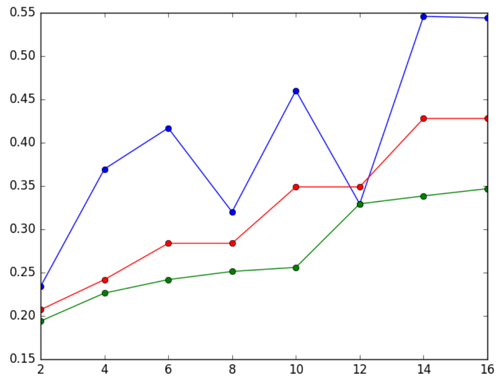

Fig. 7. Max cluster diameter. Blue curve: approximation algorithm from [2]; Red curve: distributed algorithm; Green curve: $d_r^{\max}$.

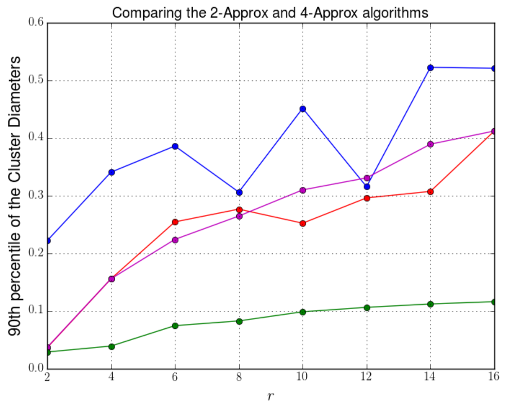

Fig. 8. 90% percentile of all cluster diameter. Blue curve: approximation algorithm from [2]; Red curve: distributed algorithm; Green curve: $d_r^{\max}$. Magenta: the distributed algorithm with the best maximal indepdenent set run over 20 iterations.

Figure 9 shows results on a larger dataset of 1500 mobile users where the distributed still performs well. Figure 10 shows the 90% percentile results.

Our main observations are:

- Our distributed algorithm usually produces better results than the 2-approximation algorithm of [2] in practice, although the approximation bound for the distributed algorithm is worse in theory.
- The distributed algorithm runs faster and therefore can be run on larger datasets
- The results (maximum cluster diameters) are close to the lower bound of $d_r^{\max}$.

VI. Conclusion

In this paper we investigated the r-gather problem, a varient of geometric clustering for the mobile settings. We improved hardness results for metrics in the Euclidean setting, and proposed algorithm for the dynamic setting when nodes move

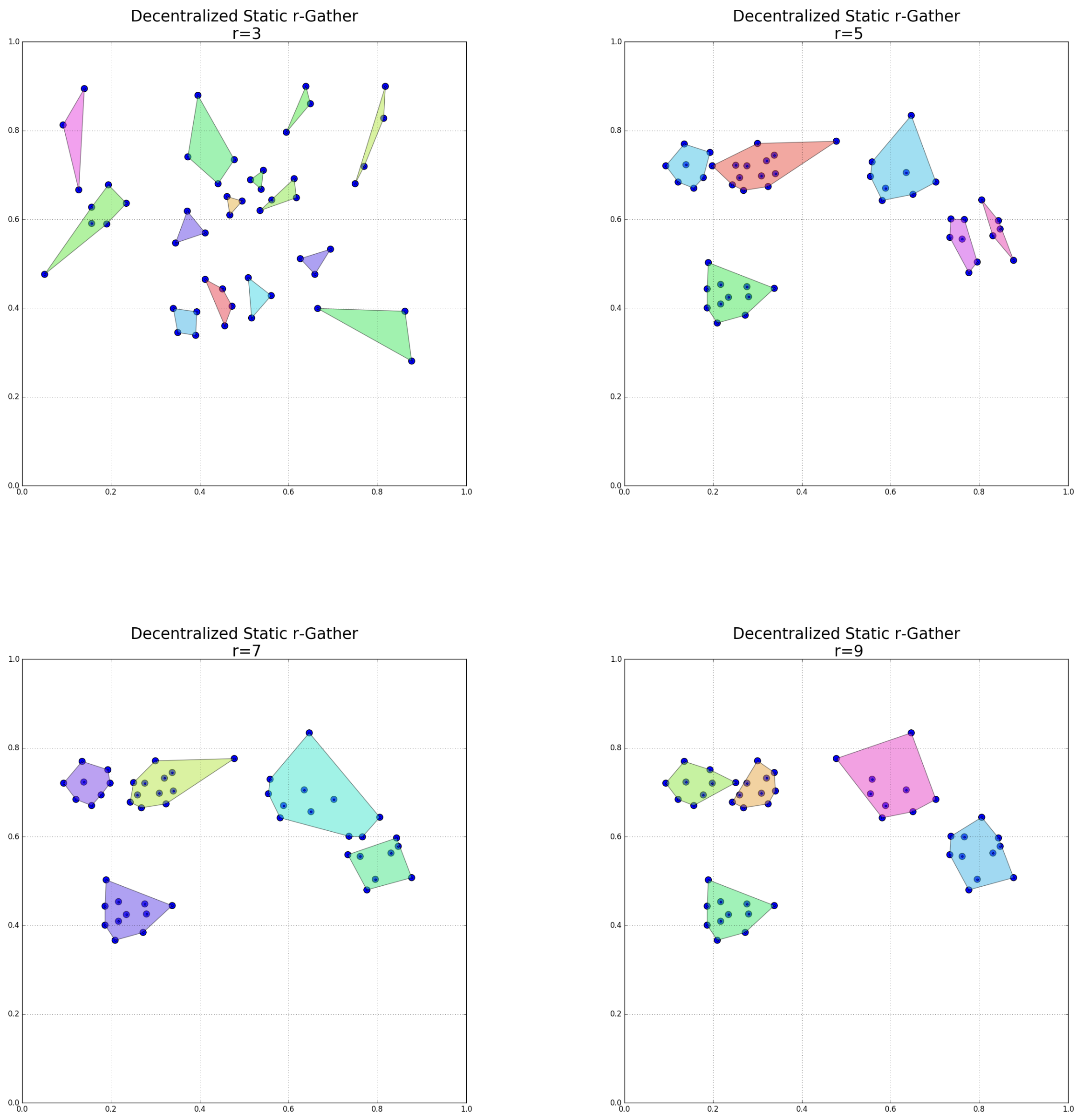

Fig. 6. The clusters produced by the distributed algorithm for a set of 50 points randomly distributed, with $r = 3$ in (top left), $r = 5$ in (top right), $r = 7$ in (bottom left) and $r = 9$ in (bottom right) respectively.

around and regrouping is allowed. Further, we proposed a distributed algorithm which uses local operations that gives a 4-approximation. We evaluated the algorithms on a real data set and show that actually the distributed algorithm (though with worst case approximation ratio worse than previous centralized algorithms) actually perform better in practice, in terms of maximum cluster diameter. We expect that the algorithms find other applications in mobile computing.

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

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

[6] S. Basagni. Distributed clustering for ad hoc networks. In *Proc. 99' International Symp. on Parallel Architectures, Algorithms, and Networks*

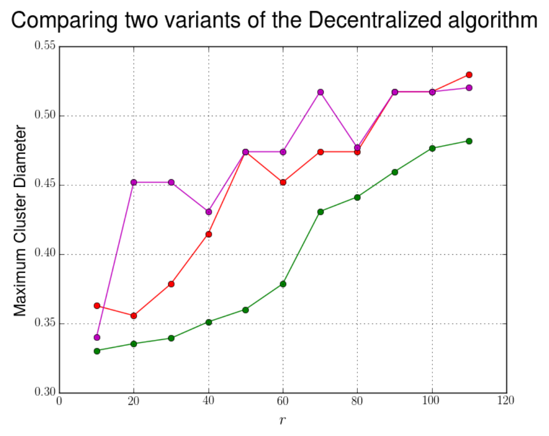

Fig. 9. Max cluster diameter on a snapshot of 1500 mobile users. Red curve: distributed algorithm; Green curve: $d_r^{\max}$. Magenta: the distributed algorithm with the best maximal indepdenent set run over 20 iterations.

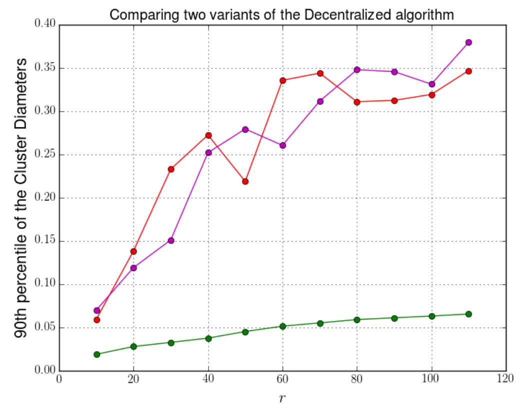

Fig. 10. 90% percentile of all cluster diameter. Red curve: distributed algorithm; Green curve: $d_r^{\max}$. Magenta: the distributed algorithm with the best maximal indepdenent set run over 20 iterations.

(I-SPAN'99), pages 310–315, June 1999.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

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

[14] 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.

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

[16] 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.

[17] R. Sarkar and J. Gao. Differential forms for target tracking and aggregate queries in distributed networks. In *ACM MobiCom '10*.

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

[19] 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.

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

[21] 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.

[22] 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.