Discrete Mobile Centers

Jie Gao[*] Leonidas J. Guibas[*] John Hershberger[†] Li Zhang[‡] An Zhu[*]

ABSTRACT

We propose a new randomized algorithm for maintaining a set of clusters among moving nodes in the plane. Given a specified cluster radius, our algorithm selects and maintains a variable subset of the nodes as cluster centers. This subset has the property that (1) balls of the given radius centered at the chosen nodes cover all the others and (2) the number of centers selected is a constant-factor approximation of the minimum possible. As the nodes move, an event-based kinetic data structure updates the clustering as necessary. This kinetic data structure is shown to be responsive, efficient, local, and compact. The produced cover is also smooth, in the sense that wholesale cluster re-arrangements are avoided. The algorithm can be implemented without exact knowledge of the node positions, if each node is able to sense its distance to other nodes up to the cluster radius. Such a kinetic clustering can be used in numerous applications where mobile devices must be interconnected into an ad-hoc network to collaboratively perform some task.

1. INTRODUCTION

Collaborating mobile devices are of interest in diverse applications, from wireless networking to sensor nets to robot exploration. In these applications there are mobile nodes that need to communicate as they move so as to accomplish the task at hand. These tasks can vary from establishing an *ad-hoc* multi-hop network infrastructure that allows point-to-point communication, to aggregating and assimilating data collected by distributed sensors, to collaboratively mapping an unknown environment. A challenge common to all these tasks is that communication is usually accomplished using low-power radio links or other short-range technologies. As a result only nodes sufficiently close to each other can communicate and therefore the communication topology of the network is strongly affected by node motion (as well as obstacle interference, etc.). The mobile networking community has been especially active in studying such problems in the context of networking protocols allowing the seamless integration of devices such as PDAs, mobile PCs, phones, pagers, etc., that can be mobile as well as switch off and on at arbitrary times. An example of such an effort is the recent *Bluetooth* specification [16].

A principle that has been discussed a number of times for enabling such collaborative tasks is the organization of the mobile nodes into *clusters* [4, 8, 12, 21]. Clustering allows hierarchical structures to be built on the mobile nodes and enables more efficient use of scarce resources, such as bandwidth and power. For example, if the cluster size corresponds roughly with the direct communication range of the nodes, much simpler protocols can be used for routing and broadcasting within a cluster; furthermore, the same time or frequency division multiplexing can be re-used across non-overlapping clusters. Clustering also allows the health of the network to be monitored and misbehaving nodes to be identified, as some nodes in a cluster can play watchdog roles over other nodes [19].

Motivated by these issues, in this paper we study the problem of maintaining a clustering for a set of n moving points or nodes in the plane. There is, of course, a huge literature on clustering, as the problem in many variations has been studied by several different communities, including operations research, statistics, and computational geometry. In our setting we assume that all the nodes are identical and each can communicate in a region around itself, which we take to be an L_p ball. For most of the paper we will focus on a ball in the L_∞ metric, that is an axis-aligned square whose side is of length r, as this makes the analysis the simplest. We will say that two nodes such that one is within the communication range of the other are *visible* to each other. We seek a minimal subset of the n nodes, the *centers*, such that every node is visible to at least one of the centers. In the mobile device setting, unlike the general facilities location context, it is appropriate to insist that the centers are located at the nodes themselves, as these are the only active elements in the system; thus we are interested in "discrete center" problems. We survey the literature on the static version of this problem in Section 2. The problem is known to be NP-complete and most of the extant work has focused on approximation algorithms.

Much less is known, however, about maintaining a clustering on mobile nodes. There have been a few papers in the mobile networking community [4, 8, 12, 21] proposing and simulating a number of distributed algorithms for cluster maintenance, but to our knowledge there has been very little prior work on a theoretical analysis of the problem. Bespamyatnikh *et al.* investigated the problem of maintaining a continuous 1-center and 1-median [7]. Their main observation is that the center or median might move faster than the

[*]Department of Computer Science, Stanford University, Stanford, CA 94305. E-mail: jgao,guibas,anzhu@cs.stanford.edu.
[†]Mentor Graphics, 8005 S.W. Boeckman Road, Wilsonville, OR 97070. E-mail: john_hershberger@mentor.com.
[‡]Compaq Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301. E-mail: l.zhang@compaq.com.

SCG'01, June 3-5, 2001, Medford, Massachusetts, USA.

points in the optimal solution. They presented a 2-approximation algorithm for both the 1-center and the 1-median cases with restricted velocity on the resulting center and median.

In this paper we present a new randomized clustering algorithm that provides a set of centers that is an $O(1)$ approximation to the optimal discrete center solution with very high probability. Our algorithm uses $O(\log\log n)$ rounds of a "center nomination" procedure in which each node nominates another node within a certain region around itself to be a center; a round of the nomination procedure can be implemented in $O(n \log n)$ time. Furthermore, we show how this approximately optimal clustering can be maintained as the nodes move continuously. The goal here is to exploit the continuity of the motion of the nodes so as to avoid recomputing and updating the clustering as much as possible. We employ the framework of *Kinetic Data Structures* (KDS) [5, 15] to provide an analysis of our method. For this analysis we assume that nodes follow posted flight plans, though they may change them at any moment by appropriately notifying the data structure. The correctness of the clustering is certified by a set of conditions, or *certificates*, whose predicted failure times are inserted as events into an event queue. At each certificate failure the KDS certification repair mechanism is invoked to repair the certificate set and possibly the clustering as well. We show that the proposed structure is responsive, efficient, local, and compact. Certificate failures and flight-plan updates can be processed in expected time $O(\log^{3.6} n)$ and $O(\log n \log\log n)$ respectively. Under the assumption of pseudo-algebraic motions for the nodes, we show that our structure processes at most $O(n^2 \log\log n)$ events (certificate failures). We also give a construction showing that for any constant $c > 1$, there is a configuration of n points moving linearly on the real line so that *any* c-approximate set of centers must change $\Omega(n^2/c^2)$ times. Thus, even though an approximate clustering is not a canonical structure [2], we can claim efficiency for our method.

Our clustering algorithm has a number of other attractive properties:

- We can show that the clustering produced is an $O(1)$ approximation with high probability *during the entire history of a pseudo-algebraic motion*—not only at a particular instant; this addresses one of the most common concerns when randomization is used in kinetic algorithms.
- The clustering generated by the algorithm is *smooth* in the sense that, degeneracies aside, clusters always change by adding or deleting a small (polylog) number of nodes; furthermore, when new centers need to appear, they are created near existing centers. This allows the incremental updating of information maintained by an application and convenient initialization of newly formed centers.
- The algorithm supports dynamic insertion and deletion of nodes, with the same update bound as for a certificate failure.
- The algorithm can be implemented in a distributed fashion: each node need only reason about the nodes visible to it.
- If a node can sense when its set of neighbors changes within certain subregions of its visibility range, the algorithm can be implemented without any knowledge of the actual positions of the nodes. This is advantageous in mobile networking applications in which a GPS-type device would be expensive to provide with every mobile node. Hybrid schemes are also possible, where a node uses kinetic-style prediction to estimate roughly when such events might occur and employs active sensing (e.g., polling) only then, so as to minimize power consumption.

The remainder of the paper is organized as follows. Section 2 summarizes previous work on discrete centers and related problems. Section 3 introduces the basic algorithm and analyzes the approximation factors for the clusterings it produces. Section 4 describes a hierarchical version of the algorithm and proves the constant approximation bound. Section 5 shows how this clustering can be maintained kinetically under node motion and analyzes the performance of the algorithm. Finally Section 6 concludes with some directions for future research.

2. PREVIOUS WORK

There is little prior work on this specific mobile clustering problem. The static version of the problem is known to be NP-complete [10] and to admit a PTAS (polynomial time approximation scheme). A variant, the connected dominating set problem, has been studied extensively as well.

The static version of the discrete clustering problem is equivalent to finding the minimum dominating set in the intersection graph of unit disks. The dominating set problem is defined as follows. Given a graph $G = (V, E)$, find a minimum size subset V' of vertices, such that every vertex in $V \setminus V'$ is adjacent to some node in V'.

For our problem we build a graph G on all the points and create an edge between two points if a disk of size r centered at one point contains the other point. The goal is to find the minimum dominating set in G. The dominating set problem on general graphs is NP-complete and hard to approximate as well. In fact, no algorithm with approximation factor better than $(1-\epsilon)\ln n$ exists unless NP $\subset$ DTIME($|V|^{\log\log|V|}$) [9]. A greedy algorithm can construct a solution of size $k^* \log n$, where k^* is the size of the optimal solution (this follows from a reduction to the set cover problem).

For the dominating set in an intersection graph, several approximate algorithms have been developed. The simple greedy algorithm gives a constant-factor approximation. Hunt *et al.* [18] gave a PTAS, providing a solution of size no more than $(1+\epsilon)k^*$, for the optimal k^* and any $\epsilon > 0$. The basic idea of the PTAS comes from an algorithm by Hochbaum and Maas [17] for the continuous variant, in which centers can be arbitrary points on the plane. Roughly speaking, the method in [18] divides the space into strips of a certain width, and a sub-problem is formed by grouping several consecutive strips together and proceeding recursively.

The connected dominating set has the extra condition that the subgraph induced by V' must be connected. This problem is NP-complete as well [11]. Guha and Khuller designed a greedy algorithm that achieves an approximation bound of $O(\log n)$ in a general graph [14]. Their algorithm is a slight modification of the natural greedy algorithm (pick the next available vertex with the maximum degree).

Unfortunately, these algorithms from the theory community do not easily extend to the mobile case. Fixed spatial subdivisions generate many updates to the clustering as points move across subdivision boundaries. Greedy algorithms are sequential by nature and highly sensitive to small changes. The networking community, on the other hand, has developed many routing protocols to deal with changing network topologies. However, no theoretical bounds have been derived for many of these heuristics. We note that our basic algorithm is similar to the Lowest-ID Cluster Algorithm proposed by Gerla and Tsai [12]. Experiments show that this scheme works well in practice. A similar idea leads to the Max-Min D-clustering scheme that was proposed by Amis *et al.* [3]. For the connected dominating set problem, Wu and Li proposed a distributed algorithm that performs badly in the worst case ($O(n)$-approximation) but works well in simulation [23].

3. BASIC ALGORITHM

Before presenting the algorithms, we first give some formal definitions. A d-cube with size r is a d-dimensional axis-aligned cube with *side length* r. When $d = 1$ or 2, a d-cube is also called an interval or a square, respectively. For two points p and q, p is said to be r-*covered* by or r-*visible* from q if p is inside the cube with size r centered at q. For a set of n points (nodes) $P = \{p_1, p_2, \ldots, p_n\}$ in the d-dimensional space, a subset of P is called an r-*cover* of P if every point in P is r-covered by some point in the subset. The points in a cover are also called (*discrete*) *centers*. A minimum r-cover of P is an r-cover that uses the minimum number of points. We denote by $\alpha_P(r)$ (or $\alpha(r)$ if P is clear from the context) the number of points in a minimum r-cover of P. An r-cover is called a c-approximate cover of P if it contains at most $c \cdot \alpha_P(r)$ points. When r is not mentioned, we understand it to be 1. In this paper, we are interested in computing and maintaining $O(1)$-approximate covers for points moving in the space. For the sake of presentation, we will discuss our algorithms for points in one and two dimensions, but our techniques can generally be extended to higher dimensions. In the rest of the paper, $\log$ is understood to be $\log_2$, and $\ln$ to be $\log_e$. This distinction is important because in a few places, $\log$ appears in exponents, and we have to make the base explicit in order to give precise asymptotic bounds.

In the following, we first present the algorithms for the static version of the problem and later describe their implementation for moving points.

3.1 Description of the basic algorithm

The algorithm, which is distributed in nature, is the following: we impose a random numbering (a permutation of $1, 2, \ldots, n$) onto the n points, so that point p_i has an index N_i. In most situations in practice each mobile node is given a unique identifier (UID) at set-up time, and these UIDs can be thought of as providing the random numbering (either directly, or via a hash function on the UIDs). Each point p_i nominates the largest indexed point in its visible range to be a center (note that a point can nominate itself if there is no other point with larger index inside its range). All points nominated are the centers in our solution. A cluster is formed by a selected center and all the points that nominated it.

First, we note that randomization is essential for the performance of our scheme. Without randomization, the only approximation bound that holds, even in the one-dimensional case, is the trivial $O(n)$ bound. For example, consider the one-dimensional case in which n points are equally spaced along a unit interval, with their indices increasing monotonically from left to right. Each point in the left half of the set has a different center, which is the rightmost point within distance $\frac{1}{2}$ of it. Thus the number of centers produced by the algorithm is $n/2$, even though the optimal covering uses only a single center.

In the following, we are able to show that for *any* configuration, if the ordering is assigned randomly, the basic algorithm yields a sub-linear approximation ($\log n$ in 1-D, and $\sqrt{n}$ in higher dimensions) with high probability.

3.2 Analysis for the basic algorithm

3.2.1 Analysis for the one-dimensional case

As a warm-up, we first present the analysis for this algorithm in the 1-D case, where points lie along the real line and the unit square corresponds to the unit interval.

LEMMA 3.1. *If V' is a subset of the points that are mutually visible to each other, then there is at most one point in V' nominated by points in V'.*

PROOF. Only the point with the maximum rank in V' can possibly be nominated by other points in V'. □

Let the optimal centers be O_i, $i = 1, 2, \ldots, k$. We partition each unit interval U_i centered at O_i into two sub-intervals with O_i as the dividing point. We define the *visible range* of an interval to be all the points on the line that are visible to at least one of the nodes in the interval and call nodes in the visible range the *visible set* for that interval.

THEOREM 3.2. *The basic algorithm has an approximation factor of $4 \ln n + 2$ in expectation.*

PROOF. It suffices to show that, for each sub-interval S, the number of centers nominated by points in S is at most $2 \ln n + 1$. The visible range of S is contained in an interval of size $\frac{3}{2}$ as shown in Figure 1. We use S_l to denote the portion of the interval to the left of S and S_r for the right portion. Note that the points in S are mutually visible. Lemma 3.1 shows that all the points in S nominate at most one center in S.

Now we calculate the expected number of centers in S_r that are nominated by points in S. Let $x = |S|$ and $y = |S_r|$ be the number of nodes in the respective subintervals. Scan all points from left to right in S_r. The i^{th} point in S_r can be nominated by a point in S only if it has the largest index compared to all points to its left in $S \cup S_r$. Therefore, the expected number of centers in S_r is no more than $\sum_{i=1}^{y} \frac{1}{x+i} < \ln n$. (Note that $\sum_{i=1}^{n} \frac{1}{i}$ is asymptotically equal to $\ln n + \gamma$, where $\gamma = 0.5772\ldots$ is Euler's constant [13]. Because our sum begins with $\frac{1}{x+1}$, our result is less than $\ln n$ if $x \geq 1$. But the expected number of centers is also bounded by x, and so we can assume that $x \geq 1$.) A similar argument works for S_l, and we can conclude that all points in S nominate at most $2 \ln n + 1$ centers. □

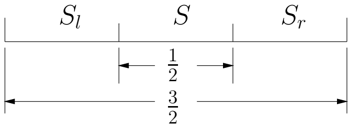

Figure 1: Visible range in 1-D

We remark that the approximation bound is asymptotically tight. Consider the following situation in Figure 2: the unit interval centered at p is divided into two sub-intervals S_l and S_r. S_l contains $\sqrt{n}$ evenly distributed points, each of which can see $\sqrt{n}$ more points in S_r from left to right. In this configuration, with probability 0.5, the leftmost point q in S_l nominates a point in the first group of $\sqrt{n}$ points in S_r. This is because q sees $2\sqrt{n}$ points ($\sqrt{n}$ in S_l and another $\sqrt{n}$ in S_r). Under a random numbering, the point with the maximum rank falls in S_r with probability 0.5. In general, a point in the i^{th} group of S_r is nominated by the i-th point in S_l with probability $\frac{1}{i+1}$. Thus the expected number of centers (in S_r alone) is $\sum_{i=1}^{\sqrt{n}} \frac{1}{i+1} = \Omega(\log n)$. But a single center at p covers all the points.

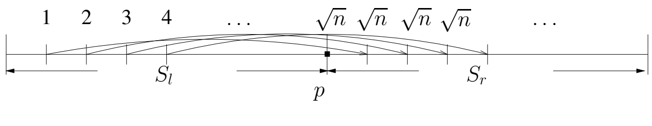

Figure 2: Lower bound for the 1-D case

We can also prove that the $O(\log n)$ upper bound holds with high probability. This fact is useful in our hierarchical algorithm, which achieves a constant approximation factor, and in our kinetic maintenance algorithms.

THEOREM 3.3. *The probability that the basic algorithm selects more than $ck^* \ln n$ centers is $O(1/n^{\Theta(c^2)})$, where k^* is the optimal number of centers.*

PROOF. We divide the optimal intervals in the same way as in the proof of Theorem 3.2. Consider a sub-interval S and its right portion S_r. We look for the fraction of random numberings such that points in S nominate not too many centers in S_r. We sort all points in $S \cup S_r$ according to their coordinates from left to right into a sequence of m points. The sequence of their indices can be viewed as a random permutation on numbers $1, 2, \ldots, m$. Each center in S_r must have a bigger index than all the other points to its left. Thus, to guarantee that points in S nominate no more than s centers in S_r, it suffices to ensure that the total number of left-to-right maximal indices in the sequence is no more than s. The number of permutations with s left-to-right maxima is known as the Stirling number $C(m, s)$, which is asymptotically equal to $m!\, e^{-\frac{\theta^2}{2}}/\sqrt{2\pi}$, for $s = \ln m + \theta\sqrt{\ln m}$, as $m \to \infty$ and $\theta/m \to 0$ [22]. Let x be the random variable of the number of left-to-right maxima in this permutation. Then we have

$$\text{Prob}(x \geq s) = \int_s^\infty P(l)\, dl \leq \int_s^\infty \frac{C(m,l)}{m!}\, dl\,.$$

If we set $s = c \ln n$, this formula becomes

$$\begin{aligned} \text{Prob}(x \geq c\ln n) &\leq \int_{(c-1)\sqrt{\ln n}}^\infty \frac{e^{-\frac{\theta^2}{2}}}{\sqrt{2\pi}} \sqrt{\ln m}\, d\theta \\ &\leq \sqrt{\frac{\ln m}{\pi}} \int_{(c-1)\frac{\sqrt{\ln n}}{\sqrt{2}}}^\infty e^{-x^2} dx \\ &\leq \frac{n^{-\frac{(c-1)^2}{2}}}{\sqrt{2\pi}(c-1)} \leq n^{-\Theta(c^2)}\,. \end{aligned}$$

(The approximation in the last line is obtained using the asymptotics of the complementary error function [1].) For $O(k^*)$ sub-intervals, since each needs to be considered only twice for its left and right points, the probability that there are more than $ck^* \ln n$ centers is less than $\Theta(n)\, n^{-\Theta(c^2)}$, which is $O(n^{-\Theta(c^2)})$. □

3.2.2 Analysis for the two-dimensional case

Unfortunately this good result does not extend to higher dimensions. We will show that in two (and higher) dimensions, the method above produces a $\Theta(\sqrt{n}\log n)$ approximate cover with high probability. The analysis is similar to the 1-D case. Again, we consider the sub-squares with side length 0.5. For such a square S, suppose that L is the visible range of S. Clearly, L is a square of side length $3/2$ and can be partitioned into 9 sub-squares where S is the center one (Figure 3). Now, we have the following lemma:

LEMMA 3.4. *Suppose that $|L| \leq m$. Then the number of centers nominated inside S is $O(\sqrt{m})$ in expectation. Furthermore, for any $c > 0$, the probability that S contains more than $8c\sqrt{m}\ln m + 1$ centers is bounded by $O(1/m^{c^2 \ln m})$.*

PROOF. We need to consider only those points inside L. It suffices to bound the number of centers nominated by points in each sub-square S' of L. If $S' = S$, since all the points are mutually visible in S, there can be at most one point nominated. For $S' \neq S$, suppose that $x = |S|$, $y = |S'|$. A point $p \in S$ can be nominated by a point $q \in S'$ if q finds that p has the largest index in its visible range. Since q sees all points in S', p must have rank higher than all the points in S'. Thus, the probability that p can be nominated is at most $\frac{1}{1+y}$. Thus, in expectation, there are at most $\frac{x}{1+y}$ points nominated. On the other hand, since there are only y points in S', there can be at most y centers nominated by points in S'. The expected total number of centers is therefore no more than $\min(y, \frac{x}{1+y}) \leq \sqrt{x+y+1} - 1 < \sqrt{m}$.

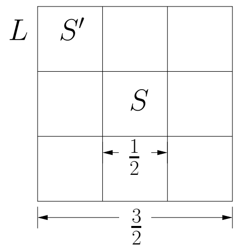

Figure 3: Visible range in 2-D

Furthermore, if $y < c\sqrt{m}\ln m$, then we know that S' cannot nominate more than $c\sqrt{m}\ln m$ points. Otherwise, S' contains $y > c\sqrt{m}\ln m$ points. In order to nominate s points in S, S must contain at least s points with higher ranks than all the points in S'. That is, S must contain the s highest ranked points in $S \cup S'$.

The probability for this to happen is:

$$\binom{x}{s} \Big/ \binom{x+y}{s} = \frac{x!(x+y-s)!}{(x+y)!(x-s)!}$$
$$< \left(\frac{x}{x+y}\right)^s < \left(1 - \frac{y}{m}\right)^s < \left(1 - \frac{c\sqrt{m}\ln m}{m}\right)^s.$$

Thus, if $s > c\sqrt{m}\ln m$, we have

$$\begin{aligned} \text{Prob}(x \geq s) &< \left(1 - \frac{c\ln m}{\sqrt{m}}\right)^{c\sqrt{m}\ln m} \\ &< \left(\frac{1}{e}\right)^{c^2\ln^2 m} = O\Big(\frac{1}{m^{c^2 \ln m}}\Big)\,. \end{aligned}$$

Summing over all the 9 sub-squares, we see that the expected number of centers nominated in S is bounded by $O(\sqrt{m})$, and with high probability, the number of centers nominated is bounded by $O(\sqrt{m}\ln m)$. □

By Lemma 3.4, it is easy to obtain

THEOREM 3.5. *For points in the plane, the algorithm has an approximation factor of $O(\sqrt{n})$ in expectation. Further, the probability that there are more than $\sqrt{n}\ln n \cdot k$ centers is $O(1/n^{\ln n - 1})$, where k is the optimal number of centers.*

PROOF. Consider an optimal covering U_i, $1 \leq i \leq k$. We partition each U_i in the optimal solution into 4 quadrant sub-squares and apply Lemma 3.4 to each sub-square. Since there are at most $O(n)$ sub-squares, the high probability result also holds. □

Again, this bound is asymptotically tight. Consider the configuration in Figure 4: the upper left sub-square S_1 has $\sqrt{n}$ points, each of which can see a distinct set of $\sqrt{n}$ points in the lower right sub-square S_2. Each point in S_1 will nominate a point in S_2 with probability $\frac{1}{2}$. Thus the expected number of centers in S_2 is $\Omega(\sqrt{n})$. We remark that in this analysis, the use of the unit square and the

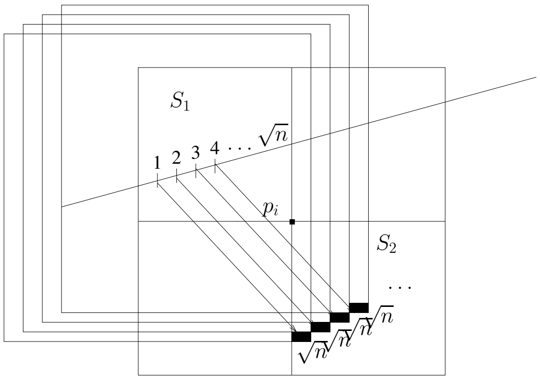

Figure 4: Lower bound for the 2-D case

dimensionality is not essential. It is easy to extend the analysis to any centrally symmetric covering shape in any dimension; the constant factors, however, depend on the covering shape and the dimensionality.

Note also that the worst-case examples that prove the tightness of the upper bounds in Theorems 3.2 and 3.5 require a significantly non-uniform distribution of the points. If the points are uniformly distributed, or within a constant factor of being uniformly distributed, then the approximation factor is $O(1)$. This observation may explain the good performance of the basic algorithm observed in practice [12].

4. HIERARCHICAL ALGORITHMS FOR CLUSTERING

The basic algorithm is simple, but it achieves only an $O(\sqrt{n})$ approximation for points in the plane. To obtain a constant-factor approximation, we will use a hierarchical algorithm that proceeds in a number of rounds. At each round we apply the basic algorithm to the *centers* produced by the previous round, using a larger covering cube. Suppose that $\delta_i = 2^i/\log n$, for $i > 0$. Initially, set P_0 to be P, the input set of points. At the i^{th} step, for $1 \leq i < \log\log n$, we apply the basic algorithm using squares with side length δ_i to the set P_{i-1} and let P_i be the output. The final output of the algorithm is $P' = P_{\log\log n - 1}$. (To make our analysis fully rigorous, we would need to use $\lfloor \log n \rfloor$ and $\lfloor \log\log n \rfloor$ instead of $\log n$ and $\log\log n$; however, in the interest of readability, we will omit the floor functions from this paper.) We claim that:

LEMMA 4.1. *P' is a 1-cover of P.*

PROOF. We actually prove a stronger statement: P_i is a $\frac{2^{i+1}}{\log n}$-cover of P.

We proceed by induction. The assertion is clearly true when $i = 0$. Suppose that it is true for i, i.e., every point $p \in P$ can be covered by a size $2^{i+1}/\log n$ square centered at a point $q \in P_i$. If q is also in P_{i+1}, then p is covered. Otherwise, there must be a q' so that q nominates q' at the $(i+1)^{\text{th}}$ step. Thus, p is covered by q' with a square with side length $2^{i+1}/\log n + \delta_{i+1} = 2^{i+2}/\log n$. That is, P_{i+1} is a $(2^{i+2}/\log n)$-cover of P. □

In the following, we bound the approximation factor for P'. To explain the intuition, we first consider the situation when P admits a single center, i.e., there is a unit square that covers all the points in P. Recall that $\alpha(x)$ denotes the number of centers of an optimal covering of P by using squares with side length x. First, we observe that

LEMMA 4.2. $\alpha(x) \leq \frac{4}{x^2}$.

PROOF. We uniformly divide the unit square into $\frac{4}{x^2}$ small squares of size $\frac{x}{2}$. We then pick one point from each non-empty small square, which gives an x-cover with at most $\frac{4}{x^2}$ centers. □

According to Theorem 3.5, the expected size of P_{i+1} is at most $c\sqrt{|P_i|}\alpha(\delta_{i+1})$, for some constant $c > 0$. Denote by n_i the size of P_i. We have the following recursive relation:

$$n_0 = n\,, \quad n_{i+1} \leq c\sqrt{n_i}\alpha(\delta_{i+1}) \leq c\sqrt{n_i}\frac{4\log^2 n}{2^{2i+2}}\,.$$

By induction, it is easy to verify that

$$n_i \leq \frac{(c^2\log^4 n)n^{\frac{1}{2^i}}}{4^{2i-4}}\,.$$

Thus $|P'| = n_{\log\log n - 1} \leq c^2 2^{14} = O(1)$.

We cannot apply this argument directly to the general case because $\alpha(x)$ can be as large as $\Theta(n)$. In order to establish a similar recursive relation, we consider points restricted to lie in squares of a certain size. For any square S with side length δ_i, let $m_i(S)$ denote the expected value of $|P_i \cap S|$. Further, let m_i denote the maximum of $m_i(S)$ over all the squares S with size δ_i. We then have the following relation between m_i's.

LEMMA 4.3. *$m_{i+1} \leq c\sqrt{m_i}$, for some constant $c > 0$ and any $0 \leq i < \log\log n - 1$.*

PROOF. Consider a square S of side length δ_{i+1}. Its visible region L, with respect to side length δ_{i+1}, is a square with side length $2\delta_{i+1} = 4\delta_i$. Thus L can be covered by $4^2 = 16$ squares with side length δ_i. That is, $|P_i \cap L| \leq 16 m_i$ in expectation. By Lemma 3.4, we know that the expected number of points inside S that survive after the $(i+1)^{\text{th}}$ step of the algorithm is $O(\sqrt{|P_i \cap L|}) = O(\sqrt{m_i})$. Thus, we have $m_{i+1} \leq c\sqrt{m_i}$, for some constant $c > 0$. □

Now, we can prove that

THEOREM 4.4. *P' is a constant approximation to the optimal covering of P with unit squares, with high probability.*

PROOF. Clearly $m_0 \leq n$. Solving the recursive relation in Lemma 4.3, we find that $m_i \leq O(c^2 n^{\frac{1}{2^i}})$. Setting $i = \log\log n - 1$, we have $m_{\log\log n - 1} = O(1)$, i.e., for a square S with side length $\frac{1}{2}$, the expected number of points of P' inside S is $O(1)$.

Now, suppose that an optimal cover uses k unit squares. We can then cover all the points by $4k$ squares with side length $\frac{1}{2}$. Since each of these squares contains $O(1)$ points in P' in expectation, the total number of points in P' is bounded by $O(k)$. Note that in the proof above, we assign an independent random ordering to each level. Since we have a high probability argument as stated in Lemma 3.4, we can define $m_i'(S) > m_i(S)$ so that $|P_i \cap S|$ is smaller than $m_i'(S)$ with high probability. We also define m_i' to be the maximum among all the $m_i'(S)$ values for squares S with size δ_i. Then we can establish the recursive relation $m_{i+1}' \leq c\sqrt{m_i'}\log m_i'$ and prove that the hierarchical algorithm achieves an $O(1)$ approximation with high probability. The details are omitted. □

5. KINETIC DISCRETE CLUSTERING

To kinetize the algorithm, we place a half-size square centered over each point. If two such squares intersect, we know the corresponding points are mutually visible. In this section when we say "squares," we refer to these half-size squares.

5.1 Standard KDS implementation

The intersection relation between two squares can change only at discrete times. If two squares of the same size intersect with each other, one square must have a corner inside the other square. Therefore, we can maintain the left and right extrema of squares in x-sorted order and the top and bottom extrema of squares in y-sorted order. The certificates of the KDS are the ordering certificates for the x- and y-sorted lists of square extrema. We maintain the lists containing the extrema of active squares for each level of the hierarchy. An event is a certificate failure. When an event happens, we first check whether it is a "real" event, i.e., whether it causes two squares to start/stop intersecting. When two squares S_1, S_2 start intersecting, we will need to check the square with the lower rank, say S_1, to see if its nomination has a lower rank than S_2. If so, we need to change S_1 to point to S_2. If S_1, S_2 stop intersecting, we need to check if S_1 nominated S_2. If so, we need to find another overlapping square with the highest rank. To answer this query efficiently, we maintain a standard range search tree [20] for the n points. For our purpose, the internal nodes of the second-level binary trees in the range tree are augmented with the maximum index of the points stored at descendants of each node. This will let us find the points within a query square that are larger than some query index in $O(\log^2 n)$ time. To maintain the range search trees kinetically, we keep sorted lists of the x- and y-coordinates of the points themselves, in addition to the sorted lists containing the extrema of the squares on each level. A range tree can be updated by deleting a point and re-inserting it in the right place [6].

For the hierarchical algorithm, we need to maintain these structures for each level. In addition, we also need to insert or delete a point to or from a level, as a consequence of an event happening at a lower level. This requires the sorted lists and range search trees used in the basic algorithm above to be dynamic. These requirements can easily be satisfied by maintaining balanced binary search trees and dynamic range search trees.

5.2 Kinetic properties

This kinetic data structure has most of the properties of a good KDS [5]. We assume the points have bounded-degree algebraic motion in the following arguments.

To analyze the efficiency, i.e., the number of events, of our algorithms, we first give some lower bound constructions.

LEMMA 5.1. *The number of changes of the optimal cover for n points in motion is $\Theta(n^3)$ in the worst case.*

PROOF. Consider the graph G in which each vertex represents a point and each edge joins a visible pair of points. Clearly, the minimum discrete covering of the points is exactly the same as the minimum dominating set of the graph. The graph can change only when two points become or cease to be visible to each other. For bounded degree algebraic motions, this can happen only $O(n^2)$ times. For each such event, the change to the minimum covering is at most $O(n)$. Thus, in the worst case, the number of changes is $O(n^3)$.

We now construct an example in which any optimal cover must change $\Theta(n^3)$ times. The construction uses $6m + 6$ static points along the perimeter of a rectangle $[0, R] \times [0, 1.6]$, where $R = 0.4(3m + 1)$. The left and right sides of the rectangle have three points apiece, located at $(0, 0.4i)$ and $(R, 0.4i)$ for $i = 1, 2, 3$. The top and bottom sides of the rectangle have $3m$ points apiece, located at $(0.4i, 0)$ and $(0.4i, 1.6)$, for $i = 1, \ldots, 3m$. We label the points counter-clockwise from 0 to $6m+5$ as shown in Figure 5. In this configuration, each point i can see the points $i - 1, i + 1$ (modulo $6m + 6$) and no other points. Thus, an optimal cover contains $2m + 2$ centers and can be realized in one of three ways by using points $3i$, $3i + 1$, or $3i + 2$, respectively, which we call types 0, 1, and 2, respectively. Clearly, to change from one type to another, we need to make $\Theta(m)$ changes to the cover.

Now consider what happens when a single point p moves linearly along the x-axis. For any i, suppose that q_j is the middle point between the pair $3i + j, 3i + j + 1$, for $0 \leq j \leq 2$. When p is located at q_j, the only points p can see are $3i + j$ and $3i + j + 1$. Thus, an optimal cover has to use either $3i + j$ or $3i + j + 1$ as a center. In other words, an optimal cover has to be of type j or $j + 1$. It is easily verified that when p moves from q_0 to q_2, an optimal cover has to change its type. Therefore, an optimal cover undergoes $\Theta(m)$ changes when p moves from q_0 to q_2. When p moves from $(0, 0)$ to $(R, 0)$, the number of changes is $\Theta(m^2)$. We repeat this procedure by sending m points along the x-axis, passing through the interval $[0, R]$ one at a time. This causes a total of $\Theta(m^3)$ changes to optimal covers. The total number of points is $n = 7m + 6$, so the total number of center changes is $\Theta(n^3)$. □

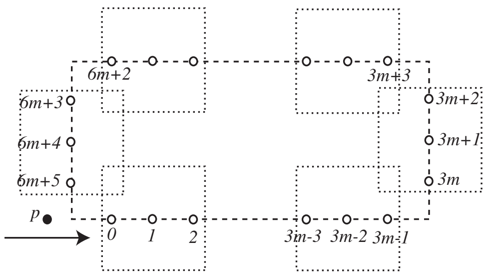

Figure 5: Lower bound for optimal coverings

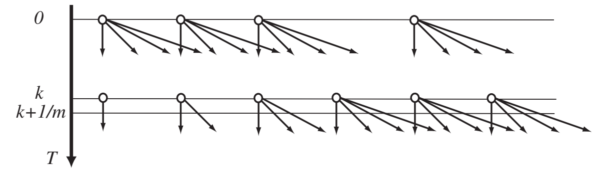

Figure 6: Lower bound approximate coverings

While the optimal cover in this construction changes $\Omega(n^3)$ times, a 2-approximate cover does not change at all—we can simply use an optimal cover for the static points and assign each moving point to be a center. However, in the following, we will show that for any constant c, there is a set of moving points that forces any c-approximate cover to change $\Omega(n^2/c^2)$ times.

THEOREM 5.2. *For any constant $c > 1$, there exists a configuration of n points moving linearly on the real line so that any c-approximate cover undergoes $\Omega(n^2/c^2)$ changes.*

PROOF. In the following, we assume that c is an integer and $n = 2cm$, where $m > 2c$ is an integer. We group n points into m groups, each containing $2c$ points. We label each point by (i, j)

where $0 \leq i < m$ is the group number, and $0 \leq j < 2c$ is the numbering within each group. Initially, all the points in the i^{th} group are located at $i \cdot 2m$, and the speed of the point (i, j) is $j \cdot 2m$. To summarize, we consider points $p(i, j, t)$ defined as $p(i, j, t) = (i + jt) \cdot 2m$, for $0 \leq i < m$, $0 \leq j < 2c$, and $t \geq 0$.

Whenever $t = k + 1/m$, for some integer $k < m$, $p(i, j, t) = (i+jk+j/m)\cdot 2m = 2(i+jk)m+2j$. For any two distinct points (i, j) and (i', j'), if $i+jk \neq i'+j'k$, then $|p(i, j, t)-p(i', j', t)| > 2m - 4c \geq 2$; if $i + jk = i' + j'k$, since (i, j) and (i', j') are distinct, $j' \neq j$ and $|p(i, j, t) - p(i', j', t)| \geq 2$. Thus, at time t, no two points are within distance 1. In other words, any covering has to have n centers (Figure 6).

On the other hand, at time $t = k$ for an integer $k < m$, since $p(i, j, k) = (i + jk) \cdot 2m$ where $0 \leq i < m$, $0 \leq j < 2c$, and $k < m$, each point has position $2sm$ for some $0 \leq s < m + 2ck$. That is, at $t = k$, the minimum covering has at most $m + 2ck$ centers (Figure 6). Thus, a c-approximate cover may have at most $c(m + 2ck)$ centers. Therefore, between times k and $k + 1/m$, there are at least $n - c(m + 2ck) = n/2 - 2c^2k$ changes to any c-approximate covering. In total, for all $0 \leq t < K$, the number of changes is at least $\sum_{0 \leq k < K}(n/2 - 2c^2k) > Kn/2 - c^2K^2$. Setting $K = \frac{n}{4c^2} < m$, we have established that the total number of changes is $\Omega(n^2/c^2)$. $\square$

LEMMA 5.3. *The number of events in our basic algorithm is* $O(n^2)$.

PROOF. An event is the failure of an ordering certificate in an x- or y-sorted list of square side coordinates or point coordinates. Since the points have bounded-degree algebraic motion, each pair of points can cause $O(1)$ certificate failures. $\square$

THEOREM 5.4. *The number of events processed by our hierarchical KDS is at most $O(n^2 \log\log n)$, and hence the KDS is efficient.*

PROOF. We maintain x- and y-ordering certificates on each of $\log\log n$ levels. As in Lemma 5.3, each pair of points can cause $O(1)$ certificate failures on each level. In addition, in the hierarchical KDS, we need to consider the events caused by maintaining the range search tree. Those events can happen when two points swap their x- or y-ordering. Such an exchange requires possible updates of the range trees on all levels where the exchanging pair is present. Again, there are $O(n^2)$ exchange events at each level. $\square$

Since the clustering stays fixed between the roughly quadratically many kinetic events, the high-probability result of Section 4 shows that, with high probability, the current clustering will be a constant-factor approximation of the optimum at every instant, throughout the entire motion history.

LEMMA 5.5. *For any bounded degree algebraic motion, the algorithm maintains an $O(1)$-approximate covering through the entire motion, with high probability.*

We now proceed to examine the cost of processing the kinetic events.

THEOREM 5.6. *The expected update cost for one event is $O(\log^{3.6} n)$. Hence the KDS is responsive in an expected sense.*

PROOF. When two points exchange in x- or y-order, only the relevant range search trees need to be updated. We need $O(\log^2 n)$ time to update each of $\log\log n$ range trees.

When two points p_i, p_j start/stop being mutually visible at any level of the hierarchy, we can update the centers involved with p_i, p_j in $O(\log^2 n)$ time, since we may need to search for a replacement center in the range tree. One new center may appear and one old center may disappear; these changes bubble up the hierarchy.

On hierarchy levels above the bottom, we divide the changes into two kinds, those caused by the motion of the points in that level and those caused by insertion or deletion of points bubbled up from lower levels. The number of changes of the first kind per event is a constant.

Let us consider the insertion of point p. The only points that may change their centers are those in p's visible range S. We divide S into four quadrants S_i, each with k_i $(i = 1, 2, 3, 4)$ points. If there is some point in S_i that nominates p to be its center, the index of p must be bigger than the indices of all the k_i points. The probability of this occurring is $\frac{1}{k_i+1}$. Therefore, the expected number of point-center changes caused by the appearance of p is at most

$$\frac{k_1}{k_1+1} + \frac{k_2}{k_2+1} + \frac{k_3}{k_3+1} + \frac{k_4}{k_4+1} + 1 \leq 5.$$

Assuming that p becomes a center, how many centers does it replace? For a given quadrant S_i, suppose the number of centers its points nominate is $m_i \leq k_i$. At most one of these centers is inside S_i. If m' points are outside S_i, the probability that p replaces j of them is at most $1/(m'+1)$. Hence the expected number of centers replaced in a single quadrant is upper bounded by either

$$\frac{1}{k_i+1}\left(1 + \frac{1 + \cdots + m_i - 1}{m_i}\right) = \frac{m_i + 1}{2(k_i + 1)} \leq \frac{1}{2},$$

if one of the centers is inside S_i, or by

$$\frac{1}{k_i+1}\left(\frac{1 + \cdots + m_i}{m_i + 1}\right) = \frac{m_i}{2(k_i + 1)} \leq \frac{1}{2}$$

if none of the centers is inside S_i. Each replaced center may stop being a center at this level of the hierarchy, if it is nominated by no points outside S. Thus the expected number of centers created/destroyed in this level (inserted/deleted at higher levels) due to the appearance of p is at most $4 \times \frac{1}{2} + 1 = 3$.

We can make a similar argument for the disappearance of a point. So the expected total number of point-center changes at all levels of the hierarchy per event is at most

$$5 \times (3^{\log\log n} + 3^{\log\log n - 1} + \cdots + 1)$$

which is $O(3^{\log\log n}) = O(\log^{1.6} n)$.

Since insertion or deletion in a range search tree costs $O(\log^2 n)$, the total expected update cost is $O(\log^{3.6} n)$. $\square$

THEOREM 5.7. *The kinetic data structure uses $O(n \log n \log\log n)$ storage, and hence it is compact.*

PROOF. Range trees take $O(n \log n)$ space per level. All other data structures use less space. $\square$

THEOREM 5.8. *Each point participates in at most $O(\log\log n)$ ordering certificates; therefore, the KDS is local.*

PROOF. Each point participates in at most $O(1)$ ordering certificates in each level. $\square$

5.3 Distributed implementation

The hierarchical algorithm can also be implemented in a distributed manner, making it appropriate for a mobile networking scenario. Each node broadcasts a "who is there" message and waits for replies. Each point that hears the request responds. The hierarchy can be implemented by having nodes broadcast with different power for each level or by other local positioning mechanisms. We emphasize that no global positioning information is needed. Therefore, each point keeps track of its neighborhood within different size ranges. This information is sufficient for each node to select a center for each level. Each node needs to sense or be informed when a neighbor enters or leaves any of its $\log \log n$ ranges. When such an event happens, each node involved checks whether it needs to update its center. When it nominates a center that is not nominated by any other node, the center will also be added to a higher level and may cause updates in that level. If a node ceases to be pointed to by any node, then it also has to be deleted from higher levels in the hierarchy. Clearly, all of these operations can be done locally without centralized control. The total storage needed is $O(sn)$, where s is the maximum number of nodes inside a node's range. In the worst case, this can be $\Theta(n^2)$, but in practice, s is often small.

Furthermore, we can restrict the storage of each node to be n^ϵ, where $0 < \epsilon < 1$, and still get a constant approximation. If we let each node keep up to $O(n^\epsilon)$ neighbors and select a center among them, then we have the following:

LEMMA 5.9. *In 2-dimensional space, the number of centers nominated inside a unit-size square S by the space-restricted one-level algorithm is $O(n^{\max(1-\epsilon, \frac{1}{2})})$ in expectation.*

PROOF. We use the same notation as Lemma 3.4. Suppose L is the visible range of S. L can be divided into 9 sub-squares of size $\frac{1}{2}$. Consider any such sub-square S', and suppose $m = |L|$, $x = |S|$, $y = |S'|$. If $y <= n^\epsilon$, from Lemma 3.4, the expected number of centers in S that are nominated by nodes in S' is bounded by $\sqrt{m} \leq \sqrt{n}$. Otherwise, a point p in S can be nominated by a point q in S' only if q finds p the largest index in its n^ϵ neighbors. (Note that q has at least $y > n^\epsilon$ neighbors.) So the probability that p can be nominated is at most $\frac{1}{n^\epsilon}$. The expected number of centers in S nominated by points in S' is no more than $x/n^\epsilon \leq n^{1-\epsilon}$. $\square$

Let c_ϵ denote $\frac{-1}{\log(\max(1-\epsilon, \frac{1}{2}))}$.

THEOREM 5.10. *The expected number of centers generated by the space-restricted hierarchical algorithm is a constant-factor approximation to the optimum, if we have $c_\epsilon \log \log n - 1$ rounds, and in each round we apply the basic space-restricted algorithm with square size $\delta_i = 2^i/(\log n)^{c_\epsilon}$.*

PROOF. The proof is the same as that of Theorem 4.4. $\square$

6. SUMMARY AND FUTURE WORK

Our randomized hierarchical algorithm can easily be extended to higher dimensions. Most of the analysis for the 2-D case works for any dimension d, except that the constant approximation factor depends exponentially on d. Our algorithms also support efficient insertion or deletion of nodes.

This work also raises several open problems. In the standard KDS setting, where we have nearly linear space, our center updating algorithm exploits the fact that the ranges are aligned congruent squares. Can we find a similar algorithm in a standard KDS setting with congruent disk ranges instead? Finally, though we have a high probability result, our algorithm is still randomized and it would be interesting to find a deterministic algorithm for the mobile centers problem.

We note that our algorithm clusters based solely on the positions of the mobile nodes. It would be interesting to develop clustering strategies that utilize additional information about the node motions, say both position and velocity. Such clusterings may be far more stable under motion, although they may require more clusters. In fact, a trade-off between the quality and stability of a clustering needs to be investigated. Besides clustering, numerous other problems for ad-hoc networks can be studied in the same style as the clustering problem, including network connectivity, route maintenance, node misbehavior detection, etc. We should also note that the constant approximation ratio in our analysis is quite large. That is partly because our analysis applies in the worst case. In practice, we would expect much better performance.

We believe that kinetic clustering is a fundamental problem for the organization of mobile devices and deserves further study. Motion models and quality measures for different application areas need to be developed further. We expect that the ideas presented will find applications in other areas, such as temporal data-bases, molecular modelling, and the large-scale tracking of people or vehicles.

Acknowledgements

The authors wish to thank Michael Segal and Samir Khuller for useful discussions. The work of J. Gao, L. Guibas, and A. Zhu was supported in part by NSF grants CCR-9623851 and CCR-9910633, US Army MURI grant DAAH04-96-1-0007 and AASERT grant DAAG55-97-0218, and a grant from the Stanford Networking Research Center. The work of A. Zhu was also supported in part by the Lucent & Bell-labs GRPW fellowship.

7. REFERENCES

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover, 1972.

[2] P. K. Agarwal, J. Basch, M. de Berg, L. J. Guibas, and J. Hershberger. Lower bounds for kinetic planar subdivisions. In *Proc. 15th ACM Symp. on Computational Geometry*, pages 247–254, 1999.

[3] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh. Max-Min D-cluster formation in wireless ad hoc networks. In *19th IEEE INFOCOM*, March 1999.

[4] 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.

[5] J. Basch, L. Guibas, and J. Hershberger. Data structures for mobile data. *J. Alg.*, 31(1):1–28, 1999.

[6] J. Basch, L. J. Guibas, and L. Zhang. Proximity problems on moving points. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 344–351, 1997.

[7] S. Bespamyatnikh, B. Bhattacharya, D. Kirkpatrick, and M. Segal. Mobile facility location. In *4th International Workshop on Discrete Algorithms and Methods for Mobile Computing & Communications*, 2000.

[8] C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings of IEEE SICON'97*, pages 197–211, April 1997.

[9] U. Feige. A threshold of $\ln n$ for approximating set cover. In *ACM Symp. on Theory of Computing*, 1996.

[10] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inform. Process. Lett.*, 12(3):133–137, 1981.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[12] M. Gerla and J. Tsai. Multicluster, mobile, multimedia radio network. *ACM-Baltzer Journal of Wireless Networks*, 1(3), 1995.

[13] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, Reading, MA, 1989.

[14] S. Guha and S. Khuller. Approximation algorithms for connected dominating set. *Algorithmica*, 20:374–387, 1998.

[15] L. J. Guibas. Kinetic data structures — a state of the art report. In P. K. Agarwal, L. E. Kavraki, and M. Mason, editors, *Proc. Workshop Algorithmic Found. Robot.*, pages 191–209. A. K. Peters, Wellesley, MA, 1998.

[16] J. Haartsen, M. Naghshineh, J. Inouye, O. Joeressen, and W. Allen. Bluetooth: Vision, goals, and architecture. *Mobile Computing and Communications Review*, 2(4):38–45, Oct 1998.

[17] D. S. Hochbaum and W. Maas. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32:130–136, 1985.

[18] H. B. Hunt, H. Marathe, V. Radhakrishnan, S. Ravi, D. Rosenkrantz, and R. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *Journal of Algorithms*, 26(2), 1998.

[19] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. 6th Annual International Conference on Mobile Computing and Networking*, pages 255–265, 2000.

[20] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[21] J. Sharony. An architecture for mobile radio networks with dynamically changing topology using virtual subnets. *ACM-Baltzer Mobile Networks and Applications Journal*, 1(1), 1996.

[22] J. H. van Lint and R. M. Wilson. *A Course in Combinatorics*. Cambridge Press, 1992.

[23] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing & Communications*, 1999.