Approximation Algorithms for the Mobile Piercing Set Problem with Applications to Clustering in Ad-hoc Networks

Hai Huang* Andréa W. Richa* Michael Segal†

Abstract

The main contributions of this paper are two-fold. First, we present a simple, general framework for obtaining efficient constant-factor approximation algorithms for the mobile piercing set (MPS) problem on unit-disks for standard metrics in fixed dimension vector spaces. More specifically, we provide low constant approximations for L_1- and L_∞-norms on a d-dimensional space, for any fixed $d > 0$, and for the L_2-norm on 2- and 3-dimensional spaces. Our framework provides a family of fully-distributed and decentralized algorithms, which adapts (asymptotically) optimally to the mobility of disks, at the expense of a low degradation on the best known approximation factors of the respective centralized algorithms: Our algorithms take $O(1)$ time to update the piercing set maintained, per "relevant" movement of a disk. We also present a family of fully-distributed algorithms for the MPS problem which either match or improve the best known approximation bounds of centralized algorithms for the respective norms and dimensions.

Second, we show how the proposed algorithms can be directly applied to provide theoretical performance analyses for two popular 1-hop clustering algorithms in ad-hoc networks: the lowest-id algorithm and the Least Cluster Change (LCC) algorithm. More specifically, we formally prove that the LCC algorithm adapts in constant time to the mobility of the network nodes, and minimizes (up to low constant factors) the number of 1-hop clusters maintained; we propose an alternative algorithm to the lowest-id algorithm which achieves a better approximation factor without increasing the cost of adapting to changes in the network topology. While there is a vast literature on simulation results for the LCC and the lowest-id algorithms, these had not been formally analysed prior to this work.

We also present an $O(\log n)$-approximation algorithm for the mobile piercing set problem for nonuniform disks (i.e., disks that may have different radii), with constant update time.

*Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287-5406, USA. This work was supported in part by NSF CAREER Award CCR–9985284. {hai, aricha}@asu.edu

†Communication Systems Engineering Department, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel. segal@cs.bgu.ac.il

1 Introduction

The *mobile piercing set (MPS)* problem is a variation of the (classical) piercing set problem that arises in dynamic distributed scenarios. In this paper, we present a family of low constant-factor approximation algorithms for the *mobile piercing set problem on unit-disks* for standard metrics in fixed-dimension vector spaces. The algorithms presented are fully distributed and decentralized. More specifically, we present fully distributed constant-factor approximation algorithms for the L_1- and L_∞ norms on $\Re^d$, for any fixed d, and for the L_2 norm in $\Re^2$ and $\Re^3$, which either match or improve the best known constant approximation factors of a centralized algorithm, or have (asymptotically) optimal update costs (we will formally define those costs later). All algorithms presented find an initial approximate piercing set in asymptotically optimal time.

In particular, the MPS problem has applications in ad-hoc networks, as for example in 1-hop clustering for these networks. In this paper we show how the proposed algorithms for the MPS problem can be directly applied to provide formal performance analyses for two well-known 1-hop clustering algorithms, which are popular in the applied ad-hoc network community: the lowest-id algorithm and the Least Cluster Change (LCC) algorithm. While simulation results for these algorithms and its variants exist in the literature, no attempt in providing a formal analysis of these algorithms had been previously done.

First, we present some definitions. A *disk D of radius r with center q* in $\Re^d$ with respect to L_p[1] is given by the set of points $D = \{z \in \Re^d \ : \ ||z - q||_p \leq r\}$. Let $q(D)$ denote the center of a disk D. A *piercing set* of a given collection of disks $\mathcal{D}$ is a set of points P such that for every disk $D \in \mathcal{D}$, there exists a point $p \in P$ such that $p \in D$ — i.e., P *pierces* every disk $D \in \mathcal{D}$. The *(classical) k-piercing set problem* seeks to find whether a piercing set P of cardinality k of $\mathcal{D}$ exists, and if so, produces it. If the value of k is minimal over all possible cardinalities of piercing sets of $\mathcal{D}$ then the set P is called a *minimum piercing set* of $\mathcal{D}$. The *minimum piercing set problem* asks for the minimum piercing set of a given collection $\mathcal{D}$.

We consider a dynamic variation of the classical piercing set problem, which arises in mobile and distributed scenarios, where disks are moving in space. In the *mobile piercing set (MPS) problem*, we would like to maintain a dynamic piercing set P of a collection of mobile disks $\mathcal{D}$ such that, at any time t, P is a minimum piercing set of the current configuration of the disks. In other words, P must adapt to the mobility of the disks. Moreover, we would like to be able to devise a distributed algorithm to solve this problem, where the individual disks can decide in a distributed fashion (with no centralized control) where to place the piercing points.

In this scenario, we assume that the disks are able to detect whether they intersect other disks. We can think about a disk as being the communication range of a given mobile device (node), which resides at the center of the disk: A disk can communicate with all of its adjacent nodes by a *broadcast* operation within $O(1)$ time. Below, we will present applications of the mobile piercing set problem in mobile networks.

In this paper, we focus on the case when the disks are all of the same radius r — or equivalently, of same *diameter* $2r$. Hence, without loss of generality, in the remainder of this paper, unless stated otherwise, we assume that $2r = 1$, and therefore that we have a collection of unit-diameter disks, or *unit-disks* for short. In Section 5, we address an extension of our algorithms to the nonuniform case, where the disks may not all have the same radius.

In recent years, the technological advances in wireless communications have led to the realization of ad-hoc mobile wireless networks, which are self-organizing and which do not rely on any sort of stationary backbone structure. These networks are expected to significantly grow in size and usage in the next few years. For scalability, specially in order to be able to handle updates due to the constant changes in network topology, clustering becomes mandatory.

As hinted above, mobile unit-disks can be used to model an ad-hoc network where all mobile wireless nodes have the same range of communication. Each mobile node's communication range is represented by a disk in $\Re^2$ (or $\Re^3$) centered at the node with radius equal to 1; a mobile node A can communicate with mobile node B if and only if B is within A's communication range. The ad-hoc network scenario is a direct application scenario for the unit-disk MPS problem, since an ad-hoc network is fully decentralized and any algorithm running on such a network must adapt to mobility in an efficient way.

If all disks are of the same size, then the k-piercing set problem is equivalent to the decision version of a well-known problem: the geometric k-center problem [2]. The k-center problem under L_p metric is defined as follows: Given a set S of n demand points in $\Re^d$, find a set P of k supply points so that the maximum L_p distance between a demand point and its nearest supply point in P is minimized. The corresponding decision problem is to determine, for a given radius r, whether S can be covered by the union of k L_p-disks of radius r, or in other words, to determine

[1] The L_p norm, for any fixed p, of a vector $z = (z_1, z_2, \cdots, z_d)$ in $\Re^d$ is given by $||z||_p = (|z_1|^p + |z_2|^p + \cdots + |z_d|^p)^{\frac{1}{p}}$; if $p = \infty$, then $||z||_\infty = \max(|z_1|, |z_2|, \cdots, |z_d|)$.

whether there exists a set of k points that pierces the set of n L_p-disks of radius r centered at the points of S. In some applications, P is required to be a subset of S, in which case the problem is referred to as the *discrete k-center* problem. When we choose the L_2 metric, the problem is called the Euclidean k-center problem, while for L_∞ metric the problem is called the rectilinear k-center problem. Since the Euclidean and rectilinear k-center problems in $\Re^2$ are NP-complete (see e.g. [26, 29]) when k is part of the input, the planar unit-disk k-piercing set problem in $\Re^2$ under L_1, L_2 or L_∞ norms is also NP-complete. Unfortunately, an approximation algorithm for the k-center problem does not translate directly into an approximation algorithm for the unit-disk piercing set problem (and vice-versa), since an algorithm for the former problem will give an approximation on the radius of the covering disks, while for the latter problem we need an approximation on the number of piercing points. However, the two approximation factors are highly related [2].

The remainder of this paper is organized as follows. In Section 1.1, we state our main contributions in this work. In Section 2, we discuss more related work in the literature. Section 3 proves some geometric properties of the piercing set problem. We use the results in Section 3 to develop the approximation algorithms presented in Sections 4 and 5: The algorithm introduced in Section 4 leads to lower approximation factors, for the norms and dimensions considered, while the one in Section 5 adapts optimally to the movement of disks. In Section 6, we relate the algorithms presented for the MPS problem to clustering in ad-hoc networks. Finally, we present some future work directions in Section 7.

1.1 Our results

In this paper we propose fully distributed (decentralized) approximation algorithms for the unit-disk MPS problem in $\Re^d$ under L_p norm, for any fixed d and p. All the approximation factors presented in this paper are with respect to the *number of points* in a minimum piercing set.

For each algorithm, we are interested in computing the cost associated with building an initial approximate piercing set for the given initial configuration of the collection of disks — which we call the *setup cost* of the algorithm — and the cost associated with updating the current piercing set due to the movement of a disk — which we call the *update cost* of the algorithm. Actually we charge the update costs per *event*, as we explain below. We assume that all the costs that do not involve communication between disks are negligible when compared to the cost of a disk communicating with its neighbors (through a broadcast operation). Therefore we will only consider communication costs when evaluating the algorithms considered.

In order to maintain an optimal or approximate piercing set of the disks, there are two situations which mandate an update on the maintained piercing set. The first situation is when the movement of a disk D results in having at least one disk D' of $\mathcal{D}$ unpierced (note that D' may be D itself). The second situation is when some piercing points in the set maintained become "redundant", and we may need to remove them from the set (When removing these "redundant" points, we may fall into the former situation, as we will see in Section 5.). Thus, we say that an *(update) event* is triggered (or happened) whenever one of the two situations just described occurs.

The main contributions of this paper are two-fold. First, we present a family of constant-factor approximation algorithms — represented by the M-algorithm — for the unit-disk MPS problem with *(asymptotically) optimal setup* and *update costs*, for all the norms and space dimensions considered. Moreover, we achieve this *without a significant increase* in the approximation factor of the corresponding best known approximation algorithms for the classical piercing set problem. Let P^* be a minimum piercing set. Note that any dynamic algorithm that approximates the minimum piercing set of a collection of mobile disks has setup cost at least $O(|P^*|)$, and update cost at least $O(1)$. More specifically, in d-dimension, we devise a 2^d-approximation algorithm under L_1 or L_∞. For L_2 norm, we devise a 7-approximation algorithm in $\Re^2$, and a 21-approximation algorithm in $\Re^3$. All these algorithms have $O(|P^*|)$ setup cost and $O(1)$ update cost. These algorithms are the first constant-approximation algorithms for the MPS problem, with asymptotically optimal setup and update costs.

We also present a second family of fully distributed algorithms — represented by the A-algorithm — for L_1 or L_∞ norms in any space $\Re^d$, and for L_2 norm in $\Re^2$ and $\Re^3$. These algorithms achieve the same, or better, constant approximation factors as the best known centralized algorithms for the corresponding norm and dimension, but have a poorer update cost of $O(|P^*|)$. These algorithms are, to the best of our knowledge, the first fully distributed (decentralized) approximation algorithms which achieve the same approximation factors as their centralized counterparts. These algorithms are of interest since, for example, they provide an alternative algorithm to the lowest-id clustering algorithm in ad-hoc networks, which would achieve a 4- (resp. 11-) approximation factor in $\Re^2$ (resp., $\Re^3$) without an increase on the setup and update costs. We summarize the results presented for 1-, 2-, and 3-dimension in Table 1 and Table 2.[2]

The simple framework presented for the M-algorithm which can handle mobility efficiently in a dynamic scenario is an important contribution of this work on its own. It avoids the use of involved data structures, which in general cannot avoid the use of some sort of "centralization" (even if implicitly). In order to be able to apply the given framework to a particular norm and dimension, one needs only to be able to compute a set of piercing points which are guaranteed to pierce the immediate neighborhood of any disk D: The number of such points will be used in bounding the approximation factor of the algorithms proposed.

The second main contribution of this work is the application of the algorithms developed for the MPS problem to the problem of finding an efficient self-organizing 1-hop underlying clustering structure for (wireless and mobile) ad-hoc networks. In fact, one can use the algorithms developed for the MPS problem to derive the *first theoretical performance analyses* of the popular *Least Cluster Change (LCC)* algorithm proposed by Chiang et al. [7], and of the *lowest-id* algorithm (discussed by Gerla and Tsai in [15]), both in terms of the number of 1-hop clusters maintained and in terms of update and setup costs, thus providing a deeper understanding of these two algorithms and validating the existing simulation results for the same, as seen in Section 6. No previous formal analysis of either algorithm exists in the literature. Namely, we show that the LCC algorithm has the same approximation factor, setup and update costs as the M-algorithm for L_2 in $\Re^2$ (or $\Re^3$), and that the lowest-id algorithm also maintains the same approximation factor as the M-algorithm, while incurring higher update costs. We propose an alternative algorithm that achieves same setup and update costs as the lowest-id algorithm, but that has better approximation factors.

Another contribution of our work addresses the MPS problem on nonuniform disks scenario, where the disks may no longer be all of the same radius. Without loss of generality, we assume that the smallest radius of a disk is equal to 1. Then, if the maximum radius of a disk is bounded by a polynomial on $n = |\mathcal{D}|$, we present an $O(\log n)$-approximation algorithm for this problem, with constant distributed update cost.

2 Related Work

The k-center and k-piercing problems have been extensively studied. In d-dimension, a brute-force approach leads to an exact algorithm for the k-center problem with running time $O(n^{dk+2})$. For the planar case of the Euclidean k-center problem Hwang et al. [24] gave an $n^{O(\sqrt{k})}$ time algorithm improving Drezner [9] solution which runs in time $O(n^{2k+1})$. An algorithm with the same runtime was presented in Hwang et al. [23] for the planar discrete Euclidean k-center problem. Recently, Agarwal and Procopiuc [1] extended and simplified the technique by Hwang et al. [24] to obtain an $n^{O(k^{1-1/d})}$ time algorithm for computing the Euclidean k-center problem.

Sharir and Welzl [32] explain a reduction from the rectilinear k-center problem to the k-piercing problem (under L_∞ metric), using a sorted matrices searching technique developed by Frederickson and Johnson [13]. Ko et al. [26] proved the hardness of the planar version of the rectilinear k-center and presented a $O(n \log n)$ time 2-approximation (on the covering radii) algorithm. (In fact, Ko et al. [26] proved that, unless $P = NP$, the best approximation factor that can be achieved in polynomial time for the rectilinear k-center problem is 2.) Several approximation results (on the radii of the disks) have been obtained in [11, 17, 20, 21]. For more results on the k-center problem, please refer to [2].

Regarding the k-piercing set problem in $\Re^d$, Fowler et al. [12] proved the NP-completeness of finding the minimal value of k for a given set of n disks. Hochbaum and Maas [19] gave an $O(l^d n^{2l^d+1})$ polynomial time algorithm with approximation factor $(1 + \frac{1}{l})^d$ for any fixed integer $l \geq 1$. Thus, for $l = 1$, their algorithm yields an $O(n^3)$ time (sequential) algorithm with performance ratio 2^d. For 1-dimension case, Katz et al. [25] presented an algorithm that maintains the exact piercing set of points for a collection of n intervals in $O(|P^*| \log n)$ time, where P^* is the minimum piercing set. Their solution can be adapted to obtain an algorithm with distributed running time $O(|P^*|)$ for computing a minimum piercing set of n intervals. Nielsen [30] proposed a 2^{d-1}-approximation algorithm that works in d-dimensional space and under L_∞ metric in $O(dn + n \log c)$ time, where c is the size of the piercing set found. This algorithm is based on the divide-and-conquer paradigm.

Although not stated explicitly, the approximation on the radius of the k-center problem in [1] implies a 4-approximation algorithm for the minimum piercing set problem for $\Re^2$ and L_2. Efrat et al. [10] introduced a dynamic data structure based on segment trees which can be used for the piercing set problem. They presented a sequential algorithm which gives a constant factor approximation for the minimum piercing set problem for "fat" objects with polynomial setup and update time. See [10] for the definition of "fatness" and more details.

[2]All the results are for unit-disks; $L_p = L_\infty$ for any p in 1-dimension.

A large number of clustering algorithms have been proposed and evaluated through simulations in the ad-hoc network domain, as for example the work in [15, 27, 31, 28, 3, 4]. Gerla and Tsai in [15] considered two distributed clustering algorithms, the *lowest-id* algorithm and the *highest-degree* algorithm, which select respectively the lowest-id mobile or the highest degree in a 1-hop neighborhood mobile as the clusterhead. A more general weight oriented clustering algorithm was introduced by Basagni [4], where 1-hop clusters are formed according to a predefined weight. In [27], Lin and Gerla described a non-overlapping clustering algorithm where clusters are able to be dynamically reconfigured.

The LCC algorithm proposed by Chiang et al. [7] aims to maintain a clustering for a mobile network with least changes of clusters, where clusters will be broken and re-clustered only when necessary. In fact our algorithm when translated to a clustering algorithm in the ad-hoc scenario is basically the LCC algorithm (when used in conjunction with the lowest-id algorithm), as discussed in Section 6.

Recently, researchers have investigated using geometric centers as clusterheads in order to minimize the maximum communication cost between a clusterhead and the cluster members. Bepamyatnikh et al. [6] discussed how to compute and maintain the 1-center and the 1-median for a given set of n moving points on the plane (1-median is defined as a point which minimizes the sum of all distances to the input points), which gives an algorithm to select clusterheads if mobiles are already partitioned into clusters. Gao et al. [14] proposed a randomized algorithm for maintaining a set of clusters among moving points on the plane. They [14] presented algorithms with expected approximation factor (on the optimal number of centers) of $c \log n$ for intervals and of $c\sqrt{n} \log n$ for squares[3]. The probability that there are more than the expected number of centers is $O(1/n^{\Theta(c^2)})$ for the case of intervals and $O(1/n^{c^2 \log n})$ for squares, for a constant c. An extension of this basic algorithm led to a hierarchical algorithm, also presented in [14], based on kinetic data structures [5] with expected constant approximation factor on the number of discrete centers, where the approximation factor also depends linearly on the constant c. The dependency of the approximation factor and the probability that the algorithm chooses more than a constant times the optimal number of centers is similar to that of the non-hierarchical algorithm for the squares case. The constant c, which has not been explicitly determined in [14], can be shown to be very large (certainly more than an order of magnitude larger than the corresponding approximation constant presented in this paper), even if we allow the probability of deviating from the constant approximation on the number of centers (which depends linearly on c) not to be close to 1. Their algorithm has an expected update time of $O(\log^{3.6} n)$ (when compared to constant update cost in our algorithm), the number of levels used in the hierarchy is $O(\log \log n)$, with $O(n \log n \log \log n)$ total space.

Har-Peled [18] found a scheme of determining centers in advance, i.e. if in the optimal solution the number of centers is k and r is the optimal radius for the points moving with degree of motion l, then his scheme guarantees a 2^{l+1}-approximation (of the radius) with k^{l+1} centers chosen from the set of input points before the points start to move.

3 Geometry for the Piercing Set Problem

In this section, we prove some geometric properties of the piercing set problem. To be more specific, we solve the piercing problem on the *neighborhood* of a disk, which will provide the basic building block for our approximation algorithms presented in the following sections. The main step of the approximation algorithms is to select an unpierced unit-disk and pierce all of its *neighbors* (defined below). By repeating this procedure, we will eventually pierce all unit-disks and form a piercing set. The approximation factors are determined by the number of piercing points chosen for each selected unit-disk.

If two disks D and D' intersect, we say that D is a *neighbor* of D'. The *neighborhood* of a disk D, denoted by $\mathcal{N}(D)$, is defined as the collection of all disks that intersect D, $\mathcal{N}(D) = \{D' : D \cap D' \neq \emptyset, D' \in \mathcal{D}\}$. Note that $D \in \mathcal{N}(D)$. We are interested in the minimum number of points that are enough for piercing all disks in the neighborhood of a given disk. However, this number may vary, depending on the distribution of the disks in the particular neighborhood in consideration. Thus, we compute the minimum number (along with the fixed positions) of points needed to pierce any possible neighborhood of a disk. This number is called the *neighborhood piercing number*. The neighborhood piercing number is tight in the sense that for any set of points with smaller cardinality, we can find some configuration of the neighborhood of a disk which has an unpierced disk. The corresponding piercing points are called the *neighborhood piercing points*. Clearly, the piercing number is a function of both dimension d and norm index p. Hence, we denote the neighborhood piercing number for dimension d and norm index p as $N(d,p)$, and we use $PN(D,d,p)$ to denote the corresponding set of neighborhood piercing points of a

[3] Disks in 1D correspond to intervals on the line; in 2D, Disks under L_∞ or L_1 are called squares.

unit-disk D. [4] We prove in this section that $N(d,1)=N(d,\infty)=2^d$ for all $d\geq 1$, and that $N(2,2)=7$. We also place an upper bound of 21 on $N(3,2)$.

First we reduce the above problem into an equivalent disk covering problem. Let $\mathcal{D}$ be a collection of unit-disks and P be a set of points. Let P' be the set of centers of all disks in $\mathcal{D}$, and $\mathcal{D}'$ be a collection of unit-disks centered at points in P. Then P pierces $\mathcal{D}$ if and only if $\mathcal{D}'$ covers P'. Moreover, P is a minimum piercing set for $\mathcal{D}$ if and only if $\mathcal{D}'$ is a minimum set of disks (with respect to cardinality) that covers P'. We define the *unit-disk covering problem* to be the problem of finding the minimum k such that there are k unit-disks whose union covers a given point set.

We reduce the problem of finding a neighborhood piercing number to a unit-disk covering problem as follows. Hence for a unit-disk D, all the centers of unit-disks in $\mathcal{N}(D)$ are located in the region $G=G(D)=\{z : ||z-q(D)||_p\leq 1\}$, where $q(D)$ is the center of D. Conversely, a disk centered at any point in G must intersect D. Therefore, we seek for the minimum number of unit-disks that cover region G. The centers of those disks serve as the set of neighborhood piercing points $PN(D)$. The tightness of N can be seen from the fact that in the disk covering problem, we cannot cover the entire region G with less than N disks.

Note that the region G is a disk of radius 1, and all the disks we use to cover it are unit-disks (i.e., of radius $\frac{1}{2}$). For any L_p-norm, in a d-dimensional space, the ratio of the area of G to the area of a unit-disk is 2^d. Thus, we need at least 2^d disks to cover G — i.e., $N\geq 2^d$ for any dimension $d\geq 1$ and any norm L_p. The lower bound of 2^d is in fact tight for $p=1$ or $p=\infty$, since in any dimension d, a unit-disk has 2^d "corners" and all unit-disks centered at those "corners" cover the region G.

The case $p=2$ is more involved since we cannot pack "spheres" as tightly as "hypercubes" without leaving uncovered points in the region G, if no intersection of disks is allowed. Figure 1-(a) suggests a 7-disk covering with disks centered at q,r,s,t,u,v and w, for the region G in a 2-dimensional space. Since we need at least six disks to cover the boundary of region G, which is a circle of perimeter 2π, those seven disks form a minimum cardinality covering. Figure 1-(a) shows $PN(D,2,2)$, where $q=(x,y)$ is the center of the unit-disk D. The coordinates of the six other points are $(x\pm\frac{3}{4}, y\pm\frac{\sqrt{3}}{4}), (x, y\pm\frac{\sqrt{3}}{2})$. This completes the proof of Lemma 1 (a more formal proof of this lemma can be found in [22]).

Lemma 1 *The neighborhood piercing number is equal to 2^d for d-dimensional space under L_1- and L_∞-norm. The neighborhood piercing number for 2-dimension and L_2 is equal to 7.*

For L_2-norm in $\Re^3$, we were only able to place an upper bound on the number of unit-disks needed to cover a disk of diameter 2, hence placing an upper bound on $N(3,2)$. A simple argument suffices to verify that the 20 unit-disks centered at the points listed in Table 4 plus a unit-disk D centered at the origin cover a disk G of diameter two also centered at the origin. Hence we have $N(3,2)\leq 21$ [22]. It remains an open problem to compute the exact value of $N(3,2)$. The neighborhood piercing number for L_2 is closely related to the *sphere packing* and *sphere covering* problems described in [8].

When compared to the known results in the literature, the approximation factors based on the neighborhood piercing points are not the best known. For example, we have shown that $N(1,\cdot)=2$, which leads to a 2-approximation algorithm for piercing unit-intervals on the line (see Section 5). In [16, p. 193] (see also [25]), an exact solution (i.e., 1-approximation) for piercing unit-intervals is proposed. The idea there, shown in Figure 2, is to start from the rightmost interval D, where only one endpoint of D — the left endpoint l — is enough for piercing all neighbors of D (since D has no neighbor to its right). In order to be able to extend and generalize this idea to other norms and higher dimensions, we need to define, the *halfspace of a disk D with orientation $\vec{n}$*, denoted by $\mathcal{H}_D(\vec{n})$: $\mathcal{H}_D(\vec{n})=\{z : (\vec{z}-q(\vec{D}))\cdot\vec{n}\geq 0\}$. For the 1-dimensional case, all of the centers of the neighboring disks of the rightmost interval D are located in the half space $\mathcal{H}_D(-1)$ (to the "left" of D), and only half of the neighborhood piercing points (i.e., only $N(1)/2$ points) are enough for piercing $\mathcal{N}(D)$. More generally, in any d-dimensional space, there exists an orientation $\vec{n}$, such that we need roughly half of the neighborhood piercing points to pierce all the neighbors of disk D located in $\mathcal{H}_D(\vec{n})$. The minimum number of piercing points needed for the halfspace $\mathcal{H}_D(\vec{n})$, over all possible orientations $\vec{n}$, is called the *halfspace neighborhood piercing number*, and is denoted by $\overline{N}$. The set of corresponding piercing points are called the *halfspace neighborhood piercing points of D* and are denoted by $\overline{PN}(D)$.

If $PN(D)$ is symmetric with respect to the center of the unit-disk D, then $\overline{N}=\lceil\frac{N}{2}\rceil$ if the center of D does not belong to PN, or $\overline{N}=\lceil\frac{N+1}{2}\rceil$ otherwise. Note that this is the case for $PN(d,1)$, $PN(d,\infty)$ and $PN(2,2)$.

[4] In general, we omit the parameters p, d, or D, whenever clear from the context.

The set of piercing points which correspond to the upper bound of 21 for $N(3,2)$ is not symmetric, but we can still find an orientation such that 11 points are enough to pierce the halfspace neighborhood of a disk with respect to the orientation. Figure 1-(b) illustrates the halfspace neighborhood piercing points — points q,r,s and t — for $\Re^2$ under L_2 norm. The orientation considered is $\vec{n} = (0,-1)$. Table 3 summarizes some values of neighborhood piercing number and that on halfspace for lower dimension d and norms L_1, L_∞ and L_2, where we denote the minimum of $\overline{N}(\vec{n})$ as $\overline{N}$ and the corresponding $\overline{PN}(\vec{n})$ as $\overline{PN}$. It follows from the upper bound on $N(3,2)$ that $\overline{N} \leq 11$ for L_2 and $\Re^3$. The corresponding halfspace neighborhood piercing points are also a subset of the points used for establishing the upper bound on $N(3,2)$. It also remains an open question to determine the exact value of $\overline{N}$ for L_2 and $\Re^3$.

For an orientation $\vec{n}$, if we order all unit-disks D in $\mathcal{D}$ according to the values $\vec{q}(D) \cdot \vec{n}$, then the unit-disk D bearing the smallest $\vec{q}(D) \cdot \vec{n}$ value satisfies the property that all its neighbors are located in the halfspace $\mathcal{H}_D(\vec{n})$. Thus, by carefully choosing the order in which we consider the neighborhoods of disks to be pierced, we can use the halfspace neighborhood piercing points as the basis of the fully-distributed algorithms for the MPS problem presented in Section 4, which match or improve the best known approximation factors of the respective centralized algorithms.

The problem of computing N for other L_p metrics is more involved and may not have many practical applications. A method to estimate an upper bound on N and compute the corresponding set of neighborhood piercing points for arbitrary L_p metrics is discussed in [22] for completeness.

4 Better Approximation Factors

In this section we present a family of constant-factor fully-distributed (decentralized) approximation algorithms for the piercing set problem, which at least match the best known approximation factors of centralized algorithms for the respectively norms and dimensions. This algorithm introduces some basic concepts which will be useful when developing the algorithms in Section 5. Also, the algorithm presented in this section directly translates into an alternative to the lowest-id clustering algorithm, discussed in Section 6, which achieves a better approximation factor. The algorithms in this section all follow a general algorithmic framework, which we call the *A-algorithm* (for having better *approximation* factors) in contrast with the slightly looser approximation factors of the other family of algorithms presented in Section 5 and represented by the *M-algorithm*, which can better handle *mobility*.

Consider a set of unit-disks in a d-dimensional space under norm L_p. As shown in Section 3, we need at most $\overline{N}$ piercing points to pierce the neighborhood of a unit-disk D bearing the smallest $\vec{q}(D) \cdot \vec{n}$ among the (unpierced) disks in its neighborhood, where $\vec{n}$ is an orientation that gives $\overline{N}$. We call such a disk D an *upmost disk*. Thus, at each step of the algorithm, each upmost unpierced disk D elects itself as a *piercing disk* and selects the points in $\overline{PN}(D)$ as piercing points. Since all the unpierced disks in $\mathcal{N}(D)$ are pierced by $\overline{PN}(D)$, we mark all the unpierced disks in $\mathcal{N}(D)$ as pierced, and repeat the procedure above. After repeating this step for at most $|P^*|$ times, all the unit-disks in $\mathcal{D}$ are pierced and a piercing set with cardinality at most $\overline{N}$ times $|P^*|$ is produced, as shown in Theorem 1. Provided that broadcasting has $O(1)$ cost, the running time of the distributed A-algorithm is $O(|P^*|)$ and no preprocessing is required. Theorem 1 states the main properties of the A-algorithm. This theorem actually extends the results in [25] and in [30] to a more general distributed scenario, and to the L_2-norm in 2- and 3-dimensional space.

We re-invoke the A-algorithm to maintain the piercing set every time an event (as defined in Section 1.1) happens. In a distributed scenario, this can be done by flooding a reset message to unpierce all disks. Thus the update cost of the A-algorithm is also $O(|P^*|)$.

Theorem 1 *The approximation factor of the distributed A-algorithm is $\overline{N}$, and its setup and update costs are both $O(|P^*|)$.*

Proof: For each piercing unit-disk D, we need at least one point in the minimum piercing set P^* to pierce D. For any two distinct piercing unit-disks D and E, the point in P^* that pierces D cannot pierce E since no two (distinct) piercing disks intersect. Thus we have at most $|P^*|$ piercing unit-disks. For each piercing unit-disk, we select $\overline{N}$ piercing points. Hence the approximation factor follows. It takes constant time to pierce the neighborhood of each piercing unit-disk using a broadcast operation. Hence the running time is $O(|P^*|)$. ■

5 Better Handling of Mobility

We now present the *M-algorithm*, a fully distributed (decentralized) constant approximation algorithm for the mobile piercing set problem that adapts optimally to the mobility of disks: The update cost of the M-algorithm is

$O(1)$. We break the M-algorithm into two parts: the M-Setup algorithm, which builds an initial piercing set, and the M-Update algorithm, which is in charge of adapting the piercing set maintained in response to the mobility of disks (we will see later that the M-Update algorithm may initiate a call to M-Setup as a subroutine at some of the disks). The M-algorithm is more suitable for the highly dynamic ad-hoc mobile network scenario described in Section 1.

The key idea behind the M-algorithm is to break the sequential running fashion of the A-algorithm. In the A-algorithm, an ordering of the unit-disks is mandatory (even if implicitly). As shown in Figure 3, in the worst-case, the movement of one disk (the rightmost one in the figure) could lead to a *global* update of all selected piercing disks, while the cardinality of the minimum piercing set does not change. In order to maintain a relatively stable piercing set, the desired algorithm needs to be able to sever this "cascading effect" — i.e., the algorithm needs to be able to keep the updates *local*. Lemma 2 shows that the cardinality of an optimal piercing set cannot change by much, due to the movement of a single disk. This property guarantees that an update can be kept local. The proof of this lemma, while trivial, is presented here for completeness.

Lemma 2 *If at one time only one unit-disk moves, then $||P^*| - |P^{**}|| \leq 1$, where P^* denotes a minimum piercing set before the movement, and P^{**} denotes a minimum piercing set after the movement.*

Proof: If the cardinality of the piercing set is changed, then it can be either increased or decreased. Since the reverse of a movement that increases the cardinality of the minimum piercing set is a movement that decreases it, we only need to show that the cardinality of the minimum piercing set cannot be increased by more than 1. Let E be the moving disk. If the cardinality of the minimum piercing set is increased after the movement of E, then at least one unit-disk D cannot be pierced by the previous minimum piercing set P^*. The disk D must be the moving unit-disk E. Let $P = P^* \cup \{\text{center of } D\}$. Then P is a piercing set after the movement. Let P^{**} be a minimum piercing set after the movement of E, then $|P^{**}| \leq |P| = |P^*| + 1$. Hence $||P^*| - |P^{**}|| \leq 1$. ■

In the M-Setup algorithm, instead of choosing a disk with respect to some ordering with respect to a given direction $\vec{n}$, we select arbitrary unpierced disks as piercing disks in each step, then pierce the neighborhood of each selected disk D using the points in $PN(D)$. By repeating this procedure at most $O(|P^*|)$ times, we will generate a piercing set for $\mathcal{D}$: Since now we use N points to pierce the neighborhood of each selected piercing disk, the approximation factor is roughly doubled compared to that of the A-algorithm. However, this small degradation in the approximation factor grants an optimal update strategy, as will be shown later.

In order to implement the above idea in a distributed fashion, we repeat the following procedure. Each disk D first checks if there are any piercing disks in its neighborhood. If so, then D marks itself as pierced. Otherwise, each unpierced disk tries to become a piercing disk itself. In order to guarantee that only one disk becomes a piercing disk in an unpierced disk's neighborhood (this is a key property for proving the approximation factor of this algorithm, as it was for the A-algorithm), a mechanism such as "lowest labeled neighbor wins" — assuming that each disk has a unique identification label — is required. Note that, unlike the A-algorithm, in the M-Setup algorithm disks do not need to know the disks' coordinates (no ordering of the $\vec{q}(D) \cdot \vec{n}$ values is required), which may be desirable in an ad-hoc network scenario. The proof of Theorem 2 is analog to that of Theorem 1, and is therefore omitted.

Theorem 2 *The M-Setup generates a piercing set of cardinality within a factor of N of $|P^*|$ in $O(|P^*|)$ time.*

As disks start moving in space, each disk needs to be able to trigger the update procedure whenever an update is necessary. To facilitate the following discussion, we call a disk that is not a piercing disk a *normal* disk. When a disk moves, the following events may make the current piercing set invalid and trigger an update: (i) the boundaries of two piercing disks D and E meet (thus D may become a redundant piercing disk); (ii) the boundaries of one piercing disk D and some normal disk D' pierced by D separate (thus at least one of the disks becomes unpierced). An M-Update procedure is started at disk D in events of type (i), or by disks D and D', for events of type (ii). The M-Update procedure can be divide into two phases: In the first phase, we will unmark some of the disks as to be now unpierced; in the second phase, we select piercing disks for those now unpierced disks. The second phase is executed by a call to M-Setup initiated at each unpierced disk.

The details of the M-Update procedure are as follows. If we have an event of type (i), the M-Update will degrade disk D to a normal disk and unpierce all disks that were currently pierced by D (including D itself). Otherwise, if case (ii) applies, the M-Update will simply unpierce disk D'. Each node that is marked unpierced by the M-Update procedure will invoke M-Setup locally. The M-Setup procedure invoked at an unpierced disk F will first check if any of its neighbors is a piercing disk. If so, it marks itself pierced. Otherwise, if F has the lowest label among its unpierced neighbors, it elects itself as a piercing disk and mark all its unpierced neighbors as pierced. As proven in Theorem 3, all unit-disks will be pierced at the end of the call to M-Setup, and the approximation factor on the size of the piercing set maintained is still guaranteed to be N.

Theorem 3 *The M-Update procedure maintains an N-approximation of the minimum piercing set, with update cost $O(1)$ for each event.*

Proof: First we show that the running time of M-Update is constant per event. Assume that at one time only one event occurs. All the disks the event may affect are located in a neighborhood of a disk D. Thus the operation of marking disks as unpierced (in the first phase) takes constant time. Since all nodes that invoked a call to M-Setup were neighbors of a former piercing disk D, it follows that the calls to M-Setup will have at most a constant number, N, of rounds of "lowest labeled neighbor wins" until a valid set of piercing disks is restored. Therefore the total time taken by each of the invoked M-Setup calls also takes constant time. If several events occur at the same time, then the final effect is the same as if a sequence of events occur in a row, and the update cost per event remains the same.

Now we show that the approximation factor maintained is equal to N. Clearly the resulting piercing set is determined by the collection of selected piercing unit-disks. We will show that the updated collection of piercing disks produced by the M-Update procedure could have been the initial collection of piercing disks produced by the M-Setup algorithm (for a given ordering of the labels of the disks), thus proving the claimed approximation factor. Let $\mathcal{E} = \{D_1, \cdots, D_m\}$ be the collection of the selected piercing unit-disks before the call to M-Update is invoked, and $\mathcal{E}'$ be collection of selected piercing unit-disks after the call to M-Update is completed at all nodes (which may involve calling the M-Setup algorithm locally) is completed. The following three cases may occur:

Case 1. *A normal unit-disk D' moves and after the movement, it is still pierced by some piercing unit-disk in $\mathcal{E}$.* In this case, the M-Update procedure never invokes M-Setup at a node, and $\mathcal{E}' = \mathcal{E}$. Thus, either $\mathcal{E}$ is the initial piercing disk configuration produced by the initial call to the M-Setup algorithm, or we fall into one of the other two cases below, depending on which case applies to the last update which generated $\mathcal{E}$.

Case 2. *A normal unit-disk D' moves, and after the movement, D' is no longer pierced by a piercing unit-disk in $\mathcal{E}$.* In this case, the M-Setup procedure invoked by the call to M-Update will upgrade D' to a piercing disk. Thus $\mathcal{E}' = \mathcal{E} \cup \{D'\}$. The invoked M-Setup takes constant time to upgrade D'.

We prove the bound on the size of the piercing set maintained by showing that $\mathcal{E}'$ could have been obtained by a call to the M-Setup algorithm to the current configuration (placement in space) of the disks, if all disks were currently unpierced. If the labels of the disks in $\mathcal{E}'$ are smaller than the labels of all other disks in $\mathcal{D}$, and if $label(D_1) < \ldots < label(D_m) < label(D')$ Thus, on or before step $i \leq m$, disk D_i will be selected to be a piercing disk (since all D_i's were unpierced disks in $\mathcal{E}$, and no two piercing disks intersect). After all disks $D_1, \ldots, D_m$ are selected, only disk D' is not pierced. Thus the M-Setup must select D' to be a piercing disk, and hence $\mathcal{E}'$ is obtained.

Case 3. *A piercing unit-disk D moves and after the movement, D is pierced by some other piercing unit-disk $E \in \mathcal{E}$.* Let $D = D_i$ where $D_i \in \mathcal{E}$. The M-Update will degrade D to a normal disk and unpierce all unit-disks previously pierced by D. Then the M-Update invokes local calls to the M-Setup at all unpierced disks. For each unit-disk D' previously pierced by D, M-Setup will first check if there is another piercing disk that pierces D'. If so, D' will be marked pierced. Otherwise, if there are neighbors of D which still remain unpierced, then the M-Setup algorithm will upgrade some normal disks to piercing disks. Let $\mathcal{E}'' = \{D_{m+1}, \cdots, D_{m+k}\}$ be the collection of those upgraded piercing disks. Then we have $\mathcal{E}' = (\mathcal{E} - \{D_i\}) \cup \mathcal{E}''$ as the new set of piercing disks. As in Case 2, if $label(D_1) < \ldots < label(D_{i-1}) < label(D_{i+1}) < \ldots label(D_{m+k}) < label(E)$, for all disks E not in $\mathcal{E}'$, the M-Setup algorithm when applied to the current configuration of the disks in $\mathcal{D}$, assuming all disks are unpierced at start, will produce $\mathcal{E}'$ as the resulting set of piercing disks, proving the claimed approximation factors on the cardinality of the set of piercing points produced.

Case 4. *A piercing unit-disk D moves and after the movement, D is not pierced by any other piercing unit-disk $E \in \mathcal{E}$.* Essentially the same as Case 3, but for the fact that we do not degrade D to a normal disk. ■

A simple extension of the M-algorithm provides a polylog approximation algorithm for the nonuniform case. If the collection contains disks of various radii, then we can guarantee an N-approximation if at each step we find the unpierced disk of smallest radii in the collection and pierce its neighborhood. However, we cannot guarantee having $O(1)$ update cost in this case. Without loss of generality, assume that the minimum radius of a disk is equal to 1. If the largest disk radius is bounded by a polynomial on $n = |\mathcal{D}|$, then we have the following Corollary:

Corollary 1 *By grouping the disks into $O(\log n)$ classes such that each class contains disks of radii in $[2^{i-1}, 2^i)$, we have an $O(\log n)$ approximation for the mobile piercing set problem on nonuniform disks with distributed update cost of $O(1)$.*

Proof: In each class, as we show below, N^2 points are enough to pierce an arbitrary neighborhood. Since we have $O(\log n)$ classes, and the piercing set for each class is a N^2-approximation of the over all minimum piercing set, the approximation factor is bounded by $O(\log n)$. Once a disk moves, it only affects the piercing set selected for one class, thus the update cost is still constant. We now show that N^2 points are in fact enough for covering a disk of diameter 2^{i+2}, using disks of diameter in $[2^i, 2^{i+1})$. In the worst case, we need to cover a region of diameter 2^{i+2} with disks of diameter 2^i. We can do this in two phases. First we cover the region using N disks of diameter 2^{i+1}. Then for each disk D of diameter 2^{i+1}, we cover D using N disks of diameter 2^i. ■

6 Applications to Clustering in Mobile Networks

For the ad-hoc network scenario described in the introduction, where all nodes have equal range of communication, the algorithms proposed for the mobile piercing set problem can be directly applied in order to obtain a 1-hop clustering of the network. A *clustering* of a network G is a partition of the nodes of G into subsets (*clusters*) C_i, where for each C_i, we elect a node $v \in C_i$ as the *clusterhead* of cluster C_i. A *1-hop clustering* of G is a clustering of G such that every node in the network can communicate in 1-hop with the clusterhead of the cluster it belongs to. If we view the network G as a collection of unit-disks in $\Re^2$ (resp., $\Re^3$) under L_2 (as discussed in the introduction), we have that $|P^*|/N(2,2) = |P^*|/7$ (resp., $|P^*|/N(3,2) \geq |P^*|/21$) is a lower bound on the minimum number of 1-hop clusters (and therefore on the number of selected clusterheads) needed to cover the entire network — since we need at least one piercing point for each of the neighborhoods of a piercing disk, and since we use at most seven (resp., 21) piercing points for each of these neighborhoods. It follows that the number of *piercing disks* selected by the algorithm in Section 5 is at most $|P^*|$. Since each of these piercing disks D corresponds uniquely to a 1-hop cluster C in the network (given by all the disks pierced by D), and since the union of all these clusters covers the entire network, we have that the number of clusters in such a collection must be near optimal — i.e., equal to $O(|P^*|)$ (more specifically, at most $7|P^*|$ and $21|P^*|$, for $\Re^2$ and $\Re^3$ respectively). Thus the algorithm in Section 5 can be used to obtain a near optimal (with respect to number of clusters) 1-hop clustering of a wireless network where all nodes have equal communication range. This algorithm is also suitable for maintaining such an optimal structure as nodes start moving in space, with optimal update costs. The algorithm tends to keep the number of changes in the set of selected clusterheads low.

In fact, the algorithm presented in Section 5 when translated to a clustering algorithm on ad-hoc networks is essentially the same as the Least Cluster Change (LCC) algorithm presented by Chiang et al. [7]. Therefore, in this paper we provide a theoretical analysis of the performance of this popular clustering algorithm, validating the simulation results that showed that the clusters maintained by this algorithm are relatively stable. Furthermore, we show that this algorithm sustains a 7-approximation on the number of 1-hop clusters maintained.

A closer look at the lowest-id algorithm, investigated by Gerla and Tsai in [15], shows that this algorithm corresponds to several applications of the M-Setup procedure of Section 5. Every time a disk becomes unpierced, or two piercing disks intersect, the lowest-id algorithm starts updating the clustering cover maintained in a fashion that may correspond to an application of the M-Setup algorithm on the current configuration of the disks if all disks were unpierced — in the worst-case, the lowest-id algorithm may generate a "cascading effect" which correspond to an application of the M-Setup algorithm on a collection of all unpierced disks, if the disk labels are given by the node ids. Thus the setup and the worst-case update costs of the lowest-id algorithm are both $O(|P^*|)$, and the approximation on the number of clusters maintained is equal to seven and 21, for $\Re^2$ and $\Re^3$ respectively.

If the mobile nodes know their coordinates in space, an alternative algorithm to the lowest-id algorithm, which achieves same setup and update costs, but which has a better guarantee (of 4 in $\Re^2$, and 11 in $\Re^3$) on the approximation factor maintained is given by the A-algorithm presented in Section 4, when translated to an ad-hoc network scenario as explained above. Since an update caused by a single event in the lowest-id algorithm may incur $\Theta(|P^*|)$ cost, we may as well take advantage of the ordering of the disks used by the A-algorithm.

7 Future work

There are many natural extensions of the work in this paper. We would like to extend the 1-hop clustering structure to a full network clustering hierarchy. One idea would be to apply the same algorithm presented to construct $O(\log n)$ clustering covers of the network: Clustering i would be obtained by assuming that all disks have radius equal to 2^i, for $i = 0, \ldots, \log n$, where $n = |\mathcal{D}|$. One problem with this strategy is that by artificially increasing the communication ranges on the nodes in the network (radius of the disks), a resulting cluster in the hierarchy may not even be connected. Other directions for future work are (i) to develop constant approximation algorithms for piercing a collection of disks of different radii; (ii) to extend any results on nonuniform radii disks to ad-hoc network clustering – note that once we have nonuniform radius disks, we can no longer guarantee symmetric communication between nodes in the network; and (iii) to determine the exact neighborhood piercing number for L_2 norm in 3- (or higher) dimensional spaces. Problem (iii) in $\Re^3$ is highly related with the sphere covering problem, which is still not fully solved. Note that, in case we can determine the neighborhood piercing number $N(p,d)$, the algorithms presented in Sections 4 and 5 could be directly applied to obtain an $N(p,d)$-approximation algorithm for the mobile piercing set problem in $\Re^d$ under L_p, for any given p and d.

Acknowledgement

We would like to express our thanks to Martin Ziegler for valuable discussions on estimating $N(3,2)$.

References

[1] Pankaj K. Agarwal and Cecilia M. Procopiuc. Exact and approximation algorithms for clustering. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 658–667, 1998.

[2] Pankaj K. Agarwal and Micha Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30:412–458, 1998.

[3] S. Basagni. Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks. In *Proceedings of the IEEE Vehicular Technology Conference (VTC), Amsterdam, The Netherlands*, pages 19–22, 9 1999.

[4] S. Basagni. Distributed clustering for ad-hoc networks. In *Proceedings of the 1999 International Symposium on Parallel Architectures*, pages 310–315, June 1999.

[5] J. Basch, Leonidas J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 747–756, 1997.

[6] S. Bepamyatnikh, B. Bhattacharya, D. Kirkpatrick, and M. Segal. Mobile facility location. In *ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 46–53, 2000.

[7] C.-C. Chiang, H.-K. 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.

[8] J. H. Conway and N. J. A. Sloane. *Sphere packings, lattices, and groups*. Springer, 1999.

[9] Zvi Drezner. The p-center problem: heuristic and optimal algorithms. *Journal of Operational Research Society*, 35:741–748, 1984.

[10] Alon Efrat, Matthew J. Katz, Franck Nielsen, and Micha Sharir. Dynamic data structures for fat objects and their applications. In *Workshop on Algorithms and Data Structures*, pages 297–306, 1997.

[11] T. Feder and D. H. Greene. Optimal algorithms for approximate clustering. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 434–444, 1988.

[12] 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.

[13] G. N. Frederickson and D. B. Johnson. Generalized selection and ranking: sorted matrices. *SIAM J. Comput.*, 13:14–30, 1984.

[14] J. Gao, L. J. Guibas, J. Hershburger, L. Zhang, and A. Zhu. Discrete mobile centers. In *17th ACM Symposium on Computational Geometry, Boston, MA.*, jun 2001. To appear.

[15] M. Gerla and J. T. C. Tsai. Multicluster mobile multimedia radio networks. *ACM-Baltzer Journal of Wireless Networks*, 1(3):255–256, 1995.

[16] M. Golumbic. *Algorithmic Graph Theory*. Academic Press, New York, 1980.

[17] T. Gonzalez. Covering a set of points in multidimensional space. *Inform. Process. Lett.*, 40:181–188, 1991.

[18] Sariel Har-Peled. Clustering motion. 2001. FOCS, to appear.

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

[20] D. S. Hochbaum and D. Shmoys. A best possible heuristic for the k-center problem. *Math. Oper. Res.*, 10:180–184, 1985.

[21] D. S. Hochbaum and D. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 33:533–550, 1986.

[22] H. Huang, A. W. Richa, and M. Segal. Approximation algorithms for the mobile piercing set problem with applications to clustering. Technical report, 2001. CSE, Arizona State University.

[23] R. Z. Hwang, R. C. Chang, and R. C. T. Lee. The generalized searching over separators strategy to solve some np-hard problems in subexponential time. *Algorithmica*, 9:398–423, 1993.

[24] R. Z. Hwang, R. C. T. Lee, and R. C. Chang. The slab dividing approach to solve the euclidean p-center problem. *Algorithmica*, 9:1–22, 1993.

[25] Matthew J. Katz, Frank Nielsen, and Michael Segal. Maintenance of a piercing set for intervals with applications. In *ISAAC, Int. Symp. on Algorithms and Complexity*, pages 552–563, December 2000.

[26] M. T. Ko, R. C. T. Lee, and J. S. Chang. An optimal approximation algorithm for the rectilinear m-center problem. *Algorithmica*, 5:341–352, 1990.

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

[28] A. B. McDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad-hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8), August 1999.

[29] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13(1):182–196, 1984.

[30] F. Nielsen. Fast stabbing of boxes in high dimensions. In *Proc. 8th Canad. Conf. Comput. Geom.*, pages 87–92, 1996.

[31] R. Ramanathan and M. Steenstrup. Hierarchically-organized, multihop mobile wireless for quality-of-service support. *Mobile Networks and Applications*, 3:101–119, 1998.

[32] Micha Sharir and Emo Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 122–132, 1996.

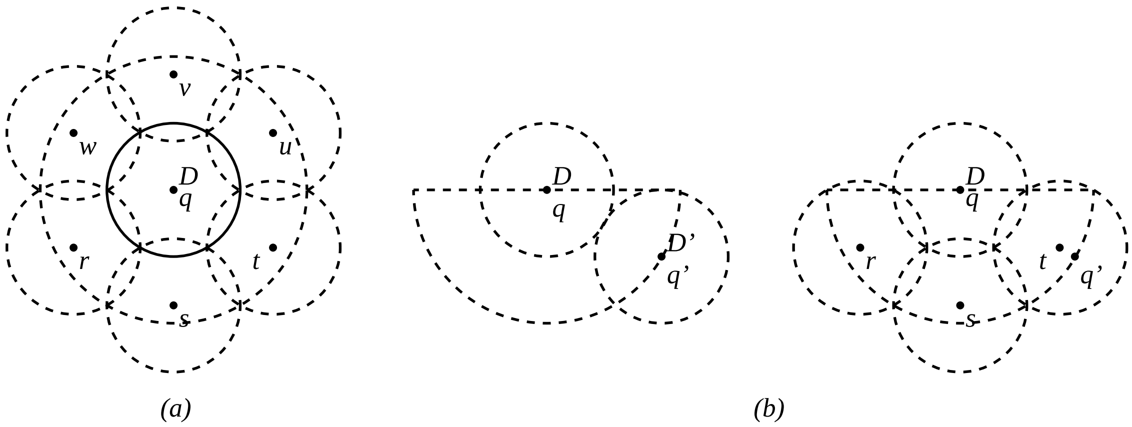

Figure 1: Piercing points for neighborhood of (a) an arbitrary disk (b) an upmost disk

Space/Norm	Approximation Factor, Setup/Update Cost
1D	2-approximation, $O(\lvert P^*\rvert)/O(1)$
2D/L_1, L_∞	4-approximation, $O(\lvert P^*\rvert)/O(1)$
2D/L_2	7-approximation, $O(\lvert P^*\rvert)/O(1)$
3D/L_1, L_∞	8-approximation, $O(\lvert P^*\rvert)/O(1)$
3D/L_2	21-approximation, $O(\lvert P^*\rvert)/O(1)$

Table 1: Main results for 1-, 2- and 3-dimensional spaces with optimal update costs

Space/Norm	Approximation Factor, Setup/Update Cost
2D/L_1, L_∞	2-approximation, $O(\lvert P^*\rvert)/O(\lvert P^*\rvert)$
2D/L_2	4-approximation, $O(\lvert P^*\rvert)/O(\lvert P^*\rvert)$
3D/L_1, L_∞	4-approximation, $O(\lvert P^*\rvert)/O(\lvert P^*\rvert)$
3D/L_2	11-approximation, $O(\lvert P^*\rvert)/O(\lvert P^*\rvert)$

Table 2: Main results for 2- and 3-dimensional spaces with better approximation factor

Figure 2: Piercing points for neighborhood of the right most interval and an arbitrary interval

	$1D$	$2D/L_1$	$2D/L_\infty$	$2D/L_2$
N	2	4	4	7
PN	2 endpoints	4 "corners"	4 "corners"	$\{(0,0),(\pm\frac{3}{4},\pm\frac{\sqrt{3}}{4}),(0,\pm\frac{\sqrt{3}}{2})\}$
$\overline{N}$	1	2	2	4
$\overline{PN}$	left endpoint	left & top "corners"	2 bottom "corners"	$\{(0,0),(\pm\frac{3}{4},-\frac{\sqrt{3}}{4}),(0,\frac{\sqrt{3}}{2})\}$
$\vec{n}$	-1	$(1,-)$	$(0,-1)$	$(0,-1)$

Table 3: Neighborhood piercing number and that on halfspace in 1D and 2D.

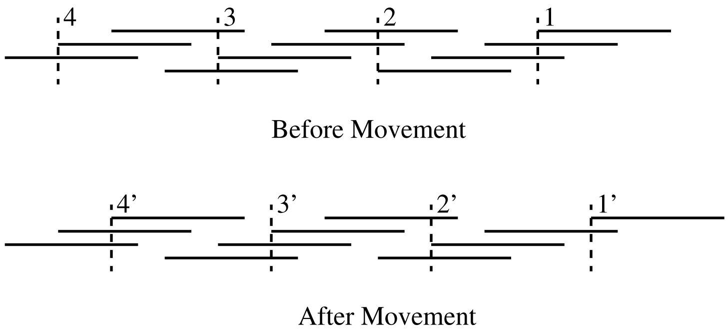

Figure 3: The movement of the right-most interval changes all piercing intervals.

1	(	0.060111,	-0.479945,	-0.718359	)
2	(	-0.559060,	0.562759,	0.347496	)
3	(	-0.164091,	-0.848680,	-0.053063	)
4	(	-0.509173,	-0.375273,	-0.591535	)
5	(	-0.14093,	0.208121,	0.828743	)
6	(	0.690970,	0.512659,	-0.098697	)
7	(	0.37763,	-0.700146,	0.342736	)
8	(	-0.250648,	-0.50408,	0.658097	)
9	(	-0743210,	0.007958,	0.444494	)
10	(	0.550076,	0.074555,	-0.664724	)
11	(	0.521328,	-0.588831,	-0.362624	)
12	(	0.088527,	0.791576,	0.339956	)
13	(	-0.158465,	0.221391,	-0.822116	)
14	(	-0.718586,	-0.473046,	0.099309	)
15	(	-0.339530,	0.726217,	-0.327609	)
16	(	0.439846,	-0.237215,	0.707294	)
17	(	0.853710,	-0.140538,	0.037800	)
18	(	0.536789,	0.335659,	0.590924	)
19	(	0.243253,	0.709202,	-0.433429	)
20	(	-0.778146,	0.197643,	-0.324694	)

Table 4: The 20 points for the N(3,2).

M-Setup

For each *unmarked* unit-disk D:

Repeat
    If there is piercing unit-disk in $\mathcal{N}(D)$ then
        `IsMarked`=True
    Elseif D bears the lowest *label* among all its neighbors which attempt to become a piercing disk then
        For each unmarked neighbor D' of D
            `D'.IsMarked`=True
        End
    End
Until the disk becomes marked

M-Update

When a unit-disk D moves

If D is a piercing unit-disk, then
    If D's boundary meets that of another piercing unit-disk E, then
        remove the neighborhood piercing points of D from P
        Unmark D and all normal unit-disks that were marked by D
    End
    If D's boundary separates from that of a normal unit-disk D', then
        Unmark D'
    End
Else (D is a normal unit-disk)
    If D's boundary separates from that of D's piercing unit-disk, then
        Unmark D
    End
Call M-Setup

Figure 4: The M-Setup and M-Update algorithm