Connected Dominating Sets on Dynamic Geometric Graphs*

Leonidas Guibas† Nikola Milosavljević‡ Arik Motskin§

Abstract

We propose algorithms for efficiently maintaining a constant-approximate minimum connected dominating set (MCDS) of a geometric graph under node insertions and deletions. Assuming that two nodes are adjacent in the graph iff they are within a fixed geometric distance, we show that an $O(1)$-approximate MCDS of a graph in $\mathbb{R}^d$ with n nodes can be maintained with polylogarithmic (in n) work per node insertion/deletion as compared with $\Omega(n)$ work to maintain the optimal MCDS, even in the weaker kinetic setting. In our approach, we ensure that a topology change caused by inserting or deleting a node only affects the solution in a small neighborhood of that node, and show that a small set of range queries and bichromatic closest pair queries is then sufficient to efficiently repair the CDS.

1 Introduction

For an undirected graph $G = (V, E)$, we say that $S \subseteq V$ is a *dominating set* of G if every node in $V \setminus S$ is adjacent to some node in S. If S also induces a connected subgraph of G, then S is a *connected dominating set (CDS)* of G. The problem of finding a CDS with the fewest vertices, or *minimum CDS (MCDS)*, in a given graph has been of interest for a long time because of its application in networking. If G represents a communication network, a small CDS S of G can be used to form a routing "backbone" [6]. The advantage of a CDS-based approach is that all nodes are close to some "backbone" node, and long distance communication is achieved within the sparser subgraph induced by the CDS; potentially expensive routing infrastructure is required only within the small set S.

The problem has also been explored in the special case of *ad-hoc wireless networks*, where *geometric graphs* are of particular interest [6, 14]. In a geometric graph, nodes are identified with points in a metric space, and two nodes are connected if they are sufficiently close to each other. More recently, there has been a growing interest in ad-hoc networks whose nodes can join and leave arbitrarily, causing the node set and the induced geometric graph to change over time. The topic of this paper is the problem of maintaining a small CDS in such dynamic graphs. The CDS may require repair when a node is added or deleted, so the challenge is to maintain a CDS that is sufficiently sparse, while also minimizing the necessary repair work.

1.1 New Results

We show that maintaining an *exact* MCDS is expensive. Not only can a single insertion or deletion require $\Omega(n)$ work (where n is the number of nodes), but the same bound also holds in the amortized sense, and even when restricted to sequences of highly correlated insertions and deletions emulating a continuous, bounded degree algebraic motion.

Our main contribution is a pair of algorithms that efficiently maintain a constant-approximate MCDS for geometric graphs in $\mathbb{R}^d$ with rectilinear (ℓ_1 and ℓ_∞) metrics. In particular, our algorithms maintain an $O(1)$-approximate MCDS with $\widetilde{O}(1)$ work[1] per operation (insertion or deletion).

For geometric graphs in $\mathbb{R}^1$, the CDS that we maintain is an *alternating independent set*. In $\mathbb{R}^d$, the CDS that we maintain comprises a maximal independent set (MIS) of nodes – which is already a dominating set – along with a small set of paths chosen to make the MIS connected while retaining the constant approximation factor. While the idea of augmenting a MIS with a small set of connectors to form a CDS is not new [2], our paper is the first to address the challenge of efficiently selecting such a sparse structure from an arbitrarily complex underlying dynamic network. Our construction ensures that graph topology changes affect the CDS solution only locally, while a small set of range queries and bichromatic closest pair queries is sufficient to repair the CDS.

All of the algorithms in this paper are *centralized*. As in the literature on maintaining approximate (but not necessarily connected) minimum dominating sets for mobile nodes [8, 11], the quantity of interest is the computational complexity of *simulating* a dynamic network on a single processor.

*This work was supported by NSF grants 0626151, 0634803, and 0914833, and a grant from Google, Inc.

†Stanford University, guibas@cs.stanford.edu

‡Max-Planck-Institut für Informatik, nikolam@mpi-inf.mpg.de

§Stanford University, amotskin@stanford.edu

[1]For any function g, the notation $\widetilde{O}(g(n))$ is equivalent to $O(g(n) \log^c n)$, for a constant c. Furthermore, the constant hidden in the big-Oh may depend on d.

1.2 Related Work

The MCDS problem is very well-studied, long known to be NP-complete [9] for general graphs, but also for unit-disk graphs [5], the context that we are considering. While difficult to approximate in general graphs [4, 10], several polynomial-time approximation schemes for MCDS in static unit-disk graphs have been developed [4, 12, 15].

However, there has been relatively little work on maintaining a CDS under node insertions and deletions. Gao *et al.* [8] and Hershberger [11] presented algorithms for the related minimum dominating set (MDS) problem, and in weaker *kinetic* setting, where nodes follow continuous (typically bounded degree algebraic) trajectories instead of entering and leaving at arbitrary locations, and the quantity of interest is the *total* computational cost for the whole motion. Gao *et al.* [8] give a randomized algorithm for maintaining an $O(1)$-approximation to the MDS, requiring $\widetilde{O}(n^2)$ cumulative work. Hershberger [11] offers a constant-approximate deterministic algorithm for maintaining a covering of nodes by axis-aligned unit boxes. Observe that we achieve the same (up to constant factors) approximation[2] and cumulative work[3] bounds as these solutions, but with the additional challenge that our structure be *connected* (a key requirement for networking applications), and that the time bound holds per-operation, instead of only in the amortized sense. Unlike Gao *et al.* [8], our solution is deterministic, but as in both previous MDS solutions, we work with rectilinear norms, so our solution comprises a covering of the nodes by a connected set of axis-aligned unit boxes, each centered at a node.

Alzoubi *et al.* [2] maintain a constant approximate MCDS solution with mobile nodes, but a change in topology can require $\Omega(\Delta)$ work in repairs (where Δ is the maximum node degree), aggregating to potentially $\Omega(n^3)$ total work over the course of the entire pseudo-algebraic motion.

1.3 Organization of the Paper

The remainder of the paper is organized as follows. In Section 2, we introduce the geometric graph model for which our results hold. In Section 3, we prove a $\Omega(n)$ lower bound on the per-operation computational cost of maintaining an *exact* MCDS, even in the kinetic setting. This motivates the problem of maintaining a constant-approximation. In Section 4 we describe geometric data structures (d.s. for short) used by both algorithms (for graphs in $\mathbb{R}^1$ and $\mathbb{R}^d$, for a constant $d \geq 2$). The actual algorithms are described in Sections 5 and 6. In Section 7, we conclude the paper and propose several directions for future work.

[2]In Sections 5 and 6, we prove the approximation bounds for our CDS by showing it is within a constant factor of the MDS.

[3]This is because over the course of a bounded degree algebraic motion there are $O(n^2)$ "events" that trigger updates to the MDS.

2 Model

In our network model, we assume a graph $G = (V, E)$ (with $|V| = n$), embedded in $\mathbb{R}^d$ for d fixed. Graph topology is determined by the unit-ball graph model with respect to the ℓ_∞ norm in the plane[4], i.e., nodes are joined by an edge if and only if their ℓ_∞-distance is at most 1. We use $d(\cdot,\cdot)$ to denote the ℓ_∞-distance, and $d_G(\cdot,\cdot)$ to denote the hop-distance in G. We denote individual dimensions (point/node coordinates) by subscripts. We assume that G remains connected as nodes are inserted and deleted.

For a hyperplane $h = \{x \in \mathbb{R}^d \mid (x-a)b = 0\}$ we define $h^+ = \{x \in \mathbb{R}^d \mid b(x-a) \geq 0\}$ and $h^- = \{x \in \mathbb{R}^d \mid b(x-a) < 0\}$. A set of d hyperplanes $H = \{h_1, h_2, \ldots, h_d\}$ (whose normals $B = \{b_1, b_2, \ldots, b_d\}$ are a basis of $\mathbb{R}^d$) defines a *(polyhedral) cone* $\bigcap_{i=1}^d h_i^+$. We use H to denote both the set of hyperplanes and the cone that they define. The *apex* of H is $\bigcap_{i=1}^d h_i$. For a cone H with apex p, the *angle* of H is $\max_{x,y \in H} \angle(x-p, y-p)$. Clearly, this depends only on normals, so we also call it the angle of B. We say that H (or B) is δ-narrow, if its angle is at most δ. We say that H *separates* a pair of points (x, y) if $p - x, y - p \in H$.

For the proofs we will need the following technical fact.

Lemma 1 *For any $\delta > 0$, one can compute in $O((\frac{d-1}{\delta})^{d-1})$ time a collection $\mathcal{B} = \mathcal{B}(\delta)$ of $O((\frac{d-1}{\delta})^{d-1})$ δ-narrow bases of $\mathbb{R}^d$ such that any vector is "well inside" the cone defined by some $B \in \mathcal{B}$, i.e., $\forall x \in \mathbb{R}^d$, $\exists B \in \mathcal{B}$, $\forall y \in \mathbb{R}^d$, $\angle(x, y) \leq \frac{1}{2}\arctan\left(\frac{1}{d-1}\tan\frac{\delta}{2}\right) \Rightarrow b_1 y, b_2 y, \ldots, b_d y \geq 0$.*

Proof. See Appendix. □

In this paper, we think of a basis as a set of hyperplane normals which define the boundary of a cone. Lemma 1 ensures that we can precompute a collection of bases $\mathcal{B}$, such that the space around *any* point p can be sufficiently covered by a collection of narrow cones, each having apex at p, and each defined by one of the bases $B \in \mathcal{B}$. Given a metric space (X, d), we call $Y \subseteq X$ an *r-packing (in (X, d))* if for any $y_1, y_2 \in Y$, $d(y_1, y_2) \geq r$, and an *r-cover (in (X, d))* if for any $x \in X$, $d(x, Y) \leq r$. The r-ball (in (X, d)) with center c and radius r is denoted by $B_{(X,d)}(c, r)$, where the subscript is omitted if clear from context.

[4]For simplicity, the results in this paper are described in terms of the ℓ_∞ norm, but also hold for the ℓ_1 norm.

3 Motivation

In this section we show that maintaining the optimal solution cannot be done significantly faster than recomputing it from scratch after each graph update. In fact, this holds even for very simple dynamics — continuous motion of vertices (the so called *kinetic* setting [3]) — and even when the speed of each vertex does not change with time. In particular, we demonstrate that shifting just 2 nodes causes $\Omega(n)$ work in repairs to the MCDS. This stands in contrast to our main result, where our $O(1)$-approximate MCDS requires just $\widetilde{O}(1)$ work to be repaired after arbitrary node insertion or deletion.

Consider n nodes in $\mathbb{R}^1$, and the graph G obtained by connecting two distinct points if and only if their coordinates differ by at most 1. As the nodes move, the graph topology changes, and so does the set of its MCDSs. First we show how to test if a given subset of nodes is the unique MCDS for a given graph. This claim will be invoked multiple times in the proof of the main result.

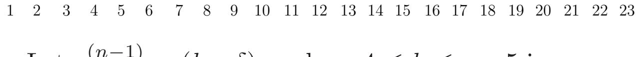
Lemma 2 *Let $v^{(1)}, v^{(2)}, \ldots, v^{(n)}$ be a fixed left-to-right ordering of nodes (ties broken arbitrarily), and let S be a CDS. S is a unique MCDS if and only if $v^{(1)}, v^{(n)} \notin S$, and there are no distinct vertices u, v, w (ordered from left to right), all pairwise connected, such that $v \in S$ and either $u \in \{v^{(1)}\} \cup S$ or $w \in \{v^{(n)}\} \cup S$.*

Proof. See Appendix. □

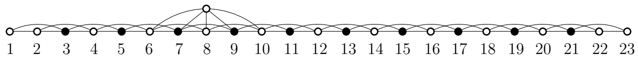
Now we can prove our main result.

Theorem 3 *For any sufficiently large n, one can assign initial positions and constant speeds to n nodes so that the MCDS undergoes $\Omega(n^3)$ total state changes during the motion.*

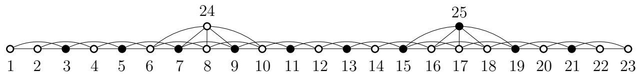
Proof. Let n be odd, and let $v^{(1)}, v^{(2)}, \ldots, v^{(n)}$ be the nodes. Suppose $v_1^{(i)} = ia$, for $i = 1, 2, \ldots, n-2$, where $a = \frac{1}{3} + \varepsilon$, and $\varepsilon > 0$ is small enough, so that $2.5a$ is smaller than the communication range 1. By Lemma 2, the unique MCDS of $\{v^{(1)}, \ldots, v^{(n-2)}\}$ is $S_1 = \{v^{(3)}, v^{(5)}, \ldots, v^{(n-4)}\}$. See figure below for an illustration with $n = 25$.



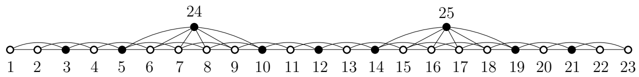
Let $v_1^{(n-1)} = (k - \delta)a$, where $4 \leq k \leq n-5$ is an even integer, and $\delta > 0$ is small enough, so that $n-1$ and k have the same neighbors (except each other). Again, Lemma 2 implies that S_1 is still the unique MCDS of $\{v^{(1)}, \ldots, v^{(n-1)}\}$. See figure below for an illustration with $n = 25$, $k = 8$.



Finally, let $v_1^{(n)} = (l - \delta)a$, where l is an odd integer "much bigger than k", and δ is small enough, so that $v^{(n)}$ and $v^{(l)}$ have the same neighbors (except each other). No MCDS can contain both $v^{(l)}$ and $v^{(n)}$ (because removing one of the two would yield a smaller CDS). It is easy to see that the CDSs that do not contain $v^{(n)}$ are exactly the CDSs for $\{v^{(1)}, \ldots, v^{(n-1)}\}$. Hence, the unique MCDS that does not contain $v^{(n)}$ is S_1. Symmetrically (exchanging the roles of $v^{(l)}$ and $v^{(n)}$), the unique MCDS that does not contain $v^{(l)}$ is $S_2 = \{v^{(3)}, v^{(5)}, \ldots, v^{(l-2)}, v^{(n)}, v^{(l+2)}, \ldots, v^{(n-4)}\}$. Since $|S_1| = |S_2|$, it follows that S_1 and S_2 are the only minimum CDSs. See figure below for an illustration with $n = 25$, $k = 8$, and $l = 17$.



Now suppose that nodes $v^{(n-1)}$ and $v^{(n)}$ move to the left by $\frac{a}{2}$, that is, $v_1^{(n-1)} = (l - \frac{1}{2})a$ and $v_1^{(n)} = (k - \frac{1}{2})a$, where k and l have the same values as before. By Lemma 2, the unique MCDS for $\{v^{(1)}, \ldots, v^{(n)}\}$ is $S_3 = \{v^{(3)}, v^{(5)}, \ldots, v^{(k-3)}, v^{(n-1)}, v^{(k+2)}, \ldots, v^{(l-3)}, v^{(n)}, v^{(l+2)}, \ldots, v^{(n-4)}\}$. See figure below for an illustration with $n = 25$, $k = 8$, and $l = 17$. Notice that all nodes between (not including) $v^{(k+2)}$ and $v^{(l-3)}$ are in S_3 if and only if they are in neither S_1 nor S_2.



Set $l - k$ to be the odd number closest to $\frac{n}{2}$. If initially $v_1^{(n-1)} = (n-1)a$, $v_1^{(n)} = (n + l - k - 1)a$, and n, $n-1$ move to the left much faster than all other nodes, any algorithm that maintains the exact minimum CDS at all times must make $\Omega(n)$ transitions between S_1 or S_2 (depending on the algorithm) and S_3, with each such transition requiring $\Omega(n)$ node state changes. Finally, we can add $\frac{n}{2}$ copies of nodes $v^{(n-1)}$ and $v^{(n)}$ with the same initial positions and speeds geometrically increasing by sufficiently large factors, to get a $2n$-node instance that undergoes $\Omega(n^3)$ total state changes over the course of the motion. □

4 Data Structures

To repair a CDS after a graph change, our algorithms use a number of range queries and bichromatic closest pair queries. In this section we precisely define queries of interest and argue that data structures that support those queries can be maintained in $\widetilde{O}(1)$ time under node insertions/deletions. In subsequent sections we assume correctness of the data structures and show how they are used to maintain correctness of the output (approximate MCDS) after graph changes.

Our algorithms reduce the problem of maintaining an $O(1)$-approximate MCDS to two fundamental problems of dynamic computational geometry. Let P be a set of n points in $\mathbb{R}^d$, where d is fixed (independent of n). An *orthogonal range searching (ORS) problem* asks to report all points in P contained in a given axis-aligned box, including its boundary. The *bichromatic closest pair (BCP) problem* asks for a closest red-blue pair, given a BCP instance (i.e., a labeling) that designates each point as either red or blue. A BCP instance *separates* point pair (x, y) if x is red and y is blue in that instance. We say that a BCP instance is *separated by a cone C* if any red-blue pair in the instance is separated by the cone C. We say that a BCP instance is *δ-narrow* if it is separated by a δ-narrow cone.

The d.s. of Willard and Lueker [13] solves the ORS problem in $O(k + \log^d n)$ time, where k is the number of reported points. It can be updated to reflect a single point insertion/deletion in $O(\log^d n)$ time, and uses $O(n \log^{d-1} n)$ space. The d.s. of Agarwal *et al.* [1] solves a certain set of BCP instances on subsets of P in time $O(1)$. It uses $O(n \log^{2d-1} n)$ space, and can be updated in time $O(\log^{2d} n)$ after insertion/deletion [7]. The set of BCP instances that it solves, defined for a given basis $B = \{b_1, b_2, \ldots, b_d\}$ of $\mathbb{R}^d$, and denoted by $\mathcal{I}(B)$, satisfies the following properties.

(i) Any instance in $\mathcal{I}(B)$ is separated by some cone with normals B.
(ii) Given a pair (x, y) separated by a cone with normals B, one can compute in $O(\log^d n)$ time an instance in $\mathcal{I}(B)$ that separates (x, y).

We prove the following generalization of property (ii), which we will use in Section 6.2.2.

Lemma 4 *Given a cone C with normals B, one can compute in $O(\log^d n)$ time a set $\mathcal{I}(B, C) \subseteq \mathcal{I}(B)$ of $O(\log^d n)$ instances, such that each pair separated by C is separated by some instance in $\mathcal{I}(B, C)$.*

Proof (sketch): See Appendix.

As defined in Section 2, V is the set of all nodes, while T (defined in Section 6) is a certain subset of V. We maintain two ORS d.s.'s, one for V and one for T. In addition, we maintain one BCP d.s. for each basis $B \in \mathcal{B}(\delta)$ (defined in Lemma 1), where δ is a constant that depends on d only. By Lemma 1 and property (i), all BCP instances that we maintain are δ-narrow.

5 Maintaining a CDS in $\mathbb{R}^1$

In this section, we show how to construct and maintain an $O(1)$-approximation to the MCDS in 1-dimensional graphs, with $\widetilde{O}(1)$ work to repair the solution after node insertions or deletions. The main idea is to maintain an *alternating independent set* that dominates the graph, which we show is a good approximation to the MCDS. For simplicity, in this section a node label u will refer both to a node's name as well as its x-coordinate.

Definition 1 *$S \subset V$ is an alternating independent set (AIS) for a graph in $\mathbb{R}^1$, if S is connected, and if for any three nodes u, v, $w \in S$ with $u \le v \le w$, u and w are not connected by an edge.*

Proposition 5 *A dominating AIS is an $O(1)$-approximation to the MCDS for a graph in $\mathbb{R}^1$.*

Proof. Let S^* denote an MCDS. Given a node $v \in S^*$, observe that it is connected to at most 4 nodes from S: if not, there are at least 5 nodes $v_1 \le v_2 \le v_3 \le v_4 \le v_5$ adjacent to v. By the AIS property, $v_3 > v_1 + 1$ and $v_5 > v_3 + 1$, so $v_5 > v_1 + 2$. Since $v_1 > v - 1$, $v_5 > v + 1$, a contradiction.

Moreover, since S^* is connected, every member of the AIS (except possibly the leftmost or rightmost nodes) are connected to at least 2 members of S^*. These facts imply $|S| \le 2|S^*| + 2$. □

The initial construction of a dominating AIS is straightforward. Initially, set $S \leftarrow \emptyset$. Select an arbitrary node u_0 and add it to S. Let u_1 be a right neighbor of u_0, and add it to S. Let u_2 be a right neighbor of u_1; add it to S, and if u_0 and u_2 are connected, remove u_1 from S. For notational purposes, suppose during the iterative construction that the elements of S are re-indexed as $u_0, u_1 \ldots\ u_i$ where $i < j \Leftrightarrow u_i < u_j$. Select a right neighbor u_{i+1} of u_i, add it to S, and if u_{i-1} and u_{i+1} are connected, remove u_i from S. Iterate until done, and repeat procedure to the left of u_0.

By construction, the leftmost and rightmost nodes are initially in S. Since S is connected and in $\mathbb{R}^1$, it is a dominating set. Furthermore, using the ORS d.s. on V described in Section 4, which is simply a binary search tree in this case, the left/right neighbor of each node is computable in $\widetilde{O}(1)$ time if it exists, so this construction takes $\widetilde{O}(n)$ time.

5.1 Maintaining Solution with Insertions and Deletions

Now, assume that a dominating AIS S has already been constructed. We show how to maintain the AIS after a node insertion or deletion.

1. If node v is added: Check if v is connected to fewer than two nodes in S (ORS query on S with $B(v, 1)$). If so, add it to S.

2. If $v \in S$ is removed: Find immediate left and right neighbors of v if they exist, and add them to S. By the connectivity assumption, this restores connectivity and domination of S. The remaining property can only be violated for triples whose "middle

node" is either one of the added nodes, or previous left and right neighbors of v in S, if they exist. Check each one of these (at most 4) possibilities, and whenever the check fails, remove the "middle node". This preserves connectivity and domination.

Immediate left and right neighbors in V (resp. S) of some node can be found by binary search for that node in the range searching d.s. on V (resp. S), which is in fact a binary search tree in case $d = 1$. *All* neighbors in S of some node can obviously be found by range searching, and there are always $O(1)$ of them. With regards to necessary data structures, as per the discussion in Section 4, each one of these operations costs $\widetilde{O}(1)$. The above procedure for maintaining a dominating AIS invokes them $O(1)$ times, so its cost is also $\widetilde{O}(1)$.

6 Maintaining a CDS in $\mathbb{R}^d$

In this section, we show how to construct and maintain an $O(1)$-approximation to the MCDS for graphs in $\mathbb{R}^d$, with $\widetilde{O}(1)$ work to fix the solution after each node insertion or deletion. Our strategy is to maintain a maximal independent set of nodes T (which is itself a dominating set), along with a small set of connecting paths, which together form a small CDS S. The challenge is to efficiently compute such a sparse structure from an arbitrarily complex underlying set of nodes.

6.1 Defining the CDS

Let V be the input node set, and let G be its unit-ball graph. Consider a *maximal 1-packing* on G, which we denote T. For each point $v \in T$, consider the set S_v of v's *connectors*, defined as follows. Let U_v be the set of *all* nodes $u \in T$ with $d_G(v, u) \leq 3$, plus some nodes $u \in T \cap B(v, 3)$ with $d_G(v, u) \leq 5$. Pick a v–u path of length ≤ 5 for each $u \in U_v$. The connectors of v comprise all points on these chosen paths, including the endpoints.

Definition 2 $S = \bigcup_{v \in T} S_v$. *Clearly,* $S \supset T$.

Lemma 6 *If G is connected, S is a CDS for G.*

Proof. By maximality, T is a 1-cover, so it is also a dominating set in G, as is its superset S. We claim that the subgraph H of G induced by S is connected. By definition of connectors, each $u \in S \setminus T$ is connected in H to some $v \in T$ via S_v. Hence it suffices to prove that any $u, v \in T$ are connected in H. Let $u_1, u_2, \ldots, u_k$ be the intermediate nodes on some u–v path in G. Since T is a dominating set, u_2 is adjacent in G to some $w_2 \in T$. As $d_G(u, w_2) \leq 3$, w_2 is connected to u in H, by the definition of connectors. If w_2 is connected to v in H, we are done. Otherwise, since T is a dominating set, u_3 is adjacent in G to some $w_3 \in T$. Since $d_G(w_2, w_3) \leq 3$, w_3 is connected to w_2, and hence to u, in H. If w_3 is connected to v in H, we are done. Otherwise, this argument is repeated until we have a node w_i connected to v in H, which is certainly true for w_{k-1}, since $d_G(w_{k-1}, v) \leq 2$. Thus, we demonstrate a connecting path in H (via nodes w_i) between u and v. □

Lemma 7 *If G is connected, S is an $O(1)$-approximate MCDS in G.*

Proof. Let S^* be an arbitrary MCDS for G. Since T is a 1-packing for $d_G(\cdot, \cdot)$, it is also a 1-packing for $d(\cdot, \cdot)$, so any unit ball contains $O(1)$ elements of T. Considering such balls centered at the points of S^*, we get $|T| = O(1)|S^*|$. For any $v \in T$, $|S_v|$ is at most six times the number of nodes in T connected to u via paths of length ≤ 5. By definition of connectors, all such nodes belong to $B(v, 4)$. By a packing argument, $B(v, 4)$ contains $O(1)$ elements of T. Hence $|S| \leq \sum_{v \in T} |S_v| = O(1)|T| = O(1)|S^*|$. □

6.2 Dynamic Maintenance

The key property of our CDS solution S is that a change in graph topology due to node insertion or deletion affects our solution only locally, so that repairs are limited to $\widetilde{O}(1)$ work. In this section, assume that a CDS S as in Definition 2 already exists; we show how to repair the maximal independent set T and connectors $\{S_v\}_{v \in T}$ according to Definition 2 after each insertion/deletion event.

For each such event, we execute the following three-step procedure. Let v denote the node that was inserted or deleted.

(A) Update V. Update the ORS d.s. on V.

(B) Update T. If node v is inserted, it may not be dominated by T. Check this by querying ORS d.s. on T with $B(v, 1)$. If query returns empty, add v to T and update ORS d.s. on T.

If node v is deleted and $v \in T$, remove v from T and update ORS d.s. on T. The new T may no longer be maximal. We must check if all nodes in $B(v, 1)$ are still adjacent to a node in T. Observe that $R = T \cap B(v, 2)$ comprises the only current nodes of T that may be adjacent to a node in $B(v, 1)$. Compute R by querying ORS d.s. on T with $B(v, 2)$. By packing, $|R| = O(1)$. Decompose $B(v, 1) \setminus \bigcup_{r \in R} B(r, 1)$ into $O(1)$ axis-aligned boxes. Query the ORS d.s. on V with each box and return a node w contained in any of them, if it exists. If not, T must be maximal, and we are done. Otherwise, add w to T (with default $S_w = \emptyset$) and update the ORS d.s. on T. Repeat, starting by recomputing R. There are $O(1)$ repetitions, since by packing there can only be $O(1)$ independent nodes in $B(v, 1)$.

(C) Update connectors. Now, we need to ensure that the connectors are built according to the definition of Section 6.1. If v was removed from T in the previous step, remove S_v from S. Any node w added to T in the previous step (including possibly v itself) does not yet have connectors (i.e., $S_w = \emptyset$), so they must be built; note that any such w is located in the ball $B(v, 1)$.

Moreover, observe that for any other node $z \in T$, the insertion or deletion of node v can only affect the correctness of S_z if $z \in B(v, 4)$. Hence, to repair all connectors according to Definition 2, it suffices to recompute S_w for all $w \in T \cap B(v, 4)$, using the subroutine in Section 6.2.2.

6.2.1 Analysis

Step (A) requires one update of the ORS d.s. on V. Step (B) requires $O(1)$ updates of T and its ORS d.s., $O(1)$ range queries on T with constant-sized result R, $O(1)$ decompositions into boxes each taking $O(1)$ time, and $O(1)$ range queries on V. Step (C) requires one range query on T and $O(1)$ invocations of the subroutine. By the discussion in Section 4, and assuming $\widetilde{O}(1)$ runtime for the subroutine (Lemma 11 in Section 6.2.2), the entire procedure takes $\widetilde{O}(1)$ time per operation. We conclude the following.

Theorem 8 *The $O(1)$-approximate MCDS S requires $\widetilde{O}(1)$ work to be maintained after a node insertion or deletion.*

6.2.2 Connecting Path Subroutine

In this section we describe the key technical result: a subroutine that efficiently computes connectors S_v for a given node $v \in T$. We begin by computing the set of "candidate nodes" $U = T \cap B(v, 3)$. Obviously, $|U| = O(1)$, and U includes all nodes of T within 3 hops of v. The subroutine tries to connect each $u \in U$ to v by a path of length ≤ 5 and, if successful, adds the nodes on the path to S_v. The procedure is the same for each u. To prove correctness of output S_v, it suffices to prove that the procedure succeeds when $d_G(u, v) \leq 3$ (Lemma 10).

The strategy is to try to identify "waypoints": intermediate nodes on the connecting paths. We know that the waypoints are contained in $B(v, 3)$. We pick a dense enough (but still constant-sized) sampling Q of this region of interest around v, comprising points of the space (and not necessarily nodes in V). Suppose $d_G(u, v) \leq 3$. Define u', v' to be the neighbors of u and v, respectively, with minimum $d(u', v')$. Notice that u' and v' may be the same node (if $d_G(u, v) = 2$). We remark that u' and v' are precisely the type of waypoints we seek (because $\{u, u', v', v\}$ is a valid connecting path). Our subroutine, however, may find two nodes which are not quite u' and v', but still suitable as waypoints: we still manage to connect u and v via these nodes. The key step is showing how a point $q \in Q$ that ε-covers the centroid of $\{u', v'\}$ can be used to find these necessary waypoints.

Lemma 9 suggests that a good choice for the waypoints is the solution to any BCP instance in which u' is red, v' is blue, and a narrow cone separates red from blue points. To find such an instance, first we must find a narrow cone C which separates u' and v'. Such a cone is found by testing the cones generated by each of the bases $B \in \mathcal{B}$, with apex at sample points $q \in Q$. We prove that the construction of $\mathcal{B}$ (in Lemma 1) and the density of Q is sufficient to identify the required cone C. Then, using C, we solve $\widetilde{O}(1)$ BCP instances drawn from $\mathcal{I}(B)$, each of which can be solved in time $\widetilde{O}(1)$ (since they are in $\mathcal{I}(B)$), and one of which we know (by Lemma 4) outputs a good solution in the sense of Lemma 9. From the solution of this good instance, we get a suitable pair of waypoints.

If $d(u', v') \leq \frac{3}{4}$, i.e., the middle hop is short, the subroutine finds a 3-hop path $\{u, z, w, v\}$ joining u and v. If, on the other hand, $d(u', v') > \frac{3}{4}$, i.e., the middle hop is long, the subroutine finds a 5-hop path $\{u, z, x, y, w, v\}$ joining u and v.

Formally, the subroutine for a fixed $u \in U$ proceeds as follows: let $\delta = \frac{1}{2} \arctan \frac{1}{d+\sqrt{d-1}}$, and $\varepsilon = \frac{3}{8} \sin\left(\frac{1}{2} \arctan\left(\frac{1}{d-1} \tan \frac{\delta}{2}\right)\right)$. Compute an ε-sample Q of $B(v, 3)$. Assume that the set of bases $\mathcal{B} = \mathcal{B}(\delta)$ (as in Lemma 1) has been computed in preprocessing, and that the set of BCP instances $\mathcal{I}(B)$ has been maintained for each $B \in \mathcal{B}$, as described in Section 4. For each $u \in U$ and $q \in Q$, try to find $w \in V \cap B(q, \frac{1}{2}) \cap B(u, 1)$ and $z \in V \cap B(q, \frac{1}{2}) \cap B(v, 1)$. If any of these is successful, we have found 3-hop connecting path $\{u, w, z, v\}$; add it to S_v. Otherwise, we look for a 5-hop path to connect u and v: for each $B \in \mathcal{B}$ and $q \in Q$, compute $\mathcal{I}(B, C)$ (as in Lemma 4), where C is the cone with normals B and apex q. For each instance in $\mathcal{I}(B, C)$, find its solution (x, y), and try to find $w \in V \cap B(x, 1) \cap B(u, 1)$ and $z \in V \cap B(y, 1) \cap B(v, 1)$. If for any of these, w and z are found and $d(x, y) \leq 1$, we have found a 5-hop connecting path $\{u, w, x, y, z, v\}$; add it to S_v.

Lemma 9 *If a δ-narrow BCP instance contains a red-blue pair (u, v) with $d(u, v) \leq 1$, then its solution (x, y) satisfies $d(u, x), d(y, v) \leq d(u, v)$.*

Proof. It suffices to prove $d(u, x) \leq d(u, v)$, because $d(y, v)$ then follows by symmetry. Let C be a δ-narrow cone separating all red-blue pairs. Without loss of generality, the apex of C is at the origin, because the claim is invariant under simultaneous translation of V and C.

By convexity of $d(u, \cdot)$ on $\mathbb{R}^d$, convexity of C, and $v \in C$, we have $d(u, 0) \leq d(u, v)$. Hence it suffices to prove $d(u,x) \le d(u,0)$. If $\angle(x-u,x) > \frac{\pi}{2}$, then $u0$ is the longest side of triangle $ux0$, so we are done. From now on we assume $\angle(x-u,x) \le \frac{\pi}{2}$.

Using $-u, -x, y \in C$ and the fact that C is closed for vector addition, we obtain $\angle(x,u), \angle(u-y,u) \le \delta$. By nearest neighbor condition, $x \in B(y, d(y,u))$. It is easy to see that for all $x \in B(y, d(y,u))$ it holds $\angle(u-x, u-y) \le \pi - \arcsin\frac{1}{\sqrt{d}}$. Therefore

$$\begin{aligned}\angle(x-u,x) &= \pi - \angle(x,u) - \angle(u-x,u)\\ &\ge \pi - \angle(x,u) - \angle(u-x,u-y)\\ &\quad -\angle(u-y,u)\\ &\ge \arcsin\frac{1}{\sqrt{d}} - 2\delta\,.\end{aligned}$$

We chose δ small enough so that $\arcsin\frac{1}{\sqrt{d}} - 2\delta > 0$, so we have proved $0 < \angle(x-u,x) \le \frac{\pi}{2}$.

$$\begin{aligned}d(u,x) &\le |u-x|\\ &= \frac{|u|\sin\angle(x,u)}{\sin\angle(x-u,x)}\\ &\le \frac{|u|\sin\delta}{\sin(\arcsin\frac{1}{\sqrt{d}} - 2\delta)}\\ &\le \frac{|u|\sin(2\delta)}{\sin(\arcsin\frac{1}{\sqrt{d}} - 2\delta)}\\ &\le \frac{|u|}{\sqrt{d}} \qquad \text{(by choice of } \delta\text{)}\\ &\le d(u,0)\,.\end{aligned}$$

□

Lemma 10 *If $d_G(u,v) \le 3$, the nodes added to S_v induce a u-v path of length ≤ 5.*

Proof. Let u', v' be defined as above, and observe that the subroutine cycles through elements of the dense covering Q. Let $q \in Q$ be an element that ε-covers the centroid of $\{u', v'\}$. If $d(u',v') \le \frac{3}{4}$, refer to Figure 1. Since $\varepsilon \le \frac{1}{8}$, $B(q, 1/2)$ contains a neighbor z of u (e.g., $z = u'$). Likewise, $B(q, 1/2)$ contains a neighbor w of v (e.g., $w = v'$). Finally, z and w are neighbors, since they are both in $B(q, \frac{1}{2})$. So w and z as described above can be found, and the nodes $\{u, w, z, v\}$ added to S_v connect u and v.

If $d(u',v') > \frac{3}{4}$, refer to Figure 2.

$$\begin{aligned}&\angle(v'-u', v'-q), \angle(v'-u', q-u')\\ &\quad\le \arcsin(2\varepsilon/|u'v'|)\\ &\quad\le \arcsin(2\varepsilon/d(u',v'))\\ &\quad\le \frac{1}{2}\arctan\left(\frac{1}{d-1}\tan\frac{\delta}{2}\right)\,. \qquad (1)\end{aligned}$$

By Lemma 1, $v' - u'$ is "well inside" some cone C with normals $B \in \mathcal{B}$ and apex at the origin. Combining this with (1), we get that $q - u', v' - q \in C$, i.e., C separates points (u', v'). By Lemma 4, some $I \in \mathcal{I}(B, C)$ separates (u', v'). Furthermore, I is δ-narrow, since all instances in $\mathcal{I}(B)$ are. By Lemma 9, I's solution (x, y) satisfies $d(x, u'), d(y, v') \le d(u', v') \le 1$. It follows that u and x have a common neighbor z (e.g. $z = u'$). Likewise, y and v have a common neighbor w (e.g. $w = v'$). Finally, x and y are neighbors, because $d(x,y) \le d(u',v') \le 1$. So w and z as described above can be found, and the nodes $\{u, w, x, y, z, v\}$ added to S_v connect u and v. □

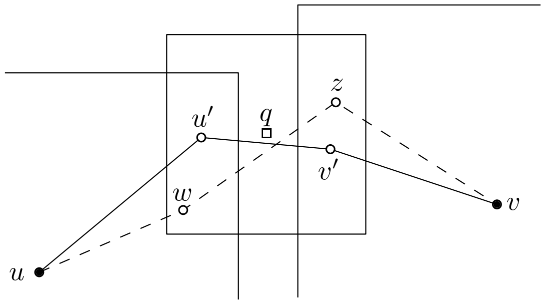


Figure 1: Connecting pairs of nodes in T using paths with a short middle hop.

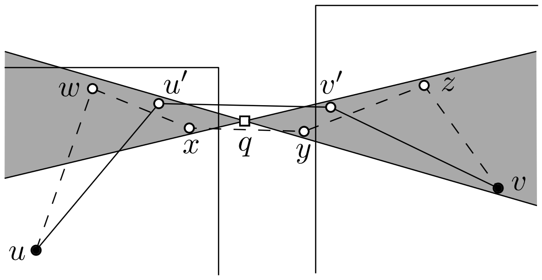


Figure 2: Connecting pairs of nodes in T using paths with a long middle hop.

Lemma 11 *The subroutine runs in $O(\log^{2d} n)$ time.*

Proof. Each range query to find w or z takes $O(\log^d n)$ time. Computation of $\mathcal{I}(B, C)$ takes $O(\log^d n)$ time, by Lemma 4. All other operations take $O(1)$ time. The loops over $u \in U$, $q \in Q$ and $B \in \mathcal{B}$ are repeated $O(1)$ times, and the loop on $I \in \mathcal{I}(B, C)$ is repeated $O(\log^d n)$ times, by Lemma 4. The total running time is therefore $O(\log^{2d} n)$. □

6.3 Initial Construction of Solution

To initially construct the maximal 1-packing T, first set $T = \emptyset$. Let $V = \{v_1, v_2, \ldots, v_n\}$. Then, iteratively for $i = 1, 2, \ldots, n$, query the ORS d.s. on T with $B(v_i, 1)$. If the query returns no nodes, then v_i is not dominated by T, so add v_i to T, and update the ORS d.s. on T. Set $i \leftarrow i + 1$. Once T is fully constructed, for each $v \in T$

call the subroutine in Section 6.2.2 to build the necessary connecting paths. Observe that this procedure to initially construct S requires $\widetilde{O}(n)$ work.

7 Conclusion

In this paper we presented algorithms for maintaining an $O(1)$-approximate MCDS for geometric graphs in $\mathbb{R}^d$ with rectilinear metrics in $\widetilde{O}(1)$ time per node insertion/removal. Open problems include establishing similar bounds for unit-ball graphs in arbitrary ℓ_p norms, improving the approximation ratio to arbitrarily small constants, and designing algorithms for distributed models of computation.

References

[1] Pankaj K. Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete Comput. Geom.*, 6(5):407–422, 1991.

[2] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *MobiHoc*, pages 157–164, 2002.

[3] Julien Basch and Leonidas J. Guibas. Data structures for mobile data. *J. Algorithms*, 31(1):1–28, 1999.

[4] Xiuzhen Cheng, Xiao Huang, Deying Li, Weili Wu, and Ding-Zhu Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003.

[5] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.

[6] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *ICC*, volume 1, pages 376–380, 1997.

[7] David Eppstein. Dynamic euclidean minimum spanning trees and extrema of binary functions. *Discrete & Computational Geometry*, 13(1):111–122, Jan 1995.

[8] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Discrete mobile centers. In *SoCG*, pages 188–196, 2001.

[9] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[10] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[11] John Hershberger. Smooth kinetic maintenance of clusters. In *SoCG*, pages 48–57, 2003.

[12] Erik Jan van Leeuwen. Approximation algorithms for unit disk graphs. In Dieter Kratsch, editor, *WG*, volume 3787 of *Lecture Notes in Computer Science*, pages 351–361. Springer, 2005.

[13] Dan E. Willard and George S. Lueker. Adding range restriction capability to dynamic data structures. *J. ACM*, 32(3):597–617, 1985.

[14] Jie Wu and Hailan Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks.

[15] Zhao Zhang, Xiaofeng Gao, Weili Wu, and Ding-Zhu Du. Ptas for minimum connected dominating set in unit ball graph. In *WASA*, pages 154–161, Berlin, Heidelberg, 2008. Springer-Verlag.

Appendix

Proof of Lemma 1: Consider the cone defined by the set of all positive linear combinations of vectors $v_1, v_2, \ldots, v_d$, where in v_i all coordinates are 1, except for the i-th coordinate which is equal to $1+x$ for some $x > 0$. Let v_0 be the vector with all coordinates equal to 1. Let

$$\alpha = \arctan \frac{x}{(d+x)\sqrt{d-1}}\,, \beta = \arctan((d-1)\tan\alpha)\,.$$

Then one can show by simple calculation that any vector within angle α from v_o is in the cone, and that any vector in the cone is within β from v_0, implying that the cone is 2β-narrow.

Now rotate this cone around the origin enough times so that for any vector v there is a rotated copy of v_0 within $\frac{\alpha}{2}$ from v. There exists a set of $O(\alpha^{-(d-1)})$ copies, computable in $O(\alpha^{-(d-1)})$ time, that satisfies this property. Then, any vector within $\frac{\alpha}{2}$ of v is inside the cone corresponding to this copy of v_0. Pick x so that

$$\alpha = \arctan\left(\frac{1}{d-1}\tan\frac{\delta}{2}\right)\,, \qquad \beta = \frac{\delta}{2}\,.$$

For $\delta \le \frac{\pi}{2}$, we have $\alpha \le \frac{\delta}{2} \le \frac{\pi}{4}$, so

$$\frac{\alpha}{\delta/2} \ge \frac{\frac{\pi}{4}\tan\alpha}{\tan\frac{\delta}{2}} = \frac{\pi}{4(d-1)}\,,$$

and the claim follows by substituting this lower bound for α in the above statement. □

Proof of Lemma 2: (Only if) If $v_1 \in S$, then $S \setminus \{v^{(1)}\} \cup \{v^{(n)}\}$ is another CDS of size no larger than $|S|$, a contradiction. Similarly if $v^{(n)} \in S$. Now suppose there exist such u, v, w. By symmetry, assume $u \in \{v^{(1)}\} \cup S$. Then $T = S \setminus \{v\} \cup \{w\}$ is a CDS. Furthermore, $T \neq S$ and $|T| \leq |S|$, which contradicts minimality and/or uniqueness of S.

(If) Assume for contradiction that S is not a unique MCDS, and let T be a different CDS of size no larger than $|S|$. Since $v_1, v_n \notin S$, points in S divide the interval $[v_1^{(1)}, v_1^{(n)}]$ into exactly $|S| + 1$ intervals of nonzero length. It is easy to see that since $|T| \leq |S|$, one of these intervals has its interior and at least one of its boundary points disjoint from T. Let v and w be the left and right boundary points, respectively, of one such interval. Furthermore, assume that $v \notin T$ (the other case is symmetric). Let u and z be the rightmost (resp. leftmost) element of $\{v^{(1)}, v^{(n)}\} \cup T$ strictly to the left (resp. right) of v. Notice that this is well defined, since $v \notin \{v^{(1)}, v^{(n)}\}$ by assumption. By choice of v, w, we have that z is either w or to the right of it. Also, u and z are adjacent, because T is a CDS (this holds even if $u = v^{(1)}$ or $z = v^{(n)}$). This implies that u, v, w are pairwise adjacent. We finish the proof by noting that $w \in S \subset \{v^{(n)}\} \cup S$. $\square$

Proof of Lemma 4 (sketch): Let q be the apex of given cone C. Desired $O(\log^d n)$ instances are defined by a cones with normals B, i.e., the red and blue points of each instance are exactly the two subsets of input point set P separated from each other by some cone. So to describe the set of instances, we show how to construct their defining cones by using a recursive procedure.

The input to the procedure is a point set (initially input point set P), and a set of hyperplanes H that defines a cone (initially empty). Let b_i be the basis vector which is not a normal of any hyperplane in H such that i is smallest (initially, b_1). Find the input point p with median projection onto b_i. Let h_i be the hyperplane through p with normal b_i, i.e., $h_i = \{x \in \mathbb{R}^d |\ b_i(x - p) = 0\}$. If $q \in h_i^+$, recurse on $P \cap h_i^+$ with H. If $q \in h_i^-$, recurse on $P \cap h_i^-$ with H. In any case, recurse on P with $H \cup \{h_i\}$. If at some level of recursion the input point set is empty, do nothing and simply return. If at some level of recursion there is no valid choice of b_i, then current H, at this point consisting of d hyperplanes, is one of the cones.

Let (x, y) be a pair of input points separated by C. We argue that one of the generated cones indeed separates (x, y). Consider the top level of recursion. Note that if x and y are on the same side of h_1, they have to be on the same side as q. In that case, the first recursive call halves halves the number of points while keeping x and y in the set. Otherwise, the second recursive call increases the number of hyperplanes in H so that x and y are on different sides of each of the hyperplanes. This proves that the procedure eventually outputs a valid cone H that separates (x, y). It is not hard to see that $O(\log^d n)$ recursive calls are made in total.

Recall that we claimed that all these instances are in the d.s. $\mathcal{I}(B)$. This is because the instances in $\mathcal{I}(B)$ are also generated by the above algorithm, except that all three recursive calls are made (since q is unavailable), resulting in $\widetilde{O}(n)$ instances in total. Because of their hierarchical structure, they can be efficiently updated. See [1, 7] for more details. $\square$