Connected Dominating Sets on Dynamic Geometric Graphs[*]

Leonidas Guibas[†] Nikola Milosavljević[‡] Arik Motskin[§]

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 correspond to 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 inserted 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 $\mathbb{R}^1$ (omitted from this abstract), 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 only locally, while a small set of range queries and bichromatic closest pair queries is sufficient to repair the CDS.

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 the 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 fac-

[*]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.

tors) 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 an $O(1)$-approximate MCDS solution with mobile nodes, but a change in topology can require $\Omega(\Delta)$ work in repairs (Δ is the maximum node degree), aggregating to potentially $\Omega(n^3)$ total work over the course of the motion.

2 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 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 $p - x, y - p \in H$.

We will require 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., $\forall x \in \mathbb{R}^d$, $\exists B \in \mathcal{B}$, $\forall y \in \mathbb{R}^d$, $\angle(x, y) \leq \frac{1}{2}\arctan\left(\frac{1}{d-1}\tan\frac{\delta}{2}\right) \Rightarrow b_1y, b_2y, \ldots, b_dy \geq 0$.*

In this paper, we think of a basis as a set of normals to hyperplanes 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.

3 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 CDS output after graph changes.

We 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. Given a basis $B = \{b_1, b_2, \ldots, b_d\}$ of $\mathbb{R}^d$, the d.s. of Agarwal *et al.* [1] maintains set $\mathcal{I}(B)$ of $O(n \log^{d-1} n)$ BCP instances on subsets of P. Each instance is separated by some cone with normals B. The d.s. uses $O(n \log^{2d-1} n)$ space, outputs the solution to an arbitrary instance in $O(1)$ time, and can be updated in time $O(\log^{2d} n)$ per insertion/deletion [7]. The following is its key property.

Lemma 2 *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 $\mathcal{I}(B, C)$.*

The set of all nodes is given by V, while T (defined in Section 4) is a 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.

[2]We prove the approximation bounds for our CDS by showing it is within a constant factor of the MDS.

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

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

By Lemma 1 and property (i), all BCP instances that we maintain are δ-narrow.

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

We first observe that maintaining the optimal MCDS 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.

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.*

In the rest of the section we describe an $O(1)$-approximate MCDS with $\widetilde{O}(1)$ update time.

4.1 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 1 $S = \bigcup_{v \in T} S_v$. *Clearly, $S \supset T$.*

Lemma 4 *If G is connected, S is a CDS for G.*

Lemma 5 *If G is connected, S is an $O(1)$-approximate MCDS in G.*

4.2 Dynamic Maintenance

In this section, assume that a CDS S as in Definition 1 already exists; we show how to repair the maximal independent set T and connectors $\{S_v\}_{v \in T}$ according to Definition 1 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 Definition 1. If v was removed from T in step (B), remove S_v from S. Any node w added to T in step (B) (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 v can only affect the correctness of S_z if $z \in B(v,4)$. Hence, to repair all connectors according to Definition 1, it suffices to recompute S_w for all $w \in T \cap B(v,4)$, using the subroutine in Section 4.2.2.

4.2.1 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 3, and assuming $O(\log^{2d} n)$ runtime for the subroutine (Lemma 9 in Section 4.2.2), the entire procedure takes $O(\log^{2d} n)$ time per operation. We conclude the following.

Theorem 6 *The $O(1)$-approximate MCDS S can be maintained in $O(\log^{2d} n)$ time per operation.*

4.2.2 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. To prove correctness of output S_v, it suffices to prove that the procedure succeeds when $d_G(u,v) \leq 3$ (Lemma 8).

The strategy is to identify "waypoints": intermediate nodes on the connecting paths. We know that the waypoints are contained in $B(v,3)$, so we pick a dense (but still constant-sized) sampling Q of this ball around v, comprising points of the space (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 $d_G(u,v) = 2$). Nodes 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 u' and v', but still suitable as waypoints: we still manage to connect u and v via these nodes.

Lemma 7 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 base $B \in \mathcal{B}$, 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 $O(\log^d n)$ BCP instances drawn from $\mathcal{I}(B)$, each of which can be solved in time $O(1)$ (since they are in $\mathcal{I}(B)$), and one of which we know (by Lemma 2) outputs a good solution in the sense of Lemma 7. 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 3. 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 for any of these, w and z are found, we have found a 3-hop connecting path $\{u, w, z, v\}$; add it to S_v. Otherwise, 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 2), 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 7 *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 $d(u,x), d(y,v) \leq d(u,v)$.*

Lemma 8 *If $d_G(u,v) \leq 3$, the nodes added to S_v induce a u-v path of length ≤ 5.*

Lemma 9 *The subroutine runs in $O(\log^{2d} n)$ time.*

5 Conclusion

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] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Disc. Comp. Geom.*, 6(5):407–422, 1991.

[2] K. M. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *MobiHoc*, pages 157–164, 2002.

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

[4] X. Cheng, X. Huang, D. Li, W. Wu, and D.-Z. Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003.

[5] B. Clark, C. Colbourn, and D. Johnson. Unit disk graphs. *Disc. 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] D. Eppstein. Dynamic euclidean minimum spanning trees and extrema of binary functions. *Disc. Comp. Geom.*, 13(1):111–122, Jan 1995.

[8] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *SoCG*, pages 188–196, 2001.

[9] M. Garey and D. 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] J. Hershberger. Smooth kinetic maintenance of clusters. In *SoCG*, pages 48–57, 2003.

[12] E. J. van Leeuwen. Approximation algorithms for unit disk graphs. In *WG*, pages 351–361, 2005.

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

[14] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIALM*, pages 7–14, 1999.

[15] Z. Zhang, X. Gao, W. Wu, and D.-Z. Du. PTAS for minimum connected dominating set in unit ball graph. In *WASA*, pages 154–161, 2008.