Approximation Algorithms for Network Connectivity

Owen Levin

Submitted under the supervision of Volkan Isler and Victor Reiner to the University Honors Program at the University of Minnesota-Twin Cities in partial fulfillment of the requirements for the degree of Bachelor Science, *summa cum laude* in Mathematics.

May 11, 2018

Abstract

Herein we discuss previous work on the Euclidean Connectivity Problem with the objective of minimizing maximum movement. We provide new algorithms which both in approximation factor and empirically outperform the state-of-the-art.

1 Introduction

While working with multiple mobile robots or sensors for data collection in an unknown environment, it is frequently beneficial to have the robots exchange and combine information collected. Quite often, mobile sensing robots have some method or instrument allowing long range communication to other robots. However, generally bandwidth constraints prevent the sharing of sensor data such as photos, audio or videos through these methods. To share this sort of data, the communication range is typically short, so the robots must move closer together in order to stably transmit sensor data to one another. Despite this, long-range communication methods are sufficient for small messages such as GPS-coordinates to be broadcast between robots.

The problem we study in this thesis addresses the issue of connectivity: how should the robots travel when they are far away from each other and they need to form a connected network as quickly as possible. We model this problem by minimizing the furthest distance travelled to reach a connected network. With the assumption that all robots move with the same speed, this corresponds to connecting robots in the shortest time.

We say a robot can communicate with another when they are within each other's connectivity radius r. For a connected communication network, the robots' positions must induce a connected graph G, where the edge lengths are the pairwise Euclidean distances between nodes, and no edge length is greater than r. Here, a connected network of robots allows for transmission of a message through intermediary robots. Let the r-disk graph $G(\mathcal{P}, r)$ on a finite set of points $\mathcal{P} \subset \mathbb{R}^d$ be the graph with vertex set $\mathcal{P}$ and edges between all points $p, q \in \mathcal{P}$ such that $|p - q| \leq r$. Since we can always normalize distance units so that the radius for connectivity is 1. Then a configuration of robots $\mathcal{P}$ is called connected exactly when $G(\mathcal{P}, 1)$ is connected.

The remainder of this thesis is organized as follows. In Section 2 we give an exact formulation of the problem we wish to solve. Afterward in Section 3 we describe some motivation for studying the problem and previous

work in the area. In Section 4 an algorithm by Anari et al. from their paper [AFGS16] using a homothety is described and we discuss how its analysis can be thought of in terms of Euclidean minimal spanning trees (EMST). We then describe an algorithm of our own more directly tied to EMSTs to approximate solutions to the Euclidean Connectivity Problem in Section 5. Along with this, we provide an analysis of the worst-case approximation factors and give a heuristic to improve the performance of both our algorithm and the Homothety algorithm in [AFGS16]. Afterward another new algorithm is proposed and analyzed which greedily connects $\mathcal{P}$ in Section 6. We conclude with an analysis of the average case approximation factor bound for each algorithm on uniformly distributed points in a square in Section 7 and a brief discussion of the run-time of each algorithm in Section 8.

2 Problem Formulation

We consider a set $\mathcal{P}$ of n **p**oints in Euclidean space $\mathbb{R}^d$. Now, given another point set $\mathcal{P}' \subset \mathbb{R}^d$ with the cardinality of $\mathcal{P}$ denoted $|P'| = n$, let

$$f^*_{\mathcal{P}'} = \underset{\text{bijective } f:\mathcal{P}\to\mathcal{P}'}{\operatorname{argmin}} \max_{p\in\mathcal{P}} |p - f(p)| \tag{1}$$

where $|p-f(p)|$ denotes the length of the vector $p-f(p)$ and argmin denotes the argument f that minimizes the quantity (as opposed to the minimum quantity itself). Informally, $f^*_{\mathcal{P}'}$ is the bijective mapping that minimizes the maximum distance travelled between points in $\mathcal{P}$ and $\mathcal{P}'$.

Then formally, the Euclidean Connectivity Problem is to find $\mathcal{P}' \subset \mathbb{R}^d$ with $|\mathcal{P}'| = n$

$$\underset{\mathcal{P}' \text{ is connected}}{\operatorname{argmin}} \max_{p\in\mathcal{P}} |p - f^*_{\mathcal{P}'}(p)| \tag{2}$$

along with its associated $f^*_{\mathcal{P}'}$. We will be working under the assumption that all connectivity radii are the same length. So we note again that we may scale the distance units for our computations such that the radius for connectivity is 1. Thus $\mathcal{P}'$ connected will always mean that the unit disk graph $G(\mathcal{P}', 1)$ is connected.

For the $\mathcal{P}'$ optimizing Equation 2, we will denote f^* by A_{opt} for the **opt**imal **a**lgorithm, and we will let $\max_{p\in\mathcal{P}} |p - A_{opt}(p)|$ be denoted by OPT for the maximum movement of the **opt**imal solution. In general, we will let

$\mathcal{P}_f$ denote the **f**inal **p**oint configuration after some (potentially non-optimal) algorithm translates points in $\mathcal{P}$ to a connected configuration.

We say that an algorithm with maximum movement M has **approximation factor** of α if $M/OPT = \alpha$. Typically, α is some function of the number of points, $\alpha(n)$. If $\lim_{n\to\infty} \alpha(n) \leq cg(n)$ for some function $g(n)$ and positive constant c, we say that the approximation factor of that algorithm is $O(g(n))$.

Anari et al. proved in [AFGS16] that there is no polynomial-time algorithm for the Euclidean Connectivity Problem with an approximation factor of less than $\left(2 - \frac{\sqrt{2}}{2}\right)$ unless $P = NP$. This was done via a reduction from the Euclidean Connectivity Problem to a variant of the Hamiltonian cycle problem on 3-regular planar graphs.

3 Motivation and Related Work

A large body of networking work studies algorithms dealing with accomplishing tasks using a connected network of mobile robots or sensors. It is a general assumption that the network is already connected in these scenarios, and the goal is to perform the task without disconnecting the network as in the work of Stump, Kumar, Jadbabaie, and Zavlanos et al. in [ZP08, ZEP11, ZTJP09, SJK08]. In [AAY07], Abasi et al. studied fault tolerance and repairing connectivity: if a point is removed from a connected $\mathcal{P}$, how can we move points to reconnect the network with minimum maximum movement. For each of these bodies of work to apply, we require a pre-constructed connected $\mathcal{P}$.

3.1 Discrete Connectivity Problem

The connectivity problem given an initially disconnected $\mathcal{P}$ was originally introduced by Demaine et al. in the discrete setting. In that context, points in $\mathcal{P}$ are a subset of vertices of a weighted graph G and may only move along the edges of G to form a subgraph H with highest weight edge at most the connectivity radius. With the optimization objective of minimizing the maximum graph distance travelled, an approximation factor of $O(\sqrt{n})$ was achieved.

The authors also proved an approximation factor of $O(n\sqrt{n})$ for points placed anywhere in $\mathbb{R}^2$[DHM$^+$09a]. To do so they use a scaled integer lattice as their graph, then prove the approximation factor to be proportional to $\sqrt{n}(1 + d(n + 1)\sqrt{2})$ for some constant $d \in \mathbb{R}$

Since then, new algorithms reducing the discrete case approximation factor to $O(1)$ were studied by Demaine, Hajiaghayi, and Marx, as well as Berman, Demaine, and Zadimoghaddam in [DHM09b, BDZ11] along with many related variants forming some other network topology and optimizing with respect to quantities beside minimization of maximum movement. Using the same method as in [DHM$^+$09a], these more recent results induce an approximation factor which is linear in $n = |\mathcal{P}'|$, but with a very large constant coefficient in the Euclidean case.

4 Euclidean Connectivity Problem

More often than not, in real scenarios robots are not constrained to move along edges of some graph. They may move essentially freely through Euclidean space. However, once we consider robots free to roam Euclidean space, algorithms consisting of movements restricted to be within some graph may no longer be as effective compared to optimal Euclidean movements. This is largely the motivation behind directly studying the Euclidean Connectivity Problem.

In the literature, apart from the work in [BDZ11, DHM09b, DHM$^+$09a] where the Euclidean case was secondary to the main problems addressed, algorithms for the Euclidean Connectivity Problem have only been discussed a few times.

The one dimensional Euclidean case was analyzed by Sharma in his master's thesis [Sha14]. In Sharma's work, an optimal algorithm was given for points lying on a single line by essentially dynamic programming. The morally correct idea here is that the maximum movement is minimized by finding the midpoint on the line between the two most distant points and centering the new configuration there. The resulting connected configuration is then essentially fully determined by the connectivity radius. We do have to be careful to make sure points do not travel if they are already connected and may not have to move.

Unfortunately, as soon as we increase the number of dimensions this

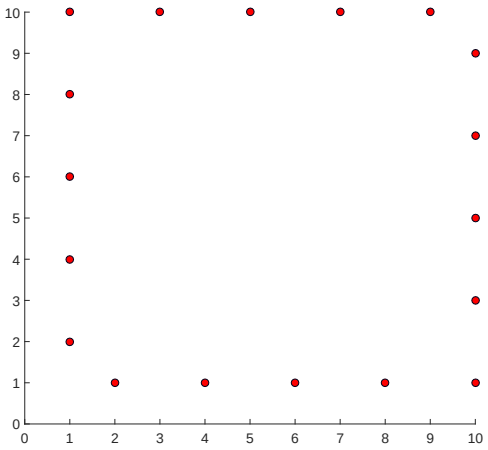

Figure 1: Here we see a configuration which is disconnected in $\mathbb{R}^2$, but connected along projections to either axis

does not generalize well. The primary issue at hand is that many disconnected configurations in $\mathbb{R}^d$ are connected when projected onto lines. In such cases, it's not even always clear how one should apply the one dimensional algorithm to a set of points. One possibility is to use a technique like principal component analysis (PCA) and use the coordinate frame of the largest eigenvectors of the covariance matrix of the set of points. This will give the dimensions along which the points have highest variance, so perhaps connecting along these axes will always give a connected configuration. See e.g. Chapter 12 of [Bis06] for more details on PCA. Still, it's not immediately clear that connecting along just these dimensions will result in a connected configuration in $\mathbb{R}^d$.

The author has not spent the time to find a truly pathological configuration where PCA then d iterations of the $\mathbb{R}^1$ optimal solution fail to connect the configuration at all. However, we suspect that for a random point cloud in $\mathbb{R}^d$, very often when given enough points, the configuration will look connected in probably any projection into a 1 dimensional euclidean subspace, but will be disconnected in R^d. In Figure 1, we give a more structured example of a box in $\mathbb{R}^2$ where along the x axis and the y axis projections, the configuration is connected yet still disconnected in $\mathbb{R}^2$. Beyond the one dimensional case, Sharma also discussed optimal solutions in $\mathbb{R}^2$ for when $|\mathcal{P}| \leq 4$ in [Sha14]. Rather, Sharma described geometric configurations that appear to be optimal, then claimed they were optimal. It was then concluded

that the general problem was hard.

There has been some discussion of algorithms for the $\mathbb{R}^2$ case of the Euclidean Connectivity problem in [LWZ+15, DMM13], though these works have differing constraints on the problem. In [LWZ+15], the total movement of all points is minimized, and in [DMM13] the connectivity problem requires using a set of fixed points and connecting all other points to them. The latter work also gives no theoretical approximation factors.

Anari et al. were the first to discuss the Euclidean connectivity problem in general with a theoretical bound on their approximation factor for $\mathbb{R}^d$ in [AFGS16]. As far as the author of this thesis is aware, Anari et al. are the only authors that have studied the theoretical side of the Euclidean connectivity problem beyond the corollaries mentioned in analyses of the discrete cases in [BDZ11, DHM09b, DHM+09a].

4.1 Connecting points with a Homothety

The idea behind the algorithm of Anari et al. was to use the affine transformation known as a homothety. A homothety is determined by a choice of center, c and a scale factor λ. Each point $p \in \mathbb{R}^d$ is scaled by a factor of λ along the ray from c to p

Explicitly, a homothety $h_{c,\lambda} : \mathbb{R}^d \to \mathbb{R}^d$ is given by $p \overset{h_{c,\lambda}}{\mapsto} c + \lambda(p - c)$.

The following is particularly useful property of homotheties which was used by Anari et al.

Lemma 4.1. *For positive λ, $h_{c,\lambda}$ scales all Euclidean lengths by λ.*

Proof. To see this fact, note that because $h_{c,\lambda}$ sends p to $c + \lambda(p - c)$ we have that

$$\begin{aligned} |h_{c,\lambda}(x) - h_{c,\lambda}(y)| &= |c + \lambda x - \lambda c - (c + \lambda y - \lambda c)| && (3)\\ &= |\lambda x - \lambda y| && (4)\\ &= |\lambda(x - y)| && (5)\\ &= |\lambda| \cdot |x - y| && (6)\\ &= \lambda\,|x - y| \text{ because } \lambda \text{ is positive} && (7) \end{aligned}$$

where Equation 6 follows from Equation 4 because the Euclidean metric on $\mathbb{R}^d$ is absolutely homogeneous. □

Corollary 4.2. *For $0 \leq \lambda \leq 1$, the distance any point p travels under the map $h_{c,\lambda}$ is given by $|h_{c,\lambda}(p) - p| = (1-\lambda)|p-c|$.*

Proof. The initial distance between the point and the center is $|p-c|$. The distance after applying $h_{c,\lambda}$ is $\lambda|p-c|$ by Lemma 4.1. It is also the case that $h_{c,\lambda}(p)$ is colinear with both c and p and lies between the two since $0 \leq \lambda \leq 1$. Thus, $|p - h_{c,\lambda}(p)| = |p-c| - \lambda|p-c| = (1-\lambda)|p-c|$ □

With these properties established, we may describe and analyze the algorithm given by Anari et al. in [AFGS16] to send a $\mathcal{P}$ to a $\mathcal{P}$ whose unit disk graph is connected.

Algorithm 1 HOMOTHETY-CONNECT
Applies a homothety, h, to $\mathcal{P}$ such that $G(h(\mathcal{P}), 1)$ is connected

1: Find the smallest R such that the R-disk graph of the $\mathcal{P}$ is connected.
2: Fix an arbitrary point $c \in \mathcal{P}$
3: Move each point $p \in \mathcal{P}$ to $h_{c,\frac{1}{R}}(p)$

Lemma 4.1 ensures that $\lambda = \frac{1}{R}$ is the largest choice λ such that the unit disk graph of the $\mathcal{P}$ is connected after Algorithm 1. Considering Corollary 4.2, we see this is the least movement a homothety with center c can produce while resulting in a connected unit disk graph. Of course, the maximum movement depends largely on c, but c is always picked to be in $\mathcal{P}$. One could often lower the maximum movement of Algorithm 1 if $c \notin \mathcal{P}$. However, Anari et al. did not consider such c, and their analysis of the approximation factor would not necessarily apply to $c \notin \mathcal{P}$.

Theorem 4.3. *([AFGS16]) There is an $O(n)$-factor approximation algorithm for the Euclidean Connectivity Problem using Algorithm 1, where $n = |\mathcal{P}|$*

To show this, one can analyze the maximum movement of Algorithm 1 by looking at paths in the Euclidean minimum spanning tree of $\mathcal{P}$. While this is not the machinery that Anari et al. used to prove Theorem 4.3, it ends up being roughly equivalent to their analysis and opens doors to a more direct algorithm.

4.2 Relation to trees

To analyze the run-time using trees, we must first build up some tools to use. We begin with a Lemma about disk graphs:

Lemma 4.4. *For any $R \geq 2OPT+1$ the R-disk graph $G(\mathcal{P}, R)$ is connected.*

Proof. We begin by considering the preimages $p, q \in \mathcal{P}$ of two points $A_{opt}(p), A_{opt}(q)$ which are adjacent in $G(A_{opt}(\mathcal{P}), 1)$. Then by the triangle inequality, we have

$$|p - q| \leq |p - A_{opt}(p)| + |A_{opt}(p) - A_{opt}(q)| + |A_{opt}(q) - q| \tag{8}$$
$$\leq |p - A_{opt}(p)| + 1 + |A_{opt}(q) - q|. \tag{9}$$

Recalling that $|p - A_{opt}(p)| \leq OPT$ for all $p \in \mathcal{P}$. The expression in (9) is less than or equal to $OPT+1+OPT = 2 \cdot OPT+1$. Thus, $G(\mathcal{P}, 2OPT+1)$ is connected. Then for $R > 2OPT + 1$, all edges in $G(\mathcal{P}, 2OPT + 1)$ are of length less than R. Thus, $G(\mathcal{P}, R)$ contains a connected subgraph, namely $G(\mathcal{P}, 2OPT + 1)$, which forces it to be connected. □

Corollary 4.5. *R_c, The minimum R such that $G(\mathcal{P}, R)$ is connected satisfies $R_c \leq 2OPT + 1$, which gives a lower bound on OPT: $\frac{R_c-1}{2} \leq OPT$*

Recall that a minimum spanning tree (MST) of an edge-weighted graph G is a connected acyclic subgraph that minimizes the sum of its edge-weights while spanning the vertices of G.

Proposition 4.6 (basis exchange)**.** *A useful property of spanning trees is that given two spanning trees T, T' of a graph G, for any edge $e \in T$, there exists an edge $f \in T'$ such that $T'' = (T - \{e\}) \cup f$ is a spanning tree.*

Now, if we let $K(\mathcal{P})$ be the complete graph on vertices in $\mathcal{P}$ with edges between $p, q \in \mathcal{P}$ weighted by $|p - q|$. Then define a Eucidean minimum spanning tree (EMST) of $\mathcal{P}$ to be a minimum spanning tree of $K(\mathcal{P})$. Then we have the following corollary of Lemma 4.4

Corollary 4.7. *The Euclidean minimum spanning tree of $\mathcal{P}$ with longest edge length R satisfies $R \leq 2OPT + 1$.*

Proof. If T is a minimum spanning tree of $K(\mathcal{P})$ with longest edge length R, then T is a subgraph of $G(\mathcal{P}, R)$. If $R > 2OPT+1$, then $G(\mathcal{P}, 2OPT+1)$ is a connected subgraph of $G(\mathcal{P}, R)$ by Lemma 4.4. Because $G(\mathcal{P}, 2OPT + 1)$ is connected, it has a spanning tree T' with longest edge length at most $2OPT + 1$. By Proposition 4.6, any edges longer than $2OPT + 1$ in T could then be replaced with edges in T' to form another spanning tree T''. Since

edges in T' were shorter than the edges they replaced in T, it must be that the sum of edge lengths in T'' is less than those of T. This contradicts the minimality of T. Thus $R \leq 2OPT + 1$. □

After stating a few more facts about trees, we can discuss the linear approximation factor of Algorithm 1. Let the **center(s)** C of T be the unique vertex or pair of vertices such that the length of the longest path in T from C to another vertex of T is minimized. That such a vertex or pair of vertices exist can be proved via induction on the number of vertices in a tree, and is done for example by Knuth in [Knu97] We define the depth of T to be the number of edges in longest path from C in T.

The diameter of T is the number of edges in the longest path in T. We define the depth as the length of the longest path in T with C as an endpoint. It's a simple exercise to show that C always lies on the longest path in T, and the diameter of T is at most twice the depth. A proof of this fact can be given by induction on the number of vertices in T by checking it for the case of 3 or fewer vertices. Then for the induction step, delete all of the degree 1 vertices (*leaves*) to reduce to a smaller case which will have the same center. For more details, see e.g. the solution of Grinberg to an assignment in his graph theory course [Gri17, Exercise 1].

Now note that any tree T on n vertices has $n-1$ edges because if it had more, then there would be a cycle, and if there were fewer, then T could not be connected. Thus, the depth of T is at most $n/2$. That corresponds to the center of any tree which is just a single path through all n vertices.

Given all that, consider translating each point $\mathcal{P}$ along paths in an EMST, T, of $\mathcal{P}$ toward the center. Anari et al. found the worst case approximation factor of Algorithm 1 to be less than or equal to $2(n-1)$ where $n = |\mathcal{P}|$. They *essentially* did this by bounding the movement in Algorithm 1 using the triangle inequality to moving all the points roughly along T. The primary quantity of interest was the "Excess" contributed by each edge e, $E_e < OPT$, as points moved travelled along e in T. In the worst case, $p \in \mathcal{P}$ moved at most $2E_e < 2OPT$ for each edge in the path between p and c in T.

In Algorithm 1, suppose we pick the homothety center $c \in \mathcal{P}$ for $h_{c,\lambda}$ to be C, the center of T. Then each point moves along at most $n/2$ edges, and thus the total Excess is less than $2OPT \cdot n/2 = n \cdot OPT$. Despite improving the worst case bound by a factor of 2, this was never actually pointed out in [AFGS16]. This is likely because both worst case bounds are linear in n, so it doesn't affect the $O(n)$ factor mentioned in Theorem 4.3.

5 The MST Connectivity Algorithm

It may seem roundabout to analyze the movement of the homothety using an EMST. If one wanted to analyze the algorithm with an EMST, why not just write an algorithm that uses an EMST? In Algorithm 2, we present an algorithm that does exactly that. The general idea of Algorithm 2 is to compute the center of the EMST of $\mathcal{P}$, then iteratively "shrink" some edges toward the center of the tree.

The protocol for our algorithm begins with computing $\mathcal{T}_\mathcal{P}$, the EMST $\mathcal{P}$. Once completed, we can find the center of $\mathcal{T}_\mathcal{P}$, either c or $\{c_1, c_2\}$ depending on the parity of the tree diameter (length of the longest path in $\mathcal{T}_\mathcal{P}$). After computing the center, it is possible there are two center vertices c_1, c_2 which are further apart than the connectivity radius r. In this case, remove the edge $e_{1,2}$ between c_1, c_2 to split $\mathcal{T}_\mathcal{P}$ into two connected components T_1, T_2 containing c_1, c_2 respectively. Let m be the midpoint of $e_{1,2}$ and $(m - c_i)$ be the vector between c_i and m. Then move every point in T_i along the vector $\frac{m-c_i}{|m-c_i|}(|m - c_i| - 1/2)$. After this process, $e_{1,2}$ will have length 1, so our center vertices are connected to one another.

Let S be the set of points guaranteed to have $G(S, 1)$ be connected. Once a point x is in S, the algorithm will never move x again. At this point, S contains only the center of $\mathcal{T}_\mathcal{P}$. Now, for each edge e_{si} incident to S with endpoints $x_s \in S, x_i \notin S$ incident to x_s, $\mathcal{T}_\mathcal{P}$ is split along e_{si} into T_s and T_i containing x_s, x_i respectively. if e_{si} was longer than 1, then every node in T_i is then moved according to the vector from x_i to x_s, but rescaled so that after movement x_i is a distance 1 away from x_s. After the movement, x_i is added to S. Iteratively repeating the above process eventually leads to S containing every point and thus all of the points being connected. We should note, that it would be silly to actually move all points exactly according to the above algorithm. Instead we just use it to compute the final locations of the points, then move each point directly to its final location. Nevertheless in our analysis, we treat each point as moving exactly as the algorithm prescribes.

The exact body of the algorithm is presented in Algorithm 2 and Algorithm 3 and a diagram is given in Figure 2. Let G be a graph on vertex set $\mathcal{P}$ with $S \subseteq \mathcal{P}$. We will use the notation $G|_S$ read G restricted to S to mean the induced subgraph of G using only nodes in S.

Algorithm 2 MST-Connect

Input: $\mathcal{P}$: a set of points where $G(\mathcal{P}, 1)$ is not connected
Output: $\mathcal{P}_f$: a final set of points where $G(\mathcal{P}_f, 1)$ is connected

$\mathcal{T}_\mathcal{P} \leftarrow$ MST of $K(\mathcal{P})$
$C \leftarrow$ the center of $T_\mathcal{P}$
if $C = \{c_1, c_2\}$ and $|c_1 - c_2| > 1$ **then**
    Apply homothety $h_{(c_1+c_2)/2,\ 1/2|c_1-c_2|}$ to $\{c_1, c_2\}$.
(This moves both c_1 and c_2 toward $\overline{c_1c_2}$'s midpoint until $|c_1 - c_2| = 1$.)
    Move other points accordingly as in Contract-To-Center
    $\mathcal{P}' \leftarrow$ the new configuration after these movements
else
    $\mathcal{P}' \leftarrow \mathcal{P}$
$S \leftarrow C$
while $S \neq \mathcal{P}$ **do**
    $\mathcal{P}' \leftarrow$ Contract-To-Center$(\mathcal{T}_\mathcal{P}, \mathcal{P}', S, r)$
    $S \leftarrow S \cup \{p' \in \mathcal{P}'$ moved from $p \in \mathcal{P} - S$ adjacent to $\mathcal{T}_\mathcal{P}|_S$'s leaves in step 11$\}$
$\mathcal{P}_f \leftarrow \mathcal{P}'$
return $\mathcal{P}_f$

Algorithm 3 Contract-To-Center (a helper for MST-Connect)

Input: T: A tree with vertex set $\mathcal{P}$, $\mathcal{P}$: locations, $S \subseteq \mathcal{P}$: set of stationary central nodes attracting outside neighbors
Output: $\mathcal{P}'$: points after contractions are performed

for all leaves q in $T|_S$ **do**
    for all points $p \in \mathcal{P} - S$ adjacent to q in T **do**
        if $|p - q| > 1$ **then**
            Comp$(p) \leftarrow$ the connected component of $T|_{\mathcal{P}-S}$ containing p
            $\vec{d} \leftarrow q - p$
            $\vec{d} \leftarrow \frac{\vec{d}}{|\vec{d}|} \cdot (|\vec{d}| - 1)$
            for all points x in Comp(p) **do**
                $x' \leftarrow x + \vec{d}$
return $\mathcal{P}' = (\{x'\}_{x \in \mathcal{P}-S}) \cup S)$

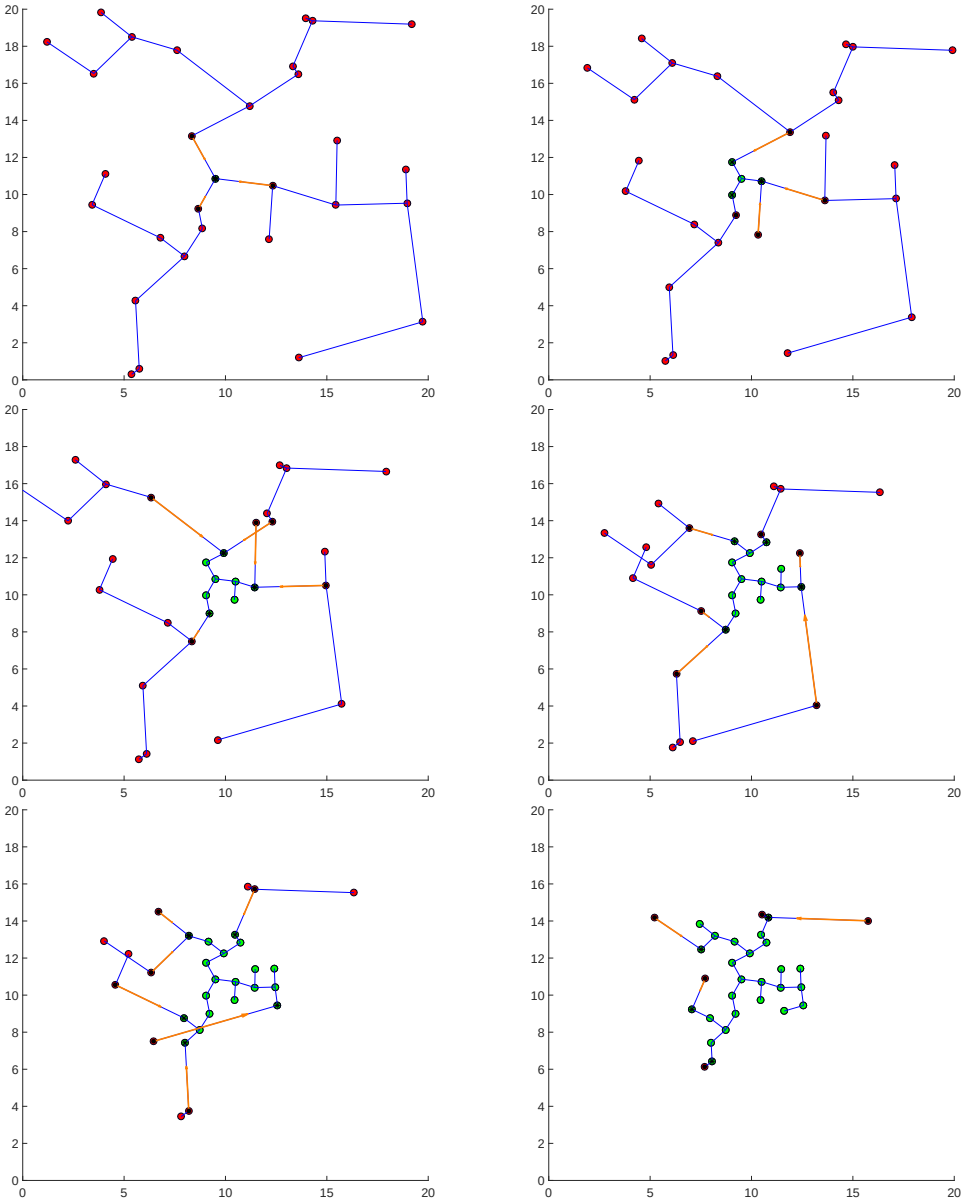

Figure 2: Here we see how the EMST (blue edges) of $\mathcal{P}$ (red points) is transformed into a (not necessarily minimal) spanning tree of $\mathcal{P}_f$ by MST-Connect. The green points are in S. The orange segments are the vectors along which connected components of $\mathcal{T}|_{\mathcal{P}'-S}$ are translated.
Top-Left- iteration 1, Top-Right- iteration 2,Mid-Left- iteration 3, Mid-Right- iteration 4, Bottom-Left- iteration 5, Bottom-Right- iteration 6 (final).

5.1 MST Algorithm Worst-Case Analysis

Let D denote the depth of an EMST of $\mathcal{P}$ and let the maximum movement of a solution produced by Algorithm 2 be denoted SOL.

Lemma 5.1. *Then we have* $SOL \leq 2D \cdot OPT$

Proof. Let R be the length of the longest edge in $\mathcal{T}_{\mathcal{P}}$. All nodes move along paths toward the center in $\mathcal{T}_{\mathcal{P}}$ during Algorithm 2. The path distance to the center is at most D. Since an edge is only contracted if it is longer than 1 and the final length is equal to 1, the very most a node can travel during the contraction of any given edge is $R-1$. Thus $D \cdot (R-1)$ is an upper bound on SOL. Applying Lemma 4.4, $SOL \leq D \cdot (2OPT + 1 - 1) = 2D \cdot OPT$ □

Theorem 5.2. *There is an $O(D)$-factor approximation algorithm for the Euclidean Connectivity Problem using Algorithm 2, where D is the depth of an EMST of $\mathcal{P}$.*

A few remarks: In the worst case, the EMST is a path, so the worst case bound is exactly the same as the worst case of Algorithm 1 when the homothety center is optimal over points in $\mathcal{P}$. However, Algorithm 1 performs a homothety which uniformly scales the tree toward a center, even edges which need not be scaled to form a connected disk graph. Algorithm 2 only rescales edges which are long enough that the result would be disconnected connected otherwise. Moreover, instead of scaling each edge based on the longest edge, Algorithm 2 rescales each edge the minimal amount necessary to leave endpoints connected in a unit disk graph. The result is that Algorithm 2 results in a final point configurations where points are spread further apart than after Algorithm 1. Due to this, one might expect the maximum movement of Algorithm 2 to be less than that of Algorithm 1. In the later discussion of average case performances, we shall see that this is typically the case, and is seen for example in Figure 3.

5.2 Heuristic improvements

One may also object that the center of the EMST need not be anywhere near the middle of the point configuration. In fact, it can even be in a corner of the smallest box that bounds the points. In such a situation, we can at least halve the maximum movement to the final point configuration of Algorithms 1 and 2 with a simple translation. This would roughly happen

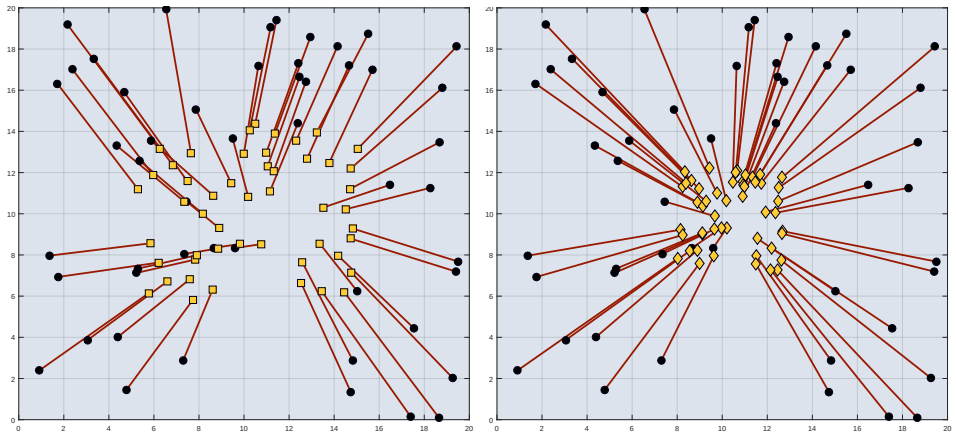

Figure 3: Here we see the movements (maroon) produced by MST-Connect (Left) and Homothety-Connect (Right)

if one of the two algorithms sends a point $p \in \mathcal{P}$ initially in one corner of the bounding box to the opposite corner.

We could for example translate a final point set $\mathcal{P}_f$ to the center of the smallest enclosing circle of $\mathcal{P}$, denoted Center($\mathcal{P}$). Doing so would cause p to only travel to Center($\mathcal{P}$). Instead of travelling potentially across the entire diameter of the smallest enclosing circle, now p is sent, at most, across the radius. This can as much as halve the movement. More typically, points don't have to travel all the way to opposite corners since they aren't usually on the edge of the box in the first place. Then this heuristic is even more of an improvement than halving the maximum movement.

For the example given in Figures 2 and 3, this improvement makes a minimal difference since the center of the tree is roughly in the middle of the environment anyway. A motivating example to keep in mind for why mean or median might be bad choices over Center($\mathcal{P}$) is when $\mathcal{P}$ consists of a large cluster of points far apart from a single outlier. Then the maximum movement is proportional to the separation of the two clusters (and is the distance the outlier point is translated). The mean and medians both tend to be within the cluster, so translating the final point set to those would not fix the situation. The smallest enclosing circle of the points will move both points from the cluster and the outlier toward the midpoint of some line between them. This is another case where the distance is roughly halved. From here on, we will assume this heuristic is applied.

Even with such improvements, the worst case of both algorithms is still a linear factor away from OPT. To see this, consider the set of $2n$ points along two parallel lines in $\mathbb{R}^2$ given by $\mathcal{P} = \{(1, 2i)\}_{i=1}^{n} \cup \{(3, 2i + 1)\}_{i=1}^{n}$. For any n, these points can be connected with maximum movement 1 by setting the y coordinate to 2 for each point. However, even after the heuristic improvement above, both Algorithms 1 and 2 are linear in the number of points. To see this, note that the minimum spanning tree with the least depth has depth about $n/4$. All but one of the edges in the longest paths are colinear with length 2, and one edge is a translation and possible reflection over the y axis of the vector $(1, 2)^\top$ which has length $\sqrt{5}$.

The actual movement performed by MST-Connect and essentially performed by Homothety-Connect is the hypotenuse of the triangle formed by [half the concatenation of all the colinear edges of some path to the center + 1] and [vertical edge of length 1/2]. That is, the maximum movement of is $\sqrt{(n/4 + 1)^2 + 1/4} \approx n/4$. Since OPT = 1, We have an approximation factor of $n/4$ which is linear. Algorithm 1 has (only slightly) larger maximum movement, since instead of scaling each of the colinear edges by 1/2, all edges are scaled by $1/\sqrt{5}$. See Figure 5.

It's perhaps now intuitive that in order to have better than a linear approximation factor, we cannot just rescale the initial configuration. It must be broken up some how. Moreover, we want an algorithm that will prevent a distant outlier from being pulled all the way to a cluster as in the example previously mentioned. In Section 6 we will discuss a greedy algorithm that does exactly that.

6 A Greedy Connectivity Algorithm

Here we propose a greedy algorithm for $\mathcal{P} \subseteq \mathbb{R}^2$ that computes the smallest enclosing circle (SEC) of $\mathcal{P}$, then moves the closest point to that circle's midpoint Center($\mathcal{P}$). That point is then considered part of the connected set. After this initial movement, the closest pair of points with one in the connected set and one outside of the connected set are computed. The point outside the connected set is moved toward the connected set until it is a distance 1 away, and then is considered part of the connected set. This process is repeated until all robots are connected. Algorithm 4 contains the details.

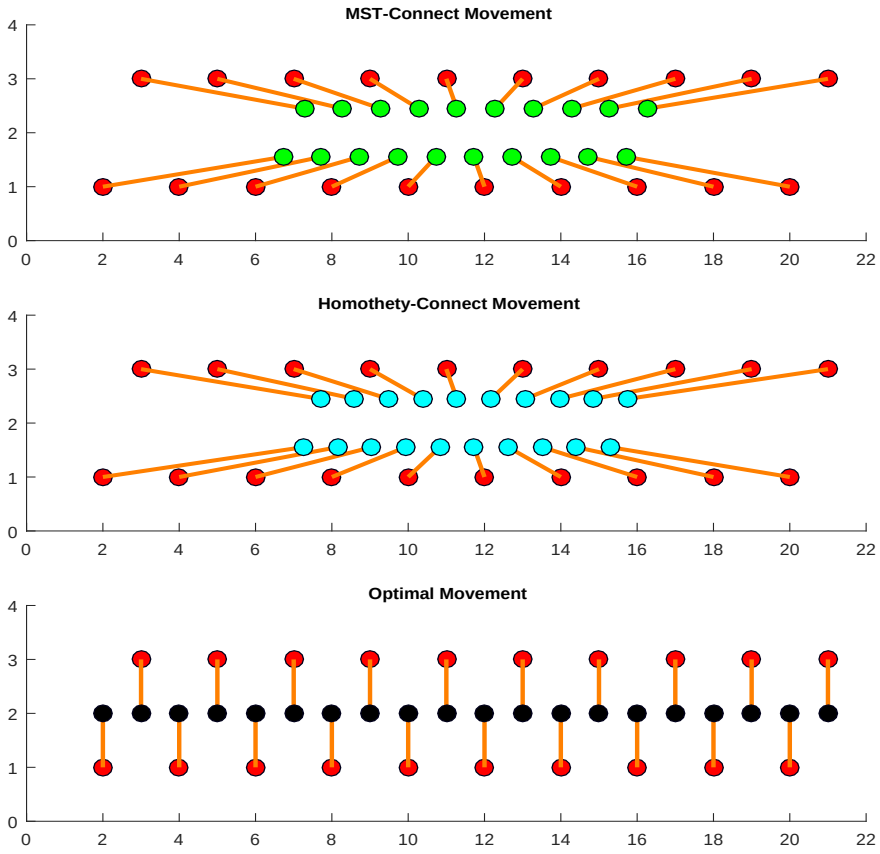

Figure 4: Here we see a point configuration (red) which results in very poor performance of both Algorithm 1 (cyan final configuration) and Algorithm 2 (green final configuration). Both have maximum movement $\sim (n/4) \cdot OPT$. The optimal solution is shown in black.

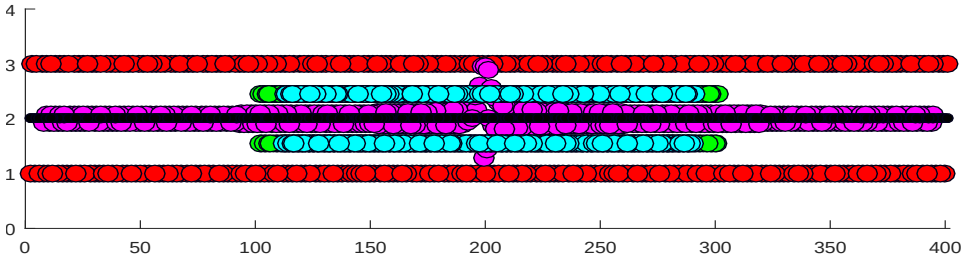

Figure 5: The parallel line configuration with final configurations of Algorithm 1 (cyan), Algorithm 2 (green), Algorithm 4 (pink), and A_{OPT}, the optimal algorithm (black).

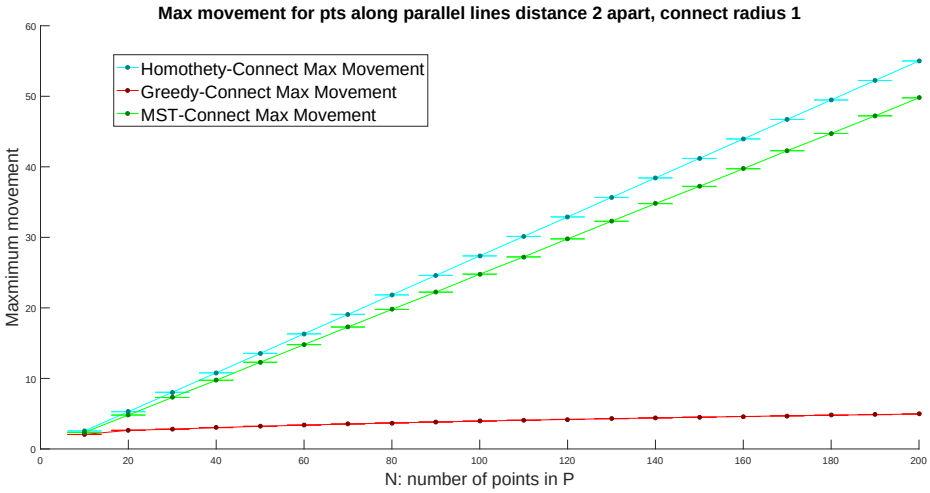

Figure 6: The performance on the parallel line configuration of Algorithm 1 (cyan), Algorithm 2 (green), and Algorithm 4 (red)

Algorithm 4 GREEDY-CONNECT

Input: $\mathcal{P}$: a set of points where $G(\mathcal{P}, 1)$ is not connected
Output: $\mathcal{P}_f$: a set of points where $G(\mathcal{P}_f, 1)$ is connected

Center($\mathcal{P}$) $\leftarrow$ The midpoint of the SEC of the nodes.
$p_c \leftarrow$ The closest robot in $\mathcal{P}$ to Center(P)
$p'_c \leftarrow c$
$S \leftarrow \{p'_c\}$
while $|S| < |\mathcal{P}|$ **do**
    $(p', q) \leftarrow$The closest pair of point with $p' \in S$ and $q \in \mathcal{P} - S$
    $\vec{d} \leftarrow \frac{p'-q}{|p'-q|} \cdot \max\{0, |p' - q| - 1\}$
    $q' \leftarrow q + \vec{d}$
    $S \leftarrow S \cup \{q'\}$
return S

(a) $\mathcal{P}$ and Center($\mathcal{P}$)

(b) $\mathcal{P}_f$ and movements

Figure 7: The blue and red dots represent $\mathcal{P}$, and $\mathcal{P}_f$ respectively. The red cross in (a) is Center($\mathcal{P}$). Algorithm 4 moves points along the black arrows to form the network in (b).

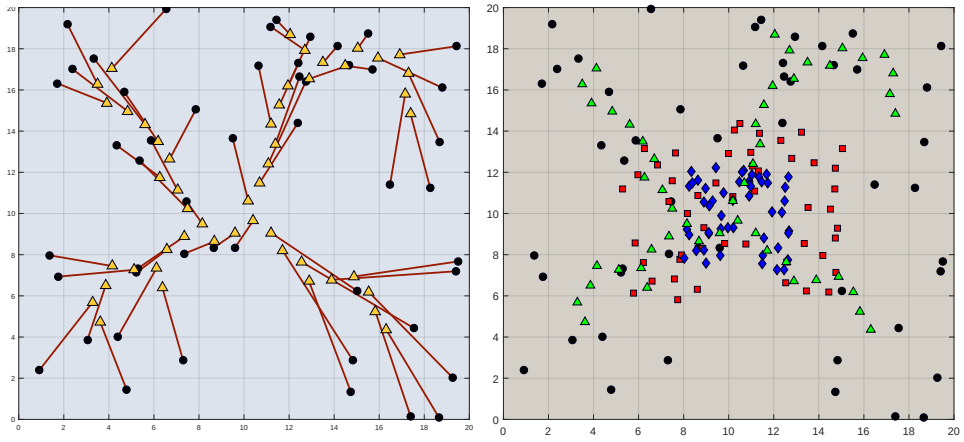

Figure 8: For the same $\mathcal{P}$ (in black) as in Figure 3, on the left we show the Greedy $\mathcal{P}_f$ (gold triangles) and greedy movement (left image in maroon). On the right we have a comparison between Greedy $\mathcal{P}_f$ (green triangles), MST-Connect ($\mathcal{P}_f$ red squares) and Homothety-Connect ($\mathcal{P}_f$ blue diamonds).

6.1 Greedy Algorithm Worst-Case Analysis

To bound the maximum movement of Algorithm 4, SOL_G, we note that the maximum possible movement would be if a node moved from somewhere on the convex hull of the robots to the midpoint Center($\mathcal{P}$). If we let d_c be the distance from the furthest point on the convex hull $\mathcal{P}$ to Center($\mathcal{P}$), it must be the case that $SOL \leq d_c$. In general, this forced movement can be arbitrarily bad. However, we can bound some of the cases where it performs worst and even prove constant factor approximations.

Suppose that we have n points, and the furthest two points are a distance ρ apart. Drager et al. proved in [DLM07] that $2d_c$, the diameter of the SEC cannot be greater than $2\rho/\sqrt{3}$. That is, $SOL_G \leq \rho/\sqrt{3}$. If $\rho > n + 1$, then the least distance the two points most distance points can need to travel would occur in the case where all other points were colinear with the two points, and each were a distance 1 apart so they were connected, centered on the midpoint of the line between the two most distant points. This yields a lower bound on OPT of $\frac{1}{2}(\rho - n - 1)$. Since SOL_G is bounded above and OPT is bounded below, we have an upper bound on the approximation factor α_G.

Theorem 6.1. *Let ρ be the distance between the furthest pair of points in $\mathcal{P}$. If $\rho > |\mathcal{P}|+1$, then the approximation factor for Algorithm 4 is less than* $\frac{2\rho}{\sqrt{3}(\rho-n-1)}$

To be sure, as $\rho \to n+1$ the approximation factor can become arbitrarily bad. This makes sense given the forced movement to the center in the algorithm. On the other hand, if there are points in between the furthest two points, then the maximum movement of Algorithm 4 will tend to be far less than Theorem 6.1 suggests. Essentially the bound is a constraint on the density of points in the domain (the SEC of $\mathcal{P}$). In $\mathbb{R}^d$ where $d > 2$, we use the smallest enclosing sphere and must remove the $\frac{1}{\sqrt{3}}$ factor in the bound, but have the same result. There is still some constant between 1 and 2, that could replace $\sqrt{3}$ but we are unsure of its exact value.

If the points are very dense (ρ very small compared to n), then minimal movement is needed to connect them and the forced movement to Center($\mathcal{P}$) can be very bad. If the environment is very sparse $\rho \gg n+1$, then moving a point to the middle might not be so bad, at least compared to the optimal movement which will also be large. Given some target approximation factor, and either n or ρ we can solve for constraints on the free variable to guarantee performance at least as good as the target approximation factor. So for sparse enough environments, we can guarantee an $O(1)$ approximation factor solution. For the parallel lines configuration with $2n$ points, ρ is $2\sqrt{1+n^2}$ which is just barely less than $2n+1$, so we cannot bound the approximation factor for this case. That said, since we know OPT is 1, we can numerically compute the approximation factor of Algorithm 4 and it appears to grow slightly quicker than, but $\ln(n)$ though remain smaller than $\ln(n)$ until $n > 300$.

As mentioned before, the worst cases for Algorithm 4 are when a point is moved all the way to the center, but minimal movement is needed to actually connect the configuration. One example of such a configuration is when $\mathcal{P}$ consists of n points equally spaced on the smallest enclosing circle with radius d_c and midpoint Center($\mathcal{P}$). Here, we can give better constraints on the approximation factor than Theorem 6.1. In this case, a homothety centered at Center($\mathcal{P}$) is actually optimal, while $SOL_G = d_c$ and comes from the first movement forcing a point to the center of the configuration. With geometric arguments, one can see that the distance between any two adjacent points on the circle will be $2d_c \sin(2\pi/2n)$. Then the scale factor of the homothety performing the optimal movement will scale this to 1. So $h_{\text{Center}(\mathcal{P}),\frac{1}{2d_c\sin(\pi/n)}}$ will be the optimal movement.

If $\sin(\pi/n) > \frac{1}{2d_c}$ we can apply Corollary 4.2 and the maximum movement of this homothety is $OPT = \left(1 - \frac{1}{2d_c \sin(\pi/n)}\right) d_c$. If the $\sin(\pi/n) \leq \frac{1}{2d_c}$, then $\mathcal{P}$ already been connected, so OPT is 0 and we would not move anything. Since the maximum movement of Algorithm 4 is d_c for this configuration.

Theorem 6.2. *When The points $\mathcal{P}$ all lie spaced equally on a circle and $\sin(\pi/n) > \frac{1}{2d_c}$, the worst-case approximation factor of Algorithm 4 is*

$$d_c/OPT = \frac{1}{1 - \frac{1}{2d_c \sin(\pi/n)}}$$

Since $\frac{1}{2d_c \sin(\pi/n)}$ ranges between 0^+ and 1, the approximation factor can be arbitrarily bad between 1 and $+\infty$ in general. The situation is not as dire as it may seem though. The converse of also holds and states that given any constant target approximation factor and some fixed $n = |\mathcal{P}|$, we have bounds on how small d_c is allowed to be (i.e. we can bound the size of the environment $\mathcal{P}$ can live in if we want to guarantee good performance.). We also have bounds on the allowed size of $\mathcal{P}$ given some fixed target approximation factor and environment size.

Theorem 6.3. *Let SOL_G denote the maximum movement of Algorithm 4. If $\mathcal{P}$ is the set of n equally spaced points around a circle of radius d_c, Then given a target constant approximation factor $\alpha > 1$, if $d_c > \frac{\alpha}{(\alpha-1)2\sin(\pi/n)}$, we have that*

$$SOL_G \leq d_c < \alpha \cdot OPT$$

.

Proof. Fix $|\mathcal{P}| = n$, and target approximation factor bound $\alpha > 1$ Then we

have from Theorem 6.2 that

$$\alpha > \frac{1}{1 - \frac{1}{2d_c \sin(\pi/n)}} \tag{10}$$

$$\alpha > \frac{1}{\frac{2d_c \sin(\pi/n) - 1}{2d_c \sin(\pi/n)}} \text{ (Common denominator)} \tag{11}$$

$$\alpha > \frac{2d_c \sin(\pi/n)}{2d_c \sin(\pi/n) - 1} \tag{12}$$

$$(2d_c \sin(\pi/n) - 1)\alpha > 2d_c \sin(\pi/n) \tag{13}$$

$$2d_c \sin(\pi/n)\alpha - \alpha > 2d_c \sin(\pi/n) \tag{14}$$

$$d_c(2 \sin(\pi/n)(\alpha - 1) - \alpha > 0 \tag{15}$$

$$d_c(2 \sin(\pi/n)(\alpha - 1) > \alpha \tag{16}$$

$$d_c > \frac{\alpha}{(\alpha - 1)2 \sin(\pi/n)} \tag{17}$$

□

A similar proof gives a bound on the number of points allowed.

Theorem 6.4. *Let SOL_G denote the maximum movement of Algorithm 4. Given some radius d_c and target constant approximation factor $\alpha > 1$, if $\mathcal{P}$ is the set of equally spaced points on a circle of radius d_c, and $|\mathcal{P}| < \frac{\pi}{\arcsin\left(\frac{\alpha}{2d_c(\alpha-1)}\right)}$, then $SOL_G \leq d_c < \alpha \cdot OPT$*

The optimal configuration of equally spaced points on higher dimensional spheres is unknown. Even solving this on just the 2 sphere in $\mathbb{R}^3$ is the seventh of Steve Smale's famous open problems [Sma98]. Due to this, we do not have a similar worst case bound for $\mathbb{R}^d$ with $d > 2$.

Still, in general for $\mathbb{R}^2$ we can give constraints on $\mathcal{P}$ to force very good approximation factors. However, if we have no constraints on $\mathcal{P}$, then we have a much looser bound. Recall that $G(\mathcal{P}, R)$ is the disk graph with connectivity radius R. In order to compare ρ to OPT, we let R_c be the minimum radius such that $G(\mathcal{P}, R_c)$ is connected. Since R_c is the longest edge length in $G(\mathcal{P}, R_c)$ by construction, $\rho \leq D_G \cdot R_c$, where $D_G := \text{diameter}(G(\mathcal{P}, R_c))$ is the maximum path length ranging over all the shortest paths between pairs of vertices in $G(\mathcal{P}, R_c)$. This follows from repeated applications of the triangle inequality to path segments of the graph's diameter. Now, an approximation factor for Algorithm 4 can be given using our bound on R_c in terms of OPT from Corollary 4.5.

To do so, we will find λ such that $1 \leq \lambda OPT$ yields a bound on the worst case approximation factor. Let $\lambda \in \mathbb{R}$ be such that $1 = \lambda \frac{R_c - 1}{2}$. Then $\lambda = \frac{2}{R_c - 1}$. Note that by Corollary 4.5, we have $1 \leq \lambda OPT$. This yields the following bound.

$$D_G \cdot R_c \leq D_G(2 + \lambda)OPT \tag{18}$$

$$= D(2 + \frac{2}{R_c - 1})OPT \tag{19}$$

$$= \frac{2R_c}{R_c - 1} D_G \cdot OPT. \tag{20}$$

Because R_c grows with the size of the environment, one can loosely think of $\frac{R_c}{R_c - 1}$ as a measure of how large the ambient space is in the following sense. When $R_c - 1$ is small, the fraction blows up which is indicative of how the forced movement to the center may be a very poor choice if the configuration is already nearly connected.

Theorem 6.5. *Using Algorithm 4, Let R_c be the smallest radius such that $G(\mathcal{P}, R_c)$ is connected and let D_G be the diameter of $G(\mathcal{P}, R_c)$. Then $SOL_G \leq \frac{2R_c}{R_c - 1} \frac{D_G}{\sqrt{3}} \cdot OPT$*

Proof. SOL_G cannot be larger than d_c. Note that $d_c \leq \frac{\rho}{\sqrt{3}} \leq D_G \cdot R_c / \sqrt{3}$, thus we have the result. □

For an arbitrary configuration, D_G can be as large as n when it's a path, but then our earlier result gives us a constant factor approximation if the points are not initially connected. Computing the actual bound on the approximation factor requires knowing the actual point configuration. Let's compute this bound to the parallel line configuration: $R_c = \sqrt{5}$, and $D_G = n + 1$. Unfortunately this bound is not only linear, but it's eight times worse than the bound on Algorithms 1 and 2. We can see in Figure 6, that this is very loose bound. A big reason for that is that Algorithm 4 never actually has to move its most distant point to the center of the configuration unless all points are on the circle, but in that case we showed how to calculate the approximation factor exactly.

More often than not, points will move only a short distance to the center, and in a very dense environment (where we have only the loose bound from Theorem 6.5), there is likely to be a point on or near the center already, points nearby everywhere between the center and the furthest point. It just so happens that OPT can be arbitrarily small in these cases. However in

these dense environments, since $\mathcal{P}$ is nearly connected anyway, even a large approximation factor is meaningless in applications as it takes minimal time no matter which of the algorithm is used.

It may be theoretically useful to run some sort of hybrid algorithm that given $\mathcal{P}$ with $|\mathcal{P}| = n$, computes bounds on the approximation factor and runs another algorithm if they're too high, but otherwise runs Algorithm 4 This guarantees a worst case approximation factor of n and gives a potentially even constant approximation factor if it's lower. Figure 9 demonstrates that in practice, this is not always necessary.

7 Average case Analyses for $\mathbb{R}^2$

With applications in mind, a set of robots that we might wish to connect may not be in some special configuration such as spaced on a circle or parallel lines. More often they will be distributed somewhat randomly before we work to connect them. This might motivate us to study the average case performance of connectivity algorithms with some distribution applied to our point set. Here we will focus on uniformly distributed points as that often models real life situations.

To analyze the average case for uniform random deployment, we assume that points in $\mathcal{P}$ are uniformly distributed in an $[0, L] \times [0, L]$ where a unit length is once again the connectivity radius. Then, we may use properties of Random Geometric Graphs (RGGs) modeling the uniformly distributed robot locations in order to bound R_c and eventually the expected value of a solution's maximum movement $E[SOL]$ for various algorithms. A RGG $\mathbf{G}(n, R, L)$ is the R-disk graph of n points uniformly distributed in $[0, L]^2$ We are restricting ourselves to the $\mathbb{R}^2$ case because very little relevant explorations random geometric graphs have been carried out for the case of $d > 2$. Since most of our analyses hinge upon these results, we have little to say until more is known about RGG in higher dimensions.

We define R_{min} to be the minimum radius above which $\mathbf{G}(n, R_{min}, L)$ is asymptotically almost surely connected as L grows large. Muthukrishnan and Pandurangan proved that such a threshold exists and gave the exact value $R_{min} \leq L\sqrt{\frac{2 \ln L}{n}}$ with equality as L goes to infinity [MP10]. To do so, they used a bin-covering argument where they bounded the largest number of boxes needed to cover space such that overlapping bins allowed for con-

nected points. Similar bounds (though with completely different arguments from bin covering) were given previously by Penrose in [Pen97] and by Gupta and Kumar in [GK99] as n goes to infinity. The sharp threshold for connectivity and other sharp thresholds for other properties of RGGs were given by Goel, Rai, and Krishnamachari in [GRK05]. A sharp threshold x_{crit} for property Prop(x) depending on x satisfies for all $\varepsilon > 0$ Prop($x_{crit} + \varepsilon$) is almost surely true, and if Prop($x_{crit} - \varepsilon$) is almost surely false. Bounds on R_{min} were later generalized to various other non-uniformly distributed sets of points by Bettstetter in [Bet04].

It follows that $E[R_c]$, the expected length of the longest edge of $G(\mathcal{P}, R_c)$, must be less than R_{min}. If this were not the case, then we could expect an edge shorter than R_c being the longest edge necessary to have a connected RGG.

7.1 Homothety-Connect and MST-Connect Average Cases

The bound on the approximation factor for these algorithms both depend on the depth of the minimum spanning tree of the set of points. Recall Theorem 5.2 which gave the approximation factor of MST-Connect as $O(D)$ where D is the depth of the minimum spanning tree of $K(\mathcal{P})$. For various distributions of points, one may be able to bound the expected depth of the minimum spanning tree far below the worst case bound of $n/2$.

Unfortunately, it seems that the literature has not yet found better bounds for D than $D \in \omega(\sqrt{n})$ and $D \in o(n)$ which mean respectively that

$$\lim_{n\to\infty} \frac{D}{\sqrt{n}} = +\infty$$

and

$$\lim_{n\to\infty} \frac{D}{n} = 0.$$

Although, simulations seem to suggest that the expected value of D is about $n^{2/3}$ asymptotically for uniformly distributed points in a square environment. This would suggest an average case bound of $2n^{2/3}OPT$ which is strictly better than the worst case for $n > 10$.

In [SSE87] it was proven that the ratio of leaves, and more generally vertices of any fixed degree, to nodes in a random euclidean minimal spanning tree converges to a constant. While the constant seems unable to be solved for analytically, Monte Carlo simulations suggest that for $n > 100$

(the maximum value simulated was $n = 2^{16}$) the ratio of leaves to nodes is 0.221. For $n < 100$ the ratio of leaves was always larger on average, and in particular when n was small, the ratio grew larger. Since only two leaves can be contained in the longest path, we can bound the average case D above also by the ratio of non-leaves to nodes. This allows for us to bound the expected value of D above by

$$D \leq \frac{1}{2}\left(\frac{779}{1000}n + 2\right).$$

Together this gives the average case upper bound $E[SOL] \leq (0.779n + 2)OPT$. For large n, $E[SOL]$ appears to be approximately $O(n^{2/3})OPT$, but there's no proof at this point.

7.2 Greedy-Connect Average Case

Given Theorem 6.5, we know that SOL_G is at most $\frac{D_G R_c}{\sqrt{3}}$, and the average value of R_c is less than $L\sqrt{\frac{2\ln L}{n}}$. Therefore, $E[SOL_G]$ is at most $\sqrt{\frac{2}{3}}\frac{D_G L\sqrt{\ln L}}{\sqrt{n}}$ if $\mathcal{P}$ is uniformly distributed in $[0, L]^2$.

While this is a more accurate bound, it does not illuminate how the solution might approximate the optimal. We shall bound the expected approximation factor for algorithm 4 by bounding ρ in terms of L directly.

First we note that ρ cannot be larger than $L\sqrt{2}$, so $d_c \leq \rho/\sqrt{3} \leq \sqrt{2/3}L$. Multiplying this expression by $1 = \sqrt{\frac{\ln(L)}{n}}\sqrt{\frac{n}{\ln(L)}}$ yields $\sqrt{2/3}L\sqrt{\frac{\ln(L)}{n}}\sqrt{\frac{n}{\ln(L)}}$ Substituting our bounds for R_c yields $\frac{1}{\sqrt{3}}\sqrt{\frac{n}{\ln(L)}}\frac{2R_c}{R_c - 1}OPT$. With another R_c substitution, we have that $SOL_G \leq \sqrt{\frac{8}{3}}\frac{L}{R_c - 1}OPT$.

To substitute for the R_c in the denominator, we utilize a lower bound on R_{min} which is also proved by Muthukrishnan and Pandurangan in [MP10]. In particular, it is shown that as $L \to \infty$ then asymptotically almost surely we have $G(\mathcal{P}, R)$ disconnected if $R < L\sqrt{\frac{c_0 \ln(L)}{n}}$ for $0 < c_0 \leq \frac{1}{2}$ with c_0 decreasing to zero as L grows large. While this result is asymptotic in L, testing over 1000000 trials with $c_0 = 1/5$, when L as small as 100, the configuration was disconnected 99.2837 percent of the time. Given these results, we can expect R_c to be larger than $L\sqrt{\frac{\ln(L)}{5n}}$.

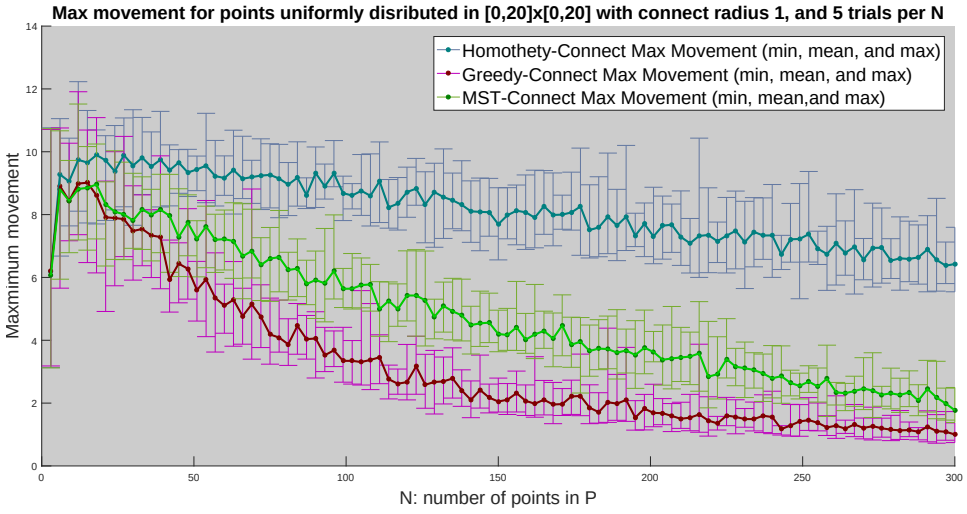

Figure 9: Some empirical trials comparing the performance of Algorithms 1, 2, and 4 on uniformly distributed points in $[0, 20]^2$.

By substituting the lower bound for R_c into the denominator and normalizing terms we arrive at the following bound.

Theorem 7.1. *If* $n < L^2 \ln L$, *then* $E[SOL_G] \leq 2\sqrt{\frac{10}{3}} \frac{L\sqrt{n}}{L\sqrt{\ln L} - \sqrt{n}} OPT$

We can see in Theorem 7.1 that if $L \gg n$ the solution is expected to be an $O(\sqrt{n/\ln L})$ factor off of the optimal solution.

If $n \gg L^2 \ln L$ it would seem that we get a negative approximation factor. In such a situation, the configuration is expected to already be connected so no algorithm would run in the first place based on the asymptotic R_{min} bounds. The heuristic picture to keep in mind be that $\mathcal{P} = \{$lattice points of $[0, L]^2\}$ is already connected, and we have at least $\ln(L) > 1$ copies of those points distributed uniformly in the box.

8 Run-time Analyses

Because the motivation for these problems is to minimize the time robots spend connecting to one another so that they can maximize their battery lives, it is important to consider the computational efficiency of these algo-

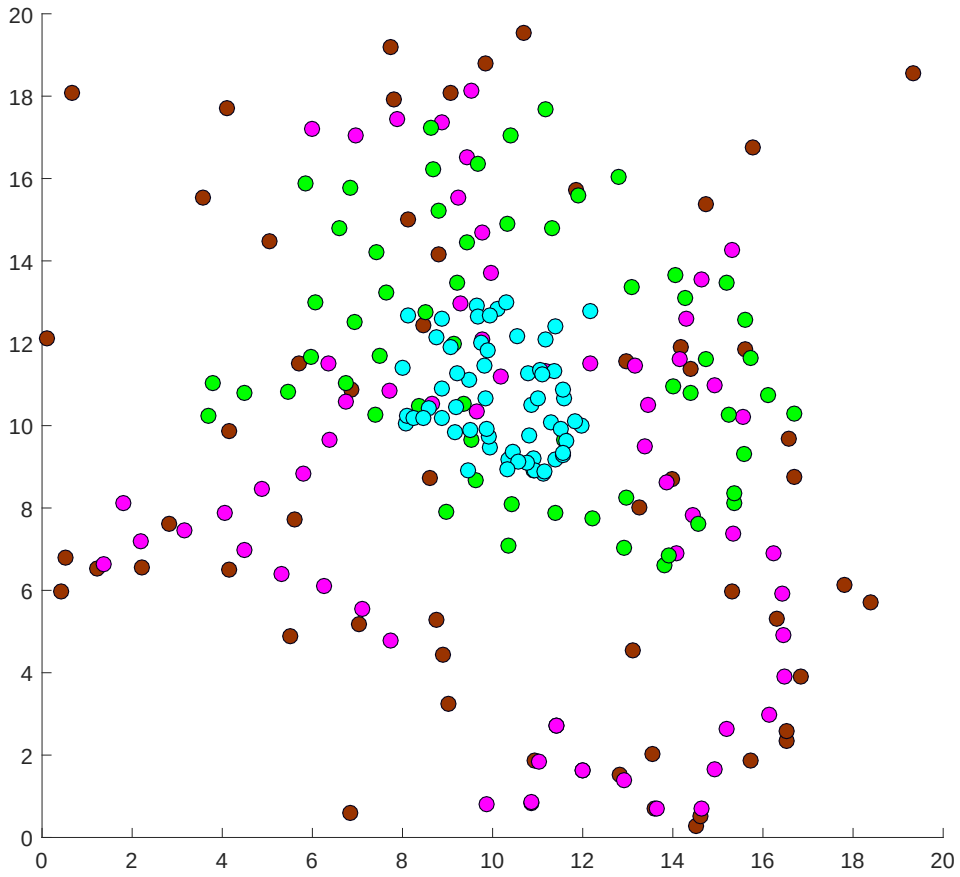

Figure 10: Another example showing $\mathcal{P}$ in red, Algorithms 1's $\mathcal{P}_f$ in cyan, Algorithm 2's $\mathcal{P}_f$ in green, and Algorithm 4's $\mathcal{P}_f$ in pink on 60 uniformly distributed points in $[0, 20]^2$. The maximum movements were Homothety-Connect: 9.2060, MST-Connect: 7.5985, and Greedy-Connect: 5.8779

rithms' run-times.

8.1 Run-time Analysis of Homothety-Connect

Let $|\mathcal{P}| = n$. Then for the heuristically optimized HOMOTHETY-CONNECT where $\mathcal{P}_f$ is given by $h_{\text{Center}\,\mathcal{P},\frac{1}{R_c}}(\mathcal{P})$, the run-time has primary contributions coming from computing the scale factor of the homothety as well as computing the smallest enclosing circle of $\mathcal{P}$ and $\mathcal{P}'$. Megiddo proved in [Meg82] that the smallest enclosing circle can be computed in $O(n)$ time. The method can be generalized to compute the smallest enclosing sphere in higher dimensions still in linear time. Finding the scale factor for the homothety is equivalent to finding the length of the longest edge R_c of the EMST of $\mathcal{P}$. This can be done with Kruskal's algorithm for computing minimum spanning trees in $O(\binom{n}{2}\log n)$ [Kru56] time after computing the $\binom{n}{2}$ pairwise distances of points in $\mathcal{P}$ which are the graph's edge weights. From here we can compute a run-time bound on HOMOTHETY-CONNECT.

Proposition 8.1. HOMOTHETY-CONNECT *has* $O(n^2 \log n)$ *run-time.*

8.2 Run-time Analysis of MST-Connect

Let $|\mathcal{P}| = n$. Consider the heuristically optimized MST-CONNECT where $\mathcal{P}_f$ is given by MST-CONNECT translated such that $\text{Center}(\mathcal{P}_f) = \text{Center}(\mathcal{P})$. The run-time has primary contributions coming from computing the EMST $\mathcal{T}_\mathcal{P}$ of $\mathcal{P}$, computing the center of $\mathcal{T}_\mathcal{P}$, then performing the movements of the connected components of $\mathcal{T}_\mathcal{P}|_{\mathcal{P}-S}$, and finally computing the smallest enclosing circles of $\mathcal{P}$ and $\mathcal{P}_f$. Computing the tree takes $O(n^2 \log n)$ time and the smallest enclosing circle takes $O(n)$ time as we saw previously. Computing the center of the tree can be done in $O(n)$ time by iteratively removing all leaves until only the center or centers are left. At each stage of movement, we have $n - |S|$ points moving, and this D times where D is the depth of $\mathcal{T}_\mathcal{P}$. Since $D \leq n/2$, all of the movements are done in $O(n^2)$ time. From here we can compute a run-time bound on MST-CONNECT.

Proposition 8.2. MST-CONNECT *has* $O(n^2 \log n)$ *run-time.*

8.3 Run-time Analysis of Greedy-Connect

Let $|\mathcal{P}| = n$. Consider Greedy-Connect. The run-time has primary contributions coming from computing the smallest enclosing circle, as well as n iterations of computing the closest pair of points between two sets of points. Computing the smallest enclosing circle takes $O(n)$ time as we saw previously. The time to compute the nearest pair of points from two sets of size k and $n-k$ with brute force takes requires $k(n-k)$ comparisons. The total runtime of doing this for each $k \in \{1, 2, \ldots, n\}$ is $\sum_{k=1}^{n-1} k(n-k)$ which is $O(n^3)$ comparisons. From here we can compute a run-time bound on Greedy-Connect.

Proposition 8.3. Greedy-Connect *has* $O(n^3)$ *run-time.*

At least in robotic applications, these algorithms are rarely used for large enough n such that n^3 is slow enough to be a constraint. Since this is the worst run-time of all three algorithms, and n is typically not so large, the run-time is a non-issue in real applications. In fact, one can usually even run all three algorithms quickly and just choose the best solution. This prevents pathological Greedy-Connect cases using the worst case linear bound from MST-Connect.

9 Conclusion

In this thesis, we have provided two new algorithms for the maximum movement minimization problem for Euclidean connectivity. We have given bounds on $\mathbb{R}^d$ case for both algorithms, as well as better bounds in $\mathbb{R}^2$ for both worst case for the greedy algorithm and average cases for both algorithms. Our algorithms empirically outperform the state-of-the-art, but there is room for improvement. The gap between our approximation factor bounds and the theoretically possible constant factor approximation algorithms is large. We hope to find better constraints on $\mathcal{P}$ or algorithms guaranteeing constant factor approximations in the future. The trickiest aspect of this project has been bounding seemingly good solutions. There are so many algorithms with multiple variants and heuristics for each one. Comparing even the best of them to an optimal solution has in practice been extremely difficult. Not all the algorithms we explored made it into this work for that very reason. Hopefully future work on this problem will allow us to find better bounds.

References

[AAY07] Ameer Ahmed Abbasi, Kemal Akkaya, and Mohamed Younis. A distributed connectivity restoration algorithm in wireless sensor and actor networks. In *Local computer networks, 2007. LCN 2007. 32nd IEEE conference on*, pages 496–503. IEEE, 2007.

[AFGS16] Nima Anari, Mohammad Amin Fazli, Mohammad Ghodsi, and Mohammad Ali Safari. Euclidean movement minimization. *J. Comb. Optim.*, 32(2):354–367, 2016.

[BDZ11] Piotr Berman, Erik Demaine, and Morteza Zadimoghaddam. O (1)-approximations for maximum movement problems. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 62–74, 2011.

[Bet04] Christian Bettstetter. On the connectivity of ad hoc networks. *The computer journal*, 47(4):432–447, 2004.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[DHM+09a] Erik D Demaine, MohammadTaghi Hajiaghayi, Hamid Mahini, Amin S Sayedi-Roshkhar, Shayan Oveisgharan, and Morteza Zadimoghaddam. Minimizing movement. *ACM Transactions on Algorithms (TALG)*, 5(3):30, 2009.

[DHM09b] Erik D Demaine, MohammadTaghi Hajiaghayi, and Dániel Marx. Minimizing movement: Fixed-parameter tractability. In *ESA*, pages 718–729. Springer, 2009.

[DLM07] Lance D. Drager, Jeffrey M. Lee, and Clyde F. Martin. On the geometry of the smallest circle enclosing a finite set of points. *Journal of the Franklin Institute*, 344(7):929 – 940, 2007.

[DMM13] R. Dai, J. Maximoff, and M. Mesbahi. Optimal trajectory generation for establishing connectivity in proximity networks. *IEEE Transactions on Aerospace and Electronic Systems*, 49(3):1968–1981, July 2013.

[GK99] Piyush Gupta and P. R. Kumar. *Critical Power for Asymptotic Connectivity in Wireless Networks*, pages 547–566. Birkhäuser Boston, Boston, MA, 1999.

[Gri17] Darij Grinberg. Math 5707 homework 3 solutions. online, March 2017. `http://www-users.math.umn.edu/~dgrinber/5707s17/hw3s.pdf`.

[GRK05] Ashish Goel, Sanatan Rai, and Bhaskar Krishnamachari. Monotone properties of random geometric graphs have sharp thresholds. *Ann. Appl. Probab.*, 15(4):2535–2552, 11 2005.

[Knu97] Donald E. Knuth. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.

[Kru56] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[LWZ+15] Z. Liao, J. Wang, S. Zhang, J. Cao, and G. Min. Minimizing movement for target coverage and network connectivity in mobile sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(7):1971–1983, July 2015.

[Meg82] N. Megiddo. Linear-time algorithms for linear programming in r3 and related problems. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 329–338, Nov 1982.

[MP10] S Muthukrishnan and Gopal Pandurangan. Thresholding random geometric graph properties motivated by ad hoc sensor networks. *Journal of Computer and System Sciences*, 76(7):686–696, 2010.

[Pen97] Mathew D. Penrose. The longest edge of the random minimal spanning tree. *Ann. Appl. Probab.*, 7(2):340–361, 05 1997.

[Sha14] Badrinath Sharma. Minimizing maximum movement to attain connectivity. Master's thesis, Indian Statistical Institute, Kolkata, 2014.

[SJK08] Ethan Stump, Ali Jadbabaie, and Vijay Kumar. Connectivity management in mobile robot teams. In *Robotics and Au-*

tomation, 2008. ICRA 2008. IEEE International Conference on, pages 1525–1530. IEEE, 2008.

[Sma98] Steve Smale. Mathematical problems for the next century. *The Mathematical Intelligencer*, 20(2):7–15, Mar 1998.

[SSE87] J Michael Steele, Lawrence A Shepp, and William F Eddy. On the number of leaves of a euclidean minimal spanning tree. *Journal of Applied Probability*, 24(4):809–826, 1987.

[ZEP11] Michael M Zavlanos, Magnus B Egerstedt, and George J Pappas. Graph-theoretic connectivity control of mobile robot networks. *Proceedings of the IEEE*, 99(9):1525–1540, 2011.

[ZP08] Michael M Zavlanos and George J Pappas. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6):1416–1428, 2008.

[ZTJP09] Michael M Zavlanos, Herbert G Tanner, Ali Jadbabaie, and George J Pappas. Hybrid control for connectivity preserving flocking. *IEEE Transactions on Automatic Control*, 54(12):2869–2875, 2009.