VERTEX ADDITION TO A BALL GRAPH WITH APPLICATION TO RELIABILITY AND AREA COVERAGE IN AUTONOMOUS SWARMS

CALUM BUCHANAN[1], PUCK ROMBACH[1], JAMES BAGROW[1], AND HAMID R. OSSAREH[2]

Abstract. A unit ball graph consists of a set of vertices, labeled by points in Euclidean space, and edges joining all pairs of points within distance 1. These geometric graphs are used to model a variety of spatial networks, including communication networks between agents in an autonomous swarm. In such an application, vertices and/or edges of the graph may not be perfectly reliable; an agent may experience failure or a communication link rendered inoperable. With the goal of designing robust swarm formations, or unit ball graphs with high reliability (probability of connectedness), in a preliminary conference paper we provided an algorithm with cubic time complexity to determine all possible changes to a unit ball graph by repositioning a single vertex. Using this algorithm and Monte Carlo simulations, one obtains an efficient method to modify a unit ball graph by moving a single vertex to a location which maximizes the reliability. Another important consideration in many swarm missions is area coverage, yet highly reliable ball graphs often contain clusters of vertices. Here, we generalize our previous algorithm to improve area coverage as well as reliability. Our algorithm determines a location to add or move a vertex within a unit ball graph which maximizes the reliability, under the constraint that no other vertices of the graph be within some fixed distance. We compare this method of obtaining graphs with high reliability and evenly distributed area coverage to another method which uses a modified Fruchterman-Reingold algorithm for ball graphs.

1. Introduction

Recent decades have seen a rise in autonomous swarms of agents of various types. For instance, autonomous swarms of satellites are likely to replace monolithic NASA missions in the near future, due increased flexibility and reduced cost [29, 2, 3, 18]. Swarms of autonomous underwater vehicles for oceanography and coastal management [13], unmanned aerial vehicles for agricultural and environmental purposes [9, 32], and swarm robotics for search and rescue missions [23, 16] (among

[1]Dept. of Mathematics and Statistics, University of Vermont, Burlington, VT USA

[2]Dept. of Electrical and Biomedical Engineering, University of Vermont, Burlington, VT USA

E-mail addresses: calumjmb@comcast.net, puck.rombach@uvm.edu, james.bagrow@uvm.edu, hossareh@uvm.edu.

2020 *Mathematics Subject Classification.* 05C85 (primary); 68R10, 93B51 (secondary).

Key words and phrases. ball graph, disk graph, vertex placement, reliability, area coverage, autonomous swarm, satellite swarm, swarm robotics, formation planning.

Implementation in Python 3 of the algorithms described herein is publicly available on GitHub at https://github.com/calum-buchanan/UDG-Neighborhoods-Reliability-AreaCoverage.

other purposes) are also the subject of significant current research. In order that the agents in a swarm work together to accomplish a task, such as Earth observation by satellites [17], each is equipped with a communication device. Without centralized control, the network induced by the agents and their communication links must remain connected; that is, any agent should be able to pass a message to any other, possibly via some intermediary agents. Otherwise, were the network to become disconnected, the swarm would not be able to function as a cohesive unit. As connectedness of the network is crucial, and as various uncertainties may render agents or the communication links between them inoperable, it is of great importance to design formations which induce communication networks whose connectedness is robust to potential failures. In the context of satellite swarms, for instance, such uncertainties may include hardware issues, radiation, space debris, or, in a lower earth orbit, clouds and other atmospheric conditions [11, 1, 22, 4].

Assuming that every agent in a swarm is equipped with the same omnidirectional communication device, the swarm's communication network can be modeled by a geometric graph known as a *unit ball graph*. Vertices of the ball graph are points in space (representing agents) and edges connect vertices within distance 1 (communication range after scaling). Unit ball graphs are also commonly used to model a wide variety of other communication networks, especially those using radio broadcast or optimal communication. The two-dimensional counterpart to a unit ball graph is known as a *unit disk graph*. These are often easier to work with and, in a number of applications such as radio broadcast networks [5], reasonable simplifications. In an application to satellite imaging of a region on the surface of an object, it is also reasonable to assume that the satellites in a formation lie approximately in a plane above the region.

In order to measure the robustness of the connectedness of a network, a theory known as network reliability has sprung forth. Commonly, one assigns a probability of operation to each edge and vertex of a graph, and asks for the probability of connectedness of the subgraph obtained by including each vertex independently with its given probability of operation, and each edge between operational vertices independently with its given probability of operation. The case in which all vertices, but not all edges, operate with probability 1 is known as *all-terminal reliability* [12]. We refer to the more general case simply as *reliability*, though in the literature it has been referred to as *residual connectedness* [33]. Unfortunately, the all-terminal reliability, even of a unit disk graph, is $\sharp$P-complete to compute [5]. A large amount of work has thus gone into efficiently estimating [14] and providing bounds [6] on the reliability of a graph.

Now, in the context of a hypothetical mission for an autonomous swarm, our goal is to ensure that the communication network has high reliability. Of course, a formation in which all agents are pairwise within communication range is optimally reliable, but mission constraints may render such a formation inefficient. For instance, suppose that a satellite swarm is assigned to image a region on the surface of an object. It is natural to assign some satellites to image the outer boundary of the region. The problem is then to assign the remaining satellites to cover the

interior of the region so that the entire region is evenly covered and so that the communication network is highly reliable.

One method to keep agents spread out is to ensure some minimum distance between any given pair. The main contribution of this paper is an efficient algorithm to enumerate all possible neighborhoods that a vertex v could have when added to a unit disk graph, under the constraint that no other vertices are within some fixed Euclidean distance b of v (Algorithm 1 of Section 3.1). We use this algorithm to move vertices one at a time, under the same constraint, to locations which maximize the reliability of the resulting graph. An alternative method is to use a spring layout- (or Fruchterman-Reingold)-type algorithm, treating vertices like similarly charged particles which repel and edges like springs which attract or repel based on the resting length; we provide a modified spring layout algorithm which avoids destroying edges in a unit ball graph in Section 3.2.

The remainder of this paper is organized as follows. First, in Section 2, we review an algorithm from our preliminary conference paper [8] to enumerate all possible neighborhoods for a vertex added to a unit ball graph. This can be used to move a vertex in a unit ball graph to a location which maximizes the network reliability of the resulting graph. In Section 3, we consider both reliability and area coverage. In Section 3.1, we describe our main algorithm in terms of unit disk graphs, noting that this algorithm can be generalized to unit ball graphs as well. In Section 3.2, we provide a variation of a classical algorithm of Fruchterman and Reingold [19] to space out vertices in a ball graph without destroying edges. In Section 4, we compare the effectiveness of the algorithms in Sections 3.1 and 3.2 in producing unit disk graphs with high reliability and evenly spread vertices over a region. The comparison shows the former algorithm to be particularly effective. We conclude with some open questions and future directions in Section 5.

1.1. **Definitions and notations.** Let V be a set of points in space, and let G be the graph with vertex set V whose edges connect pairs of distinct vertices u, v with $\|u - v\|_2 < 1$. If $V \subset \mathbb{R}^2$, we call G a *unit disk graph*, and if $V \subset \mathbb{R}^3$, we call G a *unit ball graph*. The *neighborhood* of a vertex v in V is the set of vertices adjacent to v.

Let v be a point in $\mathbb{R}^2$. For positive real numbers b and c with $b < c$, we define the (b, c)-*annulus* centered at v to be the set of points w in $\mathbb{R}^2$ such that $b < \|v - w\|_2 < c$; its boundary is the set of points w such that $\|v - w\|_2 \in \{b, c\}$.

Let G be a graph, and for each edge e in G, assign a probability of operation $p_e \in [0, 1]$. The *all-terminal reliability* of G, denoted $\mathrm{Rel}_A(G)$, is the probability that the graph G' obtained from G by deleting each edge e independently with probability $1 - p_e$ is connected. In other words, G' is the graph on the vertex set of G obtained by sampling each edge e in G independently with probability p_e. If each vertex v, along with its incident edges in G', is subsequently deleted independently with probability $1 - p_v$, the probability that the resulting graph G'' is connected is called the *residual connectedness* $\mathrm{Rel}(G)$. Equivalently, G'' is obtained by first sampling each vertex v in G independently with probability p_v, then sampling the edges e between operational vertices independently with probability p_e.

1.2. **Problem statement.** Let R be a connected region in $\mathbb{R}^2$, V a finite set of points in R, and G the unit disk graph with vertex set V. Our goal is to modify G by moving or adding vertices within R in order to increase $\mathrm{Rel}(G)$ or $\mathrm{Rel}_A(G)$ while decreasing the size of the largest empty circle (containing no points in V) whose center is contained in R. We do so in order to determine highly reliable unit disk graphs in R which do not leave large uncovered areas. We note that deleting a vertex from G can increase the reliability without increasing the size of the largest empty circle. While our results are easily adapted to accommodate this case, our application to formation planning of autonomous swarms, in which agents are not typically dispensable, makes this an undesirable option.

2. Radial sweeps and ball graphs

We begin with preliminaries on radial sweep algorithms as they apply to disk and ball graphs. We outline an algorithm from our preliminary paper [8], using radial sweeps to list all possible neighborhoods that a new vertex could have when added to a disk or ball graph. This will provide a basis for our main algorithm, described in Section 3.

2.1. **Radial Sweeps.** Radial sweeps are commonplace tools in computational geometry used to determine, for example, the maximum number of points from a finite set $V \subset \mathbb{R}^2$ which are contained in a circle of radius 1 [24]. The idea is quite simple: we imagine a unit circle with center $v + \begin{bmatrix}1 & 0\end{bmatrix}^T$ for a given point v in V. We rotate, or *sweep*, the circle 360 degrees counterclockwise, keeping v fixed on its boundary. The angles of rotation at which each point w within distance 2 of v enters and exits the circle as it sweeps can be calculated using basic trigonometry. In particular, if $v = \begin{bmatrix}x_v & y_v\end{bmatrix}^T$ and $w = \begin{bmatrix}x_w & y_w\end{bmatrix}^T$, then letting $x = x_v - x_w$, $y = y_v - y_w$, $A = \arctan(y/|x|)$, and $B = \arccos(\sqrt{x^2+y^2}/2)$, the vertex w enters the sweep around v after a rotation of $\pi - (A + B)$ degrees and exits at $\pi - (A - B)$ degrees. See Figure 1, adapted from [8].

Performing these trigonometric calculations for each vertex v in V and each vertex w in $V - v$ with $\|v - w\|_2 < 2$, one can determine all subsets N of V for which some radius-1 circle contains all of the points in N and none in $V - N$ as follows. Each time a point $w \in V$ enters or exits the circle sweeping around a point $v \in V$, we record the subset S of V contained in the interior this circle with v and w on its boundary. By shifting and/or rotating the circle slightly, one can obtain a unit circle with any of S, $S \cup v$, $S \cup w$, or $S \cup \{v, w\}$ in its interior. All of these are subsets N of V as described. On the other hand, if N is the subset of V contained in a given radius-1 circle, then the circle can be shifted until two points in V lie on its boundary. Thus, after performing a radial sweep around each vertex in V, we obtain in this manner all such subsets N of V.

Note that a subset N of V contained in the interior of some radius-1 circle which contains no other points in V would be the neighborhood of a vertex placed at the center of said circle in the corresponding unit disk graph. Conversely, any subset of V which could be the neighborhood of an appropriately placed vertex is contained in a radius-1 circle which contains no other points from V. Thus, the previously

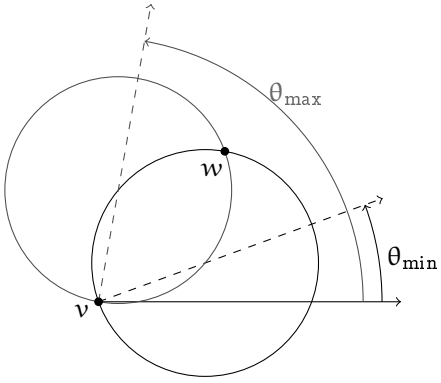


FIGURE 1. A circle of radius 1 revolves in a counterclockwise motion around a fixed point v on its boundary in a radial sweep. A point w enters and exits the circle at the angles θ_{min} and θ_{max}, respectively. (Adapted from [8].)

summarized algorithm finds all possible neighborhoods for a new vertex added to a unit disk graph on V. Although the number of possible neighborhoods for a new vertex added to an abstract graph is exponential, the number of such subsets N of V is on the order of n^2, where $n = |V|$ [36]. The algorithm described above is nearly best possible, running in $\mathcal{O}(n^2)$ time. Further, our algorithm generalizes to finite subsets V of $\mathbb{R}^3$, running in $\mathcal{O}(n^3)$ time, which is also the order of the maximum number of subsets N of V contained in a unit sphere with no other points from V in its interior [36]. These upper bounds can be improved using parameterized complexity. Letting Δ denote the maximum number of points in V contained in a ball of radius 2, our 2-D algorithm runs in time $\mathcal{O}(n\Delta)$ and the 3-D variation in time $\mathcal{O}(n^2\Delta)$ [8].

2.2. **Unit disk and unit ball graphs.** Disk and ball graphs are used to model a wide array of spatial relationships between objects. Supposing that each agent in an autonomous swarm is equipped with the same omnidirectional communication device, we can model the swarm formation by a unit ball graph[1] whose vertices represent agents in the formation and whose edges connect pairs of agents within communication distance.

Given the dynamic nature of swarm formations, it is natural to ask how the communication network changes with the formation. Generally, one might like to know which changes to the graph structure of a unit ball graph G are obtainable by repositioning vertices. This problem is difficult in general, as the recognition problem for unit ball graphs is NP-hard [7]. However, using the algorithm described in Section 2.1, we are able to enumerate in polynomial time all of the unit ball graphs obtainable by moving a single vertex in G. This can be modeled by deleting a vertex v from G and adding it back in a new location. The algorithm summarized

[1]We scale the space by a factor of $1/r$, where r is the range of the communication device.

in Section 2.1 can be used to list all of the possible neighborhoods that v could have.

3. Reliability and area coverage

Equipped with an algorithm to efficiently determine all possible neighborhoods for a new vertex added to a unit disk or unit ball graph, in [8], we used Monte Carlo simulations to determine the location to add or move a vertex which would maximize either Rel or Rel_A. Indeed, for any efficiently calculable or estimable graph parameter P, one can use our algorithm to optimize P by moving or adding a single vertex. We note that, when using our algorithm to maximize reliability, only those neighborhoods which are maximal with respect to set inclusion need be considered, for Rel and Rel_A are both monotonic with respect to deleting edges. We compared the reliabilities, over all maximal neighborhoods for a new vertex, of the graphs obtained by adding a vertex with said neighborhood. For the neighborhood N providing the most reliable estimate, we added a new vertex at the center of the smallest enclosing circle for N, which is the point which minimizes the maximum distance to a point in N. Finding this center can be done in linear time [28, 35].

We tested this method on random geometric graphs in [8], finding that repositioning a single vertex using our algorithm provides higher all-terminal reliability on average than adding ten additional vertices uniformly at random to the visible area (*i.e.*, adding vertices randomly conditioned on connectedness) with edge-operation probabilities of 90%. Applying our algorithm to add new vertices one at a time to maximize reliability, we notice that, after a number of iterations, the vertices are clustered together. This aligns with the results of [26] in which highly reliable graphs were constructed to connect a fixed set of points, and these resembled Steiner trees with thick bands of vertices.

In the context of many a swarm mission, like satellite imaging of a region, such a formation may not be desirable. A natural question then arises: *which unit disk graphs have evenly distributed vertices over a given area and have high reliability?* In the spirit of our first algorithm, we propose to add or move vertices one at a time, maximizing reliability at each iteration, but imposing the constraint that no vertex be within some fixed distance b of the vertex being added or moved. We present such an algorithm in Section 3.1, and we propose an alternative solution to the area coverage problem using a modified spring layout algorithm in Section 3.2.

3.1. Finding neighborhoods for a vertex with a buffer.

The methods we provided in [8] were not geared toward a specific autonomous swarm mission, and we acknowledge that high reliability, while important, is not the only quality desired of a swarm formation in performing a mission. A tight cluster of satellites, for instance, has a highly reliable communication network, but would not perform a task such as imaging a large area very well, where maximizing the spread of the satellites would maximize the area that can be imaged. One can imagine similar scenarios in other spatial networks, such as swarm robotic search and rescue missions. It thus seems natural to avoid tight clustering of vertices while keeping reliability high.

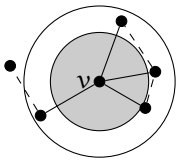


FIGURE 2. The neighbors of a vertex v added with a buffer to a disk graph are contained within an annulus with no vertices in its inner circle. The buffer is depicted by a shaded region, and existing edges in the graph before the addition of v are depicted by dashed edges.

There are a number of ways that one might handle this problem. We propose a method whereby a buffer distance is imposed around each vertex as it is added to the unit disk graph G. For a value b with $0 < b < 1$, we note that a vertex which has no neighbors within distance b must have all of its neighbors contained in the $(b, 1)$-annulus centered at that vertex (see Figure 2). In other words, the possible neighborhoods for a new vertex added with a buffer are precisely the subsets S of the vertex set V of G such that S is contained in a $(b, 1)$-annulus and all points in $V - S$ lie outside of the radius-1 circle. We use a modified radial sweep algorithm to determine all such subsets S of V. This subsection is devoted to proving the following.

Theorem 1. *Let* G *be a unit disk graph with vertex set* V*, and let* $0 < b < 1$*. The list of possible neighborhoods for a vertex* v *added to* G *such that* $\|v - w\|_2 > b$ *for all* $w \in V$ *can be obtained in* $\mathcal{O}(n\Delta)$ *time, where* Δ *is the maximum cardinality of a subset of* V *contained in a radius-*2 *circle.*

Theorem 1 follows from an analysis of Algorithm 1, which we describe in parts throughout this section. We first find all subsets contained in a $(b, 1)$-annulus with two vertices on its boundary and no vertices contained within its inner circle. To do so, we conduct radial sweeps of annuli around a fixed points on their outer and inner boundaries.

First, we sweep a $(b, 1)$-annulus around a fixed point $v \in V$ on its outer boundary (see Algorithm 2). Consider the $(b, 1)$-annulus whose center is at $v + \begin{bmatrix} 1 & 0 \end{bmatrix}^T$. We imagine a full counterclockwise rotation of the annulus around the fixed point v. Points $w \in V$ with $1 - b < \|v - w\|_2 < 1 + b$ will enter and exit both the inner and outer circles of the annulus as it rotates. Using trigonometric equations similar to those described in Section 2.1, we record the angles of rotation for each of these four moments: let θ_{min} denote the angle at which w enters the outer circle, θ_{dis} the angle at which w enters the inner circle (and disappears from the annulus), θ_{re} the angle at which w exits the inner circle (and reappears in the annulus), and θ_{max} the angle at which w exits the outer circle. We add the 5-tuple $(w, \theta_{min}, \theta_{dis}, \theta_{re}, \theta_{max})$ to a list L. Points $w \in V$ with $\|v - w\|_2 < 1 - b$ or $\|v - w\|_2 > 1 + b$ will enter and exit only the outer circle of the annulus as it rotates about v. In this case, we set $\theta_{dis} = \text{None}$ and $\theta_{re} = \text{None}$.

Algorithm 1 Enumerate neighborhoods for a new vertex added to a unit disk graph with no other vertices within distance b ($0 < b < 1$).

Input: Nonempty finite set of points $V \subset \mathbb{R}^2$ and $b \in (0, 1)$
Output: Subsets of V contained in a $(b, 1)$-annulus with no points in inner circle

function BUFFERNEIGHBORHOODS(V, b)
    Nbrhds $\leftarrow \{\}$
    for $v \in V$ **do**
        $L_1 \leftarrow$ OUTERSWEEP(V, v, b)
        for $S \in$ SETSFROMINTERVALS(L_1) **do**
            add S and $S \cup v$ to Nbrhds
        end for
        $L_2 \leftarrow$ INNERSWEEP(V, v, b)
        for $S \in$ SETSFROMINTERVALS(L_2) **do**
            add $S \cup v$ to Nbrhds
        end for
    end for
    return Nbrhds
end function

Algorithm 2 List of points and angles at which points enter and exit a $(b, 1)$-annulus swept around a fixed point v on its outer boundary

Input: Nonempty finite set of points $V \subset \mathbb{R}^2$, $v \in V$, and $b \in (0, 1)$
Output: Points $w \in V$ and angles $\theta \in (-\pi/2, \pi/2)$ of rotation at which w enters and exits a $(b, 1)$-annulus swept counterclockwise around fixed point v on outer boundary

function OUTERSWEEP(V, v, b)
    $L \leftarrow \{\}$
    for $w \in V - v$ with $\|v - w\|_2 < 2$ **do** ▷ w will enter & exit the sweep
        if $1 - b < \|v - w\|_2 < 1 + b$ **then** ▷ w will enter & exit both circles
            $\theta_{min} \leftarrow$ angle w enters outer circle
            $\theta_{dis} \leftarrow$ angle w enters inner circle
            $\theta_{re} \leftarrow$ angle w exits inner circle
            $\theta_{max} \leftarrow$ angle w exits outer circle
        else ▷ w will only enter & exit the outer circle
            $\theta_{min} \leftarrow$ angle w enters outer circle
            $\theta_{dis}, \theta_{re} \leftarrow$ None
            $\theta_{max} \leftarrow$ angle w exits outer circle
        end if
        add $(w, \theta_{min}, \theta_{dis}, \theta_{re}, \theta_{max})$ to L
    end for
    return L
end function

Algorithm 3 List of points and angles at which they enter and exit a $(b, 1)$-annulus swept around a fixed point on its inner boundary

Input: Nonempty finite set of points $V \subset \mathbb{R}^2$, $v \in V$, and $b \in (0, 1)$

Output: Points $w \in V$ and angles $\theta \in (-\pi/2, \pi/2)$ of rotation at which w enters and exits a $(b, 1)$-annulus swept counterclockwise around fixed point v on inner boundary

function InnerSweep(V, v, b)
  $L \leftarrow \{\}$
  for $w \in V - p$ with $\|v - w\|_2 < 1 + b$ **do** ▷ w will enter & exit the sweep
    if $1 - b < \|v - w\|_2 < 2b$ **then** ▷ w will enter & exit both circles
      $\theta_{min} \leftarrow$ angle w enters outer circle
      $\theta_{dis} \leftarrow$ angle w enters inner circle
      $\theta_{re} \leftarrow$ angle w exits inner circle
      $\theta_{max} \leftarrow$ angle w exits outer circle
    else if $\|v - w\|_2 > 2b$ **then** ▷ w will only enter & exit outer circle
      $\theta_{min} \leftarrow$ angle w enters outer circle
      $\theta_{dis}, \theta_{re} \leftarrow$ None
      $\theta_{max} \leftarrow$ angle w exits outer circle
    else if $\|v - w\|_2 < 1 - b$ **then** ▷ w will only enter & exit inner circle
      $\theta_{min}, \theta_{max} \leftarrow$ None
      $\theta_{dis} \leftarrow$ angle w enters inner circle
      $\theta_{re} \leftarrow$ angle w exits inner circle
    end if
    add $(w, \theta_{min}, \theta_{dis}, \theta_{re}, \theta_{max})$ to L
  end for
  return L
end function

Second, we sweep a $(b, 1)$-annulus around a fixed point $v \in V$ on its inner boundary (see Algorithm 3). Consider the $(b, 1)$-annulus centered at $v + \begin{bmatrix} b & 0 \end{bmatrix}^T$. As it rotates counterclockwise about v, points $w \in V$ within distance $1 + b$ of v will enter and exit the annulus. If $1 - b < \|v - w\|_2 < 2b$, then w will enter and exit the annulus twice in one full rotation, passing through both the inner and outer boundaries. We again record the angles θ_{min}, θ_{dis}, θ_{re}, and θ_{max} defined above, using basic trigonometry, and add the tuple $(w, \theta_{min}, \theta_{dis}, \theta_{re}, \theta_{max})$ to a list L. If $2b < \|v - w\|_2 < 1 + b$, then w only enters and exits the annulus once during the sweep, via the outer boundary. We record the angles θ_{min} and θ_{max} and set $\theta_{dis} = \text{None}$ and $\theta_{re} = \text{None}$. Finally, if $\|v - w\|_2 < 1 - b$, then w only enters and exits the annulus via the inner boundary, and we set $\theta_{min} = \text{None}$ and $\theta_{re} = \text{None}$.

We now describe how to obtain the sets S of vertices contained in a $(b, 1)$-annulus with $V - S$ outside of its outer boundary (see Algorithm 4). By Proposition 2, it suffices to find the sets S contained in a $(b, 1)$-annulus with two points on its boundary (keeping track of whether they are on the outer or inner boundary) and

Algorithm 4 Obtain sets of points contained in a $(b, 1)$-annulus from a sweep around a given point v

Input: Nonempty finite subset V of $\mathbb{R}^2$; L, the output of Algorithm 2 or Algorithm 3

Output: Collection of subsets of V contained in a $(b, 1)$-annulus with no points in its inner circle

function SETSFROMINTERVALS(L)
  $M \leftarrow [\,]$
  $R \leftarrow$ INITIALSET(L) ▷ See Algorithm 5 in Appendix B
  $B \leftarrow \{\}$ ▷ Bad vertices: contained in the inner circle
  for $(w, \theta_{\min}, \theta_{\text{dis}}, \theta_{\text{re}}, \theta_{\max}) \in L$ and **do**
    for $\theta \in \{\theta_{\min}, \theta_{\text{dis}}, \theta_{\text{re}}, \theta_{\max}\}$ **do**
      if θ is not None **then**
        add (w, θ, type) to M, where type $\in \{\min, \text{dis}, \text{re}, \max\}$
      end if
    end for
    if $\theta_{\text{re}} < \theta_{\text{dis}}$ **then** ▷ w is bad when sweep begins
      add $(w, \theta_{\text{dis}}, \text{type} = \text{dis})$ to B
    end if
  end for
  Sort M by angles θ in increasing order
  Sets $\leftarrow \{\}$
  for $(w, \theta, \text{type}) \in M$ **do**
    if type $=$ max **then** remove w from R
      if $B = \emptyset$ **then** add R and $R \cup w$ to Sets
      end if
    else if type $=$ min **then** add w to R
      if $B = \emptyset$ **then** add R and $R - w$ to Sets
      end if
    else if type $=$ dis **then** remove w from R
      if $B = \emptyset$ **then** add $R \cup w$ to Sets
      end if
      add w to B
    else if type $=$ re **then** add w to R
      remove w from B
      if $B = \emptyset$ **then** add R to Sets
      end if
    end if
  end for
  return Sets
end function

all other points in $V - S$ outside of the outer boundary. One of the points on the boundary will be the point v we are sweeping around, and the other will be a point that is entering or exiting the sweep.

Let L_v be a list of tuples $(w, \theta_{\min}, \theta_{\text{dis}}, \theta_{\text{re}}, \theta_{\max})$ obtained from a sweep around $v \in V$ using either Algorithm 2 or 3. We first determine the points contained in the annulus when the sweep begins (Algorithm 5). These are those points w which are close enough to v to enter and exit the sweep and are such that

- $\theta_{\max} < \theta_{\min}$ and $\theta_{\text{dis}} < \theta_{\text{re}}$, or
- $\theta_{\max} < \theta_{\min}$ and $\theta_{\text{dis}}, \theta_{\text{re}} = \text{None}$, or
- $\theta_{\min}, \theta_{\max} = \text{None}$ and $\theta_{\text{dis}} < \theta_{\text{re}}$.

We add these points to a "running set" R. Any points w which are not contained in the annulus when the sweep begins but which are contained in its inner circle (those with $\theta_{\text{re}} < \theta_{\text{dis}}$) we add to a "bad set" B.

From L_v, we make a new list M_v of tuples (w, θ, type) to record the angles θ, in increasing order, at which points w enter and exit the annulus during the sweep, along with whether it is entering or exiting and which boundary it is passing through. Specifically, type is one of 'min', 'dis', 're', or 'max' depending on whether θ is the angle at which w enters the outer circle, enters the inner circle, exits the inner circle, or exits the outer circle of the annulus, respectively.

Now, we go through the tuples in M_v in increasing order of θ. If a vertex w exits the outer circle (that is, if the tuple (w, θ, type) has type 'max'), we remove w from R. We then check if the set of bad vertices B is empty and, if it is, add both R and $R \cup w$ to a list Sets which will be the output of Algorithm 4. If a vertex w enters the outer circle (if the tuple has type 'min'), we add it to R, and if $B = \emptyset$, we add R and $R - w$ to Sets. If a vertex w enters the inner circle (if the tuple has type 'dis'), we remove w from R, check if $B = \emptyset$, and, if it is, add $R \cup w$ to Sets. Then, we add w to B. Finally, if a vertex w exits the inner circle (if the tuple has type 're'), we add it to R and remove it from B. If now $B = \emptyset$, we add R to Sets.

Once the list M_v is exhausted, Sets contains all sets of vertices which are contained in a $(b, 1)$-annulus with v fixed on a boundary. If v is fixed on the outer boundary, then for each set S in Sets, both S and $S \cup v$ are possible subsets of V contained in a $(b, 1)$-annulus with no points in its inner circle. On the other hand, if v was fixed on the inner boundary, then only $S \cup v$ is a possible subset of V contained in a $(b, 1)$-annulus with no points in its inner circle (see Proposition 2).

Performing Algorithms 2 and 3 each $n = |V|$ times, by Proposition 2 we obtain all subsets S of V for which there exists a $(b, 1)$-annulus containing S and with every point in $V - S$ outside of the radius-1 circle in $\mathcal{O}(n^2)$ time. This is summarized in Algorithm 1.

As with our previous algorithm in [8], the run time of Algorithm 1 can be improved using parameterized complexity. Letting Δ denote the maximum number of points contained in a radius-2 circle (the maximum number of neighbors that a vertex added to G can have), our algorithm runs in time $\mathcal{O}(n\Delta)$.

We also note that our algorithms can be generalized to unit ball graphs in $\mathbb{R}^3$ by sweeping two concentric spheres, one of radius b and one of radius 1, around the

axis formed by the line segment between two vertices $u, v \in V$ with $\|u - v\|_2 < 2$. We do not describe such an algorithm here, as it would be a lengthy description, and as unit disk graphs are reasonable models for many of the relevant applications mentioned here (including our running example of an autonomous satellite swarm imaging a region on the surface of an object).

By adjusting the value of b, we can use this algorithm to fill in a region with a reliable graph whose vertices are evenly spaced apart. In Section 4, we will compare this algorithm to another method for spreading out vertices without decreasing reliability which we presently describe.

3.2. **Fruchterman-Reingold for ball graphs.** We now describe an alternative approach for modifying a unit ball graph for reliability and area coverage purposes. In particular, we investigate a method for providing more even area coverage without decreasing the reliability of the graph obtained using Algorithm 1 or the simpler version from [8] described in Section 2.1.

The Fruchterman-Reingold, or spring layout, algorithm introduced in [19] is a standard graph drawing method. It is based on a simple idea: vertices should be reasonably spaced apart, and adjacent vertices should be closer together than pairs of nonadjacent vertices. To achieve this, vertices are treated like similarly charged particles, repelling one another, while edges are treated like springs, resisting this force. The algorithm iterates some number of times, at each iteration linearly scaling down this combination of attractive and repulsive forces (reducing the "temperature") until vertices settle into place.

Spring layout algorithms have also been used in a variety of applications. Variations of the Fruchterman-Reingold algorithm have been proposed for the deployment of large-scale wireless sensor networks [15, 27]. A related concept of virtual force algorithms has also been studied in the context of wireless sensor network coverage (see, for example, [21, 31]).

While the spring layout algorithm can provide a more even spread of vertices over a region, it can also drastically change the underlying ball graph. Of concern to us is the scenario in which edges are deleted (when vertices which were within communication distance are moved further apart), decreasing the reliability of the resulting ball graph. Extra edges, on the other hand, only increase the reliability. Luckily, this simple algorithm is not hard to modify so that it respects adjacencies in a unit disk or ball graph. We chose to modify the existing Fruchterman-Reingold algorithm from the NetworkX Python library [20].[2] There are a number of ways that one might stop edges in a unit ball graph from being broken by the Fruchterman-Reingold algorithm. Here, we do so by leveraging the temperature: at each iteration, we check whether any pair vertices which were previously within distance 1 are now of distance greater than 1 apart. If such a pair exists, we simply go back to the positions at the last iteration, reduce the temperature, and move on to the next iteration. We now describe our modified Fruchterman-Reingold algorithm for ball graphs in more detail.

[2] `https://networkx.org/documentation/stable/_modules/networkx/drawing/layout.html`

Let G be a unit ball graph of order n with vertex set V and edge set E. We consider G to lie within a *frame*, a minimum rectangle or rectangular prism containing all points in V. Let k denote the optimal distance between an average pair of vertices at the end of the algorithm. In two dimensions, it is common to set $k = \sqrt{A/n}$, where A denotes the area of the frame. Finally, let t denote the initial *temperature*, which determines how far we allow vertices to move at the first iteration. For the algorithm we chose to modify, t is $1/10$ of the maximum length of a side of the frame. For each pair of distinct vertices $u, v \in V$, we calculate the force $f_r(u,v) = k^2/d_{u,v}^2$ of repulsion. When $uv \in E$, we calculate the force $f_a(u,v) = d_{u,v}/k$ of attraction, and we let $f_a(u,v) = 0$ when $uv \notin E$. Let F denote the matrix of forces between vertices: $F_{u,v} = f_r(u,v) - f_a(u,v)$. We use these forces to calculate the initial displacement vector $\vec{disp}_v$ for each vertex v. In two dimensions, letting $v = \begin{bmatrix} x_v & y_v \end{bmatrix}^T$, the first coordinate of $\vec{disp}_v$ is $\sum_{w \neq v}(x_v - x_w)F_{v,w}$ and the second is $\sum_{w \neq v}(y_v - y_w)F_{v,w}$. We scale each displacement vector according to the temperature, obtaining a vector $\delta_v = \vec{disp}_v \cdot t/l_v$, where l_v is the length of $\vec{disp}_v$.

Classical Fruchterman-Reingold algorithms come in two types: at each iteration, either all vertices move at once according to their displacement vectors, or the vertices move one at a time, requiring a recalculation of forces at each iteration. In the former case, the new position for each vertex v is $\begin{bmatrix} x_v & y_v \end{bmatrix}^T + \delta_v$. In the latter case, we cycle through the vertices of G, moving a single vertex v according to δ_v at each iteration. Our solution to the problem of breaking edges is essentially the same in either case: if any pair of vertices previously within distance 1 would be at distance > 1 after any iteration, we avoid moving any vertices at all, lower the temperature by $t/(it+1)$, where 'it' denotes the total number of iterations, and move on to the next iteration (essentially weakening the forces $F_{u,v}$). If all pairs of vertices previously within distance 1 remain within distance 1, we keep the new positions and again lower the temperature by $t/(it+1)$ for the next iteration. As the temperature decreases to $t/(it+1)$ at the last iteration, the positions of the vertices stabilize.

4. Formation planning for autonomous swarms

In this section, we apply Algorithm 1 and the modified Fruchterman-Reingold algorithm from Section 3.2 to obtain swarm formations with even area coverage and high reliability. Returning to our running example, we consider an autonomous swarm of satellites assigned to image a region on the surface of an object. We begin by assigning satellites to image the outer boundary of the region. Our goal is to assign the remaining satellites to cover the interior of the region so that the entire region is evenly covered and so that the communication network is not only connected but highly reliable. We note that search and rescue missions by robots, imaging of an ocean bed by autonomous underwater vehicles, or use of unmanned aerial vehicles for precision agriculture may be modeled similarly. Our findings indicate that using Algorithm 1 alone (as opposed to using it in conjunction with

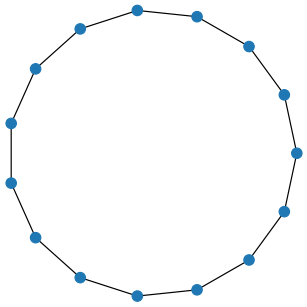


FIGURE 3. Fifteen agents cover the outer boundary of a circular region, modeled by a unit disk graph forming a regular 15-gon whose edges are of length 0.9

our modified Fruchterman-Reingold algorithm) is particularly effective in accomplishing this goal.

As a basis for our simulations, we imagine a circular disk to be covered by a swarm of 30 agents.[3] We begin with a unit disk graph on 15 vertices forming a regular 15-gon whose edges have length 0.9 (see Figure 3). These vertices outline a circular region with area about 14.717 (radius ≈ 2.16). We add 15 additional vertices one at a time to this unit disk graph using four different methods:

(M1) *Random:* Add vertices randomly to the interior of the region, conditioned on them being within unit distance of at least one existing vertex (*i.e.*, conditioned on connectedness).

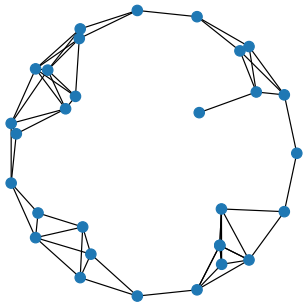
(M2) *Random & Spring:* Add vertices as in item (M1), but apply the Fruchterman-Reingold algorithm from Section 3.2 each time a vertex is added, leaving the positions of the initial 15 vertices fixed.

(M3) *Buffer:* Use Algorithm 1 and Monte Carlo simulations to add vertices which maximize the resulting reliability while being at distance at least 0.65 from any other vertex.

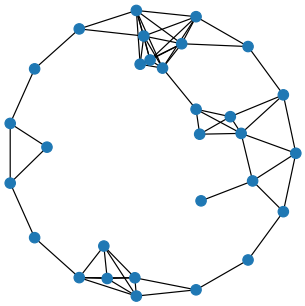
(M4) *Buffer & Spring:* Add vertices as in item (M3), but apply the Fruchterman-Reingold algorithm from Section 3.2 each time a vertex is added.

The specifics of each of these methods are discussed below, and examples of the resulting graphs are depicted in Figure 4. For a reliability measure, we use the all-terminal reliability with edge-operation probabilities all 90%. These edge-operation probabilities may be adjusted, and unreliable vertices introduced, with additional information regarding the agents in the application and their communication devices. Using the Zero-One Estimator theorem of Karp and Luby [25], or one of its improved forms (see, *e.g.*, [14]), one can easily compute upper bounds on the number of Monte Carlo simulations necessary to obtain, with probability $1-\delta$, an estimate of Rel_A with relative error ε (known as an (ε, δ)-approximation).

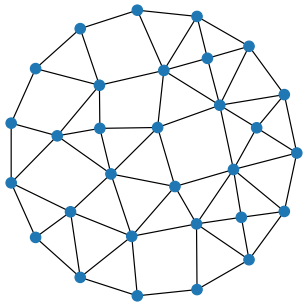
[3]This is simply for illustration; regions of other shapes, and swarms of other sizes, may be considered as well.



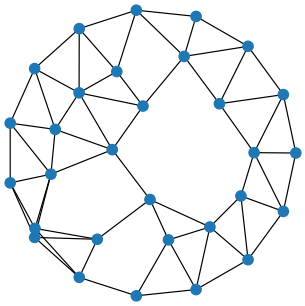
(A) An example of method (M1)



(B) An example of method (M2)



(C) An example of method (M3)



(D) An example of method (M4)

FIGURE 4. Fifteen vertices are added to the interior of a region outlined by a regular 15-gon with edge-lengths of 0.9 according to methods (M1)-(M4).

For a graph G with high reliability, however, these upper bounds can be quite large. Such Monte Carlo techniques proved inefficient for the more reliable graphs produced by the methods in items (M3) and (M4) above. Instead, we use an open-source program called $\mathcal{K}$-RelNet, introduced in [30], to obtain (ε, δ)-estimates of $\mathrm{Rel}_A(G)$. Essentially, the program reduces the problem of finding all-terminal reliability (or the more general K-terminal reliability, see [12]) to counting the number of assignments satisfying a Boolean formula in conjunctive normal form. It then obtains an (ε, δ)-approximation of this count using a state-of-the-art approximate model counter ApproxMC, introduced in [10] and now in its sixth version, ApproxMC6 [37]. We set $\varepsilon = 0.8$ and $\delta = 0.2$ for our relative estimates; these are the default values for the $\mathcal{K}$-RelNet program, and a relative error of 0.8 provides quite a small absolute error bound for the more reliable graphs in question.

As a measure of area coverage, we calculate the radius of the largest empty circle (containing no vertices of the graph) whose center is contained in the convex hull of the vertex set. The region will be well-covered if the radius of the largest empty circle is small. A largest empty circle whose center is contained in the convex hull of a set of n points can be found in $\mathcal{O}(n \log n)$ time [34]. In Table 1, we record the

	(M1)	(M2)	(M3)	(M4)
Avg. $\mathrm{Rel}_A(G)$	81.62%	85.16%	99.09%	99.20%
Avg. radius of LEC	1.2445	1.4287	0.6728	0.9697

TABLE 1. The average all-terminal reliability Rel_A with edge-operation probabilities of 90% and the average radius of the largest empty circle (LEC) over 100 graphs obtained using each method (M1)-(M4).

average all-terminal reliability and the average radius of the largest empty circle over 100 graphs obtained using each of the methods (M1)-(M4). The results in Table 1 are compared, and standard deviations depicted, in Figure 5.

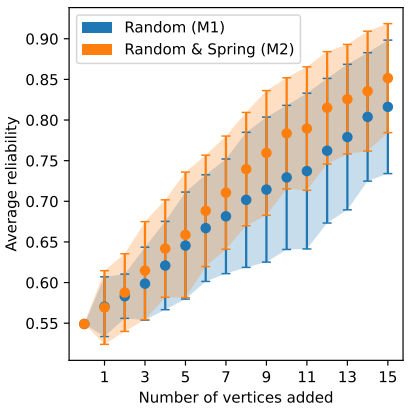
We first consider methods (M1) and (M2). Averaged over 100 trials, the all-terminal reliability of a graph obtained by adding 15 vertices one at a time uniformly at random to the interior of the region, conditioned on connectedness of the resulting graph, is about 81.62%. The average radius of the largest empty circle is about 1.245.

The Fruchterman-Reingold algorithm has quite a different effect when applied to the graphs obtained via method (M1) than one might expect: vertices are pushed away from the interior of the region towards the boundary. This increases, rather than decreases, the uncovered area of the region. On the other hand, it also has the effect of slightly increasing the reliability. We compared both types of our modified Fruchterman-Reingold algorithm from Section 3.2 (either all non-fixed vertices move at each iteration, or vertices move one at a time), and found that the former method was more effective for both reliability and area coverage purposes.

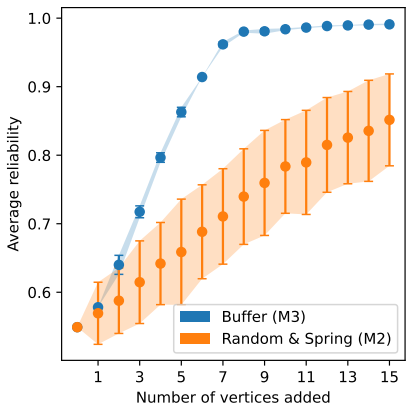
Averaged over 100 trials, the all-terminal reliability of the graph obtained by adding 15 vertices one at a time uniformly at random to the interior of the region, applying our modified Fruchterman-Reingold algorithm after each added vertex, is about 85.16%. The average radius of the largest empty circle is about 1.43. Figure 5(A) plots the average reliabilities obtained using method (M1) against those using method (M2).

Using methods (M1) and (M2) as a baseline, we now consider method (M3). Having experimented with various buffer values b, we noted that, for $b \geqslant 0.75$, the region outlined by the fifteen satellites in Figure 3 tends to fill up before fifteen vertices can be added to the interior, meaning that there are no more points within the outlined region which are of distance at least b from every vertex. Note that, if the region fills up like this, then the radius of the largest empty circle is at most b. We chose the value $b = 0.65$ as it fills the region relatively evenly and nearly fully after fifteen iterations. Of course, depending on the imaging device used by the agents in the application, this value of b may need to be altered accordingly.

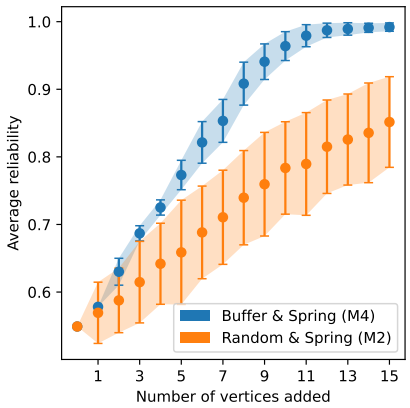
We average the all-terminal reliability of 100 graphs obtained by adding 15 vertices to the interior of a region outlined by 15 satellites using Algorithm 1 ($b = 0.65$) and choosing the most reliable candidate at each step. The average reliability after



(A) (M1) versus (M2)



(B) (M3) versus (M2)



(C) (M4) versus (M2)

FIGURE 5. A comparison of average all-terminal reliability (with edge-operation probabilities of 90%) of graphs obtained via methods (M1)-(M4). Shaded regions depict one standard deviation.

adding 15 vertices is about 99.09%. The average radius of the largest empty circle is about 0.67. The results are displayed in Figure 5(B).

Finally, we consider method (M4). Each time we add a vertex to the interior of the region to maximize reliability while being of distance at least 0.65 from every other vertex, we apply our Fruchterman-Reingold algorithm from Section 3.2. We again use the version of our algorithm in which all non-fixed vertices move at each iteration. Note that this algorithm may move vertices which were placed outside of buffer distance closer together. Indeed, the largest empty circle is, on average, larger in this case. Over 100 trials, the average radius of the largest empty circle is about 0.97. The average all-terminal reliability after adding 15 vertices is about 99.2%. The results are displayed in Figure 5(C).

5. Future directions

In this paper, we considered various possible solutions to the problem of producing unit ball graphs with high reliability whose vertices are evenly spread over a given region. We found that placing vertices one at a time in locations which are at least some fixed distance b from all other vertices (using Algorithm 1), and which maximize the reliability over all such locations (using Monte Carlo simulations) provides a better method than using a modified spring layout algorithm for unit ball graphs.

These methods focused on balancing the reliability and area coverage objective functions. Using the spring layout method, we attempted to spread out vertices while retaining the reliability of the original graph. It would be of interest to accomplish this task using a less heuristic method. We thus pose the following.

Problem 1. *Design an algorithm which takes as input a unit ball graph* G *and outputs a new unit ball graph, containing all edges of* G*, whose largest empty circle centered in the convex hull of its vertex set is as small as possible.*

We now turn to the method which uses Algorithm 1: adding vertices one at a time to maximize reliability while enforcing a buffer distance. While the estimated reliabilities of the graphs obtained using methods (M3) and (M4) were quite high, even more reliable graphs may have been obtained by adding more than one vertex at a time. Algorithms which approximate the possible neighborhoods for a set of vertices added in a given configuration are not hard to develop, but to list all such neighborhoods will require more complicated techniques.

Problem 2. *Design an algorithm to find the locations to add two or more vertices to a unit ball graph and maximize reliability.*

In a similar vein, we have not fully resolved the problem of finding the most reliable unit ball graphs of a given order which cover a given area (even for the circular area considered in Section 4). This problem is likely a difficult one, as finding most reliable (abstract) graphs of a given order and size remains an open problem. However, the constraints here are quite different, and we pose this problem for completeness.

Problem 3. *Let* G *be a unit disk graph whose vertices outline the boundary of a convex region, and let* $p \in (0, 1)$*. Given a positive integer* k *and positive real number* ℓ*, find a unit disk graph obtained by adding* k *vertices to* G *whose largest empty circle has radius at most* ℓ *(should such a graph exist) which maximizes* Rel_A *(edge-operation probabilities* p*) over all such graphs.*

Appendix A. Vertices on the boundary

In our proof of Theorem 1, we claimed that, in order to find the set of all regions of intersection of a set of annuli centered at the points v in V, it suffices to find the subsets of V contained in a $(b, 1)$-annulus with no points from V in its inner circle, and with two points from V on its boundary. This claim follows from the following proposition.

Proposition 2. *Let $V \subset \mathbb{R}^2$ be a finite set of points in general position, and let $\emptyset \neq S \subseteq V$. If S is contained in a (b, c)-annulus with all points in $V - S$ outside of its outer boundary, then there exists a second (b, c)-annulus with two points $u, v \in V$ on its boundary which contains $S - \{u, v\}$ and is such that all points in $V - (S \cup \{u, v\})$ lie outside of the outer boundary.*

Further, if u and v lie on the outer boundary of a (b, c)-annulus which contains S and no points within its inner circle, then there are (b, c)-annuli containing each of S, $S \cup \{u\}$, $S \cup \{v\}$, and $S \cup \{u, v\}$, with no points on the boundary, and the rest of the points in V outside of the outer circle. If one of u or v lies on the inner boundary, say u, then only two of these sets can be contained in such an annulus: $S \cup \{u\}$ and $S \cup \{u, v\}$. If both u and v lie on the inner boundary, then only $S \cup \{u, v\}$ is possible.

Proof. First, let A be a (b, c)-annulus with $S \subset A$ and $V - S$ lying outside of the outer boundary of A. It is not hard to see that we can shift A slightly so that some point $u \in V$ is on its outer or inner boundary. We now rotate this shifted annulus around the fixed point u until a second point v lies on its boundary to obtain the desired annulus.

On the other hand, suppose that A' is a (b, c)-annulus with $u, v \in V$ on its boundary and no points in V within its inner circle. Let S denote the set of points contained in A' (note $u, v \notin S$). Since the points in V are in general position, we may assume that u and v do not lie on a diameter of A'. First, suppose u lies on the outer boundary. We can rotate A' around u to obtain an annulus containing $S \cup \{v\}$ with only u on its boundary. Then, shifting the resulting annulus slightly, we can obtain annuli containing either $S \cup \{u, v\}$ or $S \cup \{v\}$ with all other points in V outside of the its outer circle. Similarly, if v is on the outer boundary of A', we could have rotated A' in the other direction to obtain an annulus containing S but with $V - (S \cup \{u\})$ outside of its outer boundary. In this way, we obtain annuli containing all four sets S, $S \cup \{u\}$, $S \cup \{v\}$, and $S \cup \{u, v\}$ in their interiors, and no other points in their inner circle.

On the other hand, if u is on the inner boundary, then any shift of the annulus puts u inside the inner circle if not in the annulus. Thus, only the sets $S \cup \{u\}$ and $S \cup \{u, v\}$ are possibilities in case v is on the outer boundary, and only $S \cup \{u, v\}$ is possible if both are on the inner boundary of A'. □

Appendix B. InitialSet algorithm

Algorithm 5 Determine initial set of vertices in annulus when a sweep begins

Input: $V \subset \mathbb{R}^2$ and output L of Algorithm 2 or Algorithm 3
Output: Subset of V contained in annulus when sweep begins

function InitialSet(L)
  R $\leftarrow \{\}$
  for $(w, \theta_{\text{min}}, \theta_{\text{dis}}, \theta_{\text{re}}, \theta_{\text{max}}) \in$ L **do**
    if θ_{re} is None and $\theta_{\text{max}} < \theta_{\text{min}}$ **then**
      add $(w, \theta_{\text{min}}, \text{type} = \text{min})$ to R
    else if $\theta_{\text{dis}} < \theta_{\text{re}}$ and θ_{max} is None **then**
      add $(w, \theta_{\text{re}}, \text{type} = \text{re})$ to R
    else if $\theta_{\text{dis}} < \theta_{\text{re}}$ and $\theta_{\text{max}} < \theta_{\text{min}}$ **then**
      if $\theta_{\text{dis}} < 0$ **then**
        add $(w, \theta_{\text{min}}, \text{type} = \text{min})$ to R
      else if $\theta_{\text{re}} > 0$ **then**
        add $(w, \theta_{\text{re}}, \text{type} = \text{re})$ to R
      end if
    end if
  end for
  return R
end function

Acknowledgments

This work was funded in part by NASA under cooperative agreement VT-80NSSC20M0213. C.B. acknowledges support from the Vermont Space Grant Consortium Graduate Fellowship Program.

References

[1] About space debris. https://www.esa.int/Space_Safety/Space_Debris/About_space_debris. Accessed: January 16 2024.
[2] Swarm autonomy. https://www-robotics.jpl.nasa.gov/what-we-do/research-tasks/swarm-autonomy/. Accessed: January 11 2024.
[3] 2020 NASA technology taxonomy. https://www.nasa.gov/otps/2020-nasa-technology-taxonomy/. Accessed: January 11 2024.
[4] State-of-the-art of small spacecraft technology. https://www.nasa.gov/smallsat-institute/sst-soa/soa-communications/, Jul 28 2023. Accessed: January 15 2024.
[5] H. M. AboElFotoh and C. J. Colbourn. Computing 2-terminal reliability for radio-broadcast networks. *IEEE Transactions on Reliability*, 38(5):538–555, December 1989. ISSN 1558-1721. doi: 10.1109/24.46478.
[6] Michael O. Ball, Charles J. Colbourn, and J. Scott Provan. Chapter 11 Network reliability. In *Handbooks in Operations Research and Management Science*,

volume 7 of *Network Models*, pages 673–762. Elsevier, January 1995. doi: 10.1016/S0927-0507(05)80128-8.

[7] Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Computational Geometry*, 9(1-2):3–24, 1998.

[8] Calum Buchanan, James Bagrow, Puck Rombach, and Hamid Ossareh. Node placement to maximize reliability of a communication network with application to satellite swarms. In *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3466–3473, 2023. doi: 10.1109/SMC53992.2023.10394556.

[9] Widodo Budiharto, Andry Chowanda, Alexander Agung Santoso Gunawan, Edy Irwansyah, and Jarot Sembodo Suroso. A review and progress of research on autonomous drone in agriculture, delivering items and geographical information systems (GIS). In *2019 2nd World Symposium on Communication Engineering (WSCE)*, pages 205–209. IEEE, 2019.

[10] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A scalable approximate model counter. In *Principles and Practice of Constraint Programming: 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings 19*, pages 200–216. Springer, 2013.

[11] Zhiwei Chen, Hao Zhang, Xinyue Wang, Jinling Yang, and Hongyan Dui. Reliability analysis and redundancy design of satellite communication system based on a novel Bayesian environmental importance. *Reliability Engineering & System Safety*, 243:109813, 2024.

[12] Charles J. Colbourn. *The combinatorics of network reliability*. Oxford University Press, Inc., 1987. ISBN 0-19-504920-9.

[13] Jack Connor, Benjamin Champion, and Matthew A. Joordens. Current algorithms, communication methods and designs for underwater swarm robotics: A review. *IEEE Sensors Journal*, 21(1):153–169, 2021. doi: 10.1109/JSEN.2020.3013265.

[14] Paul Dagum, Richard Karp, Michael Luby, and Sheldon Ross. An optimal algorithm for Monte Carlo estimation. *SIAM Journal on computing*, 29(5): 1484–1496, 2000.

[15] Xiaohua Deng, Zhiyong Yu, Rongxin Tang, Xin Qian, Kai Yuan, and Shiyun Liu. An optimized node deployment solution based on a virtual spring force algorithm for wireless sensor network applications. *Sensors*, 19(8):1817, 2019.

[16] Marco Dorigo, Guy Theraulaz, and Vito Trianni. Swarm robotics: Past, present, and future [point of view]. *Proceedings of the IEEE*, 109(7):1152–1165, 2021.

[17] A. Farrag, Saed Othman, Tarek Mahmoud, and Ahmed Y. ELRaffiei. Satellite swarm survey and new conceptual design for earth observation applications. *The Egyptian Journal of Remote Sensing and Space Science*, 24(1):47–54, 2021.

[18] Julianna Fishman. NASA small satellites to demonstrate swarm communications and autonomy. `https://www.nasa.gov/smallspacecraft/nasa-small-satellites-to-demonstrate-swarm-communications-and-autonomy/`, Jul 25 2023. Accessed: January 11 2024.

[19] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[20] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[21] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed autonomous robotic systems 5*, pages 299–308. Springer, 2002.

[22] Elizabeth Howell. Van Allen radiation belts: Facts & findings. `https://www.space.com/33948-van-allen-radiation-belts.html`, May 11 2018. Accessed: January 15 2024.

[23] Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):14413–14423, 2020.

[24] Aanya Jindal. Angular sweep (maximum points that can be enclosed in a circle of given radius). GeeksforGeeks [online], February 2023. URL `https://www.geeksforgeeks.org/angular-sweep-maximum-points-can-enclosed-circle-given-radius/`.

[25] Richard M. Karp and Michael Luby. Monte-Carlo algorithms for the planar multiterminal network reliability problem. *Journal of Complexity*, 1(1):45–64, October 1985. ISSN 0885064X. doi: 10.1016/0885-064X(85)90021-4.

[26] Dominik Krupke, Maximilian Ernestus, Michael Hemmer, and Sándor P. Fekete. Distributed cohesive control for robot swarms: Maintaining good connectivity in the presence of exterior forces. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 413–420, September 2015. doi: 10.1109/IROS.2015.7353406.

[27] Jiahao Li, Yuhao Tao, Kai Yuan, Rongxin Tang, Zhiming Hu, Weichao Yan, and Shiyun Liu. Fruchterman–Reingold hexagon empowered node deployment in wireless sensor network application. *Sensors*, 22(14):5179, 2022.

[28] Nimrod Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983. doi: 10.1137/0212052.

[29] Daniel Morgan, Soon-Jo Chung, Lars Blackmore, Behcet Acikmese, David Bayard, and Fred Y Hadaegh. Swarm-keeping strategies for spacecraft under J_2 and atmospheric drag perturbations. *Journal of Guidance, Control, and Dynamics*, 35(5):1492–1506, 2012.

[30] Roger Paredes, Leonardo Dueñas-Osorio, Kuldeep S. Meel, and Moshe Y. Vardi. Principled network reliability approximation: A counting-based approach. *Reliability Engineering & System Safety*, 191:106472, 2019.

[31] Xiaogang Qi, Zhinan Li, Chen Chen, and Lifang Liu. A wireless sensor node deployment scheme based on embedded virtual force resampling particle swarm

optimization algorithm. *Applied Intelligence*, 52(7):7420–7441, 2022.

[32] Fabrice Saffre, Hannu Karvonen, and Hanno Hildmann. Wild swarms: Autonomous drones for environmental monitoring and protection. In *International conference on FinDrones*, pages 1–32. Springer, 2023.

[33] Y. Shpungin. Combinatorial approach to reliability evaluation of network with unreliable nodes and unreliable edges. *International Journal of Computer Science*, 1(3):177–183, 2006.

[34] Godfried T. Toussaint. Computing largest empty circles with location constraints. *International Journal of Computer & Information Sciences*, 12: 347–358, 1983.

[35] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, volume 555 of *Lecture Notes in Computer Science*, pages 359–370. Springer, Berlin, Heidelberg, 1991.

[36] A. M. Yaglom and I. M. Yaglom. *Challenging mathematical problems with elementary solutions*, volume I, Combinatorial Analysis and Probability Theory. Holden-Day, San Francisco, 1964. Translated by J. McCawley, Jr., revised and edited by B. Gordon.

[37] Jiong Yang and Kuldeep S. Meel. Rounding meets approximate model counting. In *International Conference on Computer Aided Verification*, pages 132–162. Springer, 2023.