Node Placement to Maximize Reliability of a Communication Network with Application to Satellite Swarms

Calum Buchanan, James Bagrow, Puck Rombach
Dept. of Mathematics & Statistics
University of Vermont
Burlington, VT USA
calum.buchanan@uvm.edu
james.bagrow@uvm.edu
puck.rombach@uvm.edu

Hamid Ossareh
Dept. of Electrical and Biomedical Engineering
University of Vermont
Burlington, VT USA
hossareh@uvm.edu

***Abstract*—The structure of a mobile ad hoc network changes dynamically based on node positioning. We consider a setting in which nodes can communicate if they are within a prescribed distance of one another, giving rise to a communication network. An example is a swarm of small satellites that cooperate to perform tasks; such swarms are likely to become commonplace in space missions. In this paper, we consider the problem of adding a new node or repositioning a current node in the network while optimizing a given network parameter such as network reliability. Although there are infinitely many locations to place the new node in space, there are only finitely many possible changes to the communication network. We provide an algorithm that enumerates all possible network changes in time $O(n^2 \log n)$ or $O(n^3 \log n)$, for networks in 2- or 3-dimensional Euclidean space, respectively. We apply the proposed algorithm to a satellite swarm formation planning problem, where the goal is to maximize network reliability.**

***Index Terms*—network reliability, all-terminal reliability, geometric graph**

I. Introduction

A communication network between objects whose locations are known can often be modeled by a geometric graph whose vertices are labeled by the locations of the objects in 2- or 3-dimensional Euclidean space, and whose edges connect vertices within a certain communication radius r. When the radius is 1, these are known as *unit-disk graphs* in two dimensions [1], and *unit-ball graphs* in three (sometimes known as unit-disk or unit-ball intersection graphs [2]). By scaling, all geometric graphs considered in this paper are equivalent to unit-disk or unit-ball graphs.

As a recurring example, we consider the formation planning of satellite swarms. Swarms of small satellites have recently received considerable attention in the literature due to their increased flexibility, reliability, and reduced costs as compared to monolithic missions (see, *e.g.*, [3]). Each satellite is equipped with the ability to communicate with other satellites within a certain distance, and the swarm must work together to accomplish a certain task. The communication type may be radio communication or optical communication, but we assume that all satellites are equipped with the same communication device, and thus that any two satellites which are within distance 1 (after spatial scaling) should be able to communicate. The geometric graph induced by the satellites and the communication links between them is, of course, dynamic; the satellites are constantly moving. However, there are many natural questions to ask about a static graph, representing the formation at a given moment, which are useful in analyzing the more complicated setting of a dynamic network. For instance, the satellites and the communication links between them are not perfectly reliable: an obstructing object, communication failure, or hardware issue may render edges or vertices in the unit-ball graph inoperable. To model these uncertainties, we assign a probability of operation to each vertex and edge in the unit-ball graph given by the formation. We use this model to analyze the reliability of the formation, or the probability that a message can be passed from any operational satellite to any other. The objective of this paper is to provide an algorithm to enumerate all possible changes to the network based on the addition of a single satellite, and to apply this to find the position of a new satellite which maximizes the reliability of the communication network.

A first step towards this objective is to understand how a set of balls of equal radius partitions 2- or 3-dimensional space. Each region of intersection of a subset of the balls corresponds to a neighborhood that a new vertex could have if added to the associated unit-ball graph. It is known that n circles partition the plane into at most $n^2 - n + 2$ regions and that n spheres partition space into at most $(n^3 - 3n^2 + 8n)/3$ regions [4]. However, to the best of our knowledge, there does not appear to be an algorithm in the literature for enumerating these regions. In Section III, we provide such an algorithm with $O(n^2 \log n)$ or $O(n^3 \log n)$ complexity, depending on the dimension. We enumerate these regions via an angular sweep

This material is based upon work that was supported by NASA under cooperative agreement VT-80NSSC20M0213.

around each center p_i of a circle in $\mathbb{R}^2$, or pair of centers p_i, p_j of spheres in $\mathbb{R}^3$, with a circle or sphere of the same radius. We record other centers as they enter or exit the ball during the sweep in order to list all of the sets of centers which are contained in a ball with $p_i \in \mathbb{R}^2$ or $p_i, p_j \in \mathbb{R}^3$ on its boundary (see Figure 1). The balls which correspond to these sets of interior points, as well as the balls corresponding to the interior and boundary point(s), intersect nontrivially. We obtain all of the nontrivial intersections in this way.

Having enumerated the regions into which a set of balls partitions space, we then determine a point to add a vertex in each region. Motivated by our application, we choose the point which minimizes the maximum distance to one of its neighbors. Finding this point reduces to the smallest enclosing ball problem, which can be solved in expected linear time in the number of balls defining the region using Welzl's algorithm [5] or the methods of Gärtner and others [6]–[8]. In summary, we provide a list of all unit-ball graphs obtainable by adding a single vertex to an existing unit-ball graph G. Applying this algorithm to each of the graphs obtained by deleting a vertex from G, we list all of the unit-ball graphs obtained by repositioning a vertex of G.

We have noted that the methods proposed here are useful for designing and maintaining satellite formations with reliable communication networks. There are a number of other applications in which one would like to alter a unit-ball graph, meeting certain constraints, to maximize a quantity such as reliability. Probabilistic unit-disk and unit-ball graphs have been used to model radio-broadcast networks [9], multi-hop networks [10], wireless sensor networks [11], [12], flying ad hoc networks [13], mobile ad hoc networks [14], and mobile wireless sensor networks [15]. Our algorithm for enumerating the neighborhoods for a new vertex in a geometric graph can be used to alter any such network with mobile nodes, and, commonly, network reliability is of interest in these applications. The authors of [16] provide an algorithm to reduce the diameter of a unit-disk graph via node addition, with the motivation of increasing the reliability and efficiency of a wireless network. In comparison, our algorithm provides an exact solution to the problem of maximizing the reliability of a unit-disk or unit-ball graph via addition of a single vertex. In our application to satellite communication networks, while the locations of some satellites are predetermined to accomplish a specific task, there may be redundant satellites in the formation which can relocate in order to maximize the reliability of the network. We use Monte Carlo simulations along with our algorithm to determine the location for a single satellite to travel which maximizes the probability that any two operational satellites can communicate. In Section IV, we elaborate on Monte Carlo simulations of reliability. Finally, in Section V, we demonstrate the algorithms in Section III and IV on a case study of a satellite swarm.

II. Preliminaries and problem statement

A graph G is a pair $G = (V, E)$, where V is a set of vertices (or nodes), and E is a set of unordered pairs of vertices, called edges. The graph G is called a *unit-disk graph* if $V \subset \mathbb{R}^2$ and $uv \in E$ whenever the u and v are of distance at most 1 apart; in $\mathbb{R}^3$, this is called a *unit-ball graph*. As the edges of a unit-disk or unit-ball graph are determined by V, we will say that V *induces* the unit-disk or unit-ball graph G. We use the term *geometric graph* to mean a graph whose vertices are points and whose edges connect vertices within a specified distance r. The *open neighborhood* of a vertex v in G, denoted $N(v)$, is the set of vertices $\{u \mid uv \in E\}$; the cardinality of $N(v)$ is the *degree* of v, denoted $d(v)$. We denote by $G - v$ the graph obtained by deleting the vertex v and its incident edges from G, and, for a point p which is not in V, we denote by $G + p$ the unit-disk or unit-ball graph induced by $V \cup \{p\}$. A *path* is a graph whose edge set, under some ordering of the vertices $v_0, \ldots, v_l$, is $\{v_{i-1}v_i \mid i \in \{1, \ldots, l\}\}$; we say that vertices x and y are joined by a path if G contains a path with $x = v_0$ and $y = v_l$. If G is connected (*i.e.*, any two vertices are joined by a path), then a set of edges whose removal disconnects G is called an *edge-cut*. A graph H on V whose edge set is a subset of E is called a *spanning subgraph* of G; if H is connected and contains no cycles, it is called a *spanning tree*.

There are varying notions of the robustness of a graph G whose edges and/or vertices are not perfectly reliable (see [17] for an overview). We concern ourselves with the robustness of the connectedness of G: how likely is G to remain connected when its edges and vertices are subject to failure? In particular, we assign a probability of operation to each vertex v and edge e in G, denoted by p_v and p_e, respectively. A *probabilistic graph* $G^{\Pr}$ is a random graph sampled by deleting each vertex v in G (and its incident edges) independently with probability $1-p_v$, and each remaining edge e independently with probability $1 - p_e$. The *residual connectedness of* G, denoted $\mathrm{Rel}(G)$, is the probability that $G^{\Pr}$ is connected.[1] While $\mathrm{Rel}(G)$ is the most relevant reliability measure for many applications, such as distributed computer network performance [21], or satellite formation planning in our case, it is often difficult to work with as it lacks monotonicity. That is, the connectedness of a subgraph of $G^{\Pr}$ does not imply the connectedness of $G^{\Pr}$ itself, and the disconnectedness of $G^{\Pr}$ does not imply the disconnectedness of all of its subgraphs. For this reason, residual connectedness is less well-studied than its monotonic counterpart, the *all-terminal reliability of* G, $\mathrm{Rel}_A(G)$, which is the probability that all vertices operate (no vertex failure) and $G^{\Pr}$ is connected. We denote by $\mathrm{Fail}(G)$ and $\mathrm{Fail}_A(G)$ the values $1 - \mathrm{Rel}(G)$ and $1 - \mathrm{Rel}_A(G)$, respectively.

Letting $\mathcal{C}$ denote the collection of connected subgraphs of G and $\mathcal{S}$ the collection of connected spanning subgraphs of G, we have $\mathrm{Rel}(G) = \sum_{H \in \mathcal{C}} \mathbb{P}(G^{\Pr} = H)$ and $\mathrm{Rel}_A(G) = \sum_{H \in \mathcal{S}} \mathbb{P}(G^{\Pr} = H)$. Since $\mathcal{C}$ contains $\mathcal{S}$, it follows that $\mathrm{Rel}(G) \geq \mathrm{Rel}_A(G)$. Determining either of these quantities for an arbitrary graph G is known to be $\sharp\mathsf{P}$-complete [19], [22]. The same is true of 2-*terminal reliability*, the probability two specified vertices are joined by a path in $G^{\Pr}$, even for

[1] Residual connectedness was originally defined for graphs with perfectly reliable edges in [18] and [19], but is adapted to this more general case in [20].

unit-disk graphs [9] (and thus for unit-ball graphs as well). For this reason, a number of tractable upper and lower bounds have been determined. As a simple example, if $p_e = p$ for each edge e in G, and all vertices are reliable, then $\mathrm{Rel}(G) = \mathrm{Rel}_A(G)$, which is bounded below by $p^{n-1}(1-p)^{m-n+1}N$, where $m = |E|$ and N is the number of spanning trees in G (which can be calculated using Kirchhoff's Matrix-Tree Theorem [23]). On the other hand, if C is an edge-cut in G containing c edges, then $\mathrm{Fail}_A(G) \geq (1-p)^c$; a spanning subgraph of G cannot be connected if every edge in an edge-cut fails. While it is easier to produce upper and lower bounds for all-terminal reliability than for residual connectedness, both measures can be estimated using Monte Carlo simulation (see Section IV).

With this terminology at hand, we are able to state the main goal of this paper, which is to provide an algorithm to find the optimal location to add or move a vertex in a unit-ball graph in order to maximize $\mathrm{Rel}(G)$ or $\mathrm{Rel}_A(G)$. Our solution consists of: *i*) providing an algorithm, which has $O(n^3 \log n)$ complexity, to enumerate all of the possible graphs $G+v$ for a unit-ball graph G, described in Section III, and *ii*) comparing the reliability of these graphs using Monte Carlo methods, described in Section IV. In order to move a vertex, we simply perform steps *i* and *ii* on each of the subgraphs obtained by deleting a vertex from G.

III. Addition of a vertex to a geometric graph

As unit-disk graphs and unit-ball graphs are often used to model communication networks between objects, it is natural to ask how a graph parameter μ varies as new vertices (and corresponding edges) are added. From the answer to this problem one can deduce how μ varies when a vertex in a geometric graph G is repositioned. If an efficient means of calculating or estimating $\mu(G)$ is known, one can begin by making a list of all possible neighborhoods for a new vertex and then evaluate μ on each (abstract) graph obtained by adding a vertex with one of the neighborhoods listed. In Section III-A, we provide an algorithm to enumerate all of the possible neighborhoods for a new vertex. While the number of possible neighborhoods for a new vertex in an abstract n-vertex graph is exponential, the number of possible neighborhoods in a geometric graph is not. Our algorithm runs in $O(n^2 \log n)$ or $O(n^3 \log n)$ time, for $\mathbb{R}^2$ and $\mathbb{R}^3$ respectively, and these bounds can be improved with parameterized complexity. In Section III-B, we determine a point v_N, for each possible neighborhood N, which minimizes the maximum distance to a vertex in N. This provides a list of geometric graphs which can be obtained by adding a vertex to G. We also discuss the overall complexity of the algorithm, how it can be used to reposition a vertex in G, and how it might be extended.

A. Enumerating all possible neighborhoods for a new vertex added to a geometric graph

The problem of enumerating the neighborhoods that a new vertex can have if added to a unit-disk graph is equivalent to enumerating the non-empty intersections of a set of circles in $\mathbb{R}^2$, or spheres in $\mathbb{R}^3$ for a unit-ball graph. It is shown in [4] that n circles partition the plane into at most $n^2 - n + 2$ regions, and that n spheres partition space into at most $n(n^2-3n+8)/3$ regions. In this section, we provide a method to enumerate these regions. The 2-dimensional case is solved by Algorithms 1, 2, and 3, while the generalization of these algorithms to three dimensions is described in what follows.

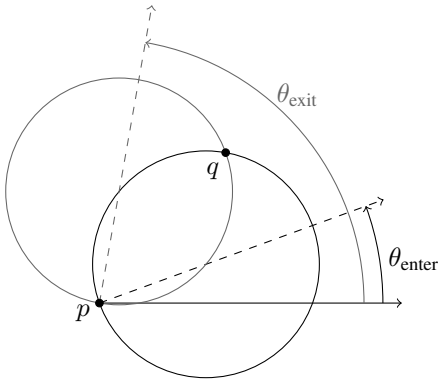


Fig. 1. A circle of radius 1 revolves in a counterclockwise motion around a fixed point p on its boundary in an angular sweep. A point q enters and exits the circle at the angles θ_{enter} and θ_{exit}, respectively (see Algorithm 1).

Algorithm 1 Determine the interval, in radians, during which a point q is contained in a circle of radius 1 during an angular sweep around p

Input: two points $p = (x_p, y_p), q = (x_q, y_q) \in \mathbb{R}^2$
Output: interval $(\theta_{\text{enter}}, \theta_{\text{exit}})$ during which q is contained in the circle of radius 1 sweeping around p

function AngSweepInterval(p, q)
  if $\mathrm{dist}(p, q) > 2$ **then**
    return "No interval."
  end if
  $x \leftarrow x_q - x_p$
  $y \leftarrow y_q - y_p$
  $A \leftarrow \arctan(y/|x|)$
  $B \leftarrow \arccos(\sqrt{x^2+y^2}/2)$
  if $x < 0$ **then**
    $\theta_{\text{enter}} \leftarrow \pi - (A + B)$
    $\theta_{\text{exit}} \leftarrow \pi - (A - B)$
  else
    $\theta_{\text{enter}} \leftarrow A - B$
    $\theta_{\text{exit}} \leftarrow A + B$
  end if
  return $(\theta_{\text{enter}}, \theta_{\text{exit}})$
end function

Let $p_1, \ldots, p_n$ denote the centers of a set of unit circles in $\mathbb{R}^2$ or a set of unit balls in $\mathbb{R}^3$. We assume the points to be generic, in that no three points lie on the boundary of a unit circle (in $\mathbb{R}^2$), and no four points lie on a unit ball (in $\mathbb{R}^3$). Here, we summarize our algorithm to list the regions into which these balls partition $\mathbb{R}^2$ or $\mathbb{R}^3$.

Algorithm 3, EnumerateRegions, takes as input the set

Algorithm 2 Obtain sets of points from angular sweep intervals

Input: a list L of pairs $(q, (\theta_{\text{enter}}, \theta_{\text{exit}}))$, where q is a point and $\theta_{\text{enter}}, \theta_{\text{exit}}$ are between $-\pi/2$ and $\pi/2$ radians

Output: a list of sets of vertices corresponding to overlapping intervals

```
1: function FindSets(L)
2:     Endpts ← []
3:     for (q, (θ_enter, θ_exit)) in L do
4:         append (θ_enter, q, type = "enter") to L
5:         append (θ_exit, q, type = "exit") to L
6:     end for
7:     sort Endpts by first coordinate
8:     Current_Set ← set of points with θ_exit < θ_enter
9:     Sets ← []
10:    for (θ, q, type) in Endpts do
11:        if type is "exit" then
12:            remove q from Current_Set
13:            append Current_Set with q appended to Sets
14:        end if
15:        if type is "enter" then
16:            append q to Current_Set
17:            append Current_Set to Sets
18:        end if
19:    end for
20:    if Sets = [] then
21:        return [∅]
22:    end if
23:    return Sets
24: end function
```

of points $V = \{p_1, \ldots, p_n\}$ and performs an angular sweep around each point p_i to determine the regions into which the circles or spheres partition space. The term angular, or radial, sweep is commonly used in 2-dimensional computational geometry to describe the effect of sweeping a circle of fixed radius around a point and recording other points as they enter and exit the circle (see Figure 1). The 2-dimensional version of our algorithm is based on such a technique, described in [24]. The angular sweep which we perform in three dimensions appears to be novel. The idea is as follows: given two points $p_i, p_j \in \mathbb{R}^3$ of distance no more than 2 apart, we choose a vector $\vec{v}_0$ which is perpendicular to the line segment joining p_i and p_j and which starts from the center c of this line segment. Associated to $\vec{v}_0$ is a unit ball for which *i*) both p_i and p_j lie on the boundary, and *ii*) $\vec{v}_0$ passes through the center of the ball. We then "rotate" this vector, and the associated unit ball, in a full revolution around the line segment joining p_i and p_j. Each time a point p_k enters or exits the ball during the sweep, we record the angle from the vector associated to this ball (passing from c through the ball's center) to $\vec{v}_0$. As such, we enumerate the unit balls which contain p_i, p_j, and one other point on their boundaries, and we keep track of the points which lie inside these balls.

Algorithm 3 Enumerate regions created by circles in $\mathbb{R}^2$

Input: points $p_1, \ldots, p_n$ in $\mathbb{R}^2$

Output: list of regions into which circles of radius 1 centered at $p_1, \ldots, p_n$ partition $\mathbb{R}^2$. Each non-empty region is denoted by the set of points whose circles define it.

```
1: function EnumerateRegions([p_1, ..., p_n])
2:     Regions ← [∅]
3:     for i ∈ {1, ..., n} do
4:         L ← []
5:         for j ∈ {1, ..., n} \ {i} do
6:             if dist(p_i, p_j) < 2 then
7:                 append AngSweepInterval(p_i, p_j) to L
8:             end if
9:         end for
10:        for R ∈ FindSets(L) do
11:            append R and {p_i} ∪ R to Regions
12:            remove the last element from R
13:            append R and {p_i} ∪ R to Regions
14:        end for
15:    end for
16:    remove duplicates from Regions
17:    return Regions
18: end function
```

We now revisit the problem in two dimensions and provide additional details. In two dimensions, we perform an angular sweep with a unit circle around each point p_i to determine the sets of points which are contained in a unit circle with p_i on its boundary. In three dimensions, we perform an angular sweep with a unit ball around each line segment of length at most 2 joining a pair of points p_i, p_j to determine the sets of points which are contained in a unit ball with p_i and p_j on its boundary. In order to determine these sets, we first make a list of the angles of rotation at which each point within sufficient distance enters and exits the circle or ball during the sweep using Algorithm 1, AngSweepInterval. Recording these angles is most easily done after a transformation which puts $p_i \in \mathbb{R}^2$ at the origin, or which puts $\overline{p_i p_j}$ on the y-axis and c at the origin in $\mathbb{R}^3$ (angles of rotation are taken with respect to the positive x-axis in both cases). We then use Algorithm 2, FindSets, to obtain a list of sets of points which are contained in the unit circle or sphere during the sweep at each moment that a point enters or exits.

In two dimensions, suppose that $S \subset V$ is the set of points contained in the interior of a unit circle with points p_i and p_j on its boundary. By shifting the circle slightly, one can obtain a unit circle which contains $S \cup \{p_i\}$, $S \cup \{p_j\}$, or $S \cup \{p_i, p_j\}$ in its interior. The existence of a circle containing a given set of points in its interior implies the existence of a region in $\mathbb{R}^2$ cut out by the unit circles around those points. Thus, for each set S of points returned by FindSets, we obtain four regions, corresponding to S, $S \cup \{p_i\}$, $S \cup \{p_j\}$, and $S \cup \{p_i, p_j\}$. Similarly, in three dimensions, a unit ball with

p_i, p_j, and p_k on its boundary and the subset S of V in its interior can be shifted so that S and any of the eight subsets of $\{p_i, p_j, p_k\}$ are in its interior; thus, we obtain all eight sets as regions formed by an intersection of unit balls around the points in question.

Having performed angular sweeps around all points in $\mathbb{R}^2$ (or pairs of points in $\mathbb{R}^3$), we thus obtain a list of regions formed by the intersections of the set of unit circles or spheres centered at these points. To show that this list is complete, we observe that the existence of a region R implies the existence of a circle or sphere of radius 1 containing all of the points in the subset S of V which defines R, and none in $V \setminus S$. We can shift such a circle until it has two points on its boundary, or a sphere until it has three; these points may or may not be contained in S. By again shifting the circle or sphere slightly, we can add any subset of the two or three points on its boundary to its interior. Thus, in at least one sweep around a point or pair of points on the boundary of a circle or sphere, we obtain S as one of the four or eight sets listed.

B. Adding or moving vertices in a geometric graph

Let G be a geometric graph on n vertices. Each region R returned by Algorithm 3 corresponds to a subset N of vertices which could be the neighbors of a new vertex. We have proposed this algorithm as a means of optimizing a graph parameter, such as network reliability, by adding or moving a single node. From a list of all possible neighborhoods that a new vertex could have, one directly obtains a list of (abstract) graphs of which to check the reliability. In order to construct a geometric graph corresponding to the optimal new graph, however, we require a specific point to place a new vertex.

A natural candidate for this point is the center of the smallest circle or sphere enclosing the points in N. This is also the point which minimizes the maximum distance to a point in N. Shamos and Hoey describe a method to determine the smallest enclosing circle for k points in $O(k \log k)$ time [25], which was improved to an $O(k)$ algorithm in [26]. In three dimensions, a randomized method is presented in [5], which solves the problem in expected $O(k)$ time, and Gärtner [6], [7] provides methods which are not linear, but are efficient in practice. For our application to reliability, we only need to apply such a method once in order to place the vertex which yields the most reliable unit-ball graph, which we find using Monte Carlo simulations of abstract graphs. For other applications, one can produce locations for new vertices corresponding to each possible neighborhood returned by Algorithm 3 in $O(n^3)$ or $O(n^4)$ time, depending on whether G lives in $\mathbb{R}^2$ or $\mathbb{R}^3$. However, we note that this is an overestimate in many cases. In these types of geometric graphs, vertex degrees do not tend to grow linearly with n. If we let Δ be the largest number of nodes contained in any ball of radius 2, then *i*) the number of regions formed by the unit balls centered at the nodes of G can be no larger than $2n\Delta$ in $\mathbb{R}^2$ or $2n^2\Delta$ in $\mathbb{R}^3$, and *ii*) the number of nodes defining any given region is at most Δ. The former observation follows from the fact that each ball intersects at most Δ others and can be shifted until it has one or two points on its boundary, in $\mathbb{R}^2$ or $\mathbb{R}^3$, respectively. Thus, the time complexity can be expressed as $O(n\Delta^2)$ or $O(n^2\Delta^2)$, depending on the dimension.

We have presented here a method to determine a list of locations which describe all of the possible geometric graphs obtainable by adding a vertex to G. Further, we place this new vertex in the location which minimizes the maximum distance to one of its neighbors. This choice is a reasonable one for many applications, but we note that this is not the only location one could place a vertex in a region. We can also use this method to move a vertex v in G by applying Algorithm 3 to $G - v$. We will revisit the significance of this in Section V.

IV. Network reliability using a Monte Carlo approach

In this section, we apply the algorithm descibed in Section III to network reliability problems on a geometric graph G. We recall the residual connectivity $\mathrm{Rel}(G)$ and the all-terminal reliability $\mathrm{Rel}_A(G)$, described in Section II. Both of these measures are ♯P-complete to compute, and so it is common to estimate them using Monte Carlo simulation. We consider a crude Monte Carlo scheme where, at each trial, we generate a probabilistic graph G^{Pr} and check if it is connected. More sophisticated Monte Carlo algorithms are available in the literature (see [20], [27]–[30]). These are "homogeneous" Monte Carlo schemes, meaning that the relative error is bounded, which perform well on highly reliable graphs. For our application and many others, the network reliability is bounded above by a constant depending on the reliability of the individual nodes and edges and on the density of the graph. Thus, the relative error is also bounded in these applications, determining the maximum number of crude Monte Carlo simulations required to estimate the reliability. This allows one to efficiently maximize the reliability of a geometric graph via node placement using a crude Monte Carlo scheme.

The "Zero-One Estimator Theorem" of Karp and Luby [29] determines the number of trials necessary to obtain an estimate to within an ϵ factor of $\mathrm{Fail}(G)$ with probability $1 - \delta$. Let Y denote the average value of N Monte Carlo simulations for $\mathrm{Fail}(G)$.

Lemma 1 (Zero-One Estimator Theorem)**.** *Let G be a graph, let μ be either $\mathrm{Fail}(G)$ or $\mathrm{Fail}_A(G)$, and let $\epsilon, \delta > 0$ with $\epsilon \leq 1 - \mu$. If $N > \frac{4.5 \ln(2/\delta)}{\mu\epsilon^2}$, then*

$$\Pr\left(\frac{|Y - \mu|}{\mu} < \epsilon\right) \geq 1 - \delta.$$

In order to make sense of Lemma 1, we require a lower bound on $\mathrm{Fail}(G)$ or $\mathrm{Fail}_A(G)$. We note that, if each edge in an edge-cut C of G fails, and if the endpoints of the edges in C are operational, then G^{Pr} is disconnected. Thus, $\mathrm{Fail}(G) \geq \prod_{e \in C}(1 - p_e) \prod_{v \in V(C)}(1 - p_v)$, where $V(C)$ denotes the set of vertices incident to an edge in C. While a graph can have exponentially many edge-cuts, an edge-cut of minimum cardinality can be found in $O(n^3)$ time [31], [32]. This does not change the order of the complexity of adding or moving a

vertex to maximize the reliability of a unit-ball graph, for our algorithm in Section III-A runs in $O(n^3 \log n)$ time. Although it is not always the case, an edge-cut with a small number of edges is typically more likely to fail than a larger one, giving a better lower bound on $\mathrm{Fail}(G)$. Since $\mathrm{Fail}_A(G) \geq \mathrm{Fail}(G)$, it suffices to conduct

$$N = \frac{4.5 \ln(2/\delta)}{c\epsilon^2}, \tag{1}$$

Monte Carlo trials to produce an estimate of $\mathrm{Fail}(G)$ or $\mathrm{Fail}_A(G)$ with relative error at most ϵ with probability $1-\delta$, where $c = \prod_{e \in C}(1-p_e)\prod_{v \in V(C)}(1-p_v)$.

In theory, as c tends to zero, the number of trials needed to estimate $\mathrm{Fail}(G)$ or $\mathrm{Fail}_A(G)$ with a fixed relative error ϵ grows large. However, for our applications, a small absolute error suffices for small $\mathrm{Fail}(G)$. For instance, if we would only like to know whether or not $\mathrm{Rel}(G) > .95$, and if Monte Carlo simulations estimate $\mathrm{Rel}(G)$ to be .96, then a relative error bound of .01 corresponds to an absolute error bound of .0004, which is much smaller needed to compare these numbers.

In our application to satellite formation planning in Section V, we compare the reliabilities of various graphs. If the difference between their reliabilities is large, we can determine which graph is most reliable using a large value of ϵ. If only a few graphs are good candidates for being most reliable, then we can increase efficiency by only applying Monte Carlo schemes with small values of ϵ to these graphs.

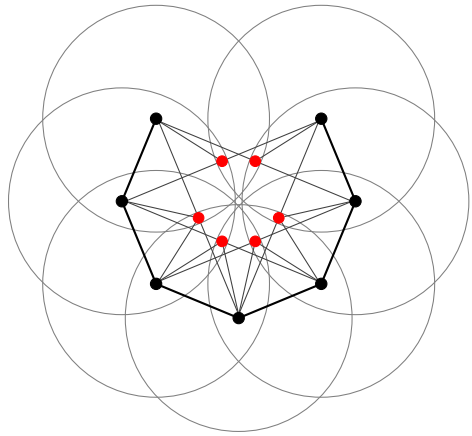
V. Application to Autonomous Satellite Swarms

In this section, we describe an application of our algorithm to satellite swarms and autonomous formation planning. Swarms of small satellites which cooperate to perform tasks are likely to become commonplace in future space missions [33]–[35].

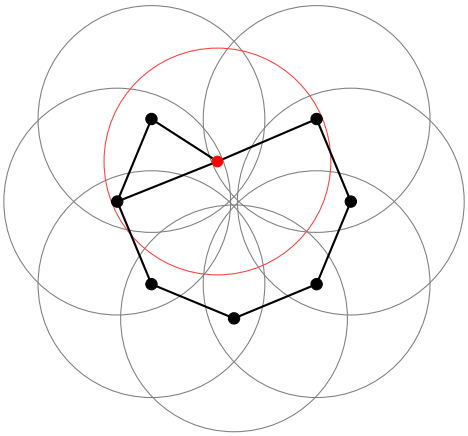
Consider a swarm of satellites, each able to communicate with other satellites within a certain distance r. The locations of the satellites at a given moment, with respect to a reference satellite and scaled down by a factor of r, induce a unit-ball graph which represents the communication network. Communication failures or hardware reliability issues may render communication links, or the satellites themselves, inoperable. It is thus important to optimize the reliability of unit-ball graphs in the event of random failures.

We recall the previously discussed reliability measures $\mathrm{Rel}(G)$ and $\mathrm{Rel}_A(G)$. The former represents the general case, in which both satellites and communication links may fail, while the latter represents the simpler case in which satellites are perfectly reliable, and only the communication links are subject to failure. Given that the swarm is to accomplish some task, such as distributed imaging or in-space assembly of an object, we assume that the positions of at least some of the satellites in the formation are fixed. The algorithm proposed in Section III and the Monte Carlo methods in Section IV provide a means of placing or moving redundant satellites in the formation in order to maximize either $\mathrm{Rel}(G)$ or $\mathrm{Rel}_A(G)$.

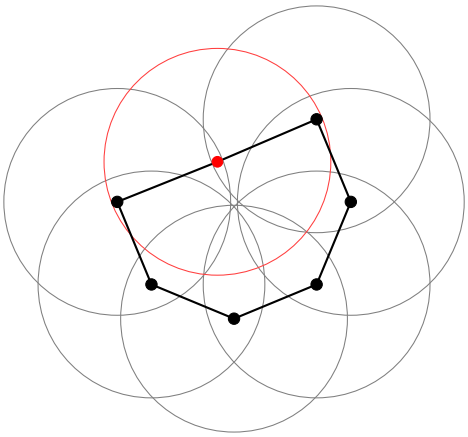
In Figure 2, an example formation is depicted by a unit-disk graph. Of course, satellite formations induce 3-dimensional unit-ball graphs, but we depict one in 2D for simplicity; the same discussion holds in generality. The black vertices correspond to satellites whose locations are fixed for a certain task. The grey circle around each black satellite is of communication radius r. We assume that all satellites are perfectly reliable and that each edge has a 99% probability of operation.



(a) Possible (maximal) neighborhoods



(b) A most reliable candidate



(c) Most reliable location to move a vertex

Fig. 2. Adding a vertex to a unit-disk graph in order to maximize reliability.

Suppose that there is an eighth satellite in the formation,

which is to be placed to maximize the reliability of the network. Using Algorithm 3, we can enumerate all of the possible neighborhoods that this new vertex can have. In Figure 2a, the neighborhoods of the red vertices are the neighborhoods returned by Algorithm 3 which are maximal with respect to subset inclusion. The locations of the red vertices are the centers of the smallest enclosing circles for these maximal neighborhoods. Using the methods from Section IV, we can determine the choice which maximizes the reliability, depicted in Figure 2b.

If H denotes the graph induced by the black vertices in Figure 2, and the probabilities of operation are as before, it is not hard to calculate that $\mathrm{Rel}_A(H) = .99^6$. Thus $\mathrm{Fail}_A(G) < .067$ for each graph G induced by H and a single red vertex in Figure 2a. Choosing $\epsilon = .05$ and $\delta = .01$ and applying Lemma 1 gives, with 99% probability, an estimate of each failure probability $\mathrm{Fail}_A(G)$ to within $.34\%$ of the true value in at most $8,650$ trials. The top two red vertices in Figure 2a each induce graphs with estimated reliability about 99.8%, while the bottom four red vertices induce graphs with estimated reliability about 96.7%. The difference between these values, compared with the bound on the error of our estimate, shows that with 99% confidence, the most reliable choices for the new vertex are the top two red vertices.

Our methods can be used to determine all of the unit-ball graphs obtainable by repositioning a single vertex. Indeed, one can do so by applying Algorithm 3 EnumerateRegions to each graph obtained by deleting a vertex and its incident edges from G and finding the points which minimize the maximum distance to a vertex in each region. In our application, this can be parallelized by having each satellite v perform Algorithm 3 and the Monte Carlo simulations described in Section IV, on the geometric graph $G - v$. By parallelizing, the swarm can determine in $O(n^3 \log n)$ time the optimal location to move a single satellite in order to maximize the reliability.

Moving the upper-leftmost black vertex to the same red position (see Figure 2c) is also the best place to move an existing vertex in H in order to maximize the all-terminal reliability. The reliability of the resulting cycle is about 99.8%, where as the optimal position to move any other vertex in H results in a graph which is about 96.2% reliable using the same values of ϵ and δ for the Monte Carlo simulations.

In general, adding nodes to a unit disk graph within a bounded region adds connections and improves reliability. We compare our method to a random addition of new nodes to a random unit disk graph. First, we compare the reliability of a random unit disk graph on n nodes, conditioned on connectedness, to the reliability of the graph obtained by repositioning one node (selected uniformly at random) using our algorithm to optimize reliability. We take a uniform random geometric graph in a square of size 2.5×2.5. For the analysis, we let $p = 90\%$, $\epsilon = .2$, $\delta = .05$, and we let n range from 6 to 20. We average over one hundred trials. The results are shown in Figure 3. For example, repositioning one randomly selected node in a unit disk graph on ten nodes is more effective than doubling the size of the swarm. Using

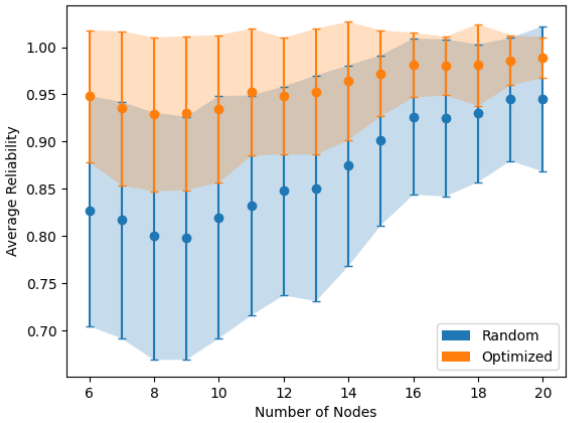


Fig. 3. A comparison of the reliability of random unit disk graphs (conditioned on connectedness) to the same graphs where one node has been repositioned for improved reliability. The standard deviation is also depicted.

our algorithm to place a single new vertex in an optimal position improves the network reliability significantly. One can achieve the same improvement in reliability by adding multiple new vertices uniformly at random within the convex hull of the graph. Based on one hundred simulations, we found that more than four random vertices were needed to match the reliability of a single optimal vertex added to a 10-vertex random geometric graph with the above parameters.

VI. Conclusions and Future Work

In this paper, we addressed the problem of altering a communication network by moving a single node in order to maximize network reliability. We did so in the context of formation planning of satellite swarms, which we represent by unit-ball graphs. Although there are infinitely many locations to place a new node in space, there are only finitely many possible changes to the network; we provided an algorithm to determine all possible changes in time $O(n^2 \log n)$ or $O(n^3 \log n)$, for graphs in 2- or 3-dimensional Euclidean space, respectively. This algorithm was used alongside existing methods from the literature to find the precise location to move a node which minimizes the maximum distance to one of its neighbors, and to compare the reliability of the possible resulting networks using Monte Carlo simulation.

It is reasonable to ask further about the movement of two or more vertices in a unit-ball graph, instead of just one. Our method can be used to add or redirect satellites one by one, while other satellites remain static. However, this is not always optimal. Consider a long path in the shape of an arc, such as the one induced by the black vertices in the unit-disk graph depicted in Figure 4. With two additional vertices, shown in red, one can connect the two endpoints of this path. The resulting cycle has a higher all-terminal reliability than any of the graphs obtainable by adding vertices one at a time to maximize the reliability at each step. New methods will be

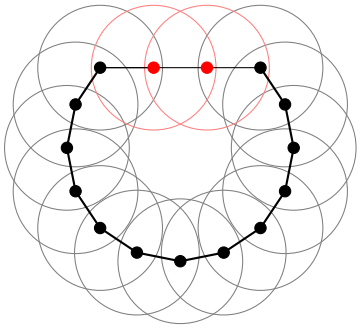


Fig. 4. A unit-disk graph (induced by the black vertices) for which adding two (red) vertices simultaneously results in a more reliable graph than iteratively maximizing the reliability by adding vertices one at a time.

required to address these types of problems when multiple satellites are in motion simultaneously.

REFERENCES

[1] A. Brandstädt, V. B. Le, and J. P. Spinrad, *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898719796

[2] P. Hliněný and J. Kratochvıl, "Representing graphs by disks and balls (a survey of recognition-complexity results)," *Discrete Mathematics*, vol. 229, no. 1-3, pp. 101–124, 2001.

[3] H. Basu, Y. Pedari, M. Almassalkhi, and H. R. Ossareh, "Fuel-optimal trajectory planning of satellites using minimum distance assignment and comparative analysis of satellite relative dynamics," in *in Proceedings of the 2022 American Control Conference (ACC)*, Atlanta, USA, 2022, pp. 1–7.

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

[5] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," in *New Results and New Trends in Computer Science*, ser. Lecture Notes in Computer Science, H. Maurer, Ed., vol. 555. Springer, Berlin, Heidelberg, 1991, pp. 359–370.

[6] B. Gärtner, "Fast and robust smallest enclosing balls," in *Algorithms-ESA'99: 7th Annual European Symposium Prague, Czech Republic, July 16–18, 1999 Proceedings 7*. Springer, 1999, pp. 325–338.

[7] B. Gärtner and S. Schönherr, "An efficient, exact, and generic quadratic programming solver for geometric optimization," in *Proceedings of the 16th Annual Symposium on Computational geometry*, 2000, pp. 110–118.

[8] K. Fischer, B. Gärtner, and M. Kutz, "Fast smallest-enclosing-ball computation in high dimensions," in *Algorithms-ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings 11*. Springer, 2003, pp. 630–641.

[9] H. AboElFotoh and C. Colbourn, "Computing 2-terminal reliability for radio-broadcast networks," *IEEE Transactions on Reliability*, vol. 38, no. 5, pp. 538–555, Dec. 1989.

[10] G. Egeland and P. Engelstad, "The availability and reliability of wireless multi-hop networks with stochastic link failures," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 7, pp. 1132–1146, Sep. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5226965/

[11] M. Ahsan, M. Hasanuzzaman, A. Olabi, and M. Hashmi, "Review of the reliability and connectivity of wireless sensor technology," in *Comprehensive Materials Processing*. Elsevier, 2014, pp. 571–588. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780080965321013273

[12] L. A. Laranjeira and G. N. Rodrigues, "Border effect analysis for reliability assurance and continuous connectivity of wireless sensor networks in the presence of sensor failures," *IEEE Transactions on Wireless Communications*, vol. 13, no. 8, pp. 4232–4246, 2014.

[13] I. Bekmezci, M. Ermis, and S. Kaplan, "Connected multi uav task planning for flying ad hoc networks," in *2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2014, pp. 28–32.

[14] N. Padmavathy and S. K. Chaturvedi, "Evaluation of mobile ad hoc network reliability using propagation-based link reliability model," *Reliability Engineering & System Safety*, vol. 115, pp. 1–9, 2013.

[15] S. Chakraborty, N. K. Goyal, and S. Soh, "On area coverage reliability of mobile wireless sensor networks with multistate nodes," *IEEE Sensors Journal*, vol. 20, no. 9, pp. 4992–5003, 2020.

[16] M. Di Ianni, L. Gualà, and G. Rossi, "Reducing the diameter of a unit disk graph via node addition," *Information Processing Letters*, vol. 115, no. 11, pp. 845–850, 2015.

[17] C. J. Colbourn, *The combinatorics of network reliability*. Oxford University Press, Inc., 1987.

[18] F. Boesch, A. Satyanarayana, and C. Suffel, "On residual connectedness network reliability," *Reliability of Computer and Communication Networks (DIMACS 5), New Brunswick, New Jersey, USA*, pp. 51–59, 1991.

[19] K. Sutner, A. Satyanarayana, and C. Suffel, "The complexity of the residual node connectedness reliability problem," *SIAM Journal on Computing*, vol. 20, no. 1, pp. 149–155, 1991.

[20] Y. Shpungin, "Combinatorial approach to reliability evaluation of network with unreliable nodes and unreliable edges," *International Journal of Computer Science*, vol. 1, no. 3, pp. 177–183, 2006.

[21] Z. He, Y. Tian, and Y. Chen, "Simulating the reliability of distributed systems with unreliable nodes," *IJ of Simulation*, vol. 3, no. 1-2, pp. 68–79, 2002.

[22] A. Rosenthal, "Computing the reliability of complex networks," *SIAM Journal on Applied Mathematics*, vol. 32, no. 2, pp. 384–393, 1977.

[23] G. Kirchhoff, "Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird," *Annalen der Physik*, vol. 148, no. 12, pp. 497–508, 1847.

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

[25] M. I. Shamos and D. Hoey, "Closest-point problems," in *16th Annual Symposium on Foundations of Computer Science*, 1975, pp. 151–162.

[26] N. Megiddo, "Linear-time algorithms for linear programming in r^3 and related problems," *SIAM journal on computing*, vol. 12, no. 4, pp. 759–776, 1983.

[27] I. B. Gertsbakh and Y. Shpungin, *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo*, 0th ed. CRC Press, Apr. 2016. [Online]. Available: https://www.taylorfrancis.com/books/9781439817421

[28] P. Dagum, R. Karp, M. Luby, and S. Ross, "An optimal algorithm for monte carlo estimation," *SIAM Journal on computing*, vol. 29, no. 5, pp. 1484–1496, 2000.

[29] R. M. Karp and M. Luby, "Monte-Carlo algorithms for the planar multiterminal network reliability problem," *Journal of Complexity*, vol. 1, no. 1, pp. 45–64, Oct. 1985. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0885064X85900214

[30] M. O. Ball, C. J. Colbourn, and J. S. Provan, "Chapter 11 Network reliability," in *Handbooks in Operations Research and Management Science*, ser. Network Models. Elsevier, Jan. 1995, vol. 7, pp. 673–762. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050705801288

[31] M. Stoer and F. Wagner, "A simple min cut algorithm," in *Algorithms—ESA'94: Second Annual European Symposium Utrecht, The Netherlands, September 26–28, 1994 Proceedings 2*. Springer, 1994, pp. 141–147.

[32] D. R. Karger and C. Stein, "A new approach to the minimum cut problem," *Journal of the ACM (JACM)*, vol. 43, no. 4, pp. 601–640, 1996.

[33] W. Truszkowski, M. Hinchey, J. Rash, and C. Rouff, "Nasa's swarm missions: The challenge of building autonomous software," *IT professional*, vol. 6, no. 5, pp. 47–52, 2004.

[34] S. Nag and L. Summerer, "Behaviour based, autonomous and distributed scatter manoeuvres for satellite swarms," *Acta Astronautica*, vol. 82, no. 1, pp. 95–109, 2013.

[35] C. J. Verhoeven, M. J. Bentum, G. Monna, J. Rotteveel, and J. Guo, "On the origin of satellite swarms," *Acta Astronautica*, vol. 68, no. 7-8, pp. 1392–1395, 2011.