Positioning Helper Nodes to Improve Robustness of Wireless Mesh Networks to Jamming Attacks

Jixin Feng, Warren E. Dixon, and John M. Shea
University of Florida, Gainesville, FL, Email:{fengjixin@ufl.edu, wdixon@ufl.edu, jshea@ece.ufl.edu}

***Abstract*—Wireless communication systems are susceptible to both unintentional interference and intentional jamming attacks. For mesh and ad-hoc networks, interference affects the network topology and can cause the network to partition, which may completely disrupt the applications or missions that depend on the network. Defensive techniques can be applied to try to prevent such disruptions to the network topology. Most previous research in this area is on improving network resilience by adapting the network topology when a jamming attack occurs. In this paper, we consider making a network more robust to jamming attacks before any such attack has happened. We consider a network in which the positions of most of the radios in the network are not under the control of the network operator, but the network operator can position a few "helper nodes" to add robustness against jamming. For instance, most of the nodes are radios on vehicles participating in a mission, and the helper nodes are mounted on mobile robots or UAVs. We develop techniques to determine where to position the helper nodes to maximize the robustness of the network to certain jamming attacks aimed at disrupting the network topology. Using our recent results for quickly determining how to attack a network, we use the harmony search algorithm to find helper node placements that maximize the number of jammers needed to disrupt the network.**

I. Introduction

Wireless networks are susceptible to both unintentional interference and intentional jamming attacks. Unintentional interference often comes from other networks using the same band and is usually mitigated through error-correction coding and medium-access control (MAC) protocols. Jamming (or intentional interference) [1]–[3] offers additional challenges because it is purposefully designed to disrupt communication and can be targeted at different layers of the protocol stack.

The vulnerability of wireless networks to jamming attacks motivates the need to develop countermeasures. Such countermeasures can be classified as providing either resilience or robustness [4]. Resilience is a reactive countermeasure, which characterizes a system's ability change its methods of operation to survive from an attack and to recover from external interference or disturbance. For instance, if a network detects a jamming attack, it can map the affected network region [5], adapt the topology to avoid those regions [6], [7], and recover network performance after the attack is finished [3].

In contrast to resilience, robustness is a proactive countermeasure. In the context of jamming attacks, a robust wireless network should be able to continue functioning under a jamming attack and not require any major system reconfiguration to adapt to the attack. For instance, one measure of robustness of a wireless network may be the number of jammers required for the network to be partitioned to some specified degree. Robustness can be achieved by adding redundancy into the network when designing the topology [8] or by reshaping the network into a new topology with more redundant routes without the knowledge of a specific jamming attack [9].

This research was supported by the National Science Foundation under grants 1217908 and 1642973.

In this paper, we propose techniques to improve the robustness of wireless networks to jamming attacks that are targeted at partitioning the network. We previously investigated jammer placement strategies to partition a wireless network in [10]–[12]. Here we consider how to make the network more robust by placing additional network "helper" nodes that create backup network links and hence increase the difficulty for the adversaries to partition the network by placing jammers. Our measure of robustness to jamming attacks is the number of jammers required to partition the network into a specified number of disconnected subnetworks. A meta-heuristic search algorithm called harmony search is used to search the space of possible helper node locations, where potential helper node placements are drawn from a probabilistic model that helps the network form additional useful routes around jammed areas.

This paper is organized as follows. The network model, jamming model, and our proposed defensive strategy are described in Section II. The algorithms used for optimizing helper node placement and evaluating our defensive strategy are formulated in Section III. The performance of our approach is assessed via simulation in Section IV, and the paper is concluded in Section V.

II. System Model and Problem Formulation

Initial network: The network model we consider is the same as in [11]. Radios are distributed in a two-dimensional plane. The radios are homogeneous, with equal transmit power, omnidirectional antennas, and equal noise figure. The channel is modeled using the exponential path-loss model. We consider the protocol model for reception, where two radios can directly communicate if the signal-to-noise ratio for communication between the radios is greater than a threshold, which corresponds to radios being within some Euclidean distance, R_c. The induced network topology is modeled as a simple graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with the radios as the vertices $\mathcal{V}$ and the communication link as the edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Each vertex V_i is associated the geographic location, $\mathrm{loc}(V_i) \in \mathbf{R}^2$, of the corresponding radio.

Jammers: We consider the jamming model from [11], in which each jammer is located near one of the nodes in the

network and blocks all communication to that node, as well as to all of that communication node's neighbors.

Helper Nodes: The network is to be augmented by a group of arbitrarily positionable radios that we call helper nodes. We assume that the helper nodes' radios have similar properties as the original network nodes. However, the communication radius for the helper nodes may be the same or larger than the other communication nodes. For instance, the helper nodes may be on UAVs, which results in better channel conditions, or may have higher transmission powers and better receivers.

A. Attack Objective

Jamming nodes are placed to partition the network. Let K be the minimum number of clusters and b_i be a bound on the number of nodes in cluster i. The problem of partitioning the network can then be viewed at a high level as finding a graph clustering scheme, which aims to find a partition

$$\Gamma = \left\{ \mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K \subset \mathcal{V} \middle| |\mathcal{V}_i| \leq b_i,\ \mathcal{V}_i \cap \mathcal{V}_j = \emptyset,\ i \neq j \right\},$$

To simplify exposition, in what follows we bound all the residual clusters by the same value $b_K = N/K$.

For non-trivial networks, searching for a minimum-cardinality edge separator or vertex separator that partitions $\mathcal{G}$ into K disconnected subsets requires a high computational complexity [13], [14]. Even searching for an approximate solution can be hard in certain cases [15]. In cases where finding the optimal network partition is not feasible, there are several viable suboptimal approaches. In our previous research [12], we developed a technique based on the combination of multilevel graph partitioning and a binary linear program, and the combination requires a tenth or even a hundredth of a second to find effective jammer placements in networks of up to 500 nodes.

B. Helper Node Placement Problem

The objective of this paper is to determine how to make the network robust to the type of jamming attacks described in Section II-A. We need a robustness metric that represents the difficulty for adversaries to perform jamming attacks aimed at disrupting the network topology. The metric should be comparable between different network topologies and sizes, and it should also be easily computed. Metrics like vertex connectivity, Fiedler value, and Cheeger constant [16], [17] have been widely used when analyzing network connectivity and robustness. All of them are able to indicate how "well-knit" the network is but also possess their own drawbacks. Vertex connectivity only measures the local property around each vertex of the network and Fiedler value can only indirectly estimate the global connectivity of the network. The Cheeger constant also represents the global connectivity of the network. Networks with small Cheeger constants have a "bottleneck", which consists of a small number of edges connecting two large connected components. However, calculation of this number is a $\mathcal{NP}$-Hard problem and the "bottleneck" only considers a single edge cut. Hence this metric has limited utility for the types of jamming attacks we consider. The robustness measurement we adopt in this paper is:

$$\eta(\mathcal{G}(\mathcal{V}, \mathcal{E})) = |\mathrm{OJS}(\mathcal{G}(\mathcal{V}, \mathcal{E}))|,$$

where $|\mathrm{OJS}(\mathcal{G})|$ is the cardinality of the Optimal Jamming Set, which is a number of jammers that can partition the network to achieve the objective in Section II-A. The details of the OJS are deferred until Section III-A.

We consider the following optimization problem for helper node placement. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be the network that is to be made more robust, and let $\mathcal{P} = \{P_1, P_2, \ldots, P_{N_h} \mid \mathrm{loc}(P_i) \in \mathbb{R}^2,\ i = 1, 2, \ldots, N_h\}$ be additional vertices that represent the N_h helpers. Each P_i has a location in $\mathbb{R}^2$. Let $\mathcal{H}(\mathcal{V}', \mathcal{E}', \mathcal{G}, \mathcal{P})$ be the reinforced network induced by placing helper nodes at the locations of the $P_i \in \mathcal{P}$. Then $\mathcal{H}$ is specified by vertices $\mathcal{V}'$ and edges $\mathcal{E}'$, which can be determined from:

$$\begin{aligned} \mathcal{V}' &= \mathcal{V} \cup \mathcal{P} \\ \mathcal{E}' &= \{(u, v) \in \mathcal{V}' \times \mathcal{V}' \mid \mathrm{dist}(u, v) \leq R_c\}. \end{aligned}$$

Then the helper node placement problem is

$$\hat{\mathcal{P}} = \arg \max_{\mathcal{P} \in (\mathbb{R}^2)^{N_h}} \eta(\mathcal{H}(\mathcal{V}', \mathcal{E}', \mathcal{G}, \mathcal{P}))). \tag{1}$$

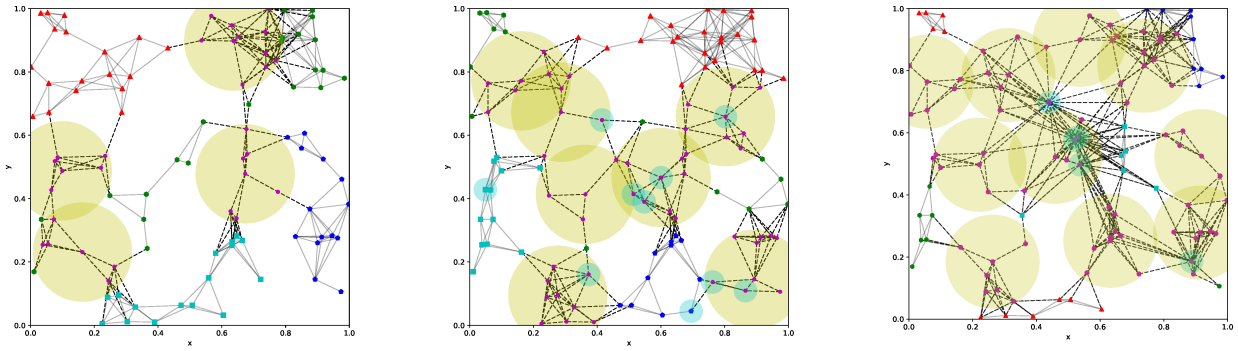
An example showing the result of placing helper nodes into a wireless network is shown in Fig. 1. The original network, shown in Fig. 1(a), is a 100-node network created as a random geometric graph [18]. This network requires 4 jammers to partition into 4 disconnected subnetworks with fewer than 25 radios in each. After adding 10 helper nodes with communication radius equal to the communication nodes in the original network, the more robust network is shown in Fig. 1(b), and requires 7 jammers to achieve the same level of partitioning. If the 10 helper nodes have twice the communication radius as the original communication nodes, then 10 jammers are required to partition the network to the same amount, as shown in Fig. 1(c).

III. Helper Node Placement Algorithms

A. Evaluation of Network Robustness

We use $\eta(\mathcal{G}(\mathcal{E}, \mathcal{V}))$ as a measure of network robustness to a jamming attack aimed at partitioning the network. The evaluation of $\eta(\mathcal{G}(\mathcal{E}, \mathcal{V}))$ can be divided into two phases: 1) finding a good edge separator $\mathcal{E}_S$ of network $\mathcal{G}$ via a computationally efficient method called multilevel balanced graph cut, and 2) finding a jammer placement solution of minimum cardinality that blocks communication along all edges in $\mathcal{E}_S$.

The multilevel balanced cut [19] aims to find a minimum edge separator $\mathcal{E}_S$ that partitions the given graph into k disconnected subgraphs with equal numbers of vertices. If the number of vertices in the original graph is not divisible by k, then the number of vertices in each subgraph may differ by



(a) Before placing helper nodes into the network, 4 jammers placed at the right locations are able to partition the network into at least 4 disconnected subnetworks with order fewer than one fourth of the original network order.

(b) New network routes are created after placing 10 helper nodes into the network. It now requires 7 jammers to partition the network with same objective.

(c) If the helper nodes have twice the communication range as the other nodes, after placing 10 helper nodes, it requires 10 jammers to partition the networks with the same objective.

Fig. 1. Network robustness against jamming attack before and after placing helper nodes. Jammed area are mapped by yellow shaded circles and jammers are placed at the center of each jammed area. A node is jammed if it is within the jammed area. Jammed nodes and network links are represented by magenta round dots and dashed lines respectively. Nodes and links not affected by jammers are represented by colored dots in various shapes and solid lines respectively.

one. Thus, the balanced k-way cut problem is

$$\begin{aligned} \min \quad & |\mathcal{E}_S| \\ \text{s.t.} \quad & \left\lfloor \frac{|\mathcal{V}|}{K} \right\rfloor \le |\mathcal{V}_i| \le \left\lceil \frac{|\mathcal{V}|}{K} \right\rceil \qquad i = 1, \dots, K \\ & \bigcup_{i=1}^{K} \mathcal{V}_i = \mathcal{V} \qquad i = 1, \dots, K \\ & \mathcal{V}_i \cap \mathcal{V}_j = \emptyset \qquad i, j \in \{1, \dots, K\},\ i \neq j \end{aligned}$$

Once an edge separator $\mathcal{E}_S$ is found, the optimal jamming set (OJS) for $\mathcal{E}_S$ in $\mathcal{G}$ can be found by solving an Integer Linear Program. Let $\mathcal{G}_S$ be the subgraph of $\mathcal{G}$ that is induced by the edge set $\mathcal{E}_s$, such that the vertex set of $\mathcal{G}_S$ is given by

$$\mathcal{V}_S = \{v \in \mathcal{V} \mid \exists e \in \mathcal{E}_S, e = (w, x) \ \ni \ v = w \cup v = x\},$$

and the edge set of $\mathcal{G}_S$ is $\mathcal{E}_S$. In our jamming model, each jammer is placed at the location of one of the vertices in $\mathcal{V}$. If $N_{\mathcal{G}_J}(v)$ denotes the neighbors of v in $\mathcal{G}_J$, then communication will be disrupted to all $u \in N_{\mathcal{G}_J}(v)$. For a vertex set $U \subset V$, let $\mathcal{N}_{\mathcal{G}_J}(U)$ be the closed neighborhood of the nodes in U, $\mathcal{N}_{\mathcal{G}_J}(U) = \{v \in V \mid v \in U \text{ or } \exists u \in U \ni v \in N_{\mathcal{G}_J}(u)\}$.

Let $\overline{\mathcal{G}_S}$ denote the subgraph of $\mathcal{G}$ induced by the vertex set $\mathcal{N}_{\mathcal{G}_J}(\mathcal{V}_S)$. Then, the minimum number of jammers required is the minimum cardinality subset $\mathcal{V}_O \subset \overline{\mathcal{G}_S}$ such that for every edge in $(u, v) \in \mathcal{E}_S$, at least one of u, v is dominated by a vertex in $\mathcal{V}_O$:

$$\begin{aligned} \mathcal{V}_O = \arg\min_{V \subset \mathcal{V}} & |V| \\ \text{s.t. } & \forall e \in \mathcal{E}_S, e = (w, x), \\ & \exists v \in V \ni \ w \in N_{\mathcal{G}_J}(v) \text{ or } x \in N_{\mathcal{G}_J}(v). \end{aligned}$$

We call $\mathcal{V}_O$ the optimal jamming set (OJS) for $\mathcal{E}_S$. The OJS can be found using a small integer linear program (ILP), as detailed in [12].

B. Search Placement Solutions

Once the robustness of the network can be quickly evaluated, it is possible to use these inside meta-heuristic search algorithms to search for optimal helper node placements. We propose to apply Harmony Search (HS) [20] to find helper node placement locations $\mathcal{P}$ for network $\mathcal{G}$ with $\text{obj}(\mathcal{P}) = \eta(\mathcal{H}(\mathcal{V}', \mathcal{E}', \mathcal{G}, \mathcal{P}))$, where $\mathcal{P}$ stands for candidate helper node placement solution – also called harmony – which has been introduced in Section II-B

The following terminology is used in the harmony search algorithm:

- HMS: Harmony Memory Size
- HM: Harmony Memory
- HMCR: Harmony Memory Considering Rate
- PAR: Pitch Adjusting Rate
- BW: Pitch Bandwidth
- ITR: Total Number of Iteration
- SOI: Current Step of Iteration i.
- $\mathcal{P}^{\text{best}|\text{worst}}$: Best/worst harmony stored in HM currently
- $\mathcal{P}^{\text{new}}$: New harmony "improvised" at current iteration

The steps of our HS can be summarized in Algorithm 1 and Algorithm 2. A series of random solutions/harmonies – placement locations – are generated to fill the HM during initialization. At iteration step i, a new harmony $\mathcal{P}_i^{\text{new}}$ is "improvised" by choosing some of its elements from HM, with a probable pitch controlled by PAR and BW, while some other elements are randomly generated with a probability controlled by HMCR. The worst harmony $\mathcal{P}^{\text{worst}}$ in HM is replaced if $\mathcal{P}_i^{\text{new}}$ is better. Hence, HM will be a pool of good solution

candidates, and their elements are more likely to appear in the new improvised harmony. HMCR, PAR and BW also added controlled randomness in the new harmony, giving HS the ability to jump away from local optima.

Data: $\mathcal{G}(\mathcal{V},\mathcal{E})$, HMS, ITR
Result: $\mathcal{P}^{\text{best}}$, $\text{obj}(\mathcal{P}^{\text{best}})$, $\mathcal{H}(\mathcal{V}',\mathcal{E}',\mathcal{G},\mathcal{P}^{\text{best}})$
begin
  Initialize $\text{HM} = [\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{\text{HMS}}]$
  while $i < \text{ITR}$ **do**
    Set $\mathcal{P} \in \text{HM}$ with $\min\ \text{obj}(\mathcal{P})$ as $\mathcal{P}^{\text{worst}}$
    New solution $\mathcal{P}^{\text{new}} = \text{Improvisation}()$
    Calculate $\mathcal{H}$, $\text{obj}(\mathcal{P}^{\text{new}})$
    if $\text{obj}(\mathcal{P}^{\text{new}}) > \text{obj}(\mathcal{P}^{\text{worst}})$ **then**
      Replace $\mathcal{P}^{\text{worst}}$ with $\mathcal{P}^{\text{new}}$
    end
  end
  Set $\mathcal{P} \in \text{HM}$ with $\max\ \text{obj}(\mathcal{P})$ as $\mathcal{P}^{\text{best}}$
  Calculate $\mathcal{H}(\mathcal{V}',\mathcal{E}',\mathcal{G},\mathcal{P}^{\text{best}})$ and $\text{obj}(\mathcal{P}^{\text{best}})$
end

Algorithm 1: Harmony Search - Main

Data: $\mathcal{G}(\mathcal{V},\mathcal{E})$, HM, HMCR, PAR, BW
Result: Improvised solution $\mathcal{P}^{\text{new}}$
begin
  Initialize $\mathcal{P}^{\text{new}}$ as an empty solution
  while $j < \text{SL}$ **do**
    Generate random number $q \sim \mathcal{U}(0,1)$
    if $q < \text{HMCR}$ **then**
      Randomly pick $\mathcal{P}$ from HM
      Randomly pick $P_j = (x_j, y_j) \in \mathcal{P}$
      Generate random number $r \sim \mathcal{U}(0,1)$
      if $r < \text{PAR}$ **then**
        Relocate P_j at $P_j'(x_j + \Delta x, y_j + \Delta y)$
        where $\Delta x, \Delta y \sim \mathcal{U}(-\text{BW}, \text{BW})$
        Append P_j' to $\mathcal{P}^{\text{new}}$
      end
      else
        Append P_j to $\mathcal{P}^{\text{new}}$
      end
    end
    else
      Append random location P_k'' to $\mathcal{P}^{\text{new}}$
    end
  end
  Return $\mathcal{P}^{\text{new}}$
end

Algorithm 2: Harmony Search - Improvisation

C. Select Random Location for Helper Node Placement

The way that the random locations for helper nodes are generated during initialization and improvisation in the Harmony Search algorithm can significantly affect the performance and convergence rate. The performance can be improved by generating the random locations using information on the current jammer placement that uses the fewest jammers to achieve the specified level of network partition. Under our jammer model, network nodes located within the jamming radius of a jammer will not be able to communicate, so placing helper node in those regions have limited positive effect on improving network robustness. Thus, when generating a random location, $P_k''(x_{\text{new}}, y_{\text{new}})$, that is a potential helper node placement, the location is generated from a continuous random variable for which the probability density at (x, y) is

- 0 if the distance between (x, y) and any jammer in OJS is smaller than R_J, and
- inversely proportional to its distance to the nearest jammer in OJS.

An example probability density generated under these rules with network and OJS in Fig. 1(a) is shown in Fig. 2

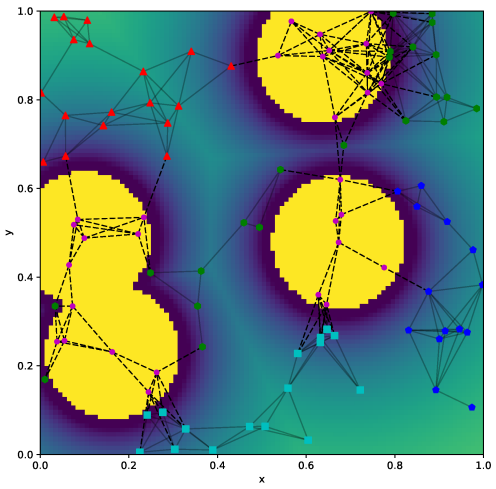


Fig. 2. Probability density for choosing helper node placement locations at the beginning of Harmony Search. Yellow (lighter) regions have zero probability density; purple (darker) region have highest probability density.

IV. Simulation Results

In this section, we evaluate the performance and running time of the proposed helper node placement algorithms via simulation of randomly generated networks. All simulations are programmed in Python with network analysis module NetworkX and integer linear programming solver provided in GLPK library (connected through Python's CVXOPT module). The Multilevel network partition solver is provided in Metis library (connected through Python's PyMetis module). The simulations were executed on computers with a 3.1GHz Intel Core i7 CPU with 8MB cache and 16GB RAM running Ubuntu 14.04 LTS. All figures show the performance averaged over 100 different network topologies.

The simulation results are all for graphs generated using the Random Geometric Graph (RGG):

$$\begin{aligned} &\mathcal{G}(\mathcal{V},\mathcal{E}) = \text{RGG}(n, R_c) \\ &\mathcal{V} = \{v_i(x_i, y_i) \mid x_i, y_i \sim U[0, \sqrt{\tfrac{n}{100}})\} \\ &\mathcal{E} = \{(u,v) \mid u,v \in \mathcal{V}, d(u,v) \le R_c\} \subset \mathcal{V} \times \mathcal{V}. \end{aligned} \tag{2}$$

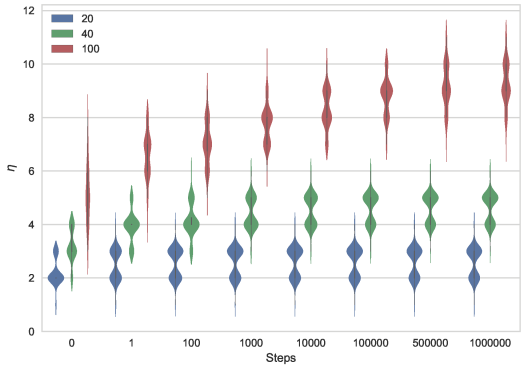


Fig. 3. Jammer placement objective function (number of jammers needed to meet the partitioning objective) for networks with different orders vs. iterations of the Harmony Search.

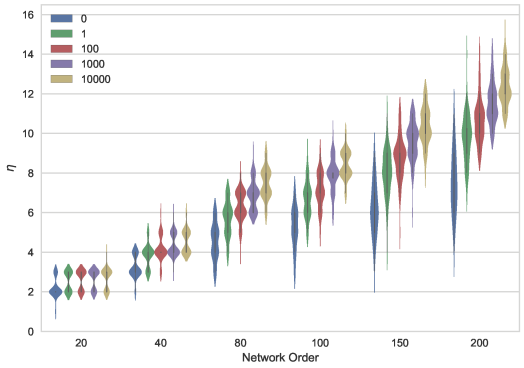


Fig. 4. Jammer placement objective function (η) at multiple iteration steps caused by placing helper nodes with the same communication radius as the original communication nodes.

Each simulation topology is generated by randomly placing nodes in the region $[0, \sqrt{\frac{n}{100}}) \times [0, \sqrt{\frac{n}{100}})$ and then creating edges for all pairs of vertices u and v if they are within the communication radius, $d(u, v) \leq R_c$. This graph models a scenario in which:

- Radios are homogeneous with: identical spatial density, equal transmit power, omnidirectional antennas, and equal noise figure.
- The channel obeys an exponential path-loss model.
- The jamming radius is equal to the communication radius ($R_J = R_c = 0.15$).

For all simulations, HMS=20, HMCR=0.8, PAR=0.8 and BW=0.05.

We first consider the tradeoff between performance and execution time in the harmony search (HS). We ran HS on networks with 20, 40, and 100 nodes with 100 different topologies for each order. The number of helper nodes is 5% of the network order for each topology. The communication radius of the helpers nodes is the same as the network nodes'. Fig. 3 is a violin plot[1] of the objective function (η, the number of jammers in the OJS) at various iterations of the HS algorithm. Here, iteration $i = 0$ corresponds to no helper nodes, and $i = 1$ means the HM has just been initialized (the placement is the $\mathcal{P}^{best}$ in 20 random solutions stored in HM).

Fig. 3 shows that for networks of 20 or 40 nodes, the objective function of the HS is saturated by 10000 iterations. For network with order 40, the average $\eta(\mathcal{H})$ increased by 41.25% (from 3.20 to 4.52) in the first 10000 iterations, and only achieved another 1.99% increment (from 4.52 to 4.61) at one million iterations. For networks with 100 nodes, the distribution continues to improve after 10000 iterations, but the majority of the gain has been achieved by 10000 iterations. Thus, in the following we consider HS with 100, 1000 and 10000 iterations. The average execution times increase linearly with the number of iterations, and so are not shown.

The performance and running times of HS on various network orders are shown in Figs. 4–6. We simulated the placement of helper nodes into networks with order 20, 40, 80, 100, 150 and 200. The number of helper nodes is equal to 5% of the network order, i.e., one helper node is used for networks with 20 nodes, five helper nodes are used for networks with 100 nodes, etc. To test the effect of placing helper nodes with different communication radii, we also have two scenarios: helper nodes with the same radius as the network nodes and helper nodes with twice the radius of the network nodes.

Fig. 4 is a violin plot that shows the density of the jammer placement objective function (η) as a function of the network order for multiple values of the number of iterations of the HS algorithm. For these results, the helper nodes' communication radius is equal to that of the network nodes. The results show that our proposed algorithm significantly increases $\eta(\mathcal{H})$, especially for the larger networks. For a 100-node network, with helper nodes the average $\eta(\mathcal{H})$ is increased by 55.35% (from 5.42 to 8.42) at 10000 iterations. Although the solution created during initialization increases η, the Harmony Search solver is able to create refined solutions by keeping the "good" parts and removing "bad" parts in the HM and can escape local optima by introducing controlled randomness.

When helper nodes with larger communication radii can be deployed, the robustness of networks increases even further, as shown by the results in Fig. 5. For a 100 node network with no helper nodes, $\eta = 5.42$; for helper nodes with the same communication radius ($1\times$) as the network nodes, $\eta = 8.42$, and for helper nodes with twice ($2\times$) that communication radius, $\eta = 9.22$. Similarly, for a 200-node network with no helpers $\eta = 7.38$, whereas with helper nodes with $1\times$ or $2\times$ communication radii, the average values of η increase to 12.36 and 13.83, respectively. We note that without helper nodes, even a 500-node network does not achieve $\eta > 12$.

In Fig. 6, the average time complexities of HS on placing helper nodes with communication radius identical to or twice ($2\times$) that of the network nodes are shown by blue and green

[1]A violin plot is similar to a box plot, except the boxes are replaced by kernel-smoothed density estimates, where the width indicates the estimated probability density.

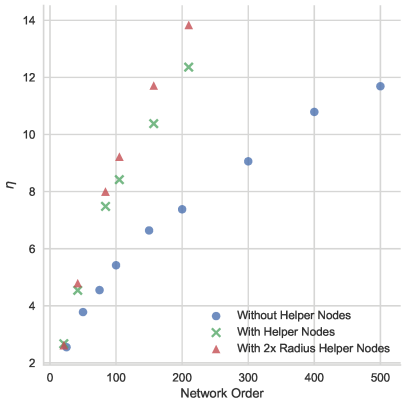


Fig. 5. Jammer placement objective function (η) for random geometric network topologies with order from 25 to 500 without helper nodes (blue dots) and with order from 20 to 200 with 5% helper nodes with either identical (green $\times$s) or 2$\times$ communication radius (red triangles).

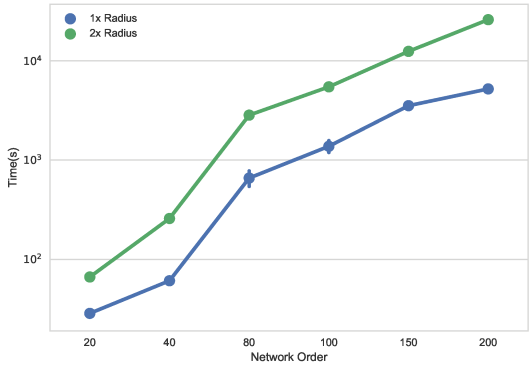


Fig. 6. Time complexity for optimizing helper node locations with 1000 iteration steps versus network order and helper node communication radius.

lines, respectively. For helper nodes with the same communication radius as the original nodes, Fig. 6 shows that the HS algorithm with 10000 iterations completes within 100 s (less than 2 min) for almost all network topologies of 40 nodes or fewer. For networks with order 100, HS algorithm with 10000 steps of iterations require about 1.4×10^3 s (approximately 23 minutes) on average and all simulations finished within 3.3×10^3 s (approximately 55 minutes). And for networks with order 200, the average time requirements for 10000 HS iterations are 5.2×10^3 s (approximately 1.5 hours). If an average run time of less than 10 minutes is required, HS with 1000 iterations can be used with average run time 500 s, but the average improvement in $\eta(\mathcal{H})$ is reduced to about 53%. Note that the additional time complexity for the helper nodes with larger communication radius is because the additional connections result in larger integer linear programs.

V. Conclusion

In this paper, we considered the problem of helper node placement to improve the robustness of wireless networks to jamming attacks aimed at partitioning the network. We propose techniques to optimize the placement of helper nodes that make the topology more robust to such jamming attacks. We use the meta-heuristic Harmony Search algorithm to search for good locations to place the helper nodes. Our simulation shows that by placing a number of helper nodes equal to 5% of the network order, our approach can significantly increase the network robustness, achieving an average robustness that would only be seen in much larger random geometric graphs.

References

[1] M. Pursley and W. Stark, "Performance of Reed-Solomon coded frequency-hop spread-spectrum communications in partial-band interference," *IEEE Trans. Commun.*, vol. 33, no. 8, pp. 767–774, Aug. 1985.

[2] M. B. Pursley and H. B. Russell, "Routing in frequency-hop packet radio networks with partial-band jamming," *IEEE Trans. Commun.*, vol. 41, pp. 1117–1124, July 1993.

[3] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in WSNs," *IEEE Commun. Surveys & Tutorials*, vol. 11, no. 4, pp. 42–56, 2009.

[4] A. Kott and T. Abdelzaher, "Resiliency and robustness of complex, multi-genre networks," *arXiv preprint arXiv:1601.06731*, 2016.

[5] A. D. Wood, J. A. Stankovic, and S. H. Son, "JAM: A jammed-area mapping service for sensor networks," in *IEEE Real-Time Systems Symp.* IEEE, 2003, pp. 286–297.

[6] W. Xu, T. Wood, W. Trappe, and Y. Zhang, "Channel surfing and spatial retreats: defenses against wireless denial of service," in *Proceedings of the 3rd ACM workshop on Wireless security*. ACM, 2004, pp. 80–89.

[7] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "JAID: An algorithm for data fusion and jamming avoidance on distributed sensor networks," *Pervasive and Mobile Computing*, vol. 5, no. 2, pp. 135–147, 2009.

[8] W. Abbas and M. Egerstedt, "Robust graph topologies for networked systems," in *3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 2012, pp. 85–90.

[9] A. Y. Yazicioglu, M. Egerstedt, and J. Shamma, "Formation of robust multi-agent networks through self-organizing random regular graphs," 2015.

[10] J. Feng, X. Li, E. L. Pasiliao, Jr., and J. M. Shea, "Jammer placement to partition wireless network," in *Proc. IEEE Global Commun. Conf. Workshop on Wireless Networking and Control for Unmanned Autonomous Vehicles*, Austin, TX, Dec. 2014, pp. 1487–1492.

[11] J. Feng, E. L. Pasiliao, Jr., W. E. Dixon, and J. M. Shea, "An optimal jamming strategy to partition a wireless network," in *Proc. IEEE Military Commun. Conf.*, Tampa, FL, Oct. 2015, pp. 978–984.

[12] J. Feng, W. E. Dixon, and J. M. Shea, "Fast algorithms for jammer placement to partition a wireless network," to appear in *Proc. 2017 IEEE Int. Conf. Commun.* [Online]. Available: http://wireless.ece.ufl.edu/jshea/pubs/jixin_icc17.pdf

[13] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis, "The complexity of multiway cuts," in *Proc. 24th Annual ACM Symp. Theory of Computing*. ACM, 1992, pp. 241–251.

[14] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theoretical Comp. Sci.*, vol. 1, no. 3, pp. 237–267, 1976.

[15] T. N. Bui and C. Jones, "Finding good approximate vertex and edge partitions is NP-hard," *Inform. Proc. Let.*, vol. 42, no. 3, pp. 153–159, 1992.

[16] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems (NIPS) 14*, vol. 2, 2002, pp. 849–856.

[17] M. X. Cheng, Y. Ling, and B. M. Sadler, "Wireless ad hoc networks connectivity assessment and relay node deployment," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 399–404.

[18] M. Penrose, *Random geometric graphs*. Oxford University Press Oxford, 2003, vol. 5.

[19] G. Karypis and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed computing*, vol. 48, no. 1, pp. 96–129, 1998.

[20] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-Inspired Harmony Search Algorithm*. Springer, 2009, pp. 1–14.