Time-Based Node Deployment Policies for Reliable Wireless Sensor Networks

Nicholas T. Boardman and Kelly M. Sullivan

***Abstract*—Wireless Sensor Networks (WSNs) are commonly used to monitor a remote environment over an extended period of time. One important design consideration is the WSNs reliability of area coverage, as sensors fail over time and functionality of the network degrades. When the WSN no longer sufficiently covers the region, maintenance actions may consider repairing failed nodes or deploying new sensors to reestablish network capability. Towards identifying an optimal maintenance policy, specifically the deployment of new sensors, we present an optimization model formulated using the network destruction spectrum (D-spectrum), that seeks to determine a time-based deployment policy balancing cost and reliability. While the benefits of using the D-spectrum in reliability are widely researched, the application of the D-spectrum to enable the modeling and solving of an optimization problem is new. With the complexity already present in estimating reliability, the significance of this optimization model is that it decouples the complexity of estimating the D-spectrum from the estimation of network reliability in the presence of a given deployment policy. This key feature allows us to quickly evaluate a wide range of time-based deployment policies. Additionally, we present an efficient destruction algorithm that performs a vital subroutine in estimating the D-spectrum, allowing for a larger of number of replications to be performed in the Monte Carlo simulation thereby reducing the variance of the resulting reliability estimate. Finally, the optimization model is illustrated through a numerical example.**

***Index Terms*—Network Maintenance Optimization, Destruction Spectrum, Network Reliability, Wireless Sensor Networks.**

NOTATION

$\mathcal{N}_1$	The set of sink and sensor nodes.
$\mathcal{N}_2$	The set of target nodes.
n_1	The number of sensor nodes.
n_2	The number of target nodes.
$\mathcal{N}$	The collection of all nodes.
d_1	The communication radius of a sensor.
d_2	The monitoring radius of a sensor.
$\mathcal{E}$	The undirected edge set created by connecting all nodes in $\mathcal{N}_1$ that can communicate with each other.
$\mathcal{A}_1$	$\{\bigcup_{\{i,j\}\in\mathcal{E}}\{(i,j),(j,i)\}\}$, The expanded directed communication edge set between sensors.
$\mathcal{A}_2$	The set of directed edges from a sensor $i \in \mathcal{N}_1 \backslash \{0\}$ to target $j \in \mathcal{N}_2$.
$\mathcal{A}$	$\{\mathcal{A}_1 \cup \mathcal{A}_2\}$, the collection of all directed arcs.
$\mathcal{G}' = (\mathcal{N}_1, \mathcal{E})$	The undirected network consisting only of sensor nodes.
$\mathcal{G} = (\mathcal{N}, \mathcal{A})$	The directed network consisting of all nodes.
$\mathcal{R}$	The region of interest.
$\mathcal{H}_i$	The indexed set of all directed paths from the sink node to node i.
h_i^j	The set of nodes on the j^{th} path from sink node to node i.
$F(t)$	The c.d.f. of sensor lifetime.
π	A random permutation of sensor failures.
q_i	The index of sensor i in order of sensor failures π.
$C[\mathcal{G}(t)]$	The coverage of the network at time $t \geq 0$.
α	The desired coverage level, $0 < \alpha \leq 1$.
η_i	The critical loss time: time at which node $i \in \mathcal{N}$ is no longer connected to sink node.
$\tilde{\eta}_{(i)}$	The sorted critical target loss times, $\tilde{\eta}_{(1)} \leq \tilde{\eta}_{(2)} \leq \cdots \leq \tilde{\eta}_{(n_2)}$.
$s_{\alpha,i}^{n_1}$	The probability that the i^{th} sensor failure results in coverage falling below α.
$r(t;\alpha,n_1)$	The reliability at time t.
δ	The time between deployment actions.
c_F	The fixed cost of deploying sensors.
c_V	The variable cost (per sensor) of deploying sensors.
$G(t;\delta)$	The c.d.f. of the stable residual life distribution for a sensor under deployment interval δ.
$r^{\infty}(t;\alpha,n_1,\delta)$	The reliability at time t under time-based deployment interval δ.
$B(x;n,p)$	The binomial c.d.f., $B(x;n,p) = \sum_{i=0}^{x} \binom{n}{i} p^i (1-p)^{n-i}$.

I. INTRODUCTION & LITERATURE REVIEW

WIRELESS sensor networks (WSNs) consist of a set of sensors, distributed over a region of interest, that monitor and report desired conditions within the region. The number of sensors in these networks can vary greatly depending on the coverage required, the detection and communication capabilities of sensors, as well as the initial effort to design and allocate sensors across the network. The network application can further influence network size ranging from areas such as fire and flood detection [1], military operations with battlefield

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-1751191.

Nicholas T. Boardman and Kelly M. Sullivan are with the Department of Industrial Engineering, University of Arkansas, Fayetteville, AR, 72701, USA; email: nboardma@uark.edu, ksulliv@uark.edu

tracking and surveillance, or environmental control in buildings [2]. Another attractive feature of WSNs is that they can be designed and constructed for a specific application (such as those previously mentioned), but also offer the flexibility to be quickly deployed as required. Whether a network is specially designed or randomly deployed can be impacted by the application or operational setting. For example, in areas with harsh environmental conditions or rough terrain, sensors can be air dropped over a desired location to achieve coverage in a given area [3]. A consequence of this approach is that sensors are randomly deployed throughout the region, but the lack of control over specifically locating sensors can be offset by deploying a larger number of sensors. The low-cost characteristic of sensors is an additional component that contributes towards the random deployment of a larger number of sensors as a feasible strategy [4].

Once established it is important that the network operate for a sufficient period of time, particularly when sensors are deployed in remote areas and difficult to access for repair. The performance of a WSN is primarily impacted by the number of operating sensors and the ability of these sensors to communicate with each other [5]. These are both characteristics of the network that decline over time as sensors start to fail. The lifetime of an individual sensor is bounded by a battery or power supply, and once diminished the sensor no longer operates [6]. Sensors additionally have components required for monitoring, processing, and routing information through the network. Software errors in any of these functions, potentially in the form of failing to properly send/receive information from nearby sensors, result in a drop in network capability and may propagate failures through the network [7]. Hardware failures also arise with physical damage and can possibly cause components to break, particularly when operating in a harsh environment where sensors are exposed to weather or other external factors [2].

The failure process has led to research on methods to extend sensor lifetimes, commonly through topology control algorithms. By modifying the communication range the power consumed by a sensor can be managed while still ensuring a message can be routed through the network [8]. Energy consumption can further be controlled by specifying which paths are used to route data, as well as aggregating data to avoid sending duplicate messages [9]. Similarly, the network topology can be dynamically controlled through periodically turning sensors on and off. Such a sleep/wake schedule results in redundant nodes conserving energy until required to help prolong network lifetime [6], [10]. These algorithms commonly aim to maximize the time until the first sensor fails, but sensor networks often have redundancy built in and can tolerate some sensor failures without losing capability [11]. Another limitation is that sensor lifetime is treated to be bounded by a battery supply that is consumed at some known rate and once depleted the sensor fails. Addressing random sensor failures that can arise (e.g., environmental interference, physical damage [4]) adds a layer of difficulty to estimating network lifetime, particularly when we are interested in the status of the network beyond the first sensor failure.

Another approach to extend network lifetime is through the use of movement based connectivity methods. With sensor nodes randomly deployed throughout the region it may be desirable to relocate sensors immediately after deployment in an effort to improve the overall network coverage and connectivity [12], [13]. We may also be interested in re-positioning sensor nodes in response to failures that occur [14], [15], [16]. One of the advantages of a movement based approach is that network topology can be dynamically controlled to prolong network lifetime. However the cost of mobile sensors is typically significantly larger than static sensors [4], [15] which introduces questions about their suitability for a large scale WSN of interest.

The attractiveness of topology control algorithms and movement based connectivity methods is their ability to extend the lifetime of a given network. The long-term operation of a WSN must also consider deploying new sensors in the network, particularly in the presence of an increasing number of sensor failures. In [17], different node replacement policies are examined to maintain a coverage requirement to maximize lifetime where a decision to replace a failed sensor or not is made immediately after observing a failure, however only a small number of sensors can be replaced. A similar problem focuses on deploying new sensors to restore some level of connectivity and/or coverage, with the additional challenge of deploying the fewest number of new sensors [18], [19], [20]. Problems related to optimal node placement commonly fall in the NP-Hard class of problems [21], which motivates the search for approximation algorithms. One of the primary limitations of current models is that they are framed in the context of single stage. That is, the deployment of new sensors is concerned with immediately preserving network functionality, but does not consider the future failure probability of sensors. As a result it is reasonable to question the reliability of the network after sensors have been deployed. To the best of our knowledge, attempts at creating a durable network with the deployment of additional sensors focus on providing a level of redundancy or k-connectivity [20], [22]. This is certainly a desirable characteristic for the network, but a node redeployment strategy should also be influenced by the residual life distribution of sensors and how frequently such a policy needs to be implemented.

Existing research has focused primarily on extending network lifetime (common in topology control and movement based methods) or maintaining a coverage/connectivity requirement (common in node redeployment methods), but there appears to be a lack of emphasis on analyzing a maintenance policy with respect to network reliability. The focus of this work is directly concentrated on evaluating and comparing the performance of time-based maintenance policies, specifically the deployment of new sensors in the network, with respect to both cost and estimated reliability. One of the difficulties of this task is that network reliability problems commonly fall in the #P-Complete class of problems [23] and can be difficult to solve exactly, particularly for larger size networks or when reliability estimation is performed as a subroutine in another algorithm. As a result network reliability problems are routinely solved by approximate solution methods.

One theme that arises with respect to approximate methods

is to bound network reliability. Compared to an exact method that explores every possible network state, carefully selecting a subset of states to evaluate can lead to more efficient algorithms that provide upper and/or lower bounds on network reliability [24]. Depending on the manner in which bounds are constructed, these algorithms still require a large amount of effort particularly for larger sized networks [25].

Closely related, and often utilized within bounding techniques, is to use a Monte Carlo method to estimate network lifetime. A naive/crude Monte Carlo approach is to randomly generate a failure time for each sensor according to its life distribution, order the sensor failures, and then examine the network state after each successive sensor failure to determine the instant of network failure. Repeating this process allows for an estimation of overall network reliability upon completion. One main drawback of this approach is the unbounded growth of the relative error for highly reliable and highly unreliable networks [26]. This issue has been addressed through the use of improved Monte Carlo methods [27] and variance reduction techniques [25], [28].

A crude Monte Carlo method can also be improved leveraging the destruction spectrum (D-spectrum) of the network, also referred to as the network signature, where the resulting reliability estimation has bounded relative error [29]. Under the assumption of independent and identically distributed sensor lifetimes, the destruction spectrum also yields an efficient representation of the network's reliability but depends only on the system structure [30]. While both the D-spectrum and crude Monte Carlo approaches require solving an embedded destruction problem in order to determine the time at which all sensors are disconnected from one or more sink nodes, we show the destruction problem for the network signature can be solved more efficiently than the one for crude Monte Carlo.

With an understanding of network reliability we can now begin to explore the impact from the deployment of new sensors in the network. The objective of deploying new sensors is directed at restoring network function (i.e. as a corrective maintenance action) or improving network capability (i.e., as a preventive maintenance action) [31]. Depending on the application of the WSN, a temporary failure of the network may lead to serious consequences making corrective deployment policies unattractive. For this reason we focus on preventive deployment policies which could be based on the number of functioning sensors, the size of the region the network covers, or the time since the last deployment action. Preventive policies have also been explored in related network maintenance models, such as power distribution networks studied in [32], [33]. Also discussed is the added difficulty in that we must now estimate the cost of such an action as well to compare different policies, in addition to estimating network reliability in the presence of a deployment policy.

Time-based (or periodic) deployment is one version of a preventive deployment policy in which new sensors are deployed at fixed time intervals. We examine a time-based deployment policy in a network consisting of n_1 sensors, where every δ time units new sensors are deployed in the network to increase the number of functioning sensors back up to n_1. An alternative action is to repair a failed sensor node, but given the low-cost characteristic of sensors combined with the potential difficulty in accessing a specific sensor for repair, the deployment of new sensor nodes is an attractive policy. Additionally, WSNs lack the requirement for a physical connection between sensors (e.g., wire, cable, etc.) which further avoids the need to repair failed sensors.

Towards identifying an optimal time-based deployment policy, the main contribution of this work is an optimization model, formulated using the D-spectrum, that seeks to determine a time-based deployment policy balancing cost and reliability. While the benefits of using the D-spectrum in reliability are widely researched, the application of the D-spectrum to enable the modeling and solving of an optimization problem is new. The inclusion of the D-spectrum in determining an optimal time-based deployment policy is of interest as the D-spectrum is a property of the network structure impacted by n_1, which can be viewed as a network design variable, but is independent of the deployment interval δ. As a result the Monte Carlo simulation for estimating the signature, which may be a large source of computational effort, is not impacted by the time-based deployment policy. This modeling approach thus decouples the complexity of estimating the network signature for evaluating reliability, and estimating reliability in the presence of a given time-based deployment policy. In a related effort, we demonstrate how to exploit random geometric graph structure commonly used to model WSNs to efficiently update the destruction spectrum estimate for networks of varying size, yielding further computational advantages in the optimization model.

Additionally, we provide an efficient destruction algorithm that performs a vital subroutine in estimating the D-spectrum and in turn reliability for a WSN. This algorithm is based on recognizing that a single iteration of a Monte Carlo algorithm (for estimating the D-spectrum) yields a maximum capacity path problem. The D-spectrum's unique characteristics enable Dial's implementation of Dijkstra's algorithm to be utilized when solving for the maximum capacity path. This improvement in the algorithm allows for a larger number of replications, thereby reducing the variance of our reliability estimate beyond the traditional approach of estimating the D-spectrum. We also discuss a simple extension that allows network reliability to be calculated for different coverage requirements without increasing the complexity of the algorithm.

The remainder of this work is organized as follows. Section II summarizes the modeling of the WSN and the methodology to estimate reliability in the network both with and without the deployment of new sensors. Section III presents a destruction algorithm necessary for estimating the network signature, which is then used to estimate reliability and evaluate various time-based deployment policies. Section IV conveys the procedure of optimizing time-based deployment policies of a WSN, which is then demonstrated in Section V. Section VI summarizes conclusions and directions for further work.

II. Modeling Fundamentals & Network Reliability

We model a WSN as a network whose node set consists of a sink node 0, sensor nodes $\{1, \ldots, n_1\}$, and target nodes $\{n_1 +$

$1, \ldots, n_1 + n_2\}$. We define $\mathcal{N}_1 = \{0, 1, \ldots, n_1\}$ as the set of sink and sensor nodes and $\mathcal{N}_2 = \{n_1+1, \ldots, n_1+n_2\}$ as the set of target nodes. The two main functions of a sensor node are to communicate with other sensor nodes and to monitor targets. Sensor nodes $i \in \mathcal{N}_1$ and $j \in \mathcal{N}_1$ are capable of communicating with one another provided they are within a given range $d_1 > 0$ of each other. Let $\mathcal{E} \subseteq \binom{\mathcal{N}_1}{2}$ denote the set of undirected edges $\{i, j\}$ created due to each pair of sensor nodes $i \in \mathcal{N}_1$ and $j \in \mathcal{N}_1$ that can communicate, and define $\mathcal{A}_1 = \bigcup_{\{i,j\}\in\mathcal{E}}\{(i, j), (j, i)\}$ as the expanded, directed edge set associated with $\mathcal{E}$. A sensor node is capable of monitoring any target within a range $d_2 > 0$. Let $\mathcal{A}_2 \subseteq \mathcal{N}_1 \times \mathcal{N}_2$ denote the set of directed edges that defines which targets are covered by which sensors. Thus, an arc $(i, j) \in \mathcal{A}_2$ exists if sensor $i \in \mathcal{N}_1\backslash\{0\}$ monitors target $j \in \mathcal{N}_2$. Without loss of generality, going forward we assume a single sink node is located in the network. We can always transform the network to one that contains a single sink by adding a new artificial sink node, and adding an arc from this new node to every sensor connected to one of the original sink nodes. The original set of sink nodes and their adjacent arcs are then removed.

In what follows, it will be useful to consider both the directed network $\mathcal{G} = (\mathcal{N}_1 \cup \mathcal{N}_2, \mathcal{A}_1 \cup \mathcal{A}_2)$ and the undirected network $\mathcal{G}' = (\mathcal{N}_1, \mathcal{E})$ as representations of the WSN. For brevity, we define $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$ and $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$. An example of both networks for $n_1 = 100$ sensor nodes randomly distributed over a $[0, 1] \times [0, 1]$ region is illustrated in Fig. 1.

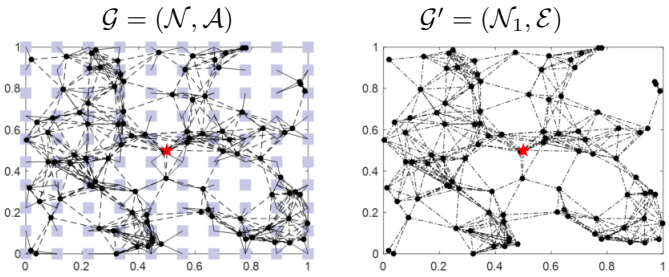

Fig. 1: Example realization of the network $\mathcal{G}$ and $\mathcal{G}'$ over the $[0, 1] \times [0, 1]$ region, $n = 100$, $d_1 = 0.2$, $d_2 = 0.1$, and a single sink node (marked by "⋆") located at $(0.5, 0.5)$. The set of 10×10 target nodes marked by "■" is defined as $\mathcal{N}_2 = \{0.00, 0.11, 0.22, \ldots, 1.00\} \times \{0.00, 0.11, 0.22, \ldots, 1.00\}$. Note that for ease of illustration in the network $\mathcal{G}$, the edge pair (i, j) and (j, i) is represented by a single dashed arc, while a solid arc represents a sensor to target arc.

Due to the failure of sensor nodes the WSN evolves over time. At any time $t \geq 0$ the network $\mathcal{G}$, so that $\mathcal{G}'(t)$ is well defined, is represented by $\mathcal{G}(t)$, and consists of only sensors that are still functioning, indicated by the set $\mathcal{N}_1(t)$, and the resulting communication edges (i.e., excludes failed nodes from $\mathcal{N}_1$ and adjacent edges). Let $T \geq 0$ represent the lifetime of a sensor, $F(t)$ denote the cumulative distribution function of T, and $\overline{F}(t) = 1 - F(t)$ the survival function of T. Upon generating a deterministic failure time for each sensor from the distribution of T, let π represent the order of sensor failures where $\pi(k) = i$ if node $i \in \mathcal{N}_1\backslash\{0\}$ is the k^{th} sensor to fail, and let $q_i \in \{1, \ldots, n_1\}, i \in \mathcal{N}_1\backslash\{0\}$, be the index such that $\pi(q_i) = i$. For the network constructed in this manner, the following assumptions are also imposed.

Assumption 1: Sensor lifetimes are independent and identically distributed (i.i.d.).

Assumption 2: Sensor capabilities are identical.

Under Assumption 1, each sensor has the same life distribution T and sensors fail independently of one another. This yields favorable theoretical properties with respect to the D-spectrum we can leverage to develop more efficient algorithms. We later discuss the impacts of relaxing the assumption that lifetimes are identically distributed, while maintaining independent failures, in the context of deploying new sensors in the network. Assumption 2 implies that all sensors also have the same communication radius d_1 and common sensing radius d_2. With identical sensors, this alleviates concerns of sensor compatibility and integrating multiple sensor types to function together.

Assumption 3: The sink node is perfectly reliable.

For a target to be covered it must be both within the coverage radius of a functioning sensor, and there must exist a communication path from this monitoring sensor back to the sink. Given this requirement, it is clear that the sink node is one of the most important nodes in the network. We can guarantee that if the sink node fails we no longer cover any of the targets, and the network fails as well. For this reason, with Assumption 3 we assume that the sink node does not fail.

The network condition is classified into one of two states, either operating or failed, and is determined by the proportion of targets that are covered denoted by $C(\mathcal{G})$. For a given α-coverage requirement, $0 < \alpha \leq 1$, the *α-failure time* is the time at which $C(\mathcal{G})$ drops below α and the network transitions to a failed state. Depending on the size of the region covered and application of the WSN, we may not require 100% coverage of targets to construct a picture of the overall status. Environmental applications such as office building climate control may allow for a smaller α-coverage requirement, while applications in target tracking or surveillance may require a larger coverage requirement [2]. It may also be that 100% coverage is too costly or impractical to maintain over the life of the network. With an α-coverage requirement specified, the network's reliability $r(t; \alpha, n_1) = \Pr[C\big(\mathcal{G}(t)\big) \geq \alpha]$ is defined to be the probability that the network's coverage is at least α at time t. In the following sections we are interested in comparing the reliability of networks of varying size, resulting in n_1 appearing in the expression $r(t; \alpha, n_1)$ to denote its dependence on the number of sensors.

Assumption 4: The initial WSN $\mathcal{G}'(0)$ is a random geometric graph (RGG) with uniform density over a bounded region $\mathcal{R} \in \mathbb{R}^2$.

The arrangement of sensors in a WSN is typically classified as either deterministic or random [3]. With the complexity already present in (i) estimating network reliability and (ii) evaluating time-based deployment policies, Assumption 4 models the the initial network as a random geometric graph with senor nodes randomly distributed over a bounded region $\mathcal{R} \in \mathbb{R}^2$. Modeling the sensors as uniformly distributed imposes no

additional loss of generality, as the results that follow hold for any density function. This removes the difficulty of designing a WSN in addition to considering these aspects, and also reflect a scenario in which a network has to be rapidly deployed in a remote area.

A. Homogeneous Network Reliability

In the absence of additional sensors being deployed in the network, the collection of sensors is homogeneous in the sense that all surviving sensors were installed at the same time and therefore have i.i.d. residual life distributions. Given the difficulty already present in estimating network reliability in this case, we first turn attention to the homogeneous network reliability using a Monte Carlo approach. Monte Carlo methods for WSN reliability evaluation give rise to a network destruction problem, an optimization problem that determines the instant of network failure given fixed sensor failure times [29]. For a network with node failures, the *destruction spectrum* is a probability distribution on the number of failed nodes required to cause network failure. The D-spectrum for a network can be estimated with steps similar to the crude Monte Carlo where the number of failed nodes corresponding to network failure is recorded instead of the time at which this occurs.

Let $s_{\alpha,i}^{n_1}$ denote the probability that in the network $\mathcal{G}$ with n_1 sensors, the i^{th} sensor failure results in $C(\mathcal{G})$ falling below α. Network reliability is then given by

$$r(t;\alpha,n_1)=\sum_{i=1}^{n_1} s_{\alpha,i}^{n_1} B(i-1;n_1,F(t)), \tag{1}$$

where $B(i-1;n_1,F(t))$ is the cumulative binomial probability of no more than $i-1$ successes in n_1 trials with probability of success $F(t)$ [34]. Although algorithms exist for computing $s_{\alpha,i}^{n_1}$ exactly, we use a Monte Carlo approach to estimate the D-spectrum which is common especially for large, complex networks [29], [35]. Therefore we use the notation $\hat{s}_{\alpha,i}^{n_1}$ to refer to the estimate of $s_{\alpha,i}^{n_1}$, leading to the reliability estimate $\hat{r}(t;\alpha,n_1)$ of $r(t;\alpha,n_1)$.

An estimate on the variance of network reliability may also be of interest, particularly if we wish to compare destruction algorithms, and is given by

$$\begin{aligned} Var(r(t;\alpha,n_1)) =& \frac{1}{M}\Big[\sum_{i=1}^{n_1} s_{\alpha,i}^{n_1}(1-s_{\alpha,i}^{n_1})B(i-1;n_1,F(t))^2 \\ &- 2\sum_{j=1}^{n_1-1}\sum_{k=j+1}^{n_1} s_{\alpha,j}^{n_1}s_{\alpha,k}^{n_1}B(j-1;n_1,F(t)) \\ &B(k-1;n_1,F(t))\Big], \end{aligned} \tag{2}$$

where M is the total number of replications [29]. Since we again use the D-spectrum estimate $\hat{s}_{\alpha,i}^{n_1}$, the estimate of the variance is denoted $\widehat{Var}(r(t;\alpha,n_1))$.

As mentioned previously an appealing aspect of the D-spectrum is that it is a property of the network structure and failure definition of the network, and does not depend on the lifetimes of the individual components [30]. With Assumption 1, the implication is that each of the $n_1!$ permutations of sensor failures are equally likely. Therefore instead of generating a random failure time from the distribution of T for each sensor, we can proceed to generate a random order of sensor failures. This is a key result that we revisit later towards identifying an efficient destruction algorithm on estimating the signature of the network.

B. A Generic Algorithm for Estimating the WSN's Destruction Spectrum

The first step towards estimating network reliability is now calculating the D-spectrum. Estimating the D-spectrum is outlined in Algorithm 1, based on the work in [29]. The driving component of Algorithm 1 is Step 5 of determining which sensor failure results in network failure.

Algorithm 1 Monte Carlo algorithm for estimating destruction spectrum

1: **function** SIGNATUREMC
2: **Set** $m_i \leftarrow 0,\ \forall\ i \in \mathcal{N}_1$.
▷ m_i = # network failures caused by ith sensor failure
3: **Generate** $\mathcal{G}$ by simulating $(x_i, y_i) \in \mathcal{R},\ \forall\ i \in \mathcal{N}_1$.
▷ (x_i, y_i) = coordinates for sensor i
4: **Simulate** random permutation π of the sensors $\{1, 2, \ldots, n_1\}$; $\pi = (i_1, i_2, \ldots, i_{n_1})$.
5: **Find** the smallest value $i^\star \in \{1, \ldots, n_1\}$ such that $C(\mathcal{G} \setminus \{\pi(1), \ldots, \pi(i^\star)\}) < \alpha$.
▷ The $i^\star$-th sensor failure causes network failure
6: **Set** $m_{i^\star} \leftarrow m_{i^\star} + 1$.
7: **Repeat** Steps 3–6 M times.
8: **Set** $\hat{s}_{\alpha,i}^{n_1} \leftarrow m_i/M,\ \forall\ i \in \mathcal{N}_1$.
9: **end function**

In Step 5, the network is subject to a destruction process where sensors are iteratively removed from the functioning set of nodes based on the order π of sensor failures. After the failure of each sensor, the network coverage $C(\mathcal{G}(t))$ is computed and compared to the α-coverage requirement. Consider a straightforward destruction algorithm for this step. For each of the networks $\mathcal{G}\backslash\{\pi(1),\ldots,\pi(i)\}, i = 0, 1, \ldots, n_1$, implement a breadth-first search algorithm in order to identify how many of the target nodes $\mathcal{N}_2$ are reachable from the sink. Dividing this count by $|\mathcal{N}_2|$ we can determine $C(\mathcal{G})\backslash\{\pi(1),\ldots,\pi(i)\}$ based on the number of targets reached. Overall this requires $O(|\mathcal{A}|n_1)$ effort per iteration of Step 5. The destruction algorithm is implemented in each of the M replications of the Monte Carlo, which motivates the search for an efficient algorithm of finding this sensor failure of interest. A binary search method can improve the complexity of this step to $O(|\mathcal{A}|\ \log(n_1))$. In Section III we present a destruction algorithm that improves on this complexity by exploiting aspects of the D-spectrum and the network construction.

Each iteration of Step 5 returns an observation on the number of failed nodes resulting in network failure. By recording this value for every iteration we are able to obtain our estimate $\hat{s}_{\alpha,i}^{n_1}$ of the D-spectrum upon completion, and finally estimate network reliability by substituting into (1).

C. Time-Based Deployment Policies

To prolong the functioning status of the WSN we are interested in periodically deploying new sensors in the region in an effort to increase the number of functioning senors, thereby increasing network coverage. We focus on a time-based deployment policy in which n_1 sensors are initially deployed over the region $\mathcal{R}$. Every δ time units thereafter, all failed sensors in the network are replaced by deploying new sensors over $\mathcal{R}$ such that the total number of functioning sensors in the network is increased back to n_1. Such a policy is identified as an (n_1, δ) policy. By replenishing the number of functioning sensors to a constant value, the present (n_1, δ) policy assumes that we have knowledge about the number of failed sensors in the network prior to any deployment action. Similar to the initial layout of sensors, by assuming that new sensors are always deployed uniformly and independently over $\mathcal{R}$ and that each sensor's location is independent of its time to failure we arrive at the following properties.

Property 1: For all $t \geq 0$, the WSN $\mathcal{G}'(t)$ is a RGG with uniform density over $\mathcal{R}$.

Property 2: For all $t \in \{k\delta : k \in \mathbb{Z}_{\geq 0}\}, |\mathcal{N}_1(t)| = n_1$.

D. Heterogeneous Network Reliability

Towards identifying an optimal time-based deployment policy, we now analyze network reliability in the presence of a given (n_1, δ) policy. Compared to Section II-A, the collection of sensors is now heterogeneous in the sense that the surviving population of sensors have different ages, and thus different residual life distributions. In [30] it was shown that the D-spectrum representation of a network remains valid in stochastic mixtures of components, provided that the components are exchangeable. With this in hand, the D-spectrum approach in the previous section can be extended to the present (n_1, δ) policy under consideration to estimate network reliability.

We refer to the time interval $[(k-1)\delta, k\delta]$ as the kth epoch, $k \in \mathbb{Z}_{\geq 0}$. Immediately after new sensors are deployed, the age X of each sensor is a random variable in the range $\{k\delta : k \in \mathbb{Z}_{\geq 0}\}$. In the presence of the (n_1, δ) policy, we are interested in the network's reliability for the infinite-horizon setting, where at the beginning of an epoch the probability distribution on the age X of a sensor does not change from one epoch to the next (i.e., there is a stable mix of sensors).

In the infinite-horizon setting, each sensor's age at the beginning of epochs $k' \in \mathbb{Z}_{>0}$ can be viewed independently as an irreducible Markov chain on the countably infinite state space $k \in \mathbb{Z}_{\geq 0}$, where state k corresponds to the sensor having age $k\delta$. In this Markov chain, each state k transitions into state $k+1$ with probability $\overline{F}((k+1)\delta)/\overline{F}(k\delta)$ and back to state 0 otherwise. This Markov chain has the unique stationary distribution

$$\rho_k = \frac{\overline{F}(k\delta)}{\sum_{j=0}^{\infty} \overline{F}(j\delta)}, \ k \in \mathbb{Z}_{\geq 0}, \tag{3}$$

provided that the denominator converges, in which case the Markov chain is ergodic.

Now, let $T_x \geq 0$ denote the residual life of a sensor at age $x > 0$, and denote its c.d.f. by $F_x(t) = [F(x+t) - F(x)]/\overline{F}(x)$. Ergodicity of the Markov chain described above also implies exchangeability of the sensors at stationarity: That is, immediately after the deployment of new sensors, a subset of sensors selected at random have i.i.d. age described by the probability distribution $\Pr\{X = k\delta\} = \rho_k$, $k \in \mathbb{Z}_{\geq 0}$. The residual lifetime of such a sensor (considering the randomness in its age), is then described (see, e.g., [36]) by the c.d.f.

$$G(t;\delta) = \mathbb{E}[F_X(t)], \tag{4a}$$

$$= \sum_{k=0}^{\infty} \frac{F(k\delta + t) - F(k\delta)}{\overline{F}(k\delta)} \rho_k, \tag{4b}$$

$$= \frac{\sum_{k=0}^{\infty} [F(k\delta + t) - F(k\delta)]}{\sum_{j=0}^{\infty} \overline{F}(j\delta)}. \tag{4c}$$

We will refer to T_X as the *stable residual life distribution* of a sensor under the (n_1, δ) policy.

Considering the above, at stationarity, the remaining life of sensors selected at random are independent and identical random variables with c.d.f. given by (4c). Further, by Property 2, at the beginning of every epoch $k \in \mathbb{Z}_{\geq 0}$ the network contains n_1 functioning sensors and the D-spectrum of the network remains applicable. Therefore, applying (1) to the i.i.d. residual life distribution, the stable network's reliability (i.e., the probability that its coverage remains at least α after $t \geq 0$ additional time units) is given by

$$r^{\infty}(t;\alpha,n_1,\delta) = \sum_{i=1}^{n_1} s_{\alpha,i}^{n_1} B(i-1; n_1, G(t;\delta)), \tag{5}$$

where the ∞-superscript has been appended to r to denote that it applies to the infinite-horizon setting. The heterogeneous network reliability estimate is represented by $\hat{r}^{\infty}(t;\alpha,n_1,\delta)$, as it again depends on the D-spectrum estimate $\hat{s}_{\alpha,i}^{n_1}$. The variance of the (n_1, δ) policy can be estimated applying (2), again substituting the residual life c.d.f $G(t;\delta)$ for $F(t)$. While we are primarily interested in the stable network reliability immediately prior to the deployment of additional sensors (i.e., at time $t = \delta$), (5) can be applied to evaluate the stable network reliability for any time $t \leq \delta$. This property maintains the ability to evaluate reliability over time, enabling system performance availability measures to be estimated as well.

Notice also that the D-spectrum is independent of the deployment interval δ. Thus, in the process of exploring the space of (n_1, δ) policies, we need only to estimate the signature once for any value of n_1 considered.

III. Destruction Algorithms

With a basis to estimate both the homogeneous and heterogeneous network reliability relying on the D-spectrum, we now revisit Step 5 of Algorithm 1 and search for an efficient destruction algorithm. Over the course of exploring (n_1, δ) policies it may be necessary to estimate the D-spectrum for assorted values of n_1. While there are efficiencies that can be gained as a result of Assumption 4 that are discussed later in Section III-A, an improved destruction algorithm allows for a larger number of Monte Carlo replications thereby reducing the variance of our resulting reliability estimate. Towards this effort, we present Algorithm 2 that seeks to identify which

sensor failure, in a predefined sequence that specifies the time of all sensor failures, causes network coverage to drop below the α-coverage requirement.

Recall that for a target to be covered it must satisfy two criteria. First the target must be within the coverage radius of a functioning sensor, and second there must be a communication path from this sensor back to the sink node. From the construction of the network $\mathcal{G}$, this equates to a directed path from the sink node to the target node using only functioning sensor nodes as internal nodes. Such a path fails as soon as one of its internal sensor node fails, and the target becomes disconnected as soon as all such paths fail. Among all directed paths from the sink to the target, define a *critical path* as one in which the time until the first failure of an internal node in the path is maximized. Thus the failure time of a critical path to a target node equals the time at which the target is no longer covered. A critical path can be similarly defined for sensor nodes, and equates to the earliest time at which a sensor node is either failed or no longer connected to the sink. With this characterization, a critical path is defined for every node $i \in \mathcal{N}$, and the failure time of this critical path is referred to as the *critical loss time*, η_i. When necessary, the critical loss time can be further distinguished as a critical sensor loss time for a node $i \in \mathcal{N}_1$, or a critical target loss time for a node $i \in \mathcal{N}_2$.

Finding a critical path to every node $i \in \mathcal{N}$ is equivalent to the maximum capacity path problem discussed by [37]. In a network with weights defined on every node, a maximum capacity path between two nodes is a path such that the weight of the smallest node on the path is maximized. Let h_i^j denote the j^{th} directed path from the sink node to node i, and $\mathcal{H}_i = \{1, 2, \ldots, H_i\}$ index the set of all directed paths from the sink node to node $i \in \mathcal{N}$. The value of a maximum capacity path to node $i \in \mathcal{N}$ is then $\max_{j\in\mathcal{H}_i}\{\min\{q_k : k \in h_i^j\}\}$. In a directed network, such as the network $\mathcal{G}$ under consideration, a maximum capacity path from a source node to every other node can be found using a slight modification to Dijkstra's algorithm while updating node labels [37]. When solving for the maximum capacity path, the label of a node is initialized as $\eta_i = 0$ for all $i \in \mathcal{N}\backslash\{0\}$, and $\eta_0 = \infty$. Nodes then have their label updated according to

$$\eta_j = \max\{\eta_j, \min\{\eta_i, q_j\}\}. \tag{6}$$

Under Assumption 3 the sink node does not fail, which is equivalent to representing the sink node as the last node to fail by $q_0 = n_1$, while target nodes are regarded in a similar fashion with $q_i = n_1$ for all $i \in \mathcal{N}_2$.

Originally introduced as a variation of the shortest path problem, the maximum capacity path is commonly defined for weights associated with every edge [37]. The network $\mathcal{G}$ can be transformed to adopt this convention by defining edge weights according to the minimum of the two adjacent nodes, with, $w_{ij} = \min\{q_i, q_j\}$ for all $(i, j) \in \mathcal{A}$. The updating of node labels in (6) can also be updated accordingly to compare the weight of the edge by $\eta_j = \max\{\eta_j, \min\{\eta_i, w_{ij}\}\}$.

A naive implementation of Dijkstra's can be accomplished in $O(|\mathcal{N}|^2)$ time, and improved to $O\big(|\mathcal{A}| + |\mathcal{N}| \log(|\mathcal{N}|)\big)$ with a heap data structure [38]. Further, the critical target loss times will be marked permanent in a non-increasing manner within Dijkstra's algorithm which means they can be sorted over the course of the algorithm, simplifying the search for the α-failure time upon completion. While possible to sort the critical loss time for all nodes, the order of critical target loss times are of particular interest as these values correspond to a change in network coverage. Therefore, let $\tilde{\eta}_{(i)}$ represent the i^{th} smallest critical target loss time for a node $i \in \mathcal{N}_2$, resulting in $\tilde{\eta}_{(1)} \leq \tilde{\eta}_{(2)} \leq \cdots \leq \tilde{\eta}_{(n_2)}$.

Algorithm 2 Coverage Destruction

1: **function** SIGNATURESUBROUTINE
2: **Initialize** $\eta_0 = \infty,\ \eta_i = 0\ \forall\ i \in \mathcal{N}\backslash\{0\}$.
3: **Initialize** $S = \emptyset, \bar{S} = \mathcal{N}$.
4: **While** $|S| < |\mathcal{N}|$.
5: **Select** node $i \in \bar{S}$ such that $\eta_i = \max\{\eta_j : j \in \bar{S}\}$.
6: **Update** $S = S \cup \{i\}$, $\bar{S} = \bar{S}\backslash\{i\}$.
7: **For** each $j : (i, j) \in \mathcal{A}$.
8: **Update** $\eta_j = \max\{\eta_j, \min\{\eta_i, q_j\}\}$.
9: **End For**.
10: **End While**.
11: **Let** $\tilde{\eta}_{(1)} \leq \tilde{\eta}_{(2)} \leq \cdots \leq \tilde{\eta}_{(n_2)}$ denote the sorted η_i-values for $i \in \mathcal{N}_2$.
12: **Find** smallest integer κ such that $\frac{n_2-\kappa}{n_2} < \alpha$.
13: **Set** $i^* = \tilde{\eta}_{(\kappa)}$.
14: **end function**

Using the D-spectrum to estimate network reliability we can improve the complexity further. Because $q_i \in \{1, 2, \ldots, n_1\}$ for all $i \in \mathcal{N}_1$, the labels $\eta_i, i \in \mathcal{N}$, are always in the range $\{0, 1, \ldots, n_1\}$. This feature motivates the use of Dial's implementation of Dijkstra's algorithm. In Dial's implementation the node to mark permanent in Step 5 of Algorithm 2 during an iteration can be found more efficiently by storing the temporary label of nodes in a sorted bucket structure. Initially, buckets $\{0, 1, \ldots, n_1\}$ are created with all nodes in bucket zero, except for the sink node which is placed in bucket n_1. Starting with bucket n_1, select the sink node to mark permanent, and update the label of adjacent nodes (i.e., by moving to the bucket numbered with the new label value) according to (6). Continuing in this manner, the node to mark permanent at each iteration can be found efficiently as the node in the largest valued non-empty bucket. Using Dial's algorithm with ordered sensor failures, the step of finding the critical path from the sink node to every other node in the network can now be accomplished in $O(|\mathcal{N}_1| + |\mathcal{A}|)$ time [38].

The overall complexity of Algorithm 2 is also $O\big(|\mathcal{N}_1| + |\mathcal{A}|\big)$, driven by Dial's implementation of Dijkstra's in Steps 4-10. Step 2 and 3 each require $O(|\mathcal{N}|) = O(|\mathcal{N}_1| + n_2)$ time, and as a result of using modified Dijkstra's algorithm nodes are marked permanent in a non-increasing manner. Therefore the sort in Step 11 can be accomplished by simply recording the order in which target nodes are marked permanent in Step 6, and the sorting of critical target loss times does not add to the complexity. Step 12 requires $O(n_2)$ time, but can be improved to $O(1)$ time. With α known, network failure occurs when $\kappa^* = \lceil n_2(1-\alpha)\rceil$ targets are no longer covered and we can simply return $\tilde{\eta}_{(\kappa^*)}$. In any case, under the assumption that

each target is initially within range of at least one functioning sensor, then $|\mathcal{A}_2| \geq n_2$, and since $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ Step 12 (and Step 2-3) does not increase the complexity.

A. Extensions of Destruction Algorithms

In the exploration of various (n_1, δ) policies, further efficiency can be gained as a result of Assumption 4. With sensors independently and randomly located in the network we have the signature relation $s^{n_1}_{\alpha,i} = s^{n_1-1}_{\alpha,i-1}$ for $i = 1, 2, \ldots, n_1$. That is, failure of one node in a RGG with n_1 nodes yields a RGG with $n_1 - 1$ nodes. This result is also previously stated as Property 1. Therefore it is not necessary to recompute the network signature for every value of n_1 desired. Utilizing this feature allows the space of (n_1, δ) policies to be explored in a more efficient manner.

We may also be interested in the impact that α has on network lifetime as this is ultimately the criteria used to classify the network as operational. It seems reasonable to expect that for a smaller α network lifetime would be longer, and at any time t the network reliability would be higher. We can make a slight modification to Algorithm 2 to explore how large of an impact this will have. Note that the critical target loss times are independent of α. If we are interested in network reliability for various coverage requirements, one option is to first specify these various levels upfront. Then in Step 12 and 13 instead of returning a single value $\tilde{\eta}_{(\kappa^*)}$, we can easily find the α-failure time for each of these levels and update the D-spectrum estimate $\hat{s}^{n_1}_{\alpha,i}$ for each different requirement. Alternatively, we can store the entire sequence of critical target loss times, specify α upon completion and then search for the α-failure times as required. This second approach may be of more interest, particularly with respect to the D-spectrum where we are more concerned with sensor failures that result in a change in coverage. By storing the entire sequence of critical target loss times we can easily determine how sensitive the network is to the next sensor failure. In either case, the complexity of Algorithm 2 does not increase as a result of estimating network reliability for multiple coverage requirements.

B. Spanning Tree Destruction Algorithm

Thus far we have modeled the WSN as a directed network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ that includes every node (sensor and target) in the network. Since the network $\mathcal{G}'$ is smaller than $\mathcal{G}$, we might find it appealing to first work with the smaller network before expanding to the larger directed network as required. Whether we work with the network $\mathcal{G}$ or $\mathcal{G}'$, the critical path for a sensor will not change. In undirected networks such as $\mathcal{G}'$, it is known that a maximum weight spanning tree contains a maximum capacity path between all pairs of nodes in the network [39]. This property gives rise to an efficient approach for solving the destruction problem in the case of K-terminal connectivity [26], [27], [35]. Such an approach can also be adapted to our problem. If we work with the smaller undirected network $\mathcal{G}'$ and proceed with the spanning tree approach, edge weights must be defined according to $w_e = \min\{q_i, q_j\}$ for every edge $e = \{i, j\} \in \mathcal{E}$. Using Prim's Algorithm, a maximum weight spanning tree over $\mathcal{G}'$ can now be found requiring $O\big(|\mathcal{E}| + |\mathcal{N}_1| \log(|\mathcal{N}_1|)\big)$ effort [38]. Once the spanning tree is constructed we are able to find the critical loss time η_i for each sensor $i \in \mathcal{N}_1$. With the critical sensor loss times, we can consider the sensor-to-target arcs $\mathcal{A}_2$ to determine the critical loss times for all targets $i \in \mathcal{N}_2$. For each target computing η_j requires $O(|\{i \in \mathcal{N}_1 : (i, j) \in \mathcal{A}_2\}|)$; therefore the total effort required for this step is $O(\sum_{j \in \mathcal{N}_2} |\{i \in \mathcal{N}_1 : (i, j) \in \mathcal{A}_2\}|) = O(|\mathcal{A}_2|)$. From $\eta_j \in \{0, \ldots, n_1\}$ for all $j \in \mathcal{N}_2$, a bucket sort algorithm can be used to sort the critical target loss times resulting in $O(|\mathcal{N}_1| + n_2)$ effort [40].

The spanning tree approach to the D-spectrum thus requires $O\big(|\mathcal{E}| + |\mathcal{N}_1| \log(|\mathcal{N}_1|) + |\mathcal{A}_2|\big)$ effort. From this we can see that it is actually advantageous to work on the entire directed network $\mathcal{G}$, as it allows us to implement a more efficient algorithm to determine the critical target loss time, and in turn the D-spectrum of the network.

IV. Optimal (n_1, δ) Policies

Section II-D provides a methodology for estimating network reliability for a given (n_1, δ) policy. We now focus on identifying values of n_1 and δ that will effectively balance cost and reliability. We assume, in the vein of *economic dependence* models in the multicomponent maintenance literature [41], that a fixed cost of $c_F > 0$ is incurred for each time at which one or more new sensors are added to the network and a variable cost of $c_V > 0$ is incurred for each new sensor added. (The fixed cost would likely be large relative to the variable cost, for instance, in a WSN that monitors a harsh/remote environment such as a glacier or another planet's atmosphere.) In the infinite-horizon setting the average cost per unit time, or long-run average cost rate, associated with an (n_1, δ) policy is given by

$$v(n_1, \delta) = \frac{c_F\{1 - [\bar{G}(\delta; \delta)]^{n_1}\} + n_1 c_V G(\delta; \delta)}{\delta}, \qquad (7)$$

where the first term in the numerator is the expected fixed cost incurred (based on the probability that at least one sensor fails), the second term is the expected variable cost incurred (based on the expected number of sensor failures), and the denominator is the time between deployment actions.

We incorporate reliability into the optimization via maximizing with respect to $\omega(n_1, \delta) = r^{\infty}(\delta; \alpha, n_1, \delta)$. The resulting bi-objective optimization model is

$$\max_{n_1, \delta} \{-v(n_1, \delta), \omega(n_1, \delta)\}. \qquad (8)$$

The two-variable model (8) can be approximately solved by enumerating combinations of n_1 and δ, allowing for an efficient frontier to be generated over the range of policies evaluated.

V. Numerical Results

We now illustrate the methodology described in Sections II–IV in an example scenario, where the lifetime T of each sensor is distributed according to a Weibull distribution with a shape parameter $\beta = 1.5$ and scale parameter $\lambda = 10$. Sensor capabilities are defined according to a communication

radius of $d_1 = 0.075$ and a sensing radius of $d_2 = 0.075$. The coverage area consists of $|\mathcal{N}_2| = 441$ targets uniformly spaced as a 21×21 grid in the region $\mathcal{R} = [0,1] \times [0,1]$.

Before proceeding to the reliability results there are two components of Algorithm 1 that are also worth discussing, those being the simulation of a RGG in Step 3 and the random permutation of failures in Step 4. Implementing these steps in a naive manner can result in significant effort. In a straightforward approach, a RGG of n_1 nodes can be generated in $O(n_1^2)$ time by randomly placing each node in $\mathcal{R}$, and then comparing the distance between all $\binom{n_1}{2}$ pairs of nodes to determine if an arc between the nodes is present. Given the complexity of Algorithm 2 is $O(|\mathcal{N}_1| + |\mathcal{A}|)$, the potential $O(n_1^2)$ cost is significant as we can now expect to spend more time generating a RGG than implementing a destruction algorithm. Efficient methods to generate a random graph have thus attracted a large amount of attention. In an attempt to reduce this source of complexity, Step 3 is implemented based on the technique described in [42] which generates a RGG by assigning each node in $\mathbb{R}^2$ to a bin, and then comparing nodes i and j from the appropriate bins to determine if the arc (i,j) is present. The complexity now depends on the number of bins as well, but if done appropriately the expected complexity is $O(|\mathcal{N}_1| + |\mathcal{E}|)$ [42].

The next step of simulating a random permutation of sensor failures is also worth examining further. A naive approach is to generate a failure time for each sensor from the distribution of T, then sort these values to determine the failure order. A number of available algorithms (e.g., [40]) can accomplish this in $O(n_1 \log(n_1))$ time. Instead, we use a modern version of the Fisher-Yates shuffle algorithm that generates a random permutation directly in $O(n_1)$ time [43].

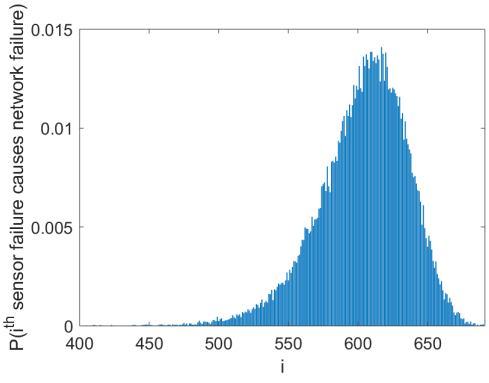

Fig. 2: Plot of $\hat{s}^{900}_{0.8,i}$

The D-spectrum was estimated using Algorithm 1 with $M = 50,000$ replications for a coverage requirement of $\alpha = 0.8$ in a network consisting of $n_1 = 900$ sensors. A plot of the resulting D-spectrum estimate is illustrated in Fig. 2. By the discussion at the beginning of Section III-A, we can also obtain an estimate of the signature for any network containing $n_1 < 900$ sensors with no additional replications. This becomes particularly useful when evaluating time-based deployment policies, as we can now examine any (n_1, δ) policy (such that $n_1 < 900$) without re-implementing Algorithm 1. As an example, Fig. 3(a) depicts the estimated D-spectrum for a network with $n_1 = 500$ sensor nodes, based on the D-spectrum estimate from the 900 node estimate from Fig. 2. To illustrate the accuracy of the signature relation, we have also utilized Algorithm 1 to estimate the signature on a network with 500 sensor nodes, which is plotted in Fig. 3(b).

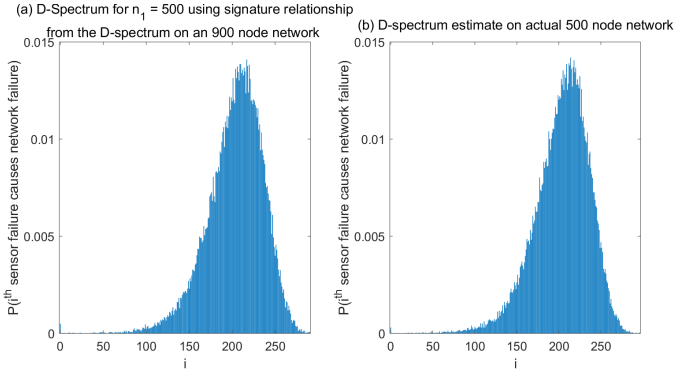

Fig. 3: Comparison of D-spectrum estimates

We can now turn to reliability, and apply (1) to estimate the homogeneous network reliability if desired. Since this information is not specifically of interest in the context of comparing (n_1, δ) policies, the plot of $\hat{r}(t; \alpha, n_1)$ has been omitted. Instead, we proceed to estimate the stable network's reliability under the presence of various (n_1, δ) policies as given by (5), and the cost of the policy as given by (7). In doing so we assume a fixed cost of $c_F = 100$ and a variable cost of $c_V = 1$. This information is plotted in Fig. 4 for networks of four different selected sizes ($n_1 \in \{450, 550, 650, 750\}$), and δ evaluated over the range $(1, 10)$ at 0.1 unit intervals. Additionally, δ at specific intervals has been identified on the plot. If there are factors that impose limitations on the network size (e.g., n_1 must be a multiple of 10) or the deployment interval, then Fig. 4 can be particularly valuable in comparing the performance of various policies. For example, both the $(550, 4.5)$ policy and the $(650, 5.9)$ policy yield a stable network reliability of 0.85. To meet a stable network reliability requirement above 0.85 the deployment interval for each policy must decrease, at which point the $(650, \delta < 5.9)$ policy dominates any $(550, \delta < 4.5)$ policy.

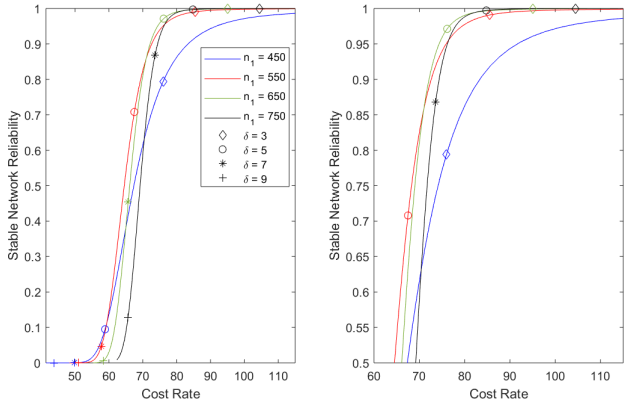

Fig. 4: Plot of stable network reliability

In the current context of a RGG and sensors randomly deployed, we can assume to freely select both n_1 and δ over a continuous range. Therefore, Fig. 5 is of more significance as it illustrates an efficient set of policies over a continuous range of (n_1, δ) of interest. The fairly intuitive result behind Fig. 5 is that to satisfy a larger requirement on the stable network reliability, in general the network should contain a larger number of sensors and the deployment interval δ should be smaller. We can also observe that near 100% reliability can be achieved with $n_1 \approx 675$ with a deployment interval of $\delta \approx 4.5$. As we deviate from this policy by either adding more sensors or decreasing the deployment interval, the cost rate increases significantly.

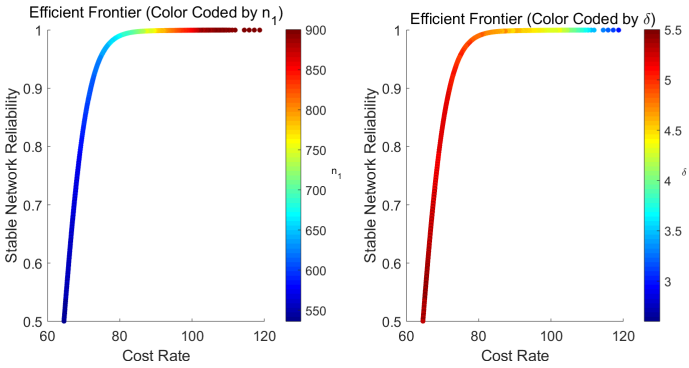

Fig. 5: Efficient Frontier for $\alpha = 0.8$

Using the methods previously described to generate a RGG and new failure order each iteration, along with the destruction algorithm from Section III, estimating the D-spectrum for a 900 node network required approximately 374 seconds (accomplished with c++ on an Intel(R) Core i7-6600U CPU with a 2.60 GHz processor and 16 GB of RAM). This D-spectrum estimate is then used to evaluate the set of (n_1, δ) policies for $n_1 \in \{500, 501, \ldots, 900\}$ and $\delta \in \{1.0, 1.1, \ldots, 10.0\}$ and plot the efficient frontier in Fig. 5. This process is far more computationally expensive, requiring approximately $5,061$ seconds (~ 84 minutes).

Delineating the time between these two step allows us to clearly see the benefit of modeling assumption 4, and the advantage of using the D-spectrum relation $s^{n_1}_{\alpha,i} = s^{n_1-1}_{\alpha,i-1}$ to estimate the signature of smaller networks. While we could use Algorithm 1 to estimate the D-spectrum for each network size under consideration, doing so would add up to $374 \times (900 - 500) = 149,600$ seconds (~ 41.5 hours) to the overall computation time. Even though we expect the time required to estimate the D-spectrum for smaller networks to decrease (take Fig. 3(b) for example, which only required 163 seconds to estimate), the additional computation time by repetitively estimating the D-spectrum remains significant (using this estimate the additional time is approximately 18 hours).

Finally, we may be interested in how sensitive the efficient frontier is to various parameters (e.g., c_F, c_V, β, λ). Fig. 6 plots the efficient frontier for two different α-coverage requirements: the original efficient frontier for $\alpha = 0.8$, along with the new efficient frontier for $\alpha = 0.9$. This plot can help illustrate the robustness of various policies and the improvement in network performance for a minor cost increase. Consider the $(600, 5)$ policy, which incurs a cost rate of 71.8. For a coverage requirement of $\alpha = 0.8$ the corresponding stable network reliability is 0.897, while for a coverage requirement of $\alpha = 0.9$ the stable network reliability drops significantly to 0.678. Clearly, the same (n_1, δ) policy will have a smaller stable network reliability for a larger coverage requirement. But now consider this relationship with respect to the $(675, 5)$ policy, which incurs a cost rate of 78.3. For a coverage requirement of $\alpha = 0.8$ the corresponding stable network reliability is now 0.984, and for a coverage requirement of $\alpha = 0.9$ the stable network reliability is 0.922. Thus by increasing the number of nodes in the network (at a minor increase to the policy cost) we can implement a policy that not only meets an α-coverage requirement of 0.8 with high probability, but also achieve a coverage requirement of 0.9 with high probability.

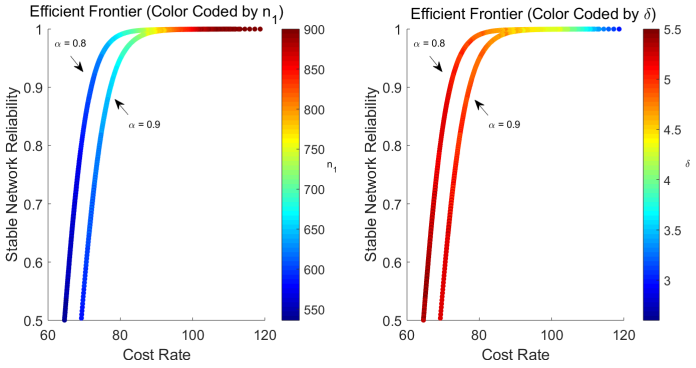

Fig. 6: Efficient Frontier for different α-coverage requirements

A similar process can be used to explore the impact of changing the associated costs of deployment actions or the sensor failure distribution parameters. The change to the efficient frontier in each scenario is similar to that illustrated in Fig. 6, the general shape of the curve remains the same but is shifted based on the direction of the parameter that is altered.

A. Confidence Interval on Stable Network Reliability

From the discussion in Section II-A we can also obtain an estimate on the network reliability variance, which in turn can be used to construct a confidence interval. Computing the confidence interval halfwidth will also help compare the performance of different destruction algorithms, illustrating the improvement that Algorithm 2 (using Dial's implementation) offers. With the variance in (2) a function of the number of replications M, the more replications we dedicate towards estimating the D-spectrum we can expect a tighter confidence interval.

To show the significance of this improvement, we consider a $n_1 = 650$ node network as a test instance. The D-spectrum is estimated using both Dial's implementation of Algorithm 2 as previously described, and also by using a naive $O(|\mathcal{N}|^2)$ implementation of Dijkstra's algorithm. For Dial's implementation we use $M = 50,000$ replications, while for Dijkstra's Algorithm we set $M = 20,000$. These values were selected so

that the total time dedicated towards estimating the D-spectrum from the two methods was approximately equal.

For each D-spectrum estimate we can again compute the stable network reliability for various (n_1, δ) policies, while in addition estimating the corresponding halfwidth. The 95% confidence interval halfwidth on the stable network reliability for the $(650, \delta)$ policy is plotted in Fig. 7. The improvement in the confidence interval halfwidth is most notable for $\delta \in (5, 8)$. If we revisit Fig. 4, this range is also where a change in δ results in a significant change to the stable network reliability. Thus, by using Dial's implementation in Algorithm 2 we can perform over twice as many replications in the same amount of time compared to the traditional Dijkstra's algorithm, which in turn results in a confidence interval halfwidth that is twice as small compared to the original Dijkstra's estimate.

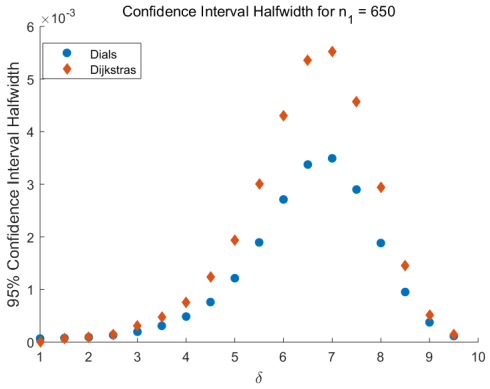

Fig. 7: Confidence Interval Halfwidth Comparison

It is also interesting to note that while the total time estimating the D-spectrum between the two methods is approximately the same, there are different steps of Algorithm 1 that dominate the complexity. For this purpose we focus primarily on Step 3 and Step 5. As previously presented, generating a RGG of n_1 nodes using [42] results in an expected complexity of $O(|\mathcal{N}_1| + |\mathcal{E}|)$, while the complexity of Algorithm 2, which accomplishes Step 5, is $O(|\mathcal{N}_1| + |\mathcal{A}|)$. With $|\mathcal{A}| = 2|\mathcal{E}| + |\mathcal{A}_2|$, the complexity of these two steps is relatively balanced. However when Dijkstra's algorithm is used as the destruction algorithm, the $O(|\mathcal{N}|^2)$ now becomes a large source of complexity and significantly more time is dedicated to Step 5. Thus, by using Algorithm 2 we are performing a larger number of replications while actually spending less time in this step of the destruction spectrum algorithm.

B. Verification of Stable Network Reliability

The stable network reliability is derived from the stable residual life distribution of a sensor, as given in (4c). This is the long run residual life distribution which is based on new sensors being deployed every δ time units. Since this applies in the infinite-horizon setting, a compelling question that arises is how long it takes to reach this steady state behavior.

To investigate this we can utilize a crude Monte Carlo simulation that implements the given (n_1, δ) policy, and check the network status at various times over the length of the simulation. This was accomplished for the $(650, 5.6)$ policy, with the resulting estimated transient network reliability illustrated in Fig. 8. The stable network reliability estimated using the methodology in Section II-D is also plotted in Fig. 8.

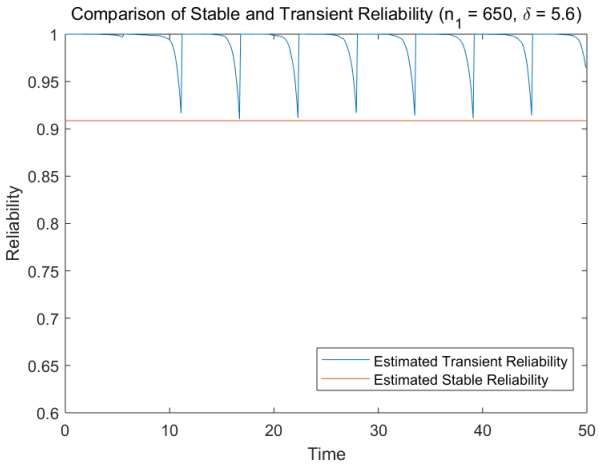

Fig. 8: Verification of Stable Network Reliability

From Fig. 8 we can observe that the stable network reliability is reached early on in the simulation, after the second or third deployment action. This helps demonstrate that while the stable network reliability is built on a long run horizon, it is reached fairly early on the process of the (n_1, δ) policy. While the transient Monte Carlo simulation is informative, particularly in plotting the change in reliability over time, it is far more computationally expensive and not conducive to optimizing time-based deployment policies. The transient reliability data in Fig. 8 is based on 10,000 replications, and required 1.7 hours. Additionally, this simulation evaluates a single value of n_1 and δ. The transient Monte Carlo simulation therefore quickly becomes intractable, particularity if we desire to evaluate the range of polices necessary to generate an efficient frontier similar to Fig. 5.

C. Verification of Long Run Cost Rate

Similarly, the Monte Carlo simulation can also help verify the long run cost rate estimated by (7). Based on the simulation of the $(650, 5.6)$ policy in Fig. 8, the estimated long run cost rate is 70.3. This is smaller than the estimate using the stable life distribution in (7), which results in an estimated cost rate of 72.5. The difference in these estimates is attributed to the very first deployment action in the Monte Carlo simulation. Referring back to Fig. 8, this is the "cheapest" deployment event in the sense that the fewest number of sensors have failed at the time of the first deployment event compared to those that occur later, which lowers the average cost rate slightly. If we omit the cost of the first deployment action each replication (i.e., only calculate the cost once the steady state behavior has been reached), we observe an average cost rate of 72.25 from the Monte Carlo simulation.

D. Multi-State Network

In Section II the network was characterized into either an operating for failed state. There are numerous applications

in which we may be interested in defining one or more intermediate states to reflect a partial degradation in network performance. An extension of the destruction spectrum to multi-state networks is discussed in [44], which can also be addressed with the current modeling framework. Since the state of the network is dependent upon a coverage requirement, multiple network states can be defined by multiple coverage levels where the state of the network is now based on network coverage falling within a given range. For example a three state network can be defined in which State 1 corresponds to $C(\mathcal{G}) \geq \alpha$, State 2 (intermediate state) in which $\alpha' \leq C(\mathcal{G}) < \alpha$, and State 3 in which $C(\mathcal{G}) < \alpha'$. For a given (n_1, δ) policy we are now interested in the probability that the network is in each of the given states. The probability the network is in State 1 can be estimated simply by $\text{Pr(State 1)} = r^{\infty}(\delta; \alpha, n_1, \delta)$. Similarly, the probability the network is in State 3 can be estimated by $\text{Pr(State 3)} = 1 - r^{\infty}(\delta; \alpha', n_1, \delta)$. The probability the network is in the intermediate State 2 can now be estimated by $\text{Pr(State 2)} = 1 - \text{Pr(State 1)} - \text{Pr(State 3)} = r^{\infty}(\delta; \alpha', n_1, \delta) - r^{\infty}(\delta; \alpha, n_1, \delta)$. Therefore, the additional work required in a multi-state model corresponds to estimating network reliability for different coverage requirements.

An example for a three state network is illustrated in Table I, where State 1 is defined by $C(\mathcal{G}) \geq 0.9$, State 2 by $0.8 \leq C(\mathcal{G}) < 0.9$, and State 3 by $C(\mathcal{G}) < 0.8$. While it is straightforward to calculate each state probability for the entire range of (n_1, δ) policies explored, the results for a smaller subset of policies are provided. Comparing the multi-state performance is of particular interest when a decision must be made to select a specific policy to implement. For example if we are comparing the $(676, 4.8)$ policy with the $(651, 5.1)$ policy, the first policy achieves a coverage of 0.9 with higher probability but is also more costly. We may also consider the second policy since it is less costly but still achieves a coverage of 0.9 with fairly high probability, and in the event coverage drops below the first coverage level is likely to be in the intermediate state. Notice that the data required to construct Table I is also previously illustrated in Fig. 6, and the table presents similar information in a slightly different manner. Modeling a larger number of network states is also possible through the introduction of new coverage levels, at the expense of an additional set of reliability calculations for each new state.

TABLE I: Multiple Network States

(n_1, δ) Policy	Cost Rate	Pr(State 1)	Pr(State 2)	Pr(State 3)
$(753, 4.5)$	88.78	0.995	0.004	0.001
$(676, 4.8)$	79.78	0.946	0.044	0.010
$(651, 5.1)$	75.59	0.853	0.113	0.034
$(615, 5.0)$	73.13	0.752	0.180	0.069
$(569, 5.1)$	68.54	0.452	0.316	0.231
$(555, 5.3)$	66.17	0.282	0.339	0.379

It is also of interest to investigate the impact a multi-state network has on optimizing a policy with respect to multiple coverage levels. With this motivation, we compare the similarity of the actual set of policies on each of the efficient frontiers in Fig. 6. Based on the results in Fig. 6, from the entire set of policies that are on the efficient frontier for at least one of the coverage requirements, 85% of these policies are on both efficient frontiers. This implies with high probability that if we select an efficient (n_1, δ) policy to achieve a coverage requirement of 0.8, this is also an efficient policy to achieve a coverage requirement of 0.9. Additionally, if we select some (n_1, δ) policy that is on the efficient frontier for a coverage requirement of 0.8 but not 0.9, the improvement (with respect to reliability) from deviating from this policy to an efficient policy for the 0.9 coverage requirement is negligible.

VI. Conclusions

As technology advances and becomes more affordable, WSNs are able to be integrated into an increasing number of applications. While the deployment of a WSN is the initial concern, the long run operating cost is an important factor to consider. This is true in terms of designing a network to meet requirements in addition to ensuring any maintenance policy preserves network functionality without an excessive cost. Existing research has emphasized methods to extend network lifetime, but does not focus on analyzing a maintenance policy with respect to network reliability.

Towards this goal, we have contributed an optimization model that determines optimal time-based deployment policies balancing cost and reliability. Of interest from this model is the inclusion of the destruction spectrum to evaluate policies, as the destruction spectrum is independent of the deployment interval parameter δ. This aspect helps decouple the complexity of estimating the destruction spectrum necessary to evaluate reliability, and evaluating the network's reliability in the presence of a given time-based deployment policy. Finally, we have presented a destruction algorithm to efficiently estimate the destruction spectrum, and illustrated the performance of the optimization model through a computational example.

In the network model we have focused on sensor failures that are identically distributed. One possible direction for extending this work involves the modeling of multiple sensor failure distributions. The incorporation of multiple failure distributions can be attractive, for example, when modeling the energy hole phenomena in which sensor nodes located closer to the sink node are relied on more often to route information through the network. As a result these nodes consume energy at a faster rate which leads to a shorter expected lifetime [45]. The destruction spectrum approach from Section II can be adapted to address this scenario (see [46]), with the drawback that the dimension of the signature increases.

Along the same direction, future work might consider the impact of load dependent failures. As sensor nodes fail in the network, messages must be routed along different paths to reach the sink. This inevitably results in various sensor nodes being relied upon in a larger capacity to route information, which can cause an increase in energy use and faster failure rate.

Section V-D discussed the impact of modeling multiple network states. A related version of this extension is to consider multi-state sensors. This problem variant introduces several

additional sources of complexity as we are now concerned not only with the initial capability of a sensor and the total number of functioning sensors in the network, but the number of sensors in each state and the capability of a sensor in a given state. A model must also be incorporated to reflect the transition of a sensor among the various states. Investigating how these components can be incorporated into a reliability estimate, and the impact they have on the current optimization model is a compelling problem to explore.

Another direction is to consider a deterministic network topology. The assumption that sensor nodes are always randomly deployed allowed us to leverage the D-spectrum relationship between networks of different sizes, saving a large amount of computation time. However in many applications we can control the network topology more precisely by locating sensors at specific points. While many of the results in Sections II–IV remain applicable in this case, it is not clear what/if there is a relationship between the D-spectrum for different networks, or if the D-spectrum can be found in a more efficient manner due to a deterministic topology.

We have also primarily considered a time-based deployment policy, in which new sensors are deployed every δ time units. Instead of scheduling a deployment based on time intervals, an improved policy could consider the condition of the network as well, such as the current number of failed sensors or the number of targets covered. Given the stochastic nature of the network evolution and the potentially enormous state and maintenance decision space, this modeling approach may be more amenable to approximate dynamic programming.

DISCLAIMER

The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

REFERENCES

[1] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 3. IEEE, 2001, pp. 1380–1387.

[2] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[3] S. Debnath, A. K. Singh, and A. Hossain, "A comprehensive survey of coverage problem and efficient sensor deployment strategies in wireless sensor networks," *Indian Journal of Science and Technology*, vol. 9, no. 45, 2016.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002.

[5] D. Deif and Y. Gadallah, "A comprehensive wireless sensor network reliability metric for critical internet of things applications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2017, no. 1, p. 145, 2017.

[6] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03. New York, NY: ACM, 2003, pp. 28–39.

[7] L. M. S. De Souza, H. Vogt, and M. Beigl, "A survey on fault tolerance in wireless sensor networks," *Interner Bericht. Fakultät für Informatik, Universität Karlsruhe*, 2007.

[8] N. Li and J. C. Hou, "Flss: a fault-tolerant topology control algorithm for wireless networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*. ACM, 2004, pp. 275–286.

[9] L. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*. IEEE, 2002, pp. 575–578.

[10] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 1, pp. 89–124, 2005.

[11] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 1, pp. 5:1–5:39, Feb. 2009.

[12] G. Wang, G. Cao, and T. F. La Porta, "Movement-assisted sensor deployment," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 640–652, 2006.

[13] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, vol. 2. IEEE, 2003, pp. 1293–1303.

[14] A. A. Abbasi, M. F. Younis, and U. A. Baroudi, "Recovering from a node failure in wireless sensor-actor networks with minimal topology changes," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 256–271, 2012.

[15] P. K. Sahoo and J.-P. Sheu, "Limited mobility coverage and connectivity maintenance protocols for wireless sensor networks," *Computer Networks*, vol. 55, no. 13, pp. 2856–2872, 2011.

[16] G. Wang, G. Cao, T. La Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 4. IEEE, 2005, pp. 2302–2312.

[17] S. Parikh, V. M. Vokkarane, L. Xing, and D. Kasilingam, "Node-replacement policies to maintain threshold-coverage in wireless sensor networks," in *2007 16th International Conference on Computer Communications and Networks*. IEEE, 2007, pp. 760–765.

[18] H. M. Almasaeid and A. E. Kamal, "On the minimum k-connectivity repair in wireless sensor networks," in

2009 IEEE International Conference on Communications. IEEE, 2009, pp. 1–5.

[19] X. Cheng, D.-Z. Du, L. Wang, and B. Xu, "Relay sensor placement in wireless sensor networks," *Wireless Networks*, vol. 14, no. 3, pp. 347–355, 2008.

[20] X. Han, X. Cao, E. L. Lloyd, and C.-C. Shen, "Fault-tolerant relay node placement in heterogeneous wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 5, pp. 643–656, 2009.

[21] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621–655, 2008.

[22] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus, "Deploying sensor networks with guaranteed capacity and fault tolerance," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2005, pp. 309–319.

[23] J. S. Provan and M. O. Ball, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 777–12, 11 1983.

[24] M. O. Ball, C. J. Colbourn, and J. S. Provan, "Network reliability," *Handbooks in Operations Research and Management Science*, vol. 7, pp. 673–762, 1995.

[25] Q. Yang and Y. Chen, "Monte carlo methods for reliability evaluation of linear sensor systems," *IEEE Transactions on Reliability*, vol. 60, no. 1, pp. 305–314, 2011.

[26] Y. Shpungin, "Networks with unreliable nodes and edges: Monte carlo lifetime estimation," *Applied Mathematics and Computer Science*, vol. 27, 01 2007.

[27] T. Elperin, I. Gertsbakh, and M. Lomonosov, "Estimation of network reliability using graph evolution models," *IEEE Transactions on Reliability*, vol. 40, no. 5, pp. 572–581, Dec 1991.

[28] H. Cancela and M. El Khadiri, "A recursive variance-reduction algorithm for estimating communication-network reliability," *IEEE Transactions on Reliability*, vol. 44, no. 4, pp. 595–602, 1995.

[29] I. B. Gertsbakh and Y. Shpungin, *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo*. Boca Raton, FL: CRC press, 2016.

[30] J. Navarro, F. J. Samaniego, N. Balakrishnan, and D. Bhattacharya, "On the application and extension of system signatures in engineering reliability," *Naval Research Logistics (NRL)*, vol. 55, no. 4, pp. 313–327, 2008.

[31] H. Wang, "A survey of maintenance policies of deteriorating systems," *European Journal of Operational Research*, vol. 139, no. 3, pp. 469–489, 2002.

[32] L. Bertling, R. Allan, and R. Eriksson, "A reliability-centered asset maintenance method for assessing the impact of maintenance in power distribution systems," *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 75–82, 2005.

[33] Y. Shang, W. Wu, J. Liao, G. Jianbo, J. Su, W. Liu, and Y. Huang, "Stochastic maintenance schedules of active distribution networks based on monte-carlo tree search," *IEEE Transactions on Power Systems*, 2020.

[34] F. J. Samaniego, "On closure of the ifr class under formation of coherent systems," *IEEE Transactions on Reliability*, vol. R-34, no. 1, pp. 69–72, April 1985.

[35] I. Gertsbakh and Y. Shpungin, "Combinatorial approaches to monte carlo estimation of network lifetime distribution," *Applied Stochastic Models in Business and Industry*, vol. 20, no. 1, pp. 49–57, 2004.

[36] M. Finkelstein and J. Vaupel, "On random age and remaining lifetime for populations of items," *Applied Stochastic Models in Business and Industry*, vol. 31, no. 5, pp. 681–689, 2015.

[37] M. Pollack, "Letter to the editor - the maximum capacity through a network," *Operations Research*, vol. 8, no. 5, pp. 733–736, 1960.

[38] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. New Jersey: Prentice Hall, 1993.

[39] T. Hu, "Letter to the editor - the maximum capacity route problem," *Operations Research*, vol. 9, no. 6, pp. 898–900, 1961.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, Massachusetts: MIT Press, 2009.

[41] R. Dekker, R. E. Wildeman, and F. A. Van der Duyn Schouten, "A review of multi-component maintenance models with economic dependence," *Mathematical Methods of Operations Research*, vol. 45, no. 3, pp. 411–435, 1997.

[42] D. Funke, S. Lamm, U. Meyer, M. Penschuck, P. Sanders, C. Schulz, D. Strash, and M. von Looz, "Communication-free massively distributed graph generation," *Journal of Parallel and Distributed Computing*, vol. 131, pp. 200–217, 2019.

[43] R. Durstenfeld, "Algorithm 235: random permutation," *Communications of the ACM*, vol. 7, no. 7, p. 420, 1964.

[44] I. B. Gertsbakh and Y. Shpungin, "Stochastic models of network survivability," *Quality Technology & Quantitative Management*, vol. 9, no. 1, pp. 45–58, 2012.

[45] J. Li and P. Mohapatra, "An analytical model for the energy hole problem in many-to-one sensor networks," in *IEEE Vehicular Technology Conference*, vol. 62, no. 4, 2005, p. 2721.

[46] F. P. Coolen and T. Coolen-Maturi, "Generalizing the signature to systems with multiple types of components," in *Complex Systems and Dependability*. Berlin: Springer, 2013, pp. 115–130.