

Designing robust network topologies for wireless sensor networks in adversarial environments

Aron Laszka^a, Levente Buttyán^a, Dávid Szeszlér^b

^a*Department of Telecommunications,
Budapest University of Technology and Economics,
www.crysys.hu*

^b*Department of Computer Science and Information Theory,
Budapest University of Technology and Economics,
Magyar tudósok körútja 2., 1117 Budapest, Hungary*

Abstract

In this paper, we address the problem of deploying sink nodes in a wireless sensor network such that the resulting network topology be robust. In order to measure network robustness, we propose a new metric, called persistence, which better captures the notion of robustness than the widely known connectivity based metrics. We study two variants of the sink deployment problem: sink selection and sink placement. We prove that both problems are NP-hard, and show how the problem of sink placement can be traced back to the problem of sink selection using an optimal search space reduction technique, which may be of independent interest. To solve the problem of sink selection, we propose efficient heuristic algorithms. Finally, we provide experimental results on the performance of our proposed algorithms.

Keywords: topology robustness, graph metric, wireless sensor network, denial-of-service attack, node placement

1. Introduction

A usual assumption on wireless sensor networks is that they consist of resource constrained and physically unprotected devices that use wireless channels for communications. These limitations make wireless sensor networks vulnerable to denial-of-service type attacks, such as physical destruction of nodes, exhaustion of their batteries, and jamming of the wireless channels. Such attacks may be addressed at different levels in the system architecture; in this work, we focus on mitigating them by controlled node deployment resulting in robust network topologies.

Email addresses: laszka@crysys.hu (Aron Laszka), buttyan@crysys.hu (Levente Buttyán), szeszler@cs.bme.hu (Dávid Szeszlér)

More specifically, we assume that the locations of the sensor nodes are pre-determined by the application requirements, which is indeed the case in most civilian applications of sensor networks (see, e.g., [1] for some supporting arguments); however, the network operator has some freedom in choosing the location of the sink nodes (or the gateways to some backbone infrastructure). We aim at determining the location of the sinks in such a way that the resulting network be resistant to node and link removal attacks (abstracting physical node destruction or exhaustion and jamming, respectively).

In order to be able to compare different sink placements in terms of robustness of the resulting network, we need to measure network robustness quantitatively. Measuring and comparing network topologies was recently listed as one of the interesting open research questions in networking [2]. Instead of the usual approach of using connectivity for this purpose, we introduce the notion of *persistence*. Roughly, the persistence of a network is defined as the minimum ratio between the cost of an attack and the gain of the attacker, where the cost of an attack is related to the difficulty of removing the attacked links or nodes, and the gain of the attacker is determined by the importance of the nodes that, as a result of the attack, get disconnected from the sinks. We explain and give some illustrative examples in Subsection 2.2 and 4.2 why persistence is a better robustness metric for wireless sensor networks than connectivity.

Using persistence as the robustness metric, we formalize and study two variants of the sink deployment problem. In the first variant, we restrict the set of possible sink locations to the set of sensor node locations (in other words, we allow some of the sensors to be extended with sink functionality); we call this variant the sink selection problem. In the second variant, we remove this restriction, and allow sinks to be placed anywhere in the deployment area; we call this the sink placement problem. In both variants, we aim at achieving a given level of persistence while minimizing the deployment cost (or equivalently, maximizing persistence under a given upper bound on the deployment budget). We prove that both sink selection with required persistence and sink placement with required persistence are NP-hard. We propose greedy and genetic heuristic algorithms to solve the sink selection problem efficiently, and we show how the problem of sink placement with required persistence can be traced back to the problem of sink selection with required persistence by an efficient search space reduction technique, which may be of independent interest. We also show how any sink selection algorithm, including our proposed heuristic algorithms, can be used to efficiently obtain solutions to the sink placement problem using our search space reduction technique. Finally, we provide experimental results on the performance of our heuristic algorithms for sink selection and our search space reduction algorithm.

This article is a follow-up work of our previous paper [3], compared to which here we prove that the problem of sink selection with required persistence is NP-hard, propose a new genetic algorithm for sink selection, formalize the problem of sink placement with required persistence and prove that it is NP-hard, propose a new search space reduction technique and show how any sink selection algorithm can be used to find a solution to the placement problem using the

reduced search space, and provide experimental results on our algorithms proposed in this article.

The organization of this paper is the following. In Section 2, we give a brief overview of placement problems in wireless sensor networks and previously proposed solutions. We also give an overview of topology robustness metrics and present a motivating example to show why we believe that vertex and edge connectivity as robustness metrics are not practical. In Section 3, we introduce (various versions of) persistence for the purpose of measuring the robustness of networks. In Section 4, we formalize the sink selection problem for a given network topology and prove that it is NP-hard. In Section 5, we present an integer programming model of the sink selection problem, which can be used to find optimal solutions. We also propose efficient greedy and genetic algorithms as heuristics to get solutions that are reasonably close to optimal. In Section 6, we formalize the sink placement problem and prove that it is NP-hard. In Section 7, we propose an algorithm for solving the sink placement problem based on an efficient search space reduction technique. We show that our search space reduction technique guarantees optimal solutions and that it can be applied to a wide range of placement problems. In Section 8, we describe the measurements that we have conducted to analyze the performance of our proposed heuristic algorithm for sink selection and to estimate the effectiveness of our search space reduction technique. Finally, in Section 9, we conclude the paper.

2. Related work

2.1. Sink placement

The optimal placement of sink nodes in wireless sensor networks is an important problem, which has been extensively studied in the literature. For a comprehensive survey on sink node placement, see [4]. For a general survey on node placement in wireless sensor networks, see [5].

In **single sink placement** problems, the network contains only a single sink node, which is often referred to as the base station. The goal is to find a sink location for which a performance metric is maximal. The most commonly used metrics include the lifetime of the network measured as the time until the most loaded node [6, 7] or until a given fraction of the nodes run out of battery [7], the number of sensors that can transmit their data [6], and the maximum throughput [8].

In **multiple sink placement** problems, the network can contain more than one sink nodes. These nodes are often referred to as gateways, because they forward the collected data to a center on longer range links. If the number of sinks that can be placed is constrained, the goal is to find a set of locations for which a performance metric is maximal. The most commonly used metrics include the worst case delay measured in the maximum number of hops between any node and the nearest sink [9, 10], the total number of hops between each node and the nearest sink [11, 10], the network lifetime measured as the time until the most loaded node runs out of battery [12, 13] or until a given fraction of nodes become unreachable [14], and the network capacity or data rate [13, 15].

If the number of sink nodes is not known in advance, the problem includes finding the minimum number of sinks that is feasible for a given constraint. The most commonly used constraints include the lifetime of the network measured as the time until a fraction of nodes become unreachable [14] or until the most loaded node dies [12], the maximum number of hops between any node and the nearest sink [16, 10], and the total number of hops between each node and the nearest sink [10].

The problem can also be defined such that the number of sinks have to be minimized and a given performance metric has to be optimized simultaneously. In this case, the goal can be to find a set of Pareto optimal solutions [17], or to find the optimal solution for a given trade-off ratio between the number of sinks and the performance metric [12].

A prevalent approach to solving multiple sink placement problems is sensor node **clustering**. In a clustering scheme, the nodes of the network are first grouped into disjoint clusters, which will be served by distinct sink nodes. Then, a sink node, which in this case is usually called a cluster-head, can be placed for each cluster using a single sink placement algorithm. While the assignment of sensor nodes to sinks after sink placement is sometimes also called clustering, here we only use the term clustering in the above sense. Naturally, the number of clusters, and therefore, the number of sinks can either be part of the objective function or be a pre-specified number. In the former case, the number of clusters has to be minimized under some constraints. For example, in [18], the size and radius of each cluster and the maximum amount of traffic each node has to relay are limited. In the latter case, the nodes have to be assigned to a fixed number of clusters. Probably the most widespread method to achieve this is *k*-means clustering, which, for example, is used by the GOALE algorithm proposed in [16]. If multi-level networks are allowed, hierarchical clustering algorithms can also be used. For example, in [19], cluster-heads aggregate data from cluster members and forward it to the next level cluster-head. A survey on clustering algorithms for wireless sensor networks can be found in [20].

One of the challenges of sink placement is the infinite size of the search space; therefore, the problem is often constrained by restricting the possible positions of the sinks to a set of candidate locations. If sinks can only be placed at the locations of the sensor nodes, such as in [18, 19], the problem is reduced to selecting a subset of nodes to be sinks, which we will refer to as a **sink selection** problem. The set of candidate locations can also be determined by an algorithm based on the geometry of sensor node positions. A wide variety of such algorithms have been proposed, for example, finding the intersections of circles centered at the nodes [6], sampling locations from the intersecting regions of discs centered at the nodes [9], creating two dimensional grids centered at the centroids of clusters after the nodes have been clustered [16], or sampling locations from dominating intersecting regions [10]. The set of candidate locations can also be pre-specified, i.e., it can be an input parameter of the problem. For example, this set is an arbitrary set in [12, 17], whereas in [8], a certain number of pre-specified locations are given from which one has to be selected as the location of the sink, while the others become the locations of the sensor nodes.

Since most sink placement problems are NP-hard, even when they are constrained to sink selection, heuristic, metaheuristic and approximation algorithms are used to solve them in practice. Greedy algorithms [18, 15] and other heuristics employing greedy decisions [12, 10] are probably the most prevalent heuristic approaches. Among metaheuristics, genetic algorithms are the most often used [9, 11, 16] as they are likely to produce good solutions, even though they have no performance guarantees. Besides genetic algorithms, other, more problem specific metaheuristics can also be used [17]. In [13], a set of procedures is proposed to design approximation algorithms for sink placement problems under any desired small error bound. Two examples are given, where this framework can be employed, placement to maximize network lifetime and placement to maximize network capacity.

What we have discussed so far is static positioning. If the nature of the application allows the sinks to be relocated and the network to be reconfigured after it has been deployed, then dynamic placement can be used to improve network performance; for example, in [21], sinks are periodically relocated to prolong network lifetime. Node failures can be anticipated by employing reactive reconfiguration schemes; for example, in [22], an efficient re-clustering mechanism is proposed to recover sensors from failed clusters.

The main difference between the prior works and our work is that we optimize for robustness against adversarial attacks measured in persistence (or deployment cost under a persistence bound) while none of the prior work did that. The similarity is that we use greedy heuristics as the problem is hard.

2.2. Robustness metrics

In the literature on wireless sensor networks, vertex- and edge-connectivity are the most frequently used metrics for measuring the robustness of network topologies [23, 5, 24, 25, 26, 27]. These metrics take the topology graph as input and return the minimum number of vertices or edges, respectively, that have to be removed in order to disconnect the graph. More precisely, a graph is said to be k -vertex-connected, if it remains connected whenever fewer than k vertices are removed, and the vertex-connectivity of a graph is the largest k for which it is k -vertex-connected. The definitions of k -edge-connectedness and edge-connectivity are similar, with the difference that they are concerned with the removal of edges, instead of vertices.

Unfortunately, connectivity as a measure of topology robustness has some weaknesses, which limit its practical usage, especially in adversarial environments. The basic problem is that the connectivity metrics are only concerned with whether a graph remains connected or not under an attack of a given maximum strength, but in practice, the strength of the attacker, in terms of number of vertices or edges that he can remove from the network, may be difficult to estimate. In addition, connectivity metrics are only concerned with the effect of the smallest effective attack, but they do not shed light on how “scattered” the graph becomes when it gets disconnected. In real scenarios, it is also important to characterize how the network fails as the strength (or budget) of the attacker increases.

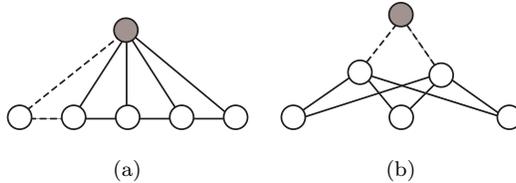


Figure 1: Illustration of why connectivity based metrics do not characterize topology robustness of wireless sensor networks well enough. The edge-connectivity of both graphs is 2, and thus, they are equally robust in terms of edge-connectivity. However, when the two dashed edges are removed, only a single vertex is separated from the sink in graph (a), while all of the vertices are separated from the sink in graph (b).

As an example, let us consider Figure 1, where two graphs are shown. Each graph represents a sensor network, in which the objective is to transfer measurement data from the nodes to the sink, represented by the shaded vertex. Both of these graphs have an edge-connectivity of 2, and therefore, they are supposed to be equally robust. Obviously, this is not true, because if the dashed edges are removed from the graphs, a single vertex is separated from the sink in graph (a), while all of the vertices are separated from the sink in graph (b).

Another known topology robustness metric, with several theoretical results, is graph *toughness* [28]. Toughness measures the minimum ratio of vertices removed to the number of components in the resulting graph. Unfortunately, the toughness of a graph is NP-hard to compute, and thus, it is not well-suited for general practical use, especially when one is concerned with large graphs.

Another similar metric is graph *strength*, which measures the minimum ratio of edges removed to the increase in the number of components in a graph [29]. The advantage of graph strength as a robustness metric is that it considers various attack strengths by default due to the fact that the minimum is taken over all possible edge removal attacks. In addition, unlike toughness, it can be computed efficiently.

3. Measuring the robustness of networks

3.1. Our proposed metrics

A common disadvantage of all robustness metrics mentioned above is that they lack the ability to incorporate the role of sink nodes. The following notion of *persistence* attempts to fill this hiatus. Its definition is obtained by an extension of the notion of *directed graph strength* introduced in [29]. However, since this notion is substantially different from *graph strength* defined above, we renamed it to avoid ambiguity. We also tailored the definition (and the corresponding computation algorithm) to the needs of sensor networks: we can allow for multiple sinks, attacks against vertices and undirected edges.

Consider a directed graph G and suppose that a subset of vertices $R \subseteq V(G)$ is given. Assume that each vertex v needs to communicate with *any* arbitrary element of R (that is, an element of R should be reachable from v through a directed path in G). Furthermore, each arc e is assigned a weight $s(e)$ that measures the cost of removing (or “attacking”) e . Finally, each node v is assigned a weight $d(v)$ that measures the loss (or “punishment”) if no element of R becomes reachable from v . When applied to model sensor networks, elements of R correspond to sink nodes, the edge weight $s(e)$ represents the difficulty of jamming the corresponding link e and the node weight $d(v)$ represents the importance of information collected by v .

For every subset of arcs $A \subseteq E(G)$ let $s(A) = \sum_{e \in A} s(e)$ and let $\lambda(A)$ be the sum of the weights $d(v)$ on those vertices v from which no element of R becomes reachable after deleting all arcs in A . Obviously, $s(A)$ and $\lambda(A)$ can be assumed to be the total attack cost and the total gain of the attacker, respectively. Accordingly, the smaller the ratio $\frac{s(A)}{\lambda(A)}$ is, the more efficient the attack of removing A is. Therefore, it makes sense to define a robustness measure as the minimum of these ratios.

Definition 1 ((Edge-)persistence). Given a directed graph G , sink nodes $R \subseteq V(G)$, edge weights $s : E(G) \rightarrow \mathbb{R}^+$ and node weights $d : V(G) \rightarrow \mathbb{R}^+$, the *persistence* (or *edge-persistence*) $\pi(G)$ is defined as

$$\pi(G) = \min \left\{ \frac{s(A)}{\lambda(A)} : A \subseteq E(G), \lambda(A) > 0 \right\}.$$

For example, consider again the two graphs of Figure 1. Assume in both cases that the shaded vertex is the (single) sink node, all edge weights $s(e)$ and node weights $d(v)$ are 1 and all edges are directed both ways. Then for both graphs the minimum in the above definition is attained at the set of edges entering the sink node. Therefore $\pi(G) = 1$ for graph (a) and $\pi(G) = \frac{2}{5}$ for graph (b). This coincides with our previous observation that graph (a) is intuitively more robust than graph (b), and thus, supports our statement that persistence is a more suitable robustness metric for wireless sensor networks than connectivity.

As mentioned in Section I, attacks against sensor networks are not restricted to destroying links between the devices (that is, edges of the graph), the devices themselves (that is, vertices of the graph) can also be the target of an attack. Therefore, in order to serve the needs of sensor networks, the above definition should be modified to allow the destruction of both edges and vertices:

Definition 2 (Edge-vertex-persistence). Given a directed graph G , sink nodes $R \subseteq V(G)$, edge and node destruction costs $s : (V(G) \cup E(G)) \rightarrow \mathbb{R}^+$ and node weights $d : V(G) \rightarrow \mathbb{R}^+$, the *edge-vertex-persistence* $\pi_v(G)$ should be defined as

$$\pi_v(G) = \min \left\{ \frac{s(A)}{\lambda(A)} : A \subseteq (V(G) \cup E(G)), \lambda(A) > 0 \right\},$$

where $s(A) = \sum_{a \in A} s(a)$ and $\lambda(A)$ is the sum of weights $d(v)$ on those vertices v from which no (remaining) element of R is reachable after deleting all edges and vertices in A . (Naturally, vertices belonging to A also become isolated from R , so these contribute to the value of $\lambda(A)$ too.)

Fortunately, computing edge-vertex-persistence can easily be reduced to computing edge-persistence by *vertex splitting*, a well-known trick in graph theory: replace each node v by two nodes v_1 and v_2 , add the arc (v_1, v_2) to G , let $s((v_1, v_2)) = s(v)$, $d(v_1) = d(v)$, $d(v_2) = 0$ and let $v_2 \in R$ if and only if $v \in R$ was originally true; finally, replace each original arc (u, v) by (u_2, v_1) and set $s((u_2, v_1)) = s((u, v))$. It is fairly easy to see that the edge-persistence of the obtained graph is the same as the edge-vertex-persistence of the original one.

Furthermore, links in wireless networks can be bidirectional. Therefore, the definition should be further modified to allow undirected graphs:

Definition 3 (Undirected persistence). Given an undirected graph G , sink nodes $R \subseteq V(G)$, edge weights $s : E(G) \rightarrow \mathbb{R}^+$ and node weights $d : V(G) \rightarrow \mathbb{R}^+$, the *undirected persistence* $\pi_u(G)$ is defined as

$$\pi_u(G) = \min \left\{ \frac{s(A)}{\lambda(A)} : A \subseteq E(G), \lambda(A) > 0 \right\},$$

where $\lambda(A)$ is the sum of the weights $d(v)$ on those vertices v from which there is no undirected path to any element of R after deleting all edges in A .

Computing undirected persistence can be reduced to computing persistence by replacing every edge with two arcs facing opposite directions, having the attack cost of the original edge. It is easy to see that the persistence of the obtained graph is the same as the undirected persistence of the original one.

Due to the arguments above, we only consider edge-persistence of directed graphs in the remainder of the paper.

3.2. Other uses of persistence

3.2.1. Persistence as a measure of robustness against random faults

Persistence can be also used to measure the robustness of a network against random link faults under some restrictive assumptions. Assume that each link e malfunctions independently with probability $p(e)$. Assign to each edge $e \in E(G)$ of the representing graph $-\log p(e)$ weight. Then, the persistence of the graph measures

$$\begin{aligned} \pi(G) &= \min \left\{ \frac{s(A)}{\lambda(A)} : A \subseteq E(G), \lambda(A) > 0 \right\} \\ &= \min \left\{ \frac{\sum_{e \in A} -\log p(e)}{\lambda(A)} : A \subseteq E(G), \lambda(A) > 0 \right\} \\ &= \min \left\{ \frac{-\log \prod_{e \in A} p(e)}{\lambda(A)} : A \subseteq E(G), \lambda(A) > 0 \right\} \\ &= \min \left\{ \frac{-\log p(A)}{\lambda(A)} : A \subseteq E(G), \lambda(A) > 0 \right\}, \end{aligned} \tag{1}$$

where $p(A)$ is the probability that all links in A malfunction. Intuitively, this means that the probability of an event decreases exponentially with its impact.

3.2.2. Persistence as a measure of network lifetime

In [30], the following model is proposed for measuring the lifetime of wireless sensor networks.

Consider a directed graph $G(N, A)$ where N is the set of all nodes and A is the set of all directed links (i, j) where $i, j \in N$. Let S_i be the set of nodes that are in the transmission range of node i . Each node has the initial battery energy of E_i , and the amount of energy consumed in transmitting a packet across link (i, j) is denoted by e_{ij} where $j \in S_i$.

The goal is to maximize the time T until which the information generated can be delivered to one of the set of gateway nodes $D \subset N$. Let $Q_i(T)$ be the number of packets generated at a sensor node $i \in N \setminus D$ until T , and let $q_{ij}(T)$ be the total number of packets routed through link $(i, j) \in A$. A time T is feasible if there exists a set of non-negative integers $q_{ij}(T)$ for all links $(i, j) \in A$ satisfying the following constraints. First, the conservation of flow constraint is

$$\sum_{j: i \in S_j} q_{ji}(T) + Q_i(T) = \sum_{k \in S_i} q_{ik}(T), \quad \forall i \in N \setminus D. \quad (2)$$

Second, the total energy constraint is

$$\sum_{j \in S_i} e_{ij} q_{ij}(T) \leq E_i, \quad \forall i \in N \setminus D. \quad (3)$$

For a set of nodes V , assume that each node $i \in V$ has the amount of information generated until T , $Q_i(T)$, which needs to be delivered out of V . For a node $i \in V$, let e_i^V be the least energy expenditure for transporting an information unit out of V . If there is no outgoing link of i through which information can be forwarded out of V , $e_i^V = \infty$ by convention. The necessary feasibility condition is

$$\sum_{i \in V} Q_i(T) \leq \sum_{i \in V} \frac{E_i}{e_i^V}. \quad (4)$$

Unfortunately, the necessary condition is not sufficient. However, if the energy expenditure through all the outgoing links of a sensor are the same, then the condition is sufficient as well. In this case, maximum lifetime becomes equivalent with undirected-vertex-persistence.

3.3. Computing persistence

It is shown in [29] that computation of persistence can be performed using a maximum flow algorithm¹. In particular, assume that besides the input data

¹In this subsection we build on the basics of network flow theory; the required background can be found in most introductory graph theory textbooks.

used above (that is, $G, R \subseteq V(G)$, $s : E(G) \rightarrow \mathbb{R}^+$ and $d : V(G) \rightarrow \mathbb{R}^+$) a constant π_0 is also given: π_0 represents a required persistence value and the task is to decide if $\pi(G) \geq \pi_0$ holds.

For any set $X \subseteq V(G)$, denote by $\delta(X)$ the set of edges leaving X and let $\delta_s(X) = \sum\{s(e) : e \in \delta(X)\}$. It is easy to see that the minimum in the definition of $\pi(G)$ is attained at a set $A = \delta(X)$ for a suitable $X \subseteq V(G) \setminus R$. (Indeed, “spare” edges could be deleted from A without increasing the ratio $s(A)/\lambda(A)$.) Of course, $A = \delta(X)$ implies $s(A) = \delta_s(X)$ and $\lambda(A) = d(X)$ (where $d(X) = \sum_{v \in X} d(v)$). Therefore $\pi(G) \geq \pi_0$ is equivalent to saying that $\delta_s(X) - \pi_0 \cdot d(X) \geq 0$ holds for all $X \subseteq V(G) \setminus R$. Adding $\pi_0 \cdot d(V(G))$ to both sides we get that $\pi(G) \geq \pi_0$ is equivalent to

$$\delta_s(X) + \pi_0 \cdot d(\bar{X}) \geq \pi_0 \cdot d(V(G)) \quad (5)$$

for all $X \subseteq V(G) \setminus R$ (where $\bar{X} = V(G) \setminus X$).

Consider the following maximum network flow problem. Add two new nodes, s^* and t^* to G ; for each $v \in V(G)$ add a new arc from s^* to v and set its capacity to $\pi_0 \cdot d(v)$; for each $v \in R$ add a new arc from v to t^* and set its capacity to infinity; finally, set the capacity of each original arc of G to $s(e)$. Denote the obtained network by G^* . According to the well-known “max-flow-min-cut” theorem of Ford and Fulkerson, the maximum flow in the obtained network from s^* to t^* is equal to the minimum cut capacity, that is, the minimum of the sum of capacities on arcs leaving a set X , where minimum is taken over all subsets $X \subseteq V(G^*)$ for which $s^* \in X$ and $t^* \notin X$. Obviously, the capacity of the cut X is $\delta_s(X) + \pi_0 \cdot d(\bar{X})$ if $X \cap R = \emptyset$ (and infinity otherwise). Comparing this with Equation 5 above, we get that $\pi(G) \geq \pi_0$ is equivalent to the existence of a flow of value $\pi_0 \cdot d(V(G))$ from s^* to t^* in the above constructed network; or, in other words, a flow that saturates all arcs leaving s^* .

Consequently, the question of $\pi(G) \geq \pi_0$ can be answered by a maximum flow algorithm. From this, the actual value of $\pi(G)$ (that is, the maximum π_0 for which the above described flow exists) can be determined by binary search (which yields a polynomial time algorithm if all input numerical data is assumed to be integer). In [29] a refinement of this approach is also given: it is shown that $\pi(G)$ can be determined by at most $|V(G)|$ maximum flow computations (even for arbitrary input data); we disregard the details here due to lack of space.

4. The sink selection problem and its complexity

Based on the above defined robustness metric $\pi(G)$, in this section, we formalize the problem of optimal selection of sink nodes in a network with a given topology.

4.1. The sink selection problem

We assume that assigning the sink role to a node v has some cost $c(v)$ resulting from the establishment of an external connection with the node, regularly

visiting the node for data collection, *etc.* We call this cost the *selection cost* of the sink, and we assume that the cost of assigning the sink role to a set of nodes is simply the sum of selection costs of the nodes in the set. We also assume that the network topology is given and our task is to select the sink vertices such that the persistence of the resulting network configuration is above a given threshold, while the total selection cost of the sink nodes is minimized. This models the design of a wireless sensor network with strict security requirements, but a flexible budget.

According to the above, the sink selection problem is formalized as follows:

Definition 4 (Sink selection with required persistence).

INSTANCE: Directed graph G , edge weights $s : E(G) \rightarrow \mathbb{R}^+$, node weights $d : V(G) \rightarrow \mathbb{R}^+$, sink selection costs $c : V(G) \rightarrow \mathbb{R}^+$, and required persistence $\pi_0 \in \mathbb{R}^+$.

SOLUTION: A subset $R \subseteq V(G)$ such that the persistence $\pi(G)$ of G is at least π_0 with R as its sink nodes.

MINIMIZE: Selection cost of subset R , i.e., $\sum_{v \in R} c(v)$.

Obviously, the variant of the sink selection problem where an upper bound on the total sink selection cost is given and the persistence of the configuration is to be maximized is also sensible. We disregard this version of the problem, we restrict ourselves to mentioning that any algorithm to solve one of the two versions can also be used to solve the other one by binary search.

4.2. Sink selection example

In this subsection, we present a motivating example of why sinks should be selected based on maximizing persistence instead of maximizing connectivity.

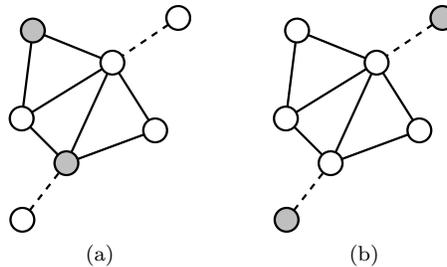


Figure 2: An example network, where sink selections based on maximizing persistence and maximizing connectivity lead to very different results. In selection (a), only one link has to be removed in order to disconnect a node, but the persistence of the network is 1, whereas in selection (b), two links have to be removed, but the persistence of the network is only 0.4.

Figure 2 shows two possible sink selections in the same network, one maximizing persistence and one maximizing connectivity. The value of every node and the attack cost of every link is one and the selected sinks are represented

by shaded nodes. The sink nodes, in addition to the links in the figure, are connected to a center as well; therefore, the connectivity of the network measures the number of links that have to be removed in order to disconnect a non-sink node. In selection (a), only one link has to be removed in order to disconnect a node, whereas in selection (b), two links have to be removed. However, the selections have a persistence of 1 and 0.4, respectively. The dashed edges are optimal attacks against the network. In the case of selection (a), only two nodes are separated if the attack is carried out, whereas in the case of selection (b), five nodes are separated. If the network application tolerates the loss of a few nodes, then selection (a) is clearly the better choice.

4.3. Complexity of the sink selection problem

In this subsection, we prove that the Sink Selection problem is NP-hard. To this end, we show that the Minimum Set Cover Problem, one of the well-known NP-hard problems, can be reduced to it. The (decision version of the) Minimum Set Cover problem is defined as follows.

Definition 5 (Minimum Set Cover).

INSTANCE: A finite set $U = \{u_1, u_2, \dots, u_n\}$, a collection of its subsets $\mathcal{A} = \{A_1, A_2, \dots, A_m\}$ ($A_i \subseteq U$ for all $1 \leq i \leq m$), and a positive integer r .

TASK: Decide if it is possible to choose at most r subsets from \mathcal{A} that cover U ; that is, if the subsets $A_{i_1}, A_{i_2}, \dots, A_{i_p}$ can be chosen such that $p \leq r$ and $\bigcup_{j=1}^p A_{i_j} = U$. (The chosen subsets A_{i_j} are said to form a *set cover of size p*.)

The decision version of the Sink Selection problem is defined in the most natural way.

Definition 6 (Sink Selection with Required Persistence (decision version)).

INSTANCE: Directed graph G , edge weights $s : E(G) \rightarrow \mathbb{R}^+$, node weights $d : V(G) \rightarrow \mathbb{R}^+$, sink selection costs $c : V(G) \rightarrow \mathbb{R}^+$, required persistence $\pi_0 \in \mathbb{R}^+$ and maximum cost $c_0 \in \mathbb{R}^+$.

TASK: Decide if a subset $R \subseteq V(G)$ exists such that $\pi(G, R) \geq \pi_0$ and $\sum_{v \in R} c(v) \leq c_0$.

In what follows, we will use the basic concepts of algorithmic complexity theory without further explanation or reference. However, to assist those unacquainted with these, we just remark that the basic idea is very simple even by common sense: roughly speaking, the above mentioned reduction shows that the Minimum Set Cover problem is a special case of the Sink Selection problem. Since the former is known to be a hard problem (whatever the precise meaning of that is), this obviously makes the latter one also hard.

Theorem 1. *The Sink Selection problem is NP-complete.*

PROOF. The problem is obviously in NP since if R is given then checking if $\pi(G, R) \geq \pi_0$ can be done in polynomial time according to [29] (and checking if $\sum_{v \in R} c(v) \leq c_0$ is trivial).

As mentioned above, we show NP-hardness by reducing the Minimum Set Cover problem to the Sink Selection problem. So assume that an instance of the Minimum Set Cover problem (that is, $U = \{u_1, \dots, u_n\}$, $\mathcal{A} = \{A_1, \dots, A_m\}$ and r) is given. Obviously, we can assume that $\bigcup_{j=1}^m A_j = U$ (otherwise there exists no set cover at all) and that $r \leq m$ (otherwise the problem is trivial). From this, we construct an instance of the Sink Selection problem in the following way:

1. Let $V(G) = U \cup \mathcal{A}$; that is, to each element $u_i \in U$ and to each given subset $A_i \in \mathcal{A}$ corresponds a vertex of G .
2. Let $E(G) = \{(u_i, A_j) : u_i \in U, A_j \in \mathcal{A}, u_i \in A_j\}$; that is, whenever $u_i \in A_j$ holds for an element $u_i \in U$ and $A_j \in \mathcal{A}$, we introduce a directed edge from u_i to A_j in G . (Consequently, G is a directed bipartite graph.)
3. Let $d(u_i) = 1$ and $c(u_i) = r + 1$ for every $u_i \in U$, let $d(A_j) = 0$ and $c(A_j) = 1$ for every $A_j \in \mathcal{A}$ and let $s(e) = 1$ for every $e \in E(G)$.
4. Finally, let $\pi_0 = 1$ and $c_0 = r$.

We have to show that there exists a set cover $\mathcal{A}_0 \subseteq \mathcal{A}$ of size at most r if and only if there exists a sink selection $R \subseteq V(G)$ such that $\pi(G, R) \geq \pi_0 = 1$ and $\sum_{v \in R} c(v) \leq c_0 = r$.

First assume that a sink selection R with the above properties is given. Since the costs of the vertices in U are all $r + 1$, R only contains vertices from \mathcal{A} . We claim that $\mathcal{A}_0 = R$ forms a set cover. Assume that this is not true. Then there is an element $u_i \in U$ not covered by R ; that is, $u_i \notin \bigcup \{A_j : A_j \in R\}$. Consequently, no element of R is reachable from u_i in G , therefore $\lambda(\emptyset) \geq 1$ holds by $d(u_i) = 1$. Since obviously $s(\emptyset) = 0$, this implies $\pi(G, R) = 0$ (by $\frac{s(\emptyset)}{\lambda(\emptyset)} = 0$), a contradiction. So R is a set cover and its size is obviously at most r (since $c(A_j) = 1$ for every $A_j \in \mathcal{A}$).

Now assume that a set cover \mathcal{A}_0 of size $p \leq r$ is given and let $R = \mathcal{A}_0$. We claim that R is a sink selection in G that fulfills the above requirements. Obviously, $\sum_{v \in R} c(v) = p \leq r$. To show $\pi(G, R) \geq 1$, let $Y \subseteq E(G)$ be any subset of edges; we need to prove that $s(Y) \geq \lambda(Y)$. Obviously, $s(Y) = |Y|$ and $\lambda(Y) = |X|$, where $X \subseteq U$ is the set of vertices in U from which no element of R is reachable after removing Y . (Observe that X is not the set of *all* vertices of G from which R is not reachable after removing Y : vertices in $\mathcal{A} \setminus R$ also have this property. However, these do not contribute to $\lambda(Y)$ as they have a weight of 0.) So $|Y| \geq |X|$ is to be proved. However, this immediately follows from the fact that since R is a set cover, for every $u_i \in X$ there exists an $A_j \in R$ such that $u_i \in A_j$, so we can assign a separate edge in Y to each element of X (namely, the one that goes from u_i to A_j).

We remark that the construction of the above proof could be modified in the following way: instead of setting $c(u_i) = r + 1$ for every $u_i \in U$, u_i could be replaced by $r + 1$ vertices, each with a cost of 1 (and each connected to

A_j , whenever $u_i \in A_j$). It is easy to verify that the proof goes through with the modified construction as well, which shows that the Sink Selection problem remains to be NP-complete even under the restriction that $c(v) = 1$ for every $v \in V(G)$.

The essence of the above proof is that in the described construction set covers of size r and proper sink selections of total cost r are basically the same. This simple fact has an important consequence on the approximability of the Sink Selection problem.

Theorem 2. *Assuming that $P \neq NP$, there exists a constant $c > 0$ such that there is no polynomial time algorithm that finds a sink selection of total cost at most $c \log \log v \cdot OPT$, where v is the number of vertices in the graph and OPT denotes the minimum total cost of a sink selection.*

We remark for the benefit of those unacquainted with algorithmic complexity theory that $P \neq NP$ is a widely accepted conjecture; if this were not true, then there would be a polynomial time algorithm for *every* NP-hard problem. An obvious corollary of the above theorem is that there is no constant factor approximation algorithm for the sink selection problem unless $P = NP$.

PROOF. Assume that there is an algorithm P that finds a sink selection of total cost at most $c \log \log v \cdot OPT$ for some constant c . Then if an instance $U = \{u_1, \dots, u_n\}$, $\mathcal{A} = \{A_1, \dots, A_m\}$ of the (optimization version of the) Minimum Set Cover problem is given, P can be applied on the construction given in the above proof. Then, if v denotes the number of vertices in the constructed graph, $v = n + m \leq n + 2^n \leq 2^{n+1}$ holds. Since, as remarked above, sink selections in the constructed graph are essentially the same as possible solutions of the given Minimum Set Cover instance, OPT also denotes the minimum size of a set cover for this. Therefore P finds a set cover of size at most $c \log \log 2^{n+1} \cdot OPT = c \log(n+1) \cdot OPT$. However, Raz and Safra [31] proved that if $P \neq NP$ then there is no polynomial time algorithm that finds a set cover of size at most $c_1 \log n \cdot OPT$, where $c_1 > 0$ is an appropriate constant. This immediately implies the existence of the required c .

5. Algorithms for solving the sink selection problem

In this section, we show that the optimal selection can be found by solving an integer program and we also introduce more efficient heuristic algorithms that approximate the optimal solution reasonably well. Performance evaluation and quantitative comparison of these algorithms is provided in Subsection 8.1.

5.1. Integer programming model for the sink selection problem

To formulate the sink selection problem as an integer program, we assign a binary variable $r(v)$ to each node v : the value of $r(v)$ is 1 or 0 if v belongs or does not belong to R , respectively. The formulation relies on the construction presented in Subsection 3.3: $\pi(G) \geq \pi_0$ is true if and only if there exists a flow in

the network G^* described there that saturates all edges (s^*, v) . Correspondingly, we assign a variable $f(e)$ to each edge $e \in E(G^*)$ to measure the flow on e . As it is natural in network flow theory, all the constraints ensuring that f is a flow (that is, capacity constraints and flow preservation constraints) can straightforwardly be formalized as linear constraints.

The only difference from the construction described in Subsection 3.3 is that the set of sink nodes R is not known in advance. Therefore we assume that an arc from v to t^* exists from each node $v \in V(G)$ and we ensure that the capacity of the arc (v, t^*) is ∞ or 0 for sink nodes and non-sink nodes, respectively. This is achieved by imposing the inequality $f((v, t^*)) \leq \mathit{bignum} \cdot r(v)$ on each edge (v, t^*) , where bignum is a sufficiently large constant (e.g., $\mathit{bignum} = \pi_0 \cdot d(V(G))$, the sum of the capacities on all arcs leaving s^* suffices, as it is an upper bound on the maximum flow value even if all vertices are assumed to be sinks).

With respect to the above, the integer program is the following:

Constants:

- bignum : a sufficiently large number
- $s((u, v))$: weight of edge (u, v)
- $d(v)$: weight of node v
- $c(v)$: selection cost of node v
- π_0 : required persistence

Variables:

- $r(v) \in \{0, 1\}$ for all $v \in V(G)$
- $f(e) \in \mathbb{R}$ for all $e \in E(G^*)$

Minimize: $\sum_{v \in V(G)} c(v) \cdot r(v)$

Constraints:

1. $\forall v \in V(G) : f((v, t^*)) \leq \mathit{bignum} \cdot r(v)$
2. $\forall e \in E(G) : f(e) \geq 0$
3. $\forall e \in E(G) : f(e) \leq s(e)$
4. $\forall v \in V(G) :$

$$\sum_{(u,v) \in E(G)} f((u, v)) = \sum_{(v,u) \in E(G)} f((v, u))$$

5. $\forall v \in V(G) : f((s^*, v)) \geq \pi_0 \cdot d(v)$

Constraints 2, 3 and 4 ensure that f is a flow: Constraints 3 and 4 correspond to capacity and flow preservation constraints, respectively. Note that capacity constraints are not imposed on (s^*, v) type arcs (as opposed to what was said in Subsection 3.3); obviously, these can be omitted as they would not affect the optimum solution. On the other hand, Constraint 5 ensures that all edges

(s^*, v) are saturated. Finally, the role of Constraint 1 was already explained above.

Obviously, the above integer program does not yield an efficient algorithm for solving the sink selection problem. However, it makes it possible to obtain the optimum solution for relatively small problem instances and thus test the heuristics presented in the following subsections and compare the search space reduction technique presented later to other techniques.

5.2. Our proposed greedy algorithm

The exponential time complexity of solving the above integer program limits its practical applicability. For this reason, in this subsection, we propose an efficient greedy algorithm as a heuristic approach to find sub-optimal, but reasonably good solution to the sink selection problem.

The algorithm starts with the set of selected sinks R as the empty set. In each step, a new vertex v is added to R ; v is chosen in a simple, but sensible way: such that the ratio of the gain in persistence by adding v to R to the selection cost $c(v)$ is maximum. The algorithm stops when the persistence $\pi(G)$ of the network with set of sinks R is at least the required persistence π_0 .

To formally describe the algorithm, denote by $\pi(G, R)$ the persistence of the network G with R as its set of sink nodes. Then our greedy algorithm for sink selection with required persistence is the following:

1. $R := \emptyset$
2. let $v \in V(G) \setminus R$ be a vertex for which the maximum

$$\max_{v \in V(G) \setminus R} \frac{\pi(G, R \cup \{v\}) - \pi(G, R)}{c(v)}$$

is attained and let $R := R \cup \{v\}$

3. If $\pi(G, R) \geq \pi_0$ then return R ; otherwise continue from Step 2.

We emphasize that, obviously, the above algorithm runs in polynomial time since it makes at most $|V(G)|$ iterations and each iteration requires at most $|V(G)|$ persistence computations.

5.3. Our proposed genetic algorithm

For a higher number of nodes, even the greedy algorithm's computational complexity may be too high; therefore, in this subsection, we propose a genetic algorithm as a more efficient alternative to the greedy algorithm.

In our proposed genetic algorithm, an individual member of the population represents a solution to the sink selection problem. To make the evolutionary process more efficient, we have chosen to encode not only the set of nodes which are selected, but also the preference for each node, including those that are not selected by the given solution. These preferences can be represented by the order in which the nodes are picked until the the given persistence value is reached. If this representation is used, then it is not necessary to record which nodes are

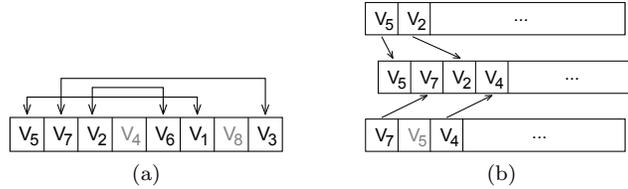


Figure 3: Illustration of how (a) mutation and (b) crossbreeding are implemented.

selected as the number of necessary nodes can be determined from the order of preference. Therefore, the solution domain is simply the set of node sequences.

We have chosen the fitness of each solution to be, naturally, the cost of the given solution. We have implemented two genetic operators (see Figure 3 for illustrations): mutation and crossover. A mutation is a reordering of the nodes, which is achieved by selecting a fixed number of random pairs from the sequence and swapping the elements of each pair. A crossover is the combination of two node sequences, which is achieved by iteratively picking the most preferred, but not yet picked node from two sequences in turns.

As the genetic operators are very simple, the key issue of our algorithm is the efficient evaluation of the fitness function. The following algorithm is proposed for this:

1. Add a source vertex s^* and a sink vertex t^* to the graph. For each $v \in V(G)$, add a new arc from s^* to v and set its capacity to $\pi_0 \cdot d_v$. Let $n = 1$.
2. Add a new arc from the n th vertex to t^* with infinity capacity.
3. Find a maximum flow in this network by augmenting the flow values of the previous iteration.
4. If the maximum flow is at least $\sum_{v \in V(G)} \pi_0 \cdot d_v$, then the cost of the given solution is $\sum_{i=1}^n c_i$. Otherwise, let $n := n + 1$ and continue from Step 2.

The construction of the graph in Step 1 is the same as in Section 3.3, except for not adding arcs from sinks to the super sink as the former are not known in advance in this case. In Step 3, the maximum flow of the previous iteration is reused, which makes the algorithm very efficient. The complexity of determining the number of necessary nodes is equal to the complexity of testing whether a graph has a given persistence, which is very simple compared to computing persistence. Therefore, we can efficiently create and evaluate a large number of solutions. The algorithm terminates after a fixed number of generations.

6. The sink placement problem and its complexity

In this section, we relax some of our previous restrictions on the design of the deployment configuration and allow sinks to be placed anywhere; however, we still consider the placement of non-sink nodes and the links between them to be given. Therefore, our goal is to design a robust sink placement for a given

network topology. By sink placement, we mean a set of locations where sink nodes have to be placed.

6.1. The sink placement problem

Before formalizing the sink placement problem, we first have to establish a sink node model and define the persistence of a placement:

Definition 7 (Persistence of a placement). Let G be a directed geometric graph, where $V(G)$ is a set of points in the Euclidean plane and $E(G)$ is an arbitrary subset of $V^2(G)$ (i.e., the edges of the graph do not have to follow any geometric rule). Let $s : E(G) \rightarrow \mathbb{R}^+$ be edge weights, let $d : V(G) \rightarrow \mathbb{R}^+$ be node weights, and let $D \in \mathbb{R}^+$ be a fixed sink transmission radius. The *persistence of a placement* R , where R is a set of points in the Euclidean plane, for graph G , denoted by $\pi_p(G, R)$, is the persistence of the graph G' with R as its sinks, where

1. $V(G') = V(G) \cup R$
2. $E(G') = E(G) \cup \{(v, r) : v \in V(G) \wedge r \in R \wedge \text{distance}(v, r) \leq D\}$
3. $\forall_{(v,r) \in V(G) \times R} s(v, r) = 1$
4. $\forall_{r \in R} d(r) = 0$.

The definition establishes our model for sink nodes, in which sinks have uniform transmission radii (2) and zero value (4) and the links connected to them have uniform weights (3). This is a realistic model for sensor networks, where typically a large number of similar nodes are deployed. The direction of the links connected to the sinks can be surprising at first since our goal is to model wireless networks, where links are bidirectional. However, we have seen that the undirected persistence of a graph is equal to the directed persistence with the undirected edges replaced by two directed edges facing opposite directions. In addition, edges leaving sinks can be omitted as no traffic has to be carried from a sink. Therefore, the direction of the sink edges is appropriate for our model. Formalizing the persistence of a placement this way simplifies our algorithm presented in Subsection 7.2.

Using the above definition of the persistence of a placement, the robust sink placement problem can be formalized as follows:

Definition 8 (Sink Placement with Required Persistence).

INSTANCE: Directed graph G , where $V(G)$ is a set of points in the Euclidean plane, edge weights $s : E(G) \rightarrow \mathbb{R}^+$, node weights $d : V(G) \rightarrow \mathbb{R}^+$, sink transmission radius D , and required persistence $\pi_0 \in \mathbb{R}^+$.

SOLUTION: A set of points R in the Euclidean plane such that $\pi_p(G, R) \geq \pi_0$.

MINIMIZE: The number of sinks required by the placement, i.e., $|R|$.

6.2. Complexity of the sink placement problem

As the sink placement problem is a generalization of the sink selection problem with uniform selection costs, it is obviously NP-hard.

To prove this, we introduce the decision version of the Sink Placement with Required Persistence problem:

Definition 9 (Sink Placement with Required Persistence (decision version)).

INSTANCE: Directed graph G , where $V(G)$ is a set of points in the Euclidean plane, edge weights $s : E(G) \rightarrow \mathbb{R}^+$, node weights $d : V(G) \rightarrow \mathbb{R}^+$, sink transmission radius D , required persistence $\pi_0 \in \mathbb{R}^+$ and maximum number of sinks $c_0 \in \mathbb{R}^+$.

TASK: Decide if it is possible to place at most c_0 sinks in the Euclidean plane such that $\pi_p(G, R) \geq \pi_0$.

Theorem 3. *The Sink Placement with Required Persistence problem is NP-hard.*

PROOF. We show NP-hardness by reducing the problem of Sink Selection with Required Persistence and Uniform Selection Costs to the problem of Sink Placement with Required Persistence. So assume that an instance of the Sink Selection problem (that is, $G, s : E(G) \rightarrow \mathbb{R}^+, d : V(G) \rightarrow \mathbb{R}^+, c_0$ and π_0) is given. We can assume that $\pi_0 < \frac{s(E(G))}{\min_{v \in V(G): d(v) > 0} \{d(v)\}}$. Otherwise, the only feasible solution to the problem is to select every $v \in V(G) : d(v) > 0$ and checking whether the cost of this selection is lower than c_0 can be done in polynomial time. From the instance of the Sink Selection problem we construct an instance of the Sink Placement problem (that is, $G', s' : E(G') \rightarrow \mathbb{R}^+, d' : V(G') \rightarrow \mathbb{R}^+, D, c'_0$ and π'_0) in the following way:

1. Let $V(G') = V(G)$, $E(G') = E(G)$, $d' = d$ and $c'_0 = c_0$.
2. Let $D = 1$ and position the vertices of G' on the plane such that the distance between each pair of vertices is greater than 2.
3. Let $s' = \beta s$, where $\beta = \frac{\min_{v \in V(G): d(v) > 0} \{d(v)\}}{s(E(G)) \sum_{v \in V(G)} d(v)}$.
4. Finally, let $\pi'_0 = \beta \pi_0$.

Let $\lambda_G(A)$ and $\lambda_{G'}(A)$ denote the value of $\lambda(A)$ in G and G' , respectively. We have to show that there exists a selection $R \subseteq V(G)$ such that $\pi(G, R) \geq \pi_0$ and $\sum_{r \in R} c(r) \leq c_0$ if and only if there exists a sink placement R' such that $\pi_p(G', R') \geq \pi'_0$ and $|R'| \leq c'_0$.

First assume that a sink placement R' with the above properties is given. Let R be the subset of vertices that have at least one sink in their proximity. We claim that R is a feasible selection, i.e., $\pi(G, R) \geq \pi_0$. Assume that this is not true. Then there is an $A \subseteq E(G)$ attack for which $\frac{s(A)}{\lambda_G(A)} < \pi_0$ holds. If a vertex can not reach any $r \in R$ sink in G when A is removed, then the same vertex can not reach any $r' \in R'$ sink in G' either as any path leading to a sink necessarily

goes through an $r \in R$. Therefore, $\lambda_G(A) \leq \lambda_{G'}(A)$. Since $s'(A) = \beta s(A)$, this implies the contradiction $\frac{s'(A)}{\lambda_{G'}(A)} \leq \frac{s'(A)}{\lambda_G(A)} = \frac{\beta s(A)}{\lambda_G(A)} < \beta \pi_0 = \pi'_0$.

Now we have to show that $\sum_{r \in R} c(r) \leq c_0$. Since the distance between each pair of vertices is greater than 2, at most one vertex is connected to every sink. Therefore, R has at most $|R'|$ vertices and $\sum_{r \in R} c(r) = \sum_{r \in R} 1 \leq |R| \leq c_0$ holds.

Now assume that a sink selection in R with the above properties is given. Let R' be a sink placement constructed from the positions of the vertices in R . We claim that R' is a feasible placement, i.e., $\pi_p(G', R') \geq \pi'_0$. Assume that this is not true. Then there is an $A \subseteq E(G')$ attack for which $\frac{s'(A)}{\lambda_{G'}(A)} < \pi'_0$ holds. First, observe that A does not contain any edges connected to sinks, i.e., $A \subseteq E(G)$. Otherwise, the $s'(A)$ cost of the attack would be at least 1 and, since $\lambda_{G'}(A) \leq \sum_{v \in V(G)} d(v)$, the ratio $\frac{s'(A)}{\lambda_{G'}(A)} \geq \frac{1}{\sum_{v \in V(G)} d(v)} = \beta \frac{s(E(G))}{\min_{v \in V(G): d(v) > 0} \{d(v)\}} > \beta \pi_0 = \pi'_0$. If a vertex v can not reach any $r' \in R'$ sink in G' when A is removed, then v can not reach any $r \in R$ sink in G either when A is removed. Otherwise, there would be a path in G from v to an $r \in R$, but then this path could be supplemented with an edge leading to a sink $r' \in R'$ and, therefore, v could reach an $r' \in R'$ sink even when A is removed. Consequently, $\lambda_{G'}(A) \leq \lambda_G(A)$. Since $s(A) = \frac{1}{\beta} s'(A)$, this implies the contradiction $\frac{s(A)}{\lambda_G(A)} \leq \frac{s(A)}{\lambda_{G'}(A)} = \frac{1}{\beta} \frac{s'(A)}{\lambda_{G'}(A)} < \frac{1}{\beta} \pi'_0 = \pi_0$.

Finally, we have to show that $|R'| \leq c_0$. This is obvious as $|R'| = |R| = \sum_{r \in R} 1 = \sum_{r \in R} c(r) \leq c_0$.

7. Algorithm for solving the sink placement problem

In this section, we introduce a technique, similar to the one in [10], which reduces the infinite search space of possible placements to a finite set that always includes an optimal solution. We also present an algorithm which can be used to find an optimal placement in the reduced search space using an arbitrary algorithm for finding an optimal selection.

7.1. Our proposed search space reduction technique

To simplify our definitions, we first introduce the concept of single sink coverable sets:

Definition 10 (Single sink coverable set). A set of points W in the Euclidean plane is *single sink coverable* for a transmission radius D , if \exists point r in the plane, such that $\forall w \in W : \text{distance}(w, r) \leq D$, i.e., the points can be covered by a disc of radius D .

We reduce the infinite search space of possible placements by restricting the positions of the sink nodes to a set of candidate locations. Contrary to most candidate location sets proposed in the literature, the subsets of our set always include an optimal solution; therefore, we call it an optimal set of candidate locations:

Definition 11 (Optimal set of candidate locations). Given a geometric graph G and a sink transmission radius D , an *optimal set of candidate locations* $R_{\text{candidate}}$ is a set of positions which includes exactly one position covering every inclusion-maximal single sink coverable subset of $V(G)$ for D , i.e., every single sink coverable set of node positions that is not included in a larger single sink coverable set.

We will prove that (1) the subsets of an optimal set of candidate locations always include an optimal placement and that (2) an optimal set of candidate locations can be found in polynomial time.

Theorem 4. *The subsets of an optimal set of candidate locations include an optimal placement for every persistence requirement.*

PROOF. Suppose that R' is an optimal sink placement for a given persistence requirement. Let the set of nodes which are covered by an $r' \in R'$ node be denoted by $N(r')$. Obviously, $N(r')$ is a single sink coverable set. If $N(r')$ is an inclusion-maximal single sink coverable set, then there is an $r \in R_{\text{candidate}}$ which covers exactly $N(r')$; otherwise, there is an $r \in R_{\text{candidate}}$ which covers all nodes covered by r' and some others as well. It is easy to see that if we substitute r' with r , then the solution still satisfies the persistence requirement.

Therefore, by substituting every $r' \notin R_{\text{candidate}}$ with an appropriate $r \in R_{\text{candidate}}$, we can obtain an $R \subseteq R_{\text{candidate}}$ which satisfies the persistence requirement and for which $|R| = |R'|$.

Before introducing our polynomial time algorithm for finding an optimal set of candidate locations, we first prove that the size of the optimal set is polynomial in the number of nodes:

Theorem 5. *There exists an optimal set of candidate locations with cardinality $O(|V(G)|^2)$.*

PROOF. (see Acknowledgements for credit): We prove the statement by assigning a pair of its points to each inclusion-maximal single sink coverable set injectively (that is, each pair of points in $V(G)$ will be assigned to at most one inclusion-maximal single sink coverable set). This will obviously imply that the number of inclusion-maximal single sink coverable sets is at most $\binom{|V(G)|}{2}$ and, therefore, choosing a corresponding center for each will provide the required optimal set of candidate locations.

So assume that an inclusion maximal single sink coverable set Z and a disc C of radius D covering Z is given. Now push C upwards (that is, to the positive direction of the y -axis) without losing any point of Z until (at least) one point of Z is on the lower semicircle of the boundary of C ; denote this point by p . Now, if p is left from the vertical diameter of C then rotate C around p counter-clockwise without losing any point of Z until (at least) one further point of Z is on the semicircle of the boundary of C that is right from the diagonal passing through p . If, on the other hand, p is right from (or on the) vertical diameter of

C then do the rotation clockwise until (at least) one further point of Z is on the semicircle of the boundary of C that is left from the diagonal passing through p . Denote the thus found second point of Z on the boundary of C by q .

Now assign the pair $\{p, q\}$ to Z . The above construction implies that out of the (at most) two discs of radius D whose boundary contains p and q , Z is contained in the one whose center is above the midpoint of the line segment p, q . Therefore p and q uniquely determine Z and thus the theorem is proved.

We remark that a fairly simple construction shows that the above theorem is best possible, that is, the size of an optimal set of candidate locations can be as large as constant times $|V(G)|^2$; we omit the details here.

Our algorithm for finding an optimal set of candidate locations is based on the ideas behind the above proof. Given a set of node positions $V(G)$ and a sink transmission radius D , the following algorithm finds an optimal set of candidate locations:

1. Let $R_{\text{circumcenters}} = \emptyset$.
2. For every $\{p, q\} \in V(G)^2$ of distance at most $2D$, out of the two intersection points of the perpendicular bisector of p and q and the circle of radius D around p , add the one to $R_{\text{circumcenters}}$ that is above (or not below) the midpoint of p and q .
3. Let $R_{\text{candidate}} = \emptyset$.
4. For every $r \in R_{\text{circumcenters}}$,
 - (a) if $\exists r' \in R_{\text{candidate}} : N(r) \subseteq N(r')$ then continue with the next iteration,
 - (b) otherwise, for every $r'' \in R_{\text{candidate}}$, if $N(r'') \subseteq N(r)$ then remove r'' from $R_{\text{candidate}}$
 - (c) and add r to $R_{\text{candidate}}$.

Obviously, the above algorithm runs in polynomial time. The correctness of the algorithm follows readily from the proof of Theorem 5.

In practice, the maximal density of the nodes (i.e., the maximal number of nodes in a given area) is usually limited. It is easy to see that the number of candidate locations is $O(|V(G)|)$ in this case. For every node, the number of nodes nearer than $2 \cdot D$ is limited; therefore, the number of maximal single sink coverable sets containing the node is less than a certain constant. Since this holds for each $|V(G)|$ node, the above claim is obviously true. Our experimental results in Subsection 8.2 show that, in practice, the number of candidate locations is indeed linear in the number of nodes.

7.2. Our proposed placement algorithm

Based on our proposed search space reduction technique, in this subsection, we introduce an algorithm which solves the problem of sink placement with required persistence using an existing algorithm for solving the problem of sink selection with required persistence.

Let us assume that each candidate location is used only once, i.e., no two sink nodes are to be placed at the same location. This is a realistic assumption

as more than one sink node is required at a single location only if the sink nodes themselves are too vulnerable. This would indicate that the required persistence goal is not met because the devices are not robust enough, not because the network topology is vulnerable.

Given an algorithm \mathcal{A} for sink selection, the following algorithm solves the sink placement problem:

1. Find an optimal set of candidate locations $R_{\text{candidate}}$.
2. Let G' be a graph defined as the following:
 - $V(G') := V(G) \cup R_{\text{candidate}}$
 - $E(G') := E(G) \cup \{(v, r) : v \in V(G) \wedge r \in R \wedge \text{distance}(v, r) \leq D\}$
 - $\forall_{(v, r) \in V(G) \times R} s(v, r) := 1.$
 - $\forall_{v \in V(G)} c(v) := \infty$
 $\forall_{v \in R_{\text{candidate}}} c(v) := 1$ and $d(v) = 0$
3. Find an optimal sink selection R_{opt} in G' with required persistence π_0 using \mathcal{A} .
4. Output R_{opt} as the optimal set of points for sink placement.

First, we prove that the set of feasible selections in G' is equal to the set of feasible placements restricted to $R_{\text{candidate}}$ in G .

Theorem 6. *Given an $R \subseteq R_{\text{candidate}}$, $\pi(G', R) \geq \pi_0$ if and only if $\pi_p(G, R) \geq \pi_0$.*

PROOF. The graph G' constructed in the above algorithm and the graph constructed in Definition 7 are identical except for the additional nodes $\bar{R} = R_{\text{candidate}} \setminus R$ in G' . By showing that the addition of these nodes does not affect the persistence of a graph, we can prove that the persistence of G' with selection R is equal to that of G with placement R . Consider an optimal attack A in G' . First, A does not contain any edge connected to a node in \bar{R} since these edges are all directed towards nodes in \bar{R} and, therefore, no path leading to a sink can contain any of these edges. Second, the nodes in \bar{R} do not affect the overall weight of nodes separated by any attack as their weights are all set to zero. Therefore, the set of optimal attacks and the ratios of overall costs to overall separated weights for these attacks are the same for the two graphs.

Since $\forall_{v \in V(G)} c(v) = \infty$, any selection in G' including a $v \notin R_{\text{candidate}}$ has infinite cost. Therefore, if there is a feasible placement for G then \mathcal{A} always selects an $R \subseteq R_{\text{candidate}}$.

As \mathcal{A} selects a minimum set of nodes, R_{opt} is an optimal solution to the problem of placement with required persistence π_0 in G constrained such that nodes can only be placed at $R_{\text{candidate}}$.

Corollary 1. *The above algorithm finds an optimal solution to the problem of sink placement with required persistence.*

The claim of this corollary readily follows from the above and Theorem 4.

The choice of the algorithm \mathcal{A} for sink selection is completely arbitrary. If the goal is to find an approximate solution only, then even polynomial time algorithms can be used. In this case, as the number of candidate locations and the time needed to enumerate them are also polynomial, the total running time of the algorithm is polynomial as well.

7.3. Other applications of our search space reduction technique

Our technique can be also applied to problems other than placement with required persistence. In fact, it can be used for any problem where sinks with fixed radii have to be placed so that a measure based only on the topology of the network is maximized.

In the following, we list a few examples where our technique could be used as an improvement:

- In [9], a similar reduction technique is proposed, which determines the set of candidate locations by sampling all possible intersection regions of the sensor nodes' transmission ranges. As the performance measure to be optimized depends only on the topology of the network, it does not matter which location is selected from a given region and, therefore, the technique guarantees an optimal solution. However, it also enumerates points from "inferior regions", which cover only strict subsets of the nodes covered by points in some neighboring regions. As our proposed technique does not enumerate such locations, it produces a smaller candidate set.
- In [11] and [16], the sensor nodes are first clustered using a genetic algorithm and then a sink node is placed for each cluster. For a given cluster, the area around its centroid is divided into a two dimensional grid. For each grid cell, the number of sensors whose transmission range covers the sink is determined and the cell with the maximum count value is selected. Clearly, there is no guarantee that the grid contains an optimal location. Therefore, searching instead the set of candidate locations determined using our technique would be an improvement.
- In [12] and [21], the set of candidate locations is assumed to be given. As the used performance measures depend only on topology, our proposed technique could be used to determine the set of candidate locations.

Unfortunately, if the distances between the nodes and the sinks covering them are also taken into consideration, then it is not guaranteed that the subsets of the candidate locations contain an optimal solution. Therefore, our technique can not be used for problems based on minimizing link lengths.

8. Experimental results

In this section, we present some of our experimental results. In our first experiment, we have evaluated the practical performance of our proposed heuristic

algorithms for sink selection. In our second experiment, we have measured the average number of candidate locations determined by our search space reduction technique in order to demonstrate that it is sufficiently low for our technique to be applicable in practice. In our third experiment, we have compared our proposed search space reduction technique to others used in the literature, based on the average cost of the best possible placement for of each technique.

8.1. Comparison of the proposed sink selection algorithms

We have studied two performance measures: (1) the ratios between the total selection costs of the sinks in the case of the heuristic algorithms and in the case of the optimal solution and (2) the running times of the heuristic algorithms and an integer programming solver.

In order to obtain reliable values, a large number of networks were generated in a probabilistic manner. The most prevalent model of a wireless sensor network is a unit disc graph, which models a wireless network where each node has the same transmission radius, and two nodes are considered to be neighbors if they are within each other's transmission range. In our simulations, we generated graphs of this type in a probabilistic manner. More precisely, a given number of nodes were placed uniformly at random on a disc of unit radius, and the transmission radius of the nodes was calculated from a given expected average node degree using the approximations given in [32]. Disconnected graphs were connected using minimum distance extra edges.

Edge attack costs, node values and node selection costs were drawn from uniform distributions on $[0.5, 1.5]$, while the required persistence and the expected average node degree were set to 1 and 4, respectively. The cost ratios were computed for each randomly generated graph, and the arithmetic means of the ratios were taken as approximate expected values. The experiments have been run for various numbers of nodes, ranging from 16 to 32.

Figure 4 shows the ratio of the selection cost of heuristic algorithms to the selection cost of the optimal solution as a function of the node count. Different curves belong to different heuristic algorithms, namely, to our greedy and to our genetic algorithm. As the optimal solution minimizes the selection cost, the cost selection ratio shown in the figure cannot be smaller than 1, and the closer it is to 1, the better the performance of the heuristic solution is.

In the case of the greedy algorithm, the excess requirement in selection cost fluctuates around 20% and the performance seems to be quite stable with respect to the number of nodes. The performance of the genetic algorithm is almost optimal if the number of nodes is low, and still better than that of the greedy algorithm for higher node counts. In the case of even higher node counts, i.e., node counts larger than 32, the performance of the genetic algorithm is only slightly better than that of the greedy algorithm. As finding optimal solutions is very hard, we have not plotted the cost ratios in this case.

Figure 5 shows the expected running times, measured on an average desktop PC, of the greedy algorithm, the genetic algorithm and the integer programming

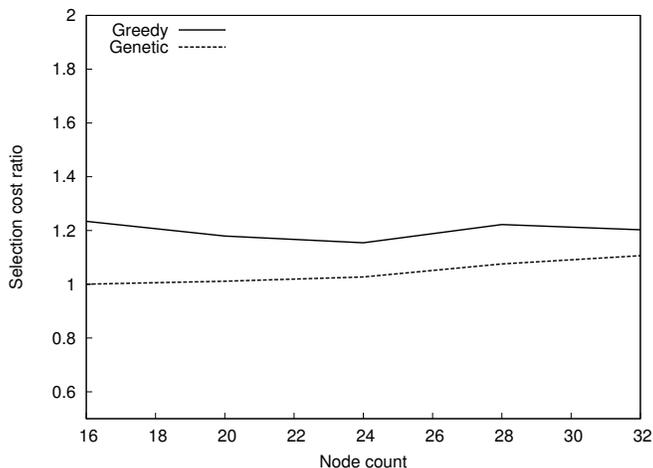


Figure 4: Ratios between the cost of the sink selection in the case of a heuristic algorithm and in the case of the optimal solution for different heuristic algorithms and node counts.

solver as a function of the node count. For solving integer programs, `lp_solve`², a free, open source mixed integer linear programming solver was used, which is based on the Branch-and-Bound method combined with the revised simplex method.

As expected, the running time of the integer programming solver is exponential and grows faster than those of the heuristic algorithms by several orders of magnitude. Of the two proposed heuristics, the genetic algorithm is faster than the greedy algorithm by an order of magnitude. For node counts higher than 32, the difference between the performance results of the heuristic algorithms is more prominent. For example, for 64 nodes, the running time of the greedy algorithm is 6834 ms, while that of the genetic algorithm is only 452 ms.

8.2. Performance of our search space reduction technique

Again, the networks on which we have conducted our measurements were generated in a probabilistic manner. Nodes were randomly placed on a disc of unit radius according to a uniform distribution. The experiment has been run for various numbers of nodes and sink radii, the former ranging from 100 to 500.

Figure 6 shows the average number of candidate locations. As the exact value of the sink radius is not very informative, the expected average number of nodes on a disc of the given radius (i.e., the average number of nodes covered by a randomly placed sink) is displayed instead. As expected, the number of candidate locations grows linearly with the number of nodes and the rate of the

²lpsolve.sourceforge.net

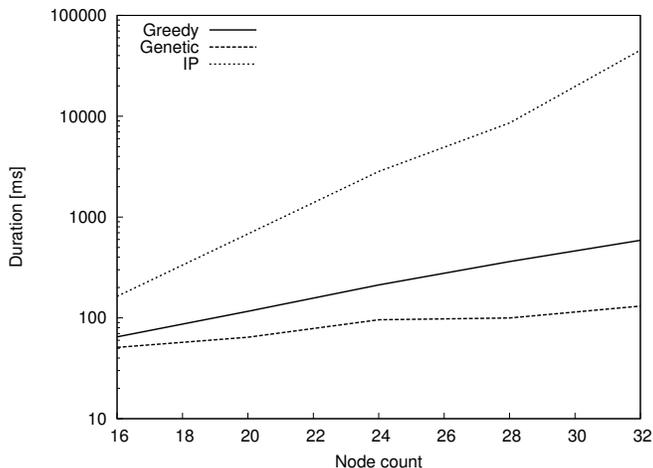


Figure 5: Expected running time of the greedy algorithm, the genetic algorithm and the integer programming based optimal solution for different node counts. Please, note the logarithmic scale on the y axis.

growth is determined by the radii of the sinks. Furthermore, the two numbers are roughly of the same order of magnitude. Therefore, the sink placement problem is practically only as hard as the sink selection problem. Surprisingly, the relationship between the sink radii and the number of candidate locations also seems to be linear.

For every combination of parameters presented above, the running time of our algorithm for enumerating candidate locations was in the order of minutes on an average desktop PC. Similarly to the number of candidate locations, the running time of the enumeration also grows linearly with the size of the network. Therefore, we can say that the running time needed to enumerate candidate locations is never going to be a bottleneck, even for large networks. For this reason, we omit the exact numerical results on these running times.

8.3. Comparison of different search space reduction techniques

The networks on which we have conducted our measurements were generated in a probabilistic manner similar to the one in Subsection 8.1. Edge attack costs and node values were drawn from uniform distributions on $[0.05, 0.15]$ and $[0.5, 1.5]$, respectively. The required persistence was set to 0.1. The transmission radii of regular nodes and sink nodes were chosen such that the expected average numbers of nodes on discs of the given radii were 4 and 8, respectively. The experiment has been run for various numbers of nodes, ranging from 16 to 32.

We have compared four different search space reduction techniques:

- Uniform grid: The area where sinks are to be placed is divided into a regular two dimensional grid and the position of each gridpoint is added

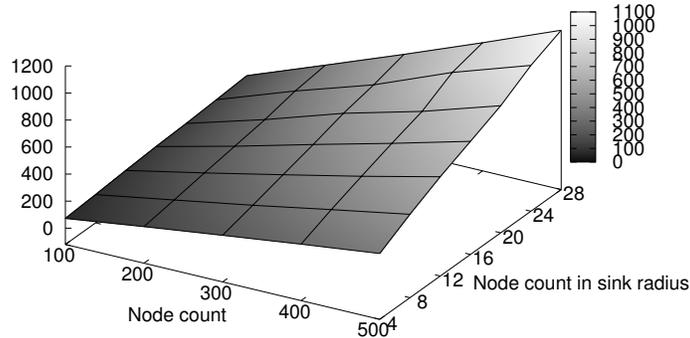


Figure 6: Average number of candidate locations for different node counts and sink radii.

to the set of candidate locations. To achieve a fair comparison, the granularity of the grid was chosen such that the number of candidate locations was roughly equal to that of the other techniques.

- “Selection”: The set of candidate locations consists of the positions of regular nodes. This technique corresponds to the case when the sink placement problem degenerates to the sink selection problem, hence the name.
- Random: Candidate locations are chosen uniformly at random from the area where sinks are to be placed. To achieve a fair comparison, the number of candidate locations was roughly equal to that of the other techniques.
- Optimal: The set of candidate locations is determined using our proposed technique.

For each network, a set of candidate locations was determined using each search space reduction technique. Then, a minimum cost sink placement with the required persistence was found using an optimal algorithm for sink selection, as it has been described in Subsection 7.2. Finally, for each search space reduction technique, the average cost of the optimal placements was calculated, where the average was taken over all generated networks.

Figure 7 shows the average cost of the best possible placement as a function of the node count. The different curves belong to the different search space reduction techniques. As expected, our proposed technique clearly outperforms the other three in terms of the average costs of placements.

9. Conclusions

In this paper, we have addressed the problem of deploying sink nodes in a wireless sensor network such that the resulting network topology be robust

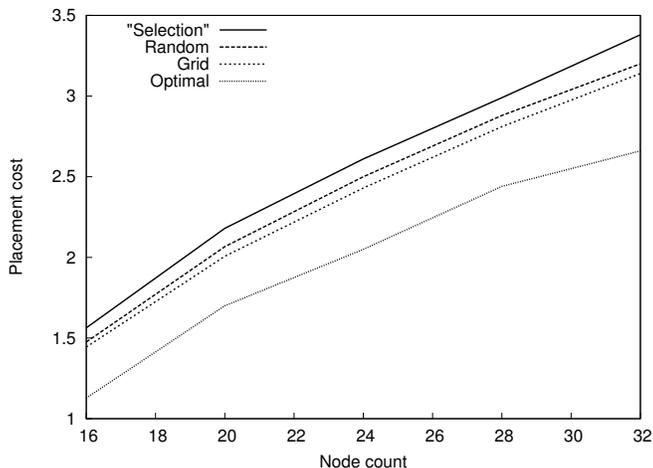


Figure 7: The average costs of best possible placements for different search space reduction techniques and node counts.

against denial-of-service type attacks such as node destruction, battery exhaustion, and jamming. Instead of the usual approach of using connectivity for measuring the robustness of network topologies, we have introduced the notion of *persistence*. We explained and gave some illustrative examples why persistence is a better robustness metric for wireless sensor networks than connectivity. Our arguments apply to a wider range of networks, including most types of access networks.

Using persistence as the robustness metric, we have formalized and studied two variants of the sink deployment problem. In the first variant, we restricted the set of possible sink locations to the set of sensor node locations; we called this variant the sink selection problem. In the second variant, we removed this restriction, and allowed sinks to be placed anywhere in the deployment area; we called this the sink placement problem. In both variants, we aimed at achieving a given level of persistence while minimizing the deployment cost, which is equally hard as maximizing persistence under a given upper bound on the deployment budget. We have proved that both sink selection with required persistence and sink placement with required persistence are NP-hard.

We have proposed greedy and genetic heuristic algorithms to solve the sink selection problem efficiently. We have shown how the infinite search space of possible placements can be reduced to a set of candidate locations, which is of polynomial size, such that the resulting set always contains an optimal solution. The proposed search space reduction technique may be of independent interest. We have also shown how any sink selection algorithm, including our proposed heuristic algorithms, can be used to find a solution in the reduced search space.

Finally, we have provided experimental results on the performance of our heuristic algorithms for sink selection and our proposed search space reduction

technique. Our results show that the proposed technique could be used to efficiently solve other problems or to enhance the performance of previously proposed algorithms.

Acknowledgements

The work presented in this paper has been carried out in the context of the WSan4CIP Project³, which receives funding from the European Community through the Seventh Framework Programme (grant agreement no. 225186). The work is also related to the internal project of the authors' hosting institution on "Talent care and cultivation in the scientific workshops of BME", which is supported by the grant TÁMOP - 4.2.2.B-10/1-2010-0009. Levente Buttyán has also been supported by the Hungarian Academy of Sciences through the Bolyai János Research Fellowship. Dávid Szeszlér is supported by grant Nr. OTKA 67651 of the Hungarian National Science Fund. Aron Laszka is supported by HSNLab, Budapest University of Technology and Economics⁴. The authors are thankful to Géza Tóth for the insights he provided on the size of the optimal set of candidate locations and his proof of Theorem 5.

References

- [1] M. Welsh, Sensor networks for the sciences, *Communications of the ACM* 53 (11).
- [2] C. Partridge, Forty data communications research questions, Tech. Rep. Report-8528, BBN (2011).
- [3] A. Laszka, L. Buttyan, D. Szeszler, Optimal selection of sink nodes in wireless sensor networks in adversarial environments, in: *Proc. of IEEE WoWMoM 2011*.
- [4] K. Akkaya, M. Younis, W. Youssef, Positioning of base stations in wireless sensor networks, *Communications Magazine, IEEE* 45 (4) (2007) 96–102.
- [5] M. Younis, K. Akkaya, Strategies and techniques for node placement in wireless sensor networks: A survey, *Ad Hoc Networks* 6 (4) (2008) 621–655.
- [6] E. M. Arkin, V. Polishchuk, A. Efrat, S. Ramasubramanian, J. Taheri, J. S. B. Mitchell, S. Sankararaman, Data transmission and base-station placement for optimizing network lifetime, in: *Proc. of ACM DIALM-POMC 2010*, pp. 23–32.

³www.wsan4cip.eu

⁴www.hsnlab.hu

- [7] J. Pan, L. Cai, Y. T. Hou, Y. Shi, S. X. Shen, Optimal base-station locations in two-tiered wireless sensor networks, *IEEE Transactions on Mobile Computing* 4 (2005) 458–473.
- [8] S. N. Muthaiah, C. Rosenberg, Single gateway placement in wireless mesh networks, in: *Proc. of IEEE ISCN 2008*, 2008.
- [9] W. Y. Poe, J. B. Schmitt, Minimizing the maximum delay in wireless sensor networks by intelligent sink placement, *Tech. Rep. 362/07*, University of Kaiserslautern, Germany (July 2007).
- [10] J. L. Wong, R. Jafari, M. Potkonjak, Gateway placement for latency and energy efficient data aggregation, in: *Proc. of IEEE LCN 2004*, 2004, pp. 490–497.
- [11] W. Youssef, M. Younis, Intelligent gateways placement for reduced data latency in wireless sensor networks, in: *Proc. of IEEE ICC 2007*, 2007, pp. 3805–3810.
- [12] A. Capone, M. Cesana, D. D. Donno, I. Filippini, Deploying multiple interconnected gateways in heterogeneous wireless sensor networks: An optimization approach, *Comput. Commun.* 33 (2010) 1151–1161.
- [13] Y. Shi, Y. T. Hou, A. Efrat, Algorithm design for base station placement problems in sensor networks, in: *Proc. of QShine 2006*, 2006.
- [14] E. I. Oyman, C. Ersoy, Multiple sink network design problem in large scale wireless sensor networks, in: *Proc. of IEEE ICC 2004*, Vol. 6, 2004, pp. 3663–3667.
- [15] A. Bogdanov, E. Maneva, S. Riesenfeld, Power-aware base station positioning for sensor networks, in: *Proc. of IEEE INFOCOM 2004*, Vol. 4, 2004.
- [16] W. Youssef, M. Younis, Optimized asset planning for minimizing latency in wireless sensor networks, *Wirel. Netw.* 16 (2010) 65–78.
- [17] M. M. Czajko, J. M. Wojciechowski, Bi-criteria gateway placement problem in wireless sensor networks, *International Journal of Electronics and Telecommunications* 56 (3) (2010) 215–222.
- [18] B. Aoun, R. Boutaba, Clustering in WSN with latency and energy consumption constraints, *Journ. of Netw. and Syst. Management* 14 (3) (2006) 415–439.
- [19] S. Bandyopadhyay, E. J. Coyle, An energy efficient hierarchical clustering algorithm for wireless sensor networks, in: *Proc. of INFOCOM 2003*, Vol. 3, 2003, pp. 1713–1723.
- [20] A. A. Abbasi, M. Younis, A survey on clustering algorithms for wireless sensor networks, *Comput. Commun.* 30 (2007) 2826–2841.

- [21] S. R. Gandham, M. Dawande, R. Prakash, S. Venkatesan, Energy efficient schemes for wireless sensor networks with multiple mobile base stations, in: Proc. of IEEE GLOBECOM 2003, Vol. 1, 2003, pp. 377–381.
- [22] G. Gupta, M. Younis, Fault-tolerant clustering of wireless sensor networks, in: Proc. of IEEE WCNC 2003, Vol. 3, 2003, pp. 1579–1584.
- [23] J. Li, L. L. H. Andrew, C. H. Foh, M. Zukerman, H. H. Chen, Connectivity, coverage and placement in wireless sensor networks, *Sensors* 9 (10) (2009) 7664–7693.
- [24] S. Misra, S. Hong, G. Xue, J. Tang, Constrained relay node placement in wireless sensor networks to meet connectivity and survivability requirements, in: Proc. of IEEE INFOCOM 2008, IEEE, 2008, pp. 281–285.
- [25] A. Kashyap, S. Khuller, M. Shayman, Relay placement for higher order connectivity in wireless sensor networks, in: Proc. of IEEE INFOCOM 2006, 2006.
- [26] W. Zhang, G. Xue, S. Misra, Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms, in: Proc. of IEEE INFOCOM 2007, 2007, pp. 1649–1657.
- [27] X. Han, X. Cao, E. L. Lloyd, C.-C. Shen, Fault-tolerant relay node placement in heterogeneous wireless sensor networks, in: Proc. of IEEE INFOCOM 2007, 2007, pp. 1667–1675.
- [28] D. Bauer, H. J. Broersma, E. Schmeichel, Toughness in graphs - A survey, *Graphs and Combinatorics* 22 (1) (2006) 1–35.
- [29] W. H. Cunningham, Optimal attack and reinforcement of a network, *Journal of the ACM* 32 (3) (1985) 549–561.
- [30] J. Chang, L. Tassiulas, Maximum lifetime routing in wireless sensor networks, *IEEE/ACM Transactions on Networking* 12 (4) (2004) 609–619.
- [31] R. Raz, S. Safra, A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP, in: Proc. of ACM STOC 1997, pp. 475–484.
- [32] C. Bettstetter, On the connectivity of ad hoc networks, *Computer Journal* 47 (4) (2004) 432–447.