Optimal Selection of Sink Nodes in Wireless Sensor Networks in Adversarial Environments

Áron Lászka and Levente Buttyán
Department of Telecommunications
Budapest University of Technology and Economics
www.crysys.hu

Dávid Szeszlér
Department of Computer Science and Information Theory
Budapest University of Technology and Economics
szeszler@cs.bme.hu

***Abstract*—In this paper, we address the problem of assigning the sink role to a subset of nodes in a wireless sensor network with a given topology such that the resulting network configuration is robust against denial-of-service type attacks such as node destruction, battery exhaustion and jamming. In order to measure robustness, we introduce new metrics based on a notion defined in [1]. We argue that our metrics are more appropriate to measure the robustness of network configurations than the widely known connectivity based metrics. We formalize the problem of selecting the sink nodes as an optimization problem aiming at minimizing the deployment budget while achieving a certain level of robustness. We propose an efficient greedy heuristic algorithm that approximates the optimal solution reasonably well.**

***Keywords*-wireless sensor networks; denial-of-service attacks; topology robustness measures**

I. Introduction

Wireless sensor networks are often envisioned to operate in hostile environments, where an adversary can try to mount different types of attacks against the network. Given that wireless sensor networks are usually assumed to consist of resource constrained and physically unprotected devices that use wireless channels for communications, one of the major concerns is to protect the network against denial-of-service type attacks based on physical destruction of devices, exhaustion of their batteries, and jamming of the wireless channels. Such attacks may be addressed at different levels in the system architecture; however, in our work, we are interested in the question of what one can do about them by carefully designing the *deployment configuration* of the network. By deployment configuration, we mean the ensemble of the topology of the network (determined by the placement of the nodes, the power and channel settings of their radios, the propagation characteristics of the environment, etc.) and the assignment of the sink role to a subset of the nodes.

This question makes sense because in most of the applications the network is static and, contrary to what is suggested in many research papers on sensor networks, the nodes are not deployed randomly, but in a systematic and controlled manner. Thus, the deployment configuration of the network can be designed before the actual deployment, taking into account the constraints imposed by the application, the deployment environment, and the available budget, and aiming at maximizing the resistance of the network against node disabling and jamming attacks. In addition, ensuring the robustness of the deployment configuration is crucial in the sense that one cannot hope for providing robustness guarantees at higher layers if certain level of robustness is not guaranteed at the lowest layer by the deployment configuration.

While our ultimate goal is to understand how to design robust deployment configurations, as a first step towards this general objective, we restrict ourselves, in this paper, to study the simpler problem of how to assign the sink role to a subset of the nodes in a given network topology such that the robustness of the resulting deployment configuration exceeds a given threshold and the deployment budget is minimized. The general problem, where, in addition, one is also required to determine the network topology, is left for future work.

An important part of the problem we are tackling is to identify an appropriate metric that can be used to measure the robustness of deployment configurations. Instead of relying on the commonly used connectivity based metrics to measure robustness, we use *persistence*, a notion obtained by an extension of *directed graph strength* defined in [1]. We will explain in Subsection II-A why we believe that persistence is a better robustness metric for our purposes than connectivity. Based on this metric, we translate deployment configuration problems into optimization problems that aim at maximizing the persistence of the graph that represents the deployment configuration while keeping the deployment cost under a certain bound, or minimizing the deployment cost while achieving a certain level of persistence.

The organization of the paper is the following: In Section II, we give an overview of previously proposed topology robustness metrics, we present a motivating example to show why we believe that vertex and edge connectivity as robustness metrics are not practical and we introduce (various versions of) persistence for the purpose of measuring the robustness of deployment configurations of wireless sensor networks. In Section III, we formalize the sink selection problem for a given network topology as a family of optimization problems, and we introduce one specific variant of the problem: sink selection with required persistence. We present an integer program formulation of the sink selection problem and we propose a greedy algorithm as a heuristics to efficiently get solutions that are reasonably close to the optimal selection of sink nodes. In Section IV, we describe the simulations that we used to analyze the performance of our greedy algorithm in terms of sink selection cost incurred and its running time. More specifically, we compare the results of our greedy algorithm to that of the optimal solutions obtained by integer programming. Finally, in Section V, we conclude the paper and sketch our plans for future work.

II. Measuring the robustness of deployment configurations

A. Related work

In the literature on wireless sensor networks, vertex- and edge-connectivity are the most frequently used metrics for measuring the robustness of network topologies [2], [3], [4], [5], [6], [7]. 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.

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).

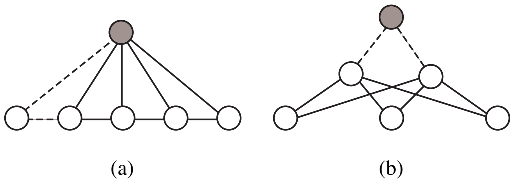

Figure 1. Illustration of why connectivity based metrics do not characterize topology robustness of wireless sensor networks well enough. The edge-connectivity of both graphs is 2, and thus, they are equally robust in terms of edge-connectivity. However, when the two dashed edges are removed, only a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

Another known topology robustness metric, with several theoretical results, is graph *toughness* [8]. 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 [1]. 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. However, unlike toughness, it can be computed efficiently.

B. 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 [1]. 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: Given a directed graph G, sink nodes $R \subseteq V(G)$, edge weights $s : E(G) \to \mathbb{R}^+$ and node weights $d : V(G) \to \mathbb{R}^+$, 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: given a directed graph G, sink nodes $R \subseteq V(G)$, edge and node destruction costs $s : (V(G) \cup E(G)) \to \mathbb{R}^+$ and node weights $d : V(G) \to \mathbb{R}^+$, 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.

We mention that handling undirected edges is also straightforward: each undirected edge $e = \{u, v\}$ has to be replaced by the directed arcs $e_1 = (u, v)$ and $e_2 = (v, u)$ and $s(e_1) = s(e_2) = s(e)$ has to be set.

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

C. Computing persistence

It is shown in [1] that computation of persistence can be performed using a maximum flow algorithm[1]. In particular, assume that besides the input data used above (that is, G, $R \subseteq V(G)$, $s : E(G) \to \mathbb{R}^+$ and $d : V(G) \to \mathbb{R}^+$) 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

$$\delta_s(X) + \pi_0 \cdot d(\overline{X}) \geq \pi_0 \cdot d(V(G)) \qquad (*)$$

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

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 $\delta_s(X) + \pi_0 \cdot d(\overline{X})$ if $X \cap R = \emptyset$ (and infinity otherwise). Comparing this with $(*)$ 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 satures 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 [1] a refinement

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

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.

III. The sink selection problem and its solution

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. We show that the optimal selection can be found by solving an integer program and we also introduce a more efficient greedy algorithm that approximates the optimal solution reasonably well.

A. 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: **Sink selection with required persistence:**

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

B. An 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 Section II-C: $\pi(G) \geq \pi_0$ 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 Section II-C 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 $f((v, t^*)) \leq bignum \cdot r(v)$ on each edge (v, t^*), where $bignum$ is a sufficiently large constant (e.g., $bignum = \pi_0 \cdot d(V(G))$, 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) $\forall v \in V(G) : f((v, t^*)) \leq bignum \cdot r(v)$
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 Section II-C); 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 next section.

C. A greedy algorithm

In this section, we propose an efficient greedy algorithm as a heuristic approach to find a sub-optimal, but reasonably good solution for 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.

IV. Experimental results

In this section, we present simulation results on the performance of our greedy algorithm described above. We study two performance measures: (1) the ratio between the total selection costs of the sinks in case of the greedy algorithm and in case of the optimal solution, and (2) the running times of the greedy algorithm and our integer programming based algorithm to compute the optimal solution.

A. Simulation settings

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 disk of unit radius, and the transmission radius of the nodes was calculated from a given expected average node degree using the approximations given in [9]. Disconnected graphs were connected using minimum distance extra edges.

In the experiments, we set all node and edge weights to one. In addition, the sink selection costs were also set to one, therefore the total sink selection cost was equal to the number of the selected sinks. This means that our first performance measure became the ratio between the numbers of the selected sinks in the two cases. This ratio was computed for each randomly generated graph, and the average of the values was taken as the approximate expected value of the given performance measure.

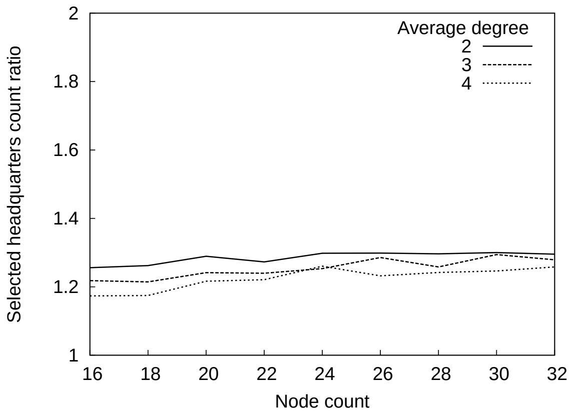

Figure 2. Sink selection with required persistence: Ratio between the numbers of necessary sink nodes in case of the greedy algorithm and in case of the optimal solution for different node counts and average degrees.

B. Simulation results

Figure 2 shows the number of sinks needed in case of the greedy algorithm compared to the number of sinks in case of the optimal solution for different node counts and expected average degrees. The comparison is done by plotting the ratio between the numbers of the necessary sinks in the two cases. Hence, the closer the ratio to 1 is, the better the performance of the greedy algorithm is. In this experiment, the required persistence was a constant 1, while the number of nodes in the network was increased from 16 to 32, and the node degree was increased from 2 to 4. The excess requirement in sink count fluctuated between 20% and 30%. We observe that the expected average degree has some impact on the ratio between the numbers of sinks required: as the average degree increases, the performance of the greedy algorithm slightly improves. This is surprising, if we consider that a higher degree means a more complex network. Nevertheless, this performance improvement is not significant, and all together, the performance of the greedy algorithm seems to be quite stable with respect to both the node count and the average degree.

Figure 3 shows the expected running times, measured on an average desktop PC, of the integer programming based optimal solution and the greedy algorithm as a function of the node count. For each node count, the running times for expected node degrees of 2, 3 and 4 were measured, and the arithmetic mean of these results was taken. For solving integer programs, lp_solve[2], a free, open source mixed integer linear programming solver was used that is

[2] lpsolve.sourceforge.net

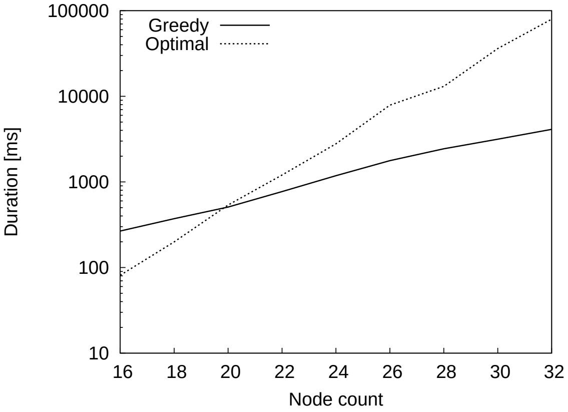

Figure 3. Sink selection with required persistence: Expected running time of the greedy algorithm and the integer programming based optimal solution for different node counts. Please, note the logarithmic scale on the y axis.

based on the Branch-and-Bound method combined with the revised simplex method. As expected, the running time of the optimal solution is exponential and grows faster than that of the greedy algorithm by an order of magnitude. For small node counts, however, the integer programming solution outperforms the greedy algorithm. This is not surprising, if we compare the complexity of computing persistence several times to that of solving only one small integer program.

V. Conclusions and future work

In this paper, we investigated the optimization problem of selecting the sink nodes, with minimal budget, in a wireless sensor network with a given topology, such that the resulting network has a certain level of robustness. We proposed to use the notion of persistence to measure robustness, we presented an integer program that yields the optimal solution, albeit inefficiently, and proposed an efficient greedy heuristic algorithm that approximates the optimal solution reasonably well.

We addressed only part of the general problem of designing robust deployment configurations in this paper. Namely, we assumed that the network topology is given, and our task was to select the sink nodes such that a given level of robustness is achieved while the incurred sink selection cost is minimized. In the future, we also intend to work on more general settings. For instance, one interesting question is how to add new nodes to an already existing network to achieve maximal increase in robustness.

Acknowledgements

The work presented in this paper has been carried out in the context of the WSAN4CIP Project[3], which receives funding from the European Community through the Seventh Framework Programme (grant agreement no. 225186). Levente Buttyán has also been supported by the Hungarian Academy of Sciences through the Bolyai János Research Fellowship. Dávid Szeszlér is supported by grants Nr. OTKA 67651 and Nr. OTKA 100238 of the Hungarian National Science Fund.

References

[1] W. H. Cunningham, "Optimal attack and reinforcement of a network," *J. ACM*, vol. 32, no. 3, pp. 549–561, 1985.

[2] J. Li, L. Andrew, C. Foh, M. Zukerman, and H. Chen, "Connectivity, coverage and placement in wireless sensor networks," *Sensors*, vol. 9, no. 10, pp. 7664–7693, 2009.

[3] 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.

[4] S. Misra, S. Hong, G. Xue, and J. Tang, "Constrained relay node placement in wireless sensor networks to meet connectivity and survivability requirements," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE.* IEEE, 2008, pp. 281–285.

[5] A. Kashyap, S. Khuller, and M. Shayman, "Relay placement for higher order connectivity in wireless sensor networks," in *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2006.

[6] W. Zhang, G. Xue, and S. Misra, "Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms," in *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2007.

[7] X. Han, X. Cao, E. L. Lloyd, and C.-C. Shen, "Fault-tolerant relay node placement in heterogeneous wireless sensor networks," in *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2007.

[8] D. Bauer, H. J. Broersma, and E. Schmeichel, "Toughness in graphs - a survey," *Graphs and Combinatorics*, vol. 22, no. 1, pp. 1–35, April 2006.

[9] C. Bettstetter, "On the connectivity of ad hoc networks," *The Computer Journal*, vol. 47, no. 4, p. 432, 2004.

[3] www.wsan4cip.eu