A Minimum Cost Source Location Problem for Wireless Sensor Networks

Levente Buttyán

MTA–BME Information Systems Research Group
Magyar Tudósok krt. 2.
Budapest, 1117, Hungary
buttyan@crysys.hu

Aron Laszka*

Department of Networked Systems and Services
Budapest University of Technology and Economics
Postafiók 91., Budapest, 1521, Hungary
laszka@crysys.hu

Dávid Szeszlér†

Department of Computer Science and Information Theory
Budapest University of Technology and Economics
Postafiók 91., Budapest, 1521, Hungary
szeszler@cs.bme.hu

Abstract: Wireless sensor networks consist of physically unprotected devices, therefore it is essential to design the network configuration in such a way that it is the most resistant against attacks. However, it is not at all obvious how to measure the robustness of a network to best serve the purposes of wireless sensor networks. In this paper we propose a metric that seems to be appropriate and then we address the problem of assigning the sink role to a subset of vertices such that the arising network is as robust with respect to the proposed metric as possible.

Keywords: wireless sensor network, robustness metric, persistence

1 Introduction

Measuring the reliability or robustness of a graph (or network) is one of the fundamental problems in graph theory that is also motivated by numerous applications. Accordingly, many metrics have been proposed in the literature with the most widespread ones being the various connectivity based metrics.

A *Wireless Sensor Network* (or *WSN*) consists of a number of sensors that measure a certain physical parameter (like temperature, sound, pressure, etc.). The sensors use wireless channels to communicate with each other and pass on the collected data. WSNs have many applications like area monitoring, environmental monitoring, forest fire detection, water quality monitoring, machine health monitoring, etc.

A WSN is modeled by a graph G with vertices corresponding to sensors and edges corresponding to pairs of sensors that can directly communicate with each other (which usually depends mainly on their physical distance). Furthermore, a subset $R \subseteq V(G)$ of the vertices, called *sink nodes* is also given: these vertices correspond to the *base stations* of the WSN that collect the data measured by all the sensors and

*Aron Laszka has been supported by HSN Lab, Budapest University of Technology and Economics, http://www.hsnlab.hu

†The work reported in the paper has been developed in the framework of the project "Talent care and cultivation in the scientific workshops of BME" project. This project is supported by the grant TÁMOP - 4.2.2.C-11/1/KONV-2012-0013

serve as a gateway between the WSN and the end user. The underlying graph G is typically undirected but in some applications directed graphs may also make sense: it is possible that a sensor u is capable of sending measurement data to a sensor v but not vice versa.

Since WSNs consist of resource constrained and physically unprotected devices, it is a major concern to protect the network against attacks (like physical destruction of the devices, exhaustion of their batteries or jamming of the wireless channels). One of the main tools of protection is to design the deployment configuration of the devices in a way that makes the arising WSN (as given by the underlying graph G and the set of sink nodes $R \subseteq V(G)$) as robust as possible.

This gives rise to the problem of suitably measuring the robustness (or reliability) of a WSN. The typical, connectivity based approach would be to measure the minimum number of edges or vertices that need to be removed to achieve that no element of R is reachable from at least one vertex of G (along a path or directed path). Of course, such a metric would be efficiently computable, however, we still claim that it would not be really suitable for the purpose of WSNs. To support this statement, consider the following examples:

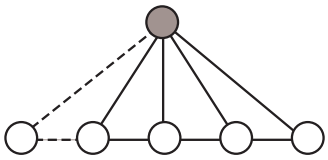

Figure 1a

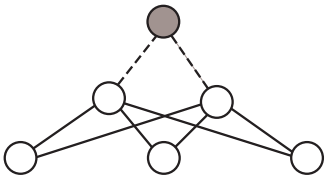

Figure 1b

The above two graphs represent WSNs with the shaded vertex being the single sink node in both cases. Obviously, if the number of sink nodes is one then the (edge version of the) above mentioned connectivity metric is simply the edge-connectivity of the graph, which is 2 in both cases (as shown by the dashed edges). However, in case of Figure 1a just a single vertex is disconnected from the sink node while it is possible to disconnect all non-sink nodes by deleting 2 edges in case of Figure 1b. In other words, removing 2 links could result in the total malfunctioning of the WSN in case of Figure 1b while it would only cause a minor problem in case of Figure 1a. This suggests that intuitively the WSN of Figure 1a is more robust than that of Figure 1b and a suitable robustness metric should capture this difference.

Generally speaking, the basic problem with connectivity metrics is that they are only concerned with whether a graph remains connected or not under an attack of a given maximum strength, but they do not shed light on how "scattered" the graph becomes when it is disconnected. In addition, connectivity metrics are only concerned with the effect of the smallest effective attack; however, it is also important to characterize how the network fails as the strength (or budget) of the attacker increases.

A more appropriate approach to measuring the robustness of a WSN could be the notion of *graph strength*. By definition, the strength $\sigma(G)$ of G is the minimum of the ratios $\frac{|A|}{\lambda(A)}$, where A is a subset of the edges of G that disconnects G and $\lambda(A)$ is the increase in the number of components after deleting A. In other words, $\sigma(G)$ measures the minimum average effort (that is, number of edges) needed for creating a new component. This notion was defined by D. Gusfield [3] and proved to be efficiently computable by W. Cunningham [2], even for the more general, weighted case. For example, the value of $\sigma(G)$ is 1.8 and 1.6 for Figure 1a and Figure 1b, respectively (in both cases the minimum is attained by deleting all edges).

Although it would be easy to incorporate the role of sink nodes in the above notion (and the same algorithm could be used for computing it after contracting all sink nodes into a single node), we still claim that graph strength is not the ideal measurement for the robustness of WSNs. The reason is that attacks against WSNs also aim the devices themselves and not only the links between them, while the notion of graph strength only allows for the removal of edges. Moreover, computing the analogous notion that concerns the removal of vertices, *graph toughness* is known to be NP-hard [1]. Finally, it is just common

sense that the efficiency of an attack against a WSN is not to be measured by the number of *components* isolated from the sink nodes, the number of *vertices* (that is, devices) would be more natural.

Fortunately, a robustness metric based on exactly this idea was defined and proved to be efficiently computable by W. Cunningham in [2]. There the notion is not given any specific name (it is referred to as the "Digraph Model"), but since we believe that it is an ideal metric for measuring the robustness of WSNs and it is the key concept of this paper, we will call it *persistence* and denote it by $\pi(G, R)$ (or simply $\pi(G)$ if R is assumed to be known).

Unfortunately, the corresponding minimum cost sink selection problem based on the notion of $\pi(G)$ was proved to be NP-hard in [5]. However, we presented a greedy heuristic in [4] and a genetic heuristic in [5] that, based on experimental data, tend to approximate the optimum surprisingly well. In this paper, we present a branch-and-bound type algorithm that computes the exact optimum; the method surpasses by far the efficiency of an integer programming based computation, but obviously it is still exponential time.

In the next section, we formally define the notion of persistence and, for the sake of completeness, describe the algorithm to compute it. We also prove the correctness of the algorithm (in a somewhat simpler way than [2]). In Section 3 we define the sink selection problem and describe the above mentioned branch-and-bound algorithm. Finally, in Section 4 we briefly mention some related results.

2 Persistence: a robustness metric for Wireless Sensor Networks

2.1 The definition of persistence

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 WSNs, 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 is the attack of removing A. Therefore it makes sense to define a robustness metric as the minimum of these ratios.

Definition 1 *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 $\pi(G, R)$ *(or $\pi(G)$) is defined as*

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

(If $R = V(G)$ then let $\pi(G, R) = \infty$ by definition.)

For example, consider the two graphs of Figure 1. Assume, like above, that the shaded vertex is the (single) sink node in both cases, edges are directed both ways (so each edge of the figures represents a pair of directed edges) and all edge weights $s(e)$ and node weights $d(v)$ are 1. 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 Figure 1a and $\pi(G) = \frac{2}{5}$ Figure 1b. This coincides with our previous observation that the graph of Figure 1a is intuitively more robust than that of Figure 1b and thus supports (at least to some extent) our statement that persistence is a more suitable robustness metric for WSNs than connectivity.

2.2 Generalizations of persistence

In the above definition, the graph underlying the WSN was supposed to be directed. The reason for that is that handling undirected edges in this model is straightforward by the usual trick: each undirected edge $e = \{u, v\}$ can be replaced by the pair of directed arcs $e_1 = (u, v)$ and $e_2 = (v, u)$ and $s(e_1) = s(e_2) = s(e)$ can be set.

As mentioned in the introduction, attacks against WSNs 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 generalized to allow for the destruction of both edges and vertices. Obviously, the only modification in the above definition would be that the cost function s would be defined on $V(G) \cup E(G)$ and A would be allowed to be a subset of $V(G) \cup E(G)$. (The notion of $\lambda(A)$ would also be suitably modified such that vertices isolated from the remainder of R as well as vertices belonging to A also contribute to $\lambda(A)$.)

However, computing this extended version of persistence is again easily reduced to the original version of Definition 1 by the well-known trick of vertex splitting: 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 $s((u_2, v_1)) = s((u, v))$.

With respect to the above, we only consider the notion defined in Definition 1 in the sequel.

2.3 Computing persistence

It is shown in [2] that persistence can be computed using a maximum flow algorithm. For the sake of completeness, we briefly describe the method here and give a (somewhat simplified) proof of its correctness.

First 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 α is also given: α represents a required persistence value and the task is to decide if $\pi(G) \geq \alpha$ is true.

Consider the following maximum 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 $\alpha \cdot d(v)$; 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 $G(\alpha)$.

Claim 2 *[2] $\pi(G) \geq \alpha$ is true if and only if there exists a flow in $G(\alpha)$ that saturates all arcs leaving s^*.*

Proof: 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 $\pi(G) \geq \alpha$ is equivalent to saying that $\delta_s(X) - \alpha \cdot d(X) \geq 0$ holds for all $X \subseteq V(G) \setminus R$. Adding $\alpha \cdot d(V(G))$ to both sides we get that $\pi(G) \geq \alpha$ is equivalent to

$$\delta_s(X) + \alpha \cdot d(\overline{X}) \geq \alpha \cdot d(V(G)) \tag{1}$$

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

By the Ford-Fulkerson Theorem, the maximum flow from s^* to t^* in $G(\alpha)$ is equal to the minimum cut capacity. Identify each cut of $G(\alpha)$ with the set of vertices in $V(G)$ on the same side as s^* (we use this convention later on too). Then the capacity of the cut X is $\delta_s(X) + \alpha \cdot d(\overline{X})$ if $X \cap R = \emptyset$ (and infinity otherwise). Comparing this with (1) above, we get that $\pi(G) \geq \alpha$ is equivalent to the existence of a flow of value $\alpha \cdot d(V(G))$ from s^* to t^*; in other words, a flow that saturates all arcs leaving s^*. □

Consequently, the question of $\pi(G) \geq \alpha$ can be answered by a single maximum flow computation. From this, the actual value of $\pi(G)$ (that is, the maximum α 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). However, the following, more efficient algorithm was given in [2] that also works for arbitrary input data.

Algorithm 3

Step 0. *Let* $X_0 = V(G) \setminus R$.

Step 1. *Assuming that* $X_i \subseteq V(G) \setminus R$ *is already given for some* $i \geq 0$, *let* $\pi_i = \dfrac{\delta_s(X_i)}{d(X_i)}$.

Step 2. *Compute a maximum flow* f_i *from* s^* *to* t^* *and a minimum cut* Z_i *in* $G(\pi_i)$.

Step 3. *If* f_i *saturates all arcs leaving* s^*, *then STOP and output* $\pi(G) = \pi_i$. *If not then let* $X_{i+1} = Z_i \cap X_i$. *Continue at* Step 1 *(with* X_{i+1} *instead of* X_i*).*

Proposition 4 *[2] Algorithm 3 terminates after at most* $|V(G) \setminus R|$ *cycles and returns the value of* $\pi(G)$.

Proof: Obviously, if the algorithm terminates with X_i, π_i and f_i then $\pi(G) = \pi_i$ is true: indeed, $\pi(G) \leq \pi_i$ follows by definition and $\pi(G) \geq \pi_i$ follows by Claim 2.

To show that the algorithm terminates after at most $|V(G) \setminus R|$ cycles, we prove by induction on i that for every $i \geq 0$, X_{i+1} is a cut of minimum capacity in $G(\pi_i)$. Obviously, this is true for $i = 0$ since $X_1 = Z_0 \cap X_0 = Z_0 \cap (V(G) \setminus R) = Z_0$ and Z_0 is a minimum cut in $G(\pi_0)$ by definition. So assume that X_i is a minimum cut in $G(\pi_{i-1})$ for some $i \geq 1$.

Denote by m_i the common value of the maximum flow f_i and the capacity of the minimum cut Z_i in $G(\pi_i)$ for all $i \geq 0$. Since the algorithm did not terminate at $G(\pi_{i-1})$, $m_{i-1} < \pi_{i-1} \cdot d(V(G))$. X_i is a minimum cut in $G(\pi_{i-1})$ by the assumption, so we get that $\delta_s(X_i) + \pi_{i-1} \cdot d(\overline{X_i}) < \pi_{i-1} \cdot d(V(G))$. Rearranging this gives $\pi_i = \dfrac{\delta_s(X_i)}{d(X_i)} < \pi_{i-1}$.

Applying the well-known submodularity of δ_s we get

$$\delta_s(X_i) + \delta_s(Z_i) \geq \delta_s(X_i \cap Z_i) + \delta_s(X_i \cup Z_i). \tag{2}$$

Furthermore,

$$\pi_{i-1} \cdot d(\overline{X_i}) + \pi_i \cdot d(\overline{Z_i}) = \pi_i \cdot d(\overline{X_i \cap Z_i}) + \pi_{i-1} \cdot d(\overline{X_i \cup Z_i}) + (\pi_{i-1} - \pi_i) \cdot d(Z_i \setminus X_i) \tag{3}$$

is easy to check (since every vertex contributes to both sides by the same amount). For any $Y \subseteq V(G) \setminus R$ and $i \geq 0$ denote by $g_i(Y)$ the capacity of the cut Y in $G(\pi_i)$; in other words, $g_i(Y) = \delta_s(Y) + \pi_i \cdot d(\overline{Y})$. Adding up (2) and (3) and using $\pi_i < \pi_{i-1}$ we get

$$g_{i-1}(X_i) + g_i(Z_i) \geq g_i(X_i \cap Z_i) + g_{i-1}(X_i \cup Z_i) \tag{4}$$

(and equation is possible only if $d(Z_i \setminus X_i) = 0$). The left hand side of (4) equals $m_{i-1} + m_i$ by the inductive assumption and the definition of Z_i. The right hand side of (4) can be bounded from below by $m_i + m_{i-1}$, because $g_i(X_i \cap Z_i) \geq m_i$ and $g_{i-1}(X_i \cup Z_i) \geq m_{i-1}$. All these together imply that all bounds above are fulfilled with equation. In particular, $g_i(X_i \cap Z_i) = m_i$, which indeed means that $X_{i+1} = X_i \cap Z_i$ is a minimum cut in $G(\pi_i)$ as stated.

Consequently, as long as the algorithm does not terminate, it keeps generating the strictly decreasing series of subsets X_i: $X_{i+1} \subset X_i$ and $X_{i+1} \neq X_i$ is true for every $i \geq 0$ (the latter by $\pi_{i+1} < \pi_i$). Hence it must terminate after at most $|V(G) \setminus R|$ cycles. □

The above imply that $\pi(G)$ can be efficiently computed by at most $|V(G) \setminus R|$ maximum flow computations. We remark that it also follows from the above proof that if d is assumed to be positive valued then the algorithm can be simplified: $Z_i \subseteq X_i$ is automatically true, so $X_{i+1} = Z_i$. Indeed, we showed above that (4) is fulfilled with equation and we also mentioned that this implies $d(Z_i \setminus X_i) = 0$; hence the positivity of d implies $Z_i \setminus X_i = \emptyset$.

3 The Minimum Cost Sink Selection Problem

Based on the above defined robustness metric $\pi(G)$, in this section we formalize the problem of optimal selection of sink nodes in a graph that models a WSN. 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 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 graph underlying the WSN is given and our task is to select the sink vertices such that the persistence of the resulting configuration is above a given threshold, while the total selection cost of the sink nodes is minimized. This models the design of a WSN with strict security requirements, but a flexible budget. According to the above, the sink selection problem is formalized as follows:

Definition 5 ***Minimum Cost 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 $\alpha \in \mathbb{R}^+$.

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

MINIMIZE: Selection cost $c(R) = \sum_{v \in R} c(v)$ of the subset R.

It was proved in [5] that this problem is NP-hard by reducing the Minimum Set Cover problem to it. Furthermore, it was also proved that the problem remains to be NP-hard even if $c(v) = 1$ for all $v \in V(G)$ is assumed. Finally, the reduction from the Minimum Set Cover problem also implied an approximability result: unless $P = NP$, the above Minimum Cost Sink Selection problem is impossible to approximate in polynomial time within a factor of $c \cdot \log \log |V(G)|$ for some constant $c > 0$.

In what follows, we describe a branch and bound type algorithm for solving the Minimum Cost Sink Selection problem. Obviously, the algorithm is exponential time and it is not intended for solving problems of the magnitude of a typical real life WSN. However, we still found it useful for testing the performance of various heuristic methods.

3.1 A Branch and Bound algorithm

The basic idea of the following algorithm is to imitate the running of a simple branch and bound method for the integer programming formulation of the Minimum Cost Sink Selection problem; however, instead of using a linear programming solver for the LP relaxation of the various subproblems arising during the process, we exploit the fact that every such problem is a very special minimum cost flow problem and thus solvable much more efficiently.

Assume that decision has already been made on certain vertices: some vertices were decided to be sink nodes – we refer to these as "black" vertices and denote their subset by B; other vertices were decided not to be sink nodes – we refer to these as "white" vertices and denote their subset by W. A decision is sought on the remaining vertices – these will be called "grey" vertices and their subset will be denoted by Y. For this, we will rely on Claim 2: $\pi(G, R) \geq \alpha$ if and only if there exists a flow in $G(\alpha)$ that saturates all arcs leaving s^*. However, the set of sink nodes R is not known in advance any more so we slightly modify the definition of $G(\alpha)$: an arc from v to t^* exists for each node $v \in Y \cup B$ with infinite capacity. However, every grey vertex $v \in Y$ for which the arc (v, t^*) carries some positive flow will have to be declared a sink node, so we will try to minimize the total selection cost of these.

For this purpose, we introduce a binary variable $r(v)$ for every grey vertex $v \in Y$: $r(v) = 1$ means that v is a sink node and $r(v) = 0$ means that it is not. We also need an easy-to-compute upper bound $L(v)$ on the maximum flow value the arc (v, t^*) needs to carry in $G(\alpha)$:

$$L(v) = \min \left\{ \alpha \cdot d(v) + \sum \{s(e) : e = (u, v), u \in Y \cup W\}; \alpha \cdot \sum \{d(u) : u \in Y \cup W\} \right\} \tag{5}$$

will do (that is, the minimum of the sum of capacities of arcs entering v and of those leaving s^* disregarding black vertices). (Of course, we could compute the maximum flow from s^* to v, but since the values $L(v)$

will be updated several times, that would immensely slow down the algorithm.) Then, obviously, imposing the constraint $f\big((v,t^*)\big) \le L(v)\cdot r(v)$ on every grey vertex $v \in Y$ is equivalent to allowing arbitrary flow values from grey vertices that become sink nodes and discarding the arc (v,t^*) from the rest.

All in all, solving the above described problem defined by the tripartition $V(G) = B \cup W \cup Y$ is equivalent to looking for a flow f in $G(\alpha)$ from s^* to t^* that saturates all arcs leaving s^* such that $f\big((v,t^*)\big) \le L(v)\cdot r(v)$ holds for every $v \in Y$ and $\sum_{v\in Y} c(v)\cdot r(v)$ is minimized.

Now consider the linear relaxation of this problem: instead of binarity, only the constraints $0 \le r(v) \le 1$ are imposed on the variables $r(v)$. Then, obviously, all inequalities $f\big((v,t^*)\big) \le L(v)\cdot r(v)$, $v \in Y$ can be assumed to be fulfilled with equation by any optimum solution. (Indeed, $r(v)$ could be decreased without increasing the objective function.) Hence the variables $r(v)$ can completely be eliminated from the problem and the objective function becomes $\displaystyle\sum_{v\in Y} \frac{c(v)}{L(v)} \cdot f\big((v,t^*)\big)$. Consequently, what we are left with is a minimum cost flow problem: let $\displaystyle w\big((v,t^*)\big) = \frac{c(v)}{L(v)}$ for all $v \in Y$ and $w(e) = 0$ for all other arcs of $G(\alpha)$; a flow f from s^* to t^* is sought that saturates all arcs leaving s^* and minimizes $\sum_{e\in E(G(\alpha))} w(e)f(e)$. Moreover, this minimum cost flow problem is a very special one: only arcs entering t^* have nonzero costs; it is easy to see that well-known, elementary algorithms for the minimum cost flow problem – like the Cycle Cancelling Algorithm or the Successive Shortest Paths Algorithm – that are exponential time in general, become polynomial time in this special case.

Of course, after solving the above linear relaxation (by computing a minimum cost flow), we have to check if the obtained solution corresponds to an integer solution or not: if the values $\displaystyle r(v) = \frac{f\big((v,t^*)\big)}{L(v)}$ all happen to be integer (that is, binary) then we succeeded in solving the problem (corresponding to the tripartition $V(G) = B \cup W \cup Y$). If not (which, obviously, is to be expected in general) then we split the problem into two subproblems in the most natural way: we pick some vertex $z \in Y$, delete z from Y and put it into W and B in the first and second subproblems, respectively. As it is usual with branch and bound type algorithms, we store the corresponding objective function value obtained from the solution of the linear relaxation with the two arising subproblems as guaranteed lower bounds on the cost of any sink selection they can give rise to.

Based on the above, we summarize (and somewhat more formally describe) the algorithm below. During its run, it maintains a list $\mathcal{L} = \big\{(B_i, W_i, Y_i, b_i) : i = 1,2,\dots\big\}$ of subproblems of the input minimum cost sink selection problem, where $V(G) = B_i \cup W_i \cup Y_i$ is a tripartition of the vertex set and b_i is a lower bound on the optimum of the corresponding subproblem. Furthermore, it stores the cheapest sink selection R^* found so far and its total cost $C^* = \sum_{v\in R^*} c(v)$.

Algorithm 6

Step 0. *Let* $\mathcal{L} = \big\{(\emptyset,\emptyset,V(G),0)\big\}$, $R^* = V(G)$, $C^* = \sum_{v\in V(G)} c(v)$.

Step 1. *If* $\mathcal{L} = \emptyset$ *then STOP and output* R^*. *If not, then choose a subproblem* (B_i, W_i, Y_i, b_i) *from* $\mathcal{L}$ *and delete it from* $\mathcal{L}$.

Step 2. *If* $b_i \ge C^*$ *then continue at Step 1.*

Step 3. *For every* $v \in Y_i$ *compute* $L_i(v)$ *defined by* (5) *above.*

Step 4. *Compute a minimum cost flow* f_i *from* s^* *to* t^* *in* $G(\alpha)$ *(corresponding to the tripartition* $B_i \cup W_i \cup Y_i$*) that saturates all arcs leaving* s^* *with respect to the cost function* $w\big((v,t^*)\big) = \frac{c(v)}{L_i(v)}$ *for all* $v \in Y_i$ *and* $w(e) = 0$ *for all other arcs. If no such flow exists then continue at Step 1.*

Step 5. *Denote the cost of* f_i *(the optimum computed in Step 4) by* $w(f_i)$ *and let* $\omega_i = w(f_i) + \sum_{v\in B_i} c(v)$. *If* $\omega_i \ge C^*$ *then continue at Step 1.*

Step 6. *If* $\displaystyle r_i(v) = \frac{f_i\big((v,t^*)\big)}{L_i(v)}$ *is binary for all* $v \in Y_i$ *then let* $R^* = B_i \cup \big\{v \in Y_i : r_i(v) = 1\big\}$ *and let* $C^* = \sum_{v\in R^*} c(v)$. *If not then choose a vertex* $z_i \in Y_i$ *and add* $(B_i \cup \{z_i\}, W_i, Y_i \setminus \{z_i\}, \omega_i)$ *and* $(B_i, W_i \cup \{z_i\}, Y_i \setminus \{z_i\}, \omega_i)$ *to* $\mathcal{L}$. *Continue at Step 1.*

Of course, before the above algorithm is implemented, certain details need to be cleared up:

1. *Choice of z_i.* According to our experiments, choosing z_i to be the element $v \in Y_i$ for which $|r_i(v) - \frac{1}{2}|$ is minimum works relatively well.

2. *Choice of the next subproblem $(B_i, W_i, Y_i, b_i) \in \mathcal{L}$.* It seems sensible to use the LIFO strategy in most of the cases, that is, to choose one of the newest subproblems attached to $\mathcal{L}$. Doing this enables us to work on with the flow values computed in the previous cycle. If the present subproblem was obtained from the previous one by turning a grey vertex black then the previous flow is still a maximum flow (that is, it saturates all arcs leaving s^*), so it is natural to use the Cycle Cancelling Algorithm starting from that flow. If, on the other hand, the present subproblem was obtained by turning a grey vertex v white then the Successive Shortest Paths Algorithm seems to be the sensible choice: introduce an arc temporarily from v back to s^* and let both its flow value and capacity be $f\big((v, t^*)\big)$ (where f is the previous flow); then, after turning v white and removing the arc (v, t^*) we have a flow that is obviously of minimum cost among flows of the same value.

   However, if the last member of $\mathcal{L}$ was not obtained from the one treated in the previous cycle then there is no point in using the LIFO strategy. In this case, choosing the subproblem (B_i, W_i, Y_i, b_i) for which b_i is minimum seems sensible.

To test the above described algorithm, we experimented on *unit disc graphs* that model WSNs relatively well: vertices corresponded to points in the plane (randomly distributed within a fixed rectangle) and edges corresponded to pairs of points within unit distance from each other. The arising graphs had an average vertex degree of 4. The required persistence value was set to $\alpha = 1$ and all edge weights $s(e)$, node weights $d(v)$ and node costs $c(v)$ were chosen from uniform distribution between 0.5 and 1.5. Our results show that the algorithm performed very well as compared to running an (open source, revised dual simplex method based) integer programming solver on the IP formulation of the problem. For example, finding the minimum cost sink selection in a graph with 36 vertices took more than 34 hours for the IP solver, while the same problem took less than 4 minutes for our algorithm. Furthermore, our algorithm terminated within acceptable running times for graphs with a vertex count of around 50, where the IP solver was already hopeless. However, finding the optimum solution for graphs corresponding to the sizes of real-life WSNs (that is, with a vertex count in the magnitude of a few thousand) is obviously intractable for our algorithm too.

4 Related Work

In [5], besides proposing various heuristics for the above introduced Minimum Cost Sink Selection problem, we also considered the *Sink Placement* problem to better model the applications arising from WSNs. Here we assumed that the vertices of a given graph G (with given edge weights $s(e)$ and node weights $d(v)$) are embedded into the plane and the task is to find a further set of points R in the plane such that connecting every point of R with all vertices of G that are within a given distance (corresponding to the transmission radius of the devices) the persistence of the resulting network (with the weights of the new edges set to 1) is above a given threshold and the cardinality of R is minimum. We gave an efficient search space reduction technique that reduces the Sink Placement problem to a Minimum Cost Sink Selection problem.

In [6] it was shown that the notion of persistence $\pi(G)$ also has an interesting game-theoretic interpretation. Assume that a connected graph G and a designated vertex $r \in V(G)$ are given. Two players, the Network Operator and the Adversary play the following game: the Network Operator chooses a spanning tree T of G and, simultaneously, the Attacker chooses (or attacks) an edge $e \in E(G)$. If $e \notin E(T)$ then there is no payoff; if, however, $e \in E(T)$ then the payoff for the Attacker is the number of vertices in the component of $T - e$ not including r. Optimum mixed strategies for both players in this two-player, zero sum game were given in [6] and it was proved that the value (that is, Nash equilibrium payoff) of the game is equal to $\dfrac{1}{\pi(G)}$. This statement is obviously strongly related to the fact that $\pi(G)$ is a useful

robustness metric: the higher the value of $\pi(G)$ is, the smaller is the average payoff the Attacker can achieve. In [7] we considered the modified version of the above game where there is no designated node r, the payoff for the Attacker is the size of the smaller component of $T - e$; optimum mixed strategies were given and the value of the game was found for this version of the game too.

References

[1] Bauer, Douglas, S. Louis Hakimi, and E. Schmeichel, Recognizing tough graphs is NP-hard *Discrete Applied Mathematics* (19), **28.3**, 191–195.

[2] Cunningham, William H., Optimal attack and reinforcement of a network. *Journal of the ACM (JACM)* (1985), **32(3)**, 549–561.

[3] Gusfield, Dan., Connectivity and edge-disjoint spanning trees. *Information Processing Letters* (1983), **16.2**, 87–89.

[4] Laszka, Aron, Levente Buttyán, and Dávid Szeszlér, Optimal selection of sink nodes in wireless sensor networks in adversarial environments *In: 2nd IEEE International Workshop on Data Security and PrivAcy in wireless Networks (D-SPAN). Lucca, Italy* (2011), 1–6.

[5] Laszka, Aron, Levente Buttyán, and Dávid Szeszlér, Designing Robust Network Topologies for Wireless Sensor Networks in Adversarial Environments, *Pervasive and Mobile Computing*, In Press, available online since May 2012.

[6] Laszka, Aron, Levente Buttyán, and Dávid Szeszlér, Game-theoretic Robustness of Many-to-one Networks, *In: Krishnamurthy V, Zhao Q, Huang M, Wen Y (ed.) Game Theory for Networks. Vancouver, Canada* (2012), Springer, 1–11.

[7] Laszka, Aron, Levente Buttyán, and Dávid Szeszlér, Linear Loss Function for the Network Blocking Game: An Efficient Model for Measuring Network Robustness and Link Criticality, *In: Jens Grossklags, Jean Walrand (ed.) Decision and Game Theory for Security. Budapest, Hungary*, (2012), Springer, 152–170.