EXTENDING NETWORK LIFETIME OF WIRELESS SENSOR NETWORKS

Ali Amiri

Department of MSIS
Spears School of Business
OklahomaStateUniversity
Stillwater, OK, 74078, USA

ABSTRACT

One critical issue in designing and managing a wireless sensor network is how to save the energy consumption of the sensors in order to maximize network lifetime under the constraint of full coverage of the monitored targets. In this paper, we adopt the common approach of creating disjoint sensor covers to prolong network lifetime. The typical goal used in the literature is to maximize the number of covers without consideration of the energy levels of the sensors. We argue that the network lifetime can be extended by maximizing the total bottleneck energy of the created covers. We formally define the problem of maximizing the total bottleneck energy of the covers, present for the first time an integer programming formulation of the problem, and develop two algorithms to solve large problem instances. Extensive experimental tests show that the use of the goal of maximizing the total bottleneck energy of the covers creates covers with substantially longer network lifetime than the lifetime of the covers created with the goal of maximizing solely the number of covers.

KEYWORDS

Wireless Sensor Network, Power Management, Bottleneck Energy, Integer Programming

1.INTRODUCTION

Wireless sensor networks have recently achieved widespread attention because of their common applications in military,environment, health, and other commercialareas [1].For instance, one recent application involves the planning and design of a sensor network for asset tracking in healthcare environments [2] [3]. Two objectives are considered: (i) identify the optimal locations of a given number of sensors to maximize the coverage of the assets to be monitored [2] [3], and (ii) determine the minimum number of sensors and their locations to achieve a desired level of coverage of the assets [3]. The application problem was formulated as a set covering problem and solved using commercial/open source optimization solvers [3] and genetic algorithms [2].

A typical wireless sensor network is composed of electronic devices, the sensors, which monitor targets and collect data about them and transmit itto the sink which acts as the network interface to serve external user applications.One significant feature of wireless sensor networks is that sensors are usually battery powered, and therefore have limited and commonly irreplaceable power supply. Thus, one issue of common interest in designing and managingawireless sensor network is how to save the overall energy consumption of the sensors in order to maximize the lifetime of the network under the constraint of full coverage of the targets being monitored [4] [5].

An adopted strategy to conserve sensor energy is based on an approximate data collection approach [6]. This approach attempts to exploit the spatial correlations among the sensor readings by selecting a subset of representative sensor nodes (referred to as R-nodes) to report their readings to the sink and estimating the readings of the remaining sensors as the readings of the corresponding R-nodes [7] [6] [8] [9] [10] [11] [12]. The quality of this approach depends on the number of the selected representative nodes and their energy consumption. Kotidis [8] developed a method called snapshot query that constructs the set of R-nodes based on the spatial correlations among the one-hop neighboring nodes. Liu et al. [9] formulated the problem as a clique-covering problem that partition the sensor nodes into cliques with similar readings. It is reported in [6] that the clique-covering based method, named EEDC, performs better than the snapshot query in term of the number of R-nodes selected. More recently, Hung et al. [6] viewed the problem of selecting R-nodes as an energy-aware set covering problem. In this problem, sensor nodes with higher energy and wider data coverage ranges are selected as the R-nodes for approximate data collection. Hung et al. [6] proposed a centralized algorithm, called DCglobal (standing for Data Coverage with global information) to construct the set of R-nodes based on available sensor energy and data coverage ranges. They conducted computational tests that show that the algorithm outperforms in general other prior methods such as snapshot query [8], EEDC [9] and DomSet [12].

Anothercommon approach for prolonging network lifetime consists of scheduling sensors to switch between active and inactive modes to reduce energy consumption while maintaining full coverage of the targets [13] [14]. As explained by Ting and Liao [4], this approach basically divides all sensors in the network into disjoint sensor subsets, or sensor covers (or just covers) each of which must satisfy the full coverage constraint. At any time of the network lifespan, only one sensor cover is in active mode providing network functionality, and the other sensor covers are inactive to save energy. As soon as any sensor in the active cover depletes its energy, and can no longer provide full coverage, one of the inactive sensor covers is selected to become active and continue functionality. Research studies [13] [15] show that this approach is effective in both reducing energy consumption and extending network lifetime.

The premise has been that identifying a higher number of sensor covers allows the network lifetime to be extended further. The problem of determining the maximum number of covers to extend network lifetime has been modeled as the SET K-COVER problem by Slijepcevic and Potkonjak [16]. Several algorithms have been proposed to solve the SET K-COVER problem.

First, Slijepcevic and Potkonjak [16] proposed a heuristic to solve the problem, called the Most Constrained Minimally Constraining Covering heuristic (MCMCC). The basic idea of the heuristic is to minimize the coverage of sparsely covered areas within one cover. The heuristic is fast, but often produces unsatisfactory results [4]. Cardei and Due [17] developed the Maximum Covers using Mixed Integer Programming (MC-MIP) heuristic, which computes the number of covers by first transforming the SET K-COVER problem into an integer maximum flow problem, and second constructing the covers from the solution to the flow problem. The MC-MIP heuristic can require an exhaustive search of the solution space to find the optimal solution.
Hence, MC-MIP is impractical in large-scale applications due to its exponential time complexity.

Lai et al. [18] developed a genetic algorithm (called iGA) to solve the SET K-COVER problem. The authors reported computational results that show that this algorithm outperforms MCMCC [16] in terms of the number of covers and speed. Recently, Ting and Liao [4] developed a memetic algorithm which uses the Darwinian evolutionary scheme and Lamarckian local enhancement to search for optima given the considerations of global exploration and local exploitation. They reported results of computational tests which show that the algorithm outperforms several heuristicssuch as MCMCC [16] and iGA [18].

Other works have studied other variations of the problem of maximizing network lifetime [19] [20] [7] [21] [1] [22]. Cerulli et al. [20] addressed the problem of determining the activation times of sensors with adjustable sensing ranges to cover a set of targets. The authors presented some heuristic approaches and an exact method based on the column generation technique to solve the problem. They reported results of computational tests that show the advantage of using adjustable range sensors over fixed range sensors. Rossi et al. [22] incorporated a genetic algorithm within a column generation scheme to solve the same problem faster. Behdani et al.

[19] considered a version of the problem where the sink node is mobile and the transmission of sensor readings to the sink can be delayed. Li and Aneja [1] addressed a broadcasting problem which involves determining a transmission route between a specified source node and every other node in the network in order to minimize total energy consumption. Lai et al. [21] studied the problem of arranging sensor cluster sizes and transmission ranges of sensor nodes in order to balance energy consumption among clusters and consequently prolong network lifetime.

A close analysis of the literature dealing with the SET K-COVER problem as the basis for prolonging the lifetime of the wireless sensor network reveals that a key aspect of the problem has been ignored, namely the energy levels of the sensors. Indeed, two factors have direct impact on the lifetime of the network: (i) the number of covers in the network, and (ii) the bottleneck energy of each cover. The bottleneck energy of a cover is the minimum energy level among the energy levels of all sensors in the cover. Once the sensor with this bottleneck energy runs out of energy, the cover looses full coverage of the target area regardless of the energy levels of the other sensors.

The example [16] shown in fig. 1 illustrates the importance of incorporating the sensor energy in solving the SET K-COVER problem. There are 6 sensors and 5 targets in the network. If we solve the SET K-COVER problem with the goal of maximizing the number of covers, then the optimal solution consists of three covers C_1={3,5}, C_2={1,4}, and C_3={2,6}. The total bottleneck energy of the covers is 10 (i.e., 2+6+2). On the other hand, if we solve the SET K-COVER problem with the goal of maximizing the total bottleneck energy of the covers, then the optimal solution consists of two covers C_1={2,3} and C_2={1,4}. The total bottleneck energy of the solution is 13 (i.e., 7+6}. Hence, by explicitly considering the sensor energy in the SET K-COVER problem, the network lifetime can be further extended.

Sensor coverages and energy levels for the network example

Sensor	Energy	Coverage
1	8	1,2,3,4
2	8	1,2,5
3	7	2,3,4,5
4	6	2,3,5
5	2	1,3,5
6	2	3,4,5

Set of covers based on maximinzing the number of covers

Cover	Sensors
1	3,5
2	1,4
3	2,6

Set of covers based on maximinzing the bottleneck energy of covers

Cover	Sensors
1	2,3
2	1,4

Fig. 1. Importance of considering sensor energy in the set k-cover problem

Since the common goal in designing and managing a wireless sensor network is to extend the network lifetime, we argue that the selection of the covers should take into consideration not only the target coverage of the sensors, but also their energy levels. Indeed, if a cover with lower bottleneck energy is constructed, the network lifetime will be reduced. Conversely, it is likely that including only sensors with higher energy levels may increase the number of sensors required to form a cover because sensors with more energy may have smaller target coverage ranges. Therefore, we formulate the problem of maximizing the network lifetime as an energy-aware SET K-COVER problem, which is an extension of the well-known SET K-COVER problem.

Three aspects of the problem of extending the lifetime of a wireless sensor network stand out when reviewing existing literature: (i) as can be seen from the body of recent work, the network lifetime maximization problem remains a very important one, (ii) the formulation of the problem as a SET K-COVER problem is incomplete as it disregards the energy levels of the sensors, and (iii) there has been no attempt to formally define and formulate the problem mathematically.

This paper fills this gap in the literature. We frame the problem of extending the network lifetime as an energy-aware SET K-COVER problem, present a formal definition of the problem and develop for the first time, to the best of our knowledge, an integer programming formulation of the problem. While this formulation can be used to solve optimally only relatively small instances of the problem, it can form the basis of solution algorithms in future research studies. We also present an effective heuristic algorithm to solve the problem. Results of extensive computational tests show that our algorithm is very robust and extends the network lifetime longer.

2. PROBLEM DEFINITION AND INTEGER PROGRAMMING FORMULATION

A wireless sensor network is composed of a set of sensors (S) and a set of targets (T). A target is said to be covered by a sensor if it lies within the sensing range of the sensor. Each sensor $i \in S$ is characterized by its energy level e_i and its target coverage. The SET K-COVER problem consists of finding the set of disjoint covers for T that prolongs the network lifetime the most. A cover $C_i$for T is a subset of sensors, $C_i \subset S$, such that every target of T is covered by at least one sensor in C_i. The covers have to be disjoint, that is, for any two covers C_i and C_j, $C_i \cap C_j = \emptyset$.

Two features of this set cover have direct impact on the network lifetime: (i) the bottleneck energy of each cover, and (ii) the number of the sensors in each cover. Obviously, (i) a cover should have a large level of bottleneck energy because as soon as the sensor with this bottleneck energy runs out of energy, the cover looses full coverage of the targets and (ii) the number of sensors in each cover should be minimized to create the maximum number of covers in order to prolong the overall network lifetime. The first priority to prolong network lifetime is to maximize the energy levels of the sensors in the covers and the second priority is to minimize the number of sensors in each cover. More precisely, the first priority is to maximize the bottleneck energy of each cover. This bottleneck energy of a cover is the minimum energy level among the energy levels of all sensors included in the cover. The problem of maximizing lifetime of a wireless sensor network can now be formulated as follows.

Parameters

S	the set of sensors in the network
T	the set of targets in the network
M	the set of covers in the network
e_j	energy of sensor $j \in S$

$$a_{ij} = \begin{cases} 1 & if sensor j \in S covers target node i \in T \\ 0 & otherwise \end{cases}$$

α	large positive number
β	large positive number

Decision variables

$$Z_k = \begin{cases} 1 & if\ set\ cover\ k \in M is created \\ 0 & otherwise \end{cases}$$

$$Y_{jk} = \begin{cases} 1 & if sensor j \in S is included\ in\ set\ cover\ k \in M \\ 0 & otherwise \end{cases}$$

$$X_{ijk} = \begin{cases} 1 & if target i \in T is covered by\ sensor\ j\ of\ set\ cover\ k \\ 0 & otherwise \end{cases}$$

V_k = Bottleneck energy of set cover $k \in M$

Problem*MNL:*

$$Max\alpha \sum_{k\in M} V_k - \sum_{k\in T}\sum_{j\in T} Y_{jk} \tag{1}$$

Subject to

$$\sum_{j\in S} a_{ij} X_{ijk} = Z_k \forall i \in T, k \in M \tag{2}$$

$$\sum_{k\in M} Y_{jk} \leq 1 \qquad \forall j \in S \tag{3}$$

$$X_{ijk} \leq Y_{jk} \forall i \in T, j \in S, k \in M \;\; such\ that\ a_{ij} = 1 \tag{4}$$

$$V_k \leq e_j Y_{jk} + \beta(1 - Y_{jk}) \forall j \in S, k \in M \tag{5}$$

$$V_k \leq \beta Z_k \forall j \in S, k \in M \tag{6}$$

$$Z_k, V_k, Y_{jk}, X_{ijk}\ binary \forall i \in T, j \in S, k \in M \tag{7}$$

The objective function maximizes the bottleneck energies of the selected covers while minimizing the number of sensors used. Constraint set (2) ensures that a target is covered by sensor *j* of cover k only if cover k is created. Constraint set (3) ensures that a sensor can be included in at most one cover. Constraint set (4) requires that if a target *i* is covered by a sensor *j* of cover *k*, then sensor *j* should be included in cover *k*. Constraint set (5) defines the bottleneck energy V_k of the covers; if a sensor *j* is included in cover *k*, then its energy should be greater than or equal to the bottleneck energy V_k and if sensor *j* is not included in cover *k*, then its energy is not considered in cover *k*. Constraint set (6) allows the bottleneck energy V_k of the cover *k* to be positive only if cover k is created. Constraint set (7) requires that the decision variables be binary. Note that α and β are defined as generic large numbers. However, any value of $\alpha > |S|$ is valid. Similarly, any value of $\beta > max_{i\in S} e_i$ is acceptable.

3. SOLUTION ALGORITHMS

The Best-Sensor-Fit algorithm, as its name indicates, constructs the set of covers in a greedy fashion so as to maximize the total bottleneck energy of the covers. For each cover to construct, it is important to select the fewest sensors with the highest bottleneck energy in order to cover all targets in the network. Each cover is initially empty, and then sensors are added consecutively until all targets in the network are covered, while simultaneously maximizing the bottleneck energy of the selected sensors and minimizing their number. The algorithm uses a quantity called, *score(j)*, to guide the search for the "winner" sensor to include in the current cover. The quantity, *score(j)*, is computed for each sensor (not included in any cover already) as follows: $score(j) = e_j \sum_{i \in \tilde{T}} a_{ij}$, where $\tilde{T}$ is the set of targets which are not covered yet. Here, the quantity *score(i)* is used to identify the next sensor to include in the current cover under construction. It is the sensor with the highest score; that is, the sensor that has the best combination of energy and target coverage range.

We let ub be an upper bound on the number of covers *K*. ub is determined in [18] as $ub = min_{t \in T} |S(t)|$, where *S(t)* is the set of sensors that cover target *t* and |.| denotes the cardinality.

The Best-Sensor-Fit algorithm is outlined below.

Input: *T* the set of targets and *S* the set of sensor nodes with target coverages and energy levels.
Output: *M* the set of covers

```
Step 1: Initialize
    • Each cover $C_k = \emptyset \forall k \in \{1, \dots, ub\}$
    • set of covers $M = \emptyset$
    • Set of non-selected sensors: $\tilde{S} = S$
    • done=false
    • k=1
Step 2: Generate set of covers
    While $k \le ub$ and done=false do
        set of covered targets: $T^c = \emptyset$; set uncovered targets: $\tilde{T} = T$
        while $T^c \neq T$ and done = false do
            Calculate $score(j) = e_j \sum_{i \in \tilde{T}} a_{ij} \forall j \in \tilde{S}$
            Let $j^* = \text{argmax}_{j \in \tilde{T}} \{score(j) > 0\}$
            if $j^*$ exists then
            Update $C_k := C_k \cup \{j^*\}$
            Update $\tilde{S} := \tilde{S} \setminus \{j^*\}$
            for each target node $i \in \tilde{T}$ such that $a_{ij^*} = 1$ do
            Update $T^c := T^c \cup \{i\}; \tilde{T} := \tilde{T} \setminus \{i\}$
            end for
            else
              done=true
            end if
        end while
        if done=false then
        Update $M = M \cup \{k\}$
        Update $k := k+1$
        end if
    end while
```

Basically, the algorithm works as follows. The covers are constructed one at a time. Initially, none of the targets is covered (i.e., $T^c = \emptyset$). Each cover is initially empty (i.e., $C_k = \emptyset$) and then sensors are iteratively added to the current cover under construction until either(i) all targets are covered or (ii) none of the un-used sensors can cover the remaining targets. In the first case, a cover is constructed and in the second case, it is not and the algorithm terminates.

Greedy heuristics can get trapped in a local optimum. To overcome this problem, we add a randomization feature to our Best-Sensor-Fit algorithm similar to the one used in [24] to solve the set covering problem. The idea of randomization is to slightly perturb the values of *score(j)*of the sensors evaluated in steps 2 as follows: $score(j) = (1 + \varepsilon_j)e_j \sum_{i\in\tilde{T}} a_{ij}\ \forall j \in \tilde{S}$, where ε_j is a small random number. In our computational tests, ε_j is drawn from the uniform random distribution *U*[0.05, 0.15]. This perturbation technique assumes implicitly that the optimal solution is not sensitive to small variations of the values of *score(j)* which are computed in terms of the values of the problem parameters e_j's and a_{ij}'s. The whole procedure made of steps 1 and 2 has to be repeated a number of times, set to 500 in this study. The hope of the technique is to diversify the search space, potentially leading to better solutions. As noted in [24], the perturbation technique has been incorporated successfully in local search methods to solve other complex problems.

The Randomized Best-Sensor-Fit Algorithm (RBSF) to solve the problem of maximizing network lifetime (*MNL*) can now be outlined as follows.

```
Randomized Best-Sensor-Fit Algorithm (RBSF):

Let $F$* be the best feasible solution found so far

**Repeat**$P$ times ($P$=500 in this study)

**Step 1: Initialize**
        •  Each cover $C_k = \emptyset \forall k \in \{1, \dots, ub\}$
        •  set of covers $M = \emptyset$
        •  done=false
        •  $k$=1
        •  $F$ is the current solution (made of the created covers and their sensors)

**Step 2: Generate set of covers**
    **While** $k \le ub$and done=false **do**
        set of covered targets: $T^c = \emptyset$; set uncovered targets: $\tilde{T} = T$
        **while**$T^c \neq T$and done = false**do**
                **Calculate**$score(j) = (1 + \varepsilon_j)e_j \sum_{i\in\tilde{T}} a_{ij}\ \forall j \in \tilde{S}$
            **Let**$j^* = \text{argmax}_{j\in\tilde{T}}\{score(j) > 0\}$
            **if**$j^*$ exists **then**
            **Update**$C_k \coloneqq C_k \cup \{j^*\}$
            **Update**$\tilde{S} \coloneqq \tilde{S} \setminus \{j^*\}$
            **for** each target node $i \in \tilde{T}$ such that $a_{ij^*} = 1$ do
```

```
                    **Update** $T^c \coloneqq T^c \cup \{i\}; \tilde{T} \coloneqq \tilde{T} \setminus \{i\}$
                **end for**
                **else**
                    done=true
                **end if**
            **end while**
            **if** done=false **then**
                **Update** $k{:}{=}k{+}1$
            **end if**
        **end while**
```

Step 3: Update best feasible solution

If the current solution *F* is better than *F** in term of the objective function, then *F**:=*F*.

End Repeat

Output *F** as the best feasible solution and stop.

4.PERFORMANCE EVALUATION

Extensive computational testing was conducted on synthesis data sets to evaluate the performance of the proposed algorithms for solving the problem of maximizing network lifetime (*MNL*). This section describes the data used for computational testing, discusses the parameters used, and analyzes the results.

4.1.Data generation

As in [4], the application scenario is based on static sensor networks. That is, the communication connectivity graphs are given, where a target *i* is covered by sensor *j* if the target is located within the sensing range of the sensor. The sensing range (*r*) of a sensor *j* is defined as the maximum distance between the sensor and a target *i* for this target to be considered as covered by the sensor.

That is, $a_{ij} = \begin{cases} 1 & if distance(i,j) \leq r \\ 0 & otherwise \end{cases}$

The experiments were conducted using networks with varying sizes. As in [4], we generated several groups of networks randomly but systematically to capture a wide range of problem structures. One hundred problem instances from each group with the same structure were generated in order to achieve a reasonable level of confidence about the performance of the solution procedures on that problem structure. The numbers of sensors and targets vary between 50 and 500. To evaluate the performance of the algorithmsas a result of changes in the problem parameters, a total of 6000 problem instances were solved. Similar to [4], the sensors and targets in the network are deployed randomly over a 500 x 500 area. The energy levels of the sensors are generated randomly from the uniform distribution between 100 and 200.

Two factors ρ_t and ρ_s [4] are used to characterize the problems instances. ρ_t represents the average number of targets covered by a sensor and ρ_s represents the average number of sensors which cover a target. The factors ρ_t and $\rho_s$are computed as follows (where |.| denotes cardinality):

$$\rho_t = \frac{\sum_{j\in S}\sum_{i\in T} a_{ij}}{|S|}$$

$$\rho_s = \frac{\sum_{j\in S}\sum_{i\in T} a_{ij}}{|T|}$$

An upper bound on the number of covers that can be formed is given by [18] [4]:

$$ub = min_{j\in T} \sum_{i\in S} a_{ij}$$

We used performance metrics similar to those adopted by Ting et al. [4] to evaluate the performance results, namely the number of covers which are created and the gap between this number and the upper bound (ub). In addition, we report the total bottleneck energy of the created covers which determines the network lifetime, and the running time (in seconds) which is the time it takes to solve the problem on Windows 7 Intel Core i5-2500 3.3 GHz machine.

4.2. Computational results with different network design goals

First, we run computational tests to study the effects of the goals of the network design on the lifetime of the network. The first goal is the typical goal adopted in prior studies [4] which is to maximize the number of covers in the network with no consideration of sensor energy levels (i.e., goal of the classic SET K-COVER problem). The second goal is the one adopted in our study which is to maximize the total bottleneck energy of the covers. We considered small networks with the number of targets varying between 10 and 30 and the number of sensors fixed to 50.

We used the commercial optimization software CPLEX to solve optimally the SET K-COVER problem with the first goal. CPLEX failed to solve this problem with larger networks (i.e., networks with more than 30 targets). In addition, CPLEX failed to solve the problem with the second goal using the small networks. For this reason, we used our Randomized Best-Sensor-Fit Algorithm (RBSF) to solve that problem.

Table 1 shows the effects of the goals of the network design on the network lifetime. Both goals generated the optimal number of covers. However, the goal of maximizing the total bottleneck energy of the covers produced covers with substantially longer network lifetime than the lifetime of the covers produced with the goal of maximizing the number of covers. Indeed, using the total bottleneck energy of the covers as the measure of network lifetime, the computational results show that network lifetime using the second goal is on average 14.7% longer than that using the first goal. The outperformance using the second goal is achieved by succeeding in employing a higher number of sensors in creating the covers. Indeed, the covers created based on the second goal employed on average 12.4% more sensors than the covers created based on the first goal.

Hence, these results clearly show the merit of using the goal of maximizing the total bottleneck energy of the covers to prolong the lifetime of a wireless sensor network.

Table 1. Computational results obtained for different network design goals

					Goal: maximize number of covers				Goal: maximize bottleneck energy			
$\|T\|$	$\|S\|$	ρ_t	ρ_s	ub	# of Covers Created	Bottleneck Energy	# of Active Sensors	CPU Time	# of Covers Created	Bottleneck Energy	# of Active Sensors	CPU Time
10	50	5.7	28.6	18	18	2212	36	49	18	2423	40	38
15	50	8.2	27.3	15	15	1873	31	6292	15	2027	38	34
20	50	11.0	27.5	15	15	1836	38	789	15	2012	41	34
25	50	13.3	26.6	15	15	1771	39	1823	15	2121	44	34
30	50	15.3	25.4	11	11	1302	33	2258	11	1646	36	27

4.3. Computational results with different sensing ranges

Table 2 show the effects of changes in the sensing range (r) of the sensors. For a given network, we varied the sensing range r from 50 to 500. We used two network sizes: The smaller networks consist of 100 sensors and 100 targets and the larger networks consist of 300 sensors and 500 targets, this larger size has also been used in [4]. The computational results show that a larger sensing range of the sensors results in a larger number of covers created and higher total bottleneck energy of those covers, indicating a longer network lifetime. The results also show that the Randomized Best-Sensor-Fit Algorithm (RBSF) outperforms the Best-Sensor-Fit Algorithm (BSF) in terms of the number of covers obtained and the total bottleneck energy of the covers. The number of covers obtained using RBSF is on average 1.43% higher than the number of covers obtained using BSF. Similarly, the total bottleneck energy of the covers obtained using RBSF is on average 1.36% higher than the total bottleneck energy of the covers obtained using BSF.Thus, RBSF produces solutions that extend the network lifetime much longer than those produced by BSF. This outperformance of RBSF over BSF shows the effectiveness of the randomization feature embedded in RBSF in prolonging the network lifetime. The statistical paired t-tests of the numbers of covers and the total bottleneck energy of the covers obtained using the two algorithms confirm the superiority of RBSF over BSF at 99% significance level.

However, this superiority seems to be more significant for the smaller networks.

Table 2. Computational results obtained for different sensing ranges (r)

						BSF algorithm				RBSF algorithm			
$\|T\|$	$\|S\|$	ρ_t	ρ_s	ub	r	# of Covers Created	Bottleneck Energy	# of Active Sensors	CPU	# of Covers Created	Bottleneck Energy	# of Active Sensors	CPU
100	100	3.1	3.1	0.2	50	0.2	21	5.8	0	0.2	21	5.6	13
100	100	10.4	10.4	2.8	100	2.8	369	39.2	0	2.8	378	40	17
100	100	21.1	21.1	7.8	150	7.4	994	58.8	0	7.8	1074	63	23
100	100	33.8	33.8	12.4	200	12.0	1691	69.2	0	12.4	1768	71.6	30
100	100	47.2	47.2	20.2	250	19.6	2779	79.2	0	20	2872	81.2	40
100	100	60.9	60.9	28.4	300	27.0	3896	82.4	0	28.2	4110	85.8	51
100	100	73.3	73.3	37.6	350	36.8	5400	86.6	0	37.6	5560	88	63
100	100	84.2	84.2	49.2	400	47.2	7051	92.8	0	48.2	7215	93	77
100	100	92.4	92.4	64.8	450	58.0	8752	98.0	0	59.2	8922	98.6	92
100	100	97.3	97.3	80.8	500	70.2	10655	99.0	0	70.6	10713	99	106
500	300	14.6	8.8	1.2	50	1.0	123	51.6	0	1.2	149	62.8	167
500	300	52.2	31.3	9.6	100	9.2	1267	164.0	0	9.6	1320	172.8	171
500	300	106.3	63.8	22.8	150	21.4	3078	202.0	0	22.6	3260	209.4	178
500	300	170.6	102.4	37.2	200	36.8	5491	248.4	0	37.2	5551	252.4	186
500	300	239.4	143.6	61.8	250	59.4	8359	273.8	0	60.6	8574	276.4	194
500	300	307.2	184.3	86.2	300	83.2	11953	274.8	0	84.8	12195	280.8	204
500	300	369.0	221.4	115.2	350	113.0	16523	277.6	0	114.2	16725	279.6	215
500	300	421.9	253.1	151.0	400	144.2	21545	288.4	0	145.8	21765	291.2	234
500	300	462.4	277.4	190.0	450	173.8	26225	297.4	0	175	26396	298.4	265
500	300	486.4	291.9	235.2	500	204.8	31225	299.4	0	206.2	31118	299.6	298

The results reported in table 2 show also the performance of the two algorithms based on the comparison between the number of covers created and its upper bound (*ub*). Based on these results, the average gap between the number of covers created using BSF and the upper bound (*ub*) is 4.6% and the average gap between the number of covers created using RBSF and the upper bound (*ub*) is 2.4%. This is another indication of the good performance of RBSF and to a lesser extend BSF.

4.4. Computational results with different numbers of targets

The second part of the computational study evaluated the performance of the two algorithms with different numbers of targets. As in [4], the study considered two settings for the number of sensors and sensing range of each. The first involves networks with 100 sensors and a sensing range of *r*=250. The second involves networks with 300 sensors and a sensing range of *r*=400. Table 3 shows the characteristics of the problem instances for different numbers of targets, varying from 50 to 500. These characteristics are similar to those observed in [4].

Specifically, varying the number of targets does not affect ρ_s, but causes ρ_t to increase and *ub* to decrease. The same table shows the effects of changes in the number of targets on the number of covers created and their total bottleneck energy. An increase in the number of targets causes, in general, the number of created covers and their total bottleneck energy to decrease. However, this may not be the case for some problem instances because the algorithms are not exact and hence are not guaranteed to generate the optimal solutions.

Indeed, the comparison between the number of covers created and its upper bound (*ub*) confirms the approximate nature of the algorithm. However, this comparison also shows the two algorithms are quite effective in generating good solutions to the problem. Based on the results reported in the table, the average gap between the number of covers created using BSF and the upper bound (*ub*) is 3.1% and the average gap between the number of covers created using RBSF and the upper bound (*ub*) is 1.6%. Again, this is an indication of the good performance of RBSF and to a lesser extend BSF. In addition, one should keep in my mind that *ub* is only an upper bound on the optimal number of covers and there is no measure that can indicate exactly how close or far these two values are.The statistical paired t-tests of the numbers of covers and the total bottleneck energy of the covers obtained using the two algorithms confirm the superiority of RBSF over BSF at 99% significance level. However, this superiority seems to be more significant for the smaller networks.

Table 3. Computational results obtained for different numbers of targets ($|T|$)

					BSF algorithm				RBSF algorithm			
$\|T\|$	$\|S\|$	ρ_t	ρ_s	ub	# of Covers Created	Bottleneck Energy	# of Active Sensors	CPU Time	# of Covers Created	Bottleneck Energy	# of Active Sensors	CPU Time
50	100	24.2	48.4	23.4	22.6	3212	80.0	0	23.2	3330	81.8	44
100	100	47.2	47.2	20.2	19.6	2779	79.2	0	20.0	2872	81.2	40
150	100	70.4	46.9	18.4	17.8	2488	77.8	0	18.2	2577	79.4	38
200	100	93.3	46.7	17.2	17.2	2381	76.4	0	17.2	2421	77.0	37
250	100	116.6	46.6	17.2	17.0	2346	79.0	0	17.2	2428	79.2	37
300	100	140.7	46.9	17.2	16.4	2262	75.6	0	17.2	2377	78.0	36
350	100	164.6	47.0	17.2	17.0	2370	78.0	0	17.2	2452	78.4	36
400	100	188.9	47.2	17.2	16.8	2314	79.8	0	17.2	2397	80.2	36
450	100	212.1	47.1	17.2	16.8	2336	79.6	0	17.2	2421	80.2	37
500	100	235.9	47.2	17.2	17.0	2334	79.6	0	17.2	2389	80.2	37
50	300	42.1	252.4	167.2	159.6	23391	288.6	0	161.2	23654	292.2	230
100	300	84.6	253.7	160.0	154.2	22533	287.4	0	156.2	22786	292.0	223
150	300	126.7	253.5	155.6	149.6	21957	289.4	0	151.8	22262	292.2	218
200	300	169.6	254.4	155.2	151.2	22137	290.4	0	152.8	22385	293.2	219
250	300	211.4	253.7	155.0	148.8	21915	290.4	0	150.2	22120	293.4	216
300	300	253.6	253.6	155.0	148.8	21774	290.8	0	150.8	22053	293.8	217
350	300	295.6	253.4	153.0	146.8	21428	287.6	0	148.4	21644	289.8	213
400	300	337.7	253.3	152.6	145.4	21265	288.6	0	146.8	21499	289.4	215
450	300	379.7	253.1	152.6	145.4	21524	285.4	0	147.4	21788	288.4	222
500	300	421.9	253.1	151.0	144.2	21545	288.4	0	145.8	21765	291.2	237

4.5. Computational results with different numbers of sensors

Table 4 shows the effects of changes in the number of sensors. As in [4], the study considered two settings for the number of targets and sensing range of each. The first involves networks with 100 targets and a sensing range of *r*=250. The second involves networks with 500 targets and a sensing range of *r*=400. Table 4 shows the characteristics of the problem instances for different numbers of sensors, varying from 50 to 500. These characteristics are similar to those observed in [4]. Note that although ρ_t remains almost unchanged, ρ_s and ub increase with the number of sensors.

The table shows the effects of changes in the number of sensors on the number of created covers and their total bottleneck energy. An increase in the number of sensors in the networks results in the increase in the number of covers created and their total bottleneck energy. Table 4 shows that RBSF outperforms BSF in terms of the obtained number of covers and the total bottleneck energy. Indeed, the number of covers obtained using RBSF is on average 1.53% higher than the number of covers obtained using BSF. Similarly, the total bottleneck energy of the covers obtained using RBSF is on average 1.95% higher than the total bottleneck energy of the covers obtained using BSF. Thus, RBSF produces solutions that extend the network lifetime much longer than those produced by BSF. The statistical paired t-tests of the numbers of covers and the total bottleneck energy of the covers obtained using the two algorithms confirms the superiority of RBSF over BSF at 99% significance level. However, this superiority seems to be more significant for the smaller networks.

Table 4. Computational results obtained for different numbers of sensors ($|S|$)

					BSF algorithm				RBSF algorithm			
$\|T\|$	$\|S\|$	ρ_t	ρ_s	ub	# of Covers Created	Bottleneck Energy	# of Active Sensors	CPU Time	# of Covers Created	Bottleneck Energy	# of Active Sensors	CPU Time
100	50	47.1	23.6	8.8	8.4	1209	33.4	0	8.6	1254	34.6	25
100	100	47.2	47.2	20.2	19.6	2779	79.2	0	20.0	2872	81.2	40
100	150	47.3	70.9	28.4	28.2	3931	119.4	0	28.4	4027	119.6	52
100	200	47.1	94.2	39.0	38.4	5404	159.0	0	39.0	5542	161.4	66
100	250	47.2	118.1	52.0	51.4	7103	211.2	0	52.0	7253	212.6	84
100	300	47.9	143.6	67.2	64.8	9008	266.4	0	67.2	9351	272.4	103
100	350	48.5	169.9	77.2	75.6	10642	303.6	0	77.0	10908	307.0	117
100	400	49.0	196.0	92.2	90.2	12670	353.4	0	91.4	12881	356.8	137
100	450	48.3	217.5	106.0	102.0	14142	408.8	0	102.6	14301	412.0	152
100	500	48.0	239.8	116.8	114.8	16029	456.0	0	116.0	16240	461.4	170
500	50	418.4	41.8	20.4	19.4	2794	43.8	0	19.8	2853	43.6	40
500	100	419.3	83.9	43.0	41.4	6084	89.0	0	42.6	6277	90.2	70
500	150	422.4	126.7	70.2	68.8	10124	138.8	0	70.2	10323	140.6	107
500	200	421.9	168.8	98.2	94.0	13693	187.4	0	95.4	13917	189.4	141
500	250	422.0	211.0	126.0	121.8	18016	241.2	0	123.4	18265	242.8	191
500	300	421.9	253.1	151.0	144.2	21545	288.4	0	145.8	21765	291.2	235
500	350	422.6	295.8	176.6	170.8	25295	338.4	0	172.8	25596	341.6	301
500	400	423.5	338.8	204.2	196.8	29041	386.8	1	198.6	29284	390.0	399
500	450	423.0	380.7	233.8	225.2	32976	440.6	1	226.6	33175	442.4	500
500	500	423.3	423.3	261.2	250.6	36666	485.8	1	252.2	36861	489.8	604

The results reported in table 4 show also the performance of the two algorithms based on the comparison between the number of created covers and its upper bound (*ub*). Based on the results, the average gap between the number of covers created using BSF and the upper bound (*ub*) is 3.1% and the average gap between the number of covers created using RBSF and the upper bound (*ub*) is 1.6%. This is an indication for the good performance of RBSF and to a lesser extend BSF.

5. CONCLUSION

The problem of maximizing lifetime of a wireless sensor network remains an important problem. In this paper, we adopt the common approach of creating disjoint sensor covers to prolong network lifetime. Based on this approach, at any time of the network lifespan, only one cover is in active mode providing network functionality, and the other sensor covers are inactive to save energy. The typical goal used in the literature is to maximize the number of covers with no consideration of the energy levels of the sensors. We argue that the network lifetime can be extended further by maximizing the total bottleneck energy of the created covers because once the sensor with the bottleneck energy in a cover runs out of energy, the cover looses full coverage of the target area regardless of the energy levels of the other sensors. In this paper, the maximization of the lifetime of a wireless sensor network is guided by two priorities: The first is to maximize the total bottleneck energy of the covers and the second is to minimize the number of sensors included in the covers. The combination of these guiding priorities would create as many covers as possible with the maximum overall bottleneck energy of the covers, resulting in a longer network lifetime.

We formally defined the problem of maximizing the total bottleneck energy of the covers and presented for the first time an integer programming formulation of the problem. Because of the NP-hard nature of the problem, the formulation can be used to solve only small instances of the problem optimally. Consequently, we have developed two algorithms to solve the problem: (i) the Best-Sensor-Fit algorithm (BSF) and (ii) the Randomized Best-Sensor-Fit algorithm (RBSF).

Extensive experimental tests on large wireless sensor networks show that the use of the goal of maximizing the total bottleneck energy of the covers creates covers with substantially longer network lifetime than the lifetime of the covers created with the goal of maximizing solely the number of covers. The results of the tests also show that the RBSF algorithm outperforms the BSF algorithm in key performances metrics, namely the network lifetime and the number of created covers.

REFERENCES

[1] B. Behdani, Y. S. Yun, J. Cole Smith, Y. Xia, "Decomposition algorithms for maximizing the lifetime of wireless sensor networks with mobile sinks", Computers & Operations Research, vol. 39, no. 5, pp. 1054-1061 (2012).

[2] Q. Cao, T.F. Abdelzaher, T. He, J.A. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection", Proc. Int'l Conf. Information Processing in Sensor Networks (IPSN), pp. 20-27 (2005).

[3] M. Cardei, D.Z. Du, "Improving wireless sensor network lifetime through power aware organization", Wireless Networks, vol. 11, no. 3, pp. 333–340 (2005).

[4] R. Cerulli, R. De Donato, A. Raiconi, "Exact and heuristic methods to maximize network lifetime in wireless sensor networks with adjustable sensing ranges", European Journal of Operational Research, vol. 220, no. 1, pp. 58-66 (2012).

[5] D. Chu, A. Deshpande, J.M. Hellerstein, W. Hong, "Approximate data collection in sensor networks using probabilistic models", Proc. 22nd Int'l Conf. Data Eng. (ICDE) pp. 48-59 (2006).

[6] M. Haouari, J. S. Chaouachi, "A probabilistic greedy search algorithm for combinatorial optimisation with application to the set covering problem", Journal of the Operational Research Society, vol. 53, no. 7, pp. 792-799 (2002).

[7] C. C. Hung, W. C. Peng, W. C. Lee, "Energy-aware set-covering approaches for approximate data collection in wireless sensor networks", IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 11, pp. 1993-2007 (2012).

[8] C. C. Hung, W. C. Peng, "Optimizing in-network aggregate queries in wireless sensor networks for energy saving", Data and Knowledge Engineering, vol. 70, no. 7, pp. 617-641 (2011).

[9] Y. Kotidis, "Snapshot queries: towards data-centric sensor networks", Proc. 21st Int'l Conf. Data Eng. (ICDE), pp. 131-142 (2005).

[10] C. C. Lai, C.K. Ting, R.S. Ko, "An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications", Proceedings of the 2007 Congress on Evolutionary Computation, pp. 3531-3538 (2007).

[11] W. K. Lai, C. S. Fan, L. Y. Lin, "Arranging cluster sizes and transmission ranges for wireless sensor networks", Information Sciences, vol. 183, no. 1, pp. 117-131 (2012).

[12] X. Li, Y. P. Aneja , "A Branch-and-Cut Approach for the Minimum-Energy Broadcasting Problem in Wireless Networks", INFORMS Journal on Computing, vol. 24, no. 3, pp. 443–456 (2012).

[13] C. Liu, K. Wu, J. Pei, "An energy-efficient data collection framework for wireless sensor Networks by exploiting spatiotemporal correlation", IEEE Transactions on Parallel and Distributed System, vol. 18, no. 7, pp. 1010-1023, 2007.
[14] G. Lu, N. Sadagopan, B. Krishnamachari, A. Goel, "Delay efficient sleep scheduling in wireless sensor networks", Proc. INFOCOM, pp. 2470-2481 (2005).
[15] A. Oztekin, F. M. Pajouh, D. Delen, L. K. Swim, "An RFID network design methodology for asset tracking in healthcare", Decision Support Systems, vol. 49, pp. 100–109 (2010).
[16] A. Pietrabissa, C. Poli, D. G. Ferriero, M. Grigioni, "Optimal planning of sensor networks for asset tracking in hospital environments", Decision Support Systems, vol. 55 pp. 304–313 (2013).
[17] V. Raghunathan, C. Schurgers, S. Park, M.B. Srivastava, "Energy-aware wireless microsensor networks", IEEE Signal Processing , vol. 19, no. 2, pp. 40–50 (2002).
[18] A. Rossi, A. Singh, M. Sevaux, "An exact approach for maximizing the lifetime of sensor networks with adjustable sensing ranges", Computers & Operations Research, vol. 39, no. 1, pp. 3166-3176 (2012).
[19] C. Schurgers, V. Tsiatsis, S. Ganeriwal, M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space", IEEE Transactions on Mobile Computing, vol. 1, no. 1, pp. 70–80 (2002).
[20] S. Slijepcevic, M. Potkonjak, "Power efficient organization of wireless sensor networks", Proceedings of the IEEE international conference on communications, vol. 2 pp. 472–476 (2001).
[21] J.A. Stine, G.D. Veciana, "Improving energy efficiency of centrally controlled wireless data networks", Wireless Networks, vol. 8, no. 6, pp. 681–700 (2002).
[22] X. Tang, J. Xu, "Adaptive data collection strategies for lifetime-constrained wireless sensor networks", IEEE Transactions on Parallel and Distributed Systems, vol. 19, no. 6, pp. 721-734 (2008).
[23] X. Tang, J. Xu, "Extending network lifetime for precision-constrained data aggregation in wireless sensor networks", Proc. INFOCOM, pp. 1-12 (2006).
[24] C. K. Ting, C. C. Liao, "A memetic algorithm for extending wireless sensor network lifetime", Information Sciences, vol. 180, no. 24, pp. 4818-483 (2010).
[25] H. Y. Yang, C. H. Lin, M. J. Tsai, "Distributed algorithm for efficient construction and maintenance of connected k-hop dominating sets in mobile Ad Hoc networks", IEEE Transactions on Mobile Computing, vol.7, no. 4, pp. 444-457 (2008).
[26] H. Zhang, J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks", Ad Hoc & Sensor Wireless Networks, vol. 1, pp. 89–124 (2005).

Biography

Ali Amiri received the BS degree in business administration from The Institut des Hautes Etudes Commerciales, Tunisia, in 1985, the MBA and PhD degrees in Management Science/Information Systems from The Ohio State University, Columbus, OH, in 1988 and 1992, respectively. He is a Professor of Management Science and Information Systems at Oklahoma State University. His research interests include data communications, electronic commerce, data mining, and database management. His papers have appeared in a variety of journals including the IEEE Transactions On Communications, European Journal of Operational Research, Computers and Operations Research, INFORMS Journal on Computing, Decision Support Systems, ACM Transactions on Internet Technology, Information Sciences, and Naval Research Logistics.