# EFFICIENT ENERGY MANAGEMENT IN SENSOR NETWORKS

P. BERMAN*, G. CALINESCU†, C. SHAH‡, AND A. ZELIKOVSKY‡

**Abstract.**
Optimizing the energy consumption in wireless sensor networks has recently become the most important performance objective. We assume the sensor network model in which sensors can interchange idle and active modes. Given monitoring regions, battery life and energy consumption rate for each of $n$ sensors, we formulate the problem of maximizing sensor network lifetime, i.e., time during which the monitored area is (partially or fully) covered. We give the first provably good algorithm for finding monitoring schedule with the approximation ratio $1 + logn$.

Our contributions also include (1) an efficient data structure to represent the monitored area with at most $n^2$ points guaranteeing the full coverage which is superior to the previously used approach based on grid points, (2) efficient provably good centralized algorithms for sensor monitoring schedule maximizing the total lifetime for the case when a $q$-portion of the monitored area is required to cover, e.g., for the 90% area coverage our schedule guarantees to be at most 3.3 times shorter than the best full coverage lifetime, (3) efficient provably good approximation algorithm for sensor network lifetime problem which takes in account the (partial) monitoring and communication cost in case when the communication range is at least twice larger than monitoring range, (4) a family of efficient distributed protocols with trade-off between communication and monitoring power consumption, (5) extensive experimental study of the proposed algorithms showing significant advantage in quality, scalability and flexibility.

**Key words.** Sensor networks, energy efficient scheduling, packing linear programs.

**1. Introduction.** Wireless sensor networks have been the focus of considerable research during the past few years. The research issues currently addressed in wireless sensor networks are hardware constraints, communication and routing issues, data management problems, and software engineering principles. One of the most important issue apart from the above mentioned ones is energy optimization in wireless sensor networks.

A sensor network is composed of a large number of sensor nodes that are densely deployed either inside the environment or close to it. The position of sensor nodes need not be engineered or predetermined. This allows random deployment in inaccessible terrains or hazardous environments. Some of the most important application areas of sensor networks include military, natural calamities, health, and home. When compared to traditional ad hoc networks, the most noticeable point about sensor networks is that, they are limited in power, computational capacities, and memory. Hence optimizing the energy consumption in wireless sensor networks has recently become the most important performance objective.

The wireless sensor node, being a microelectronic device, can only be equipped with a limited power source. In some application scenarios, replenishment of power resources might be impossible. As noted in [3], the energy density of batteries has only doubled every 5 to 20 years, depending on the particular chemistry, and prolonged refinement of any chemistry yields diminishing returns. This shows that power management will be as critical in future sensor networks as it now.

The main task of a sensor node in a sensor network is to monitor events, i.e.,

*Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802-6106, E-mail: berman@cse.psu.edu

†Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616. E-mail: calinesc@cs.iit.edu

‡Department of Computer Science, Georgia State University, Atlanta, GA 30303. E-mail: chintan@alla.cs.gsu.edu, alexz@cs.gsu.edu

collect data, perform quick local data aggregation, and then transmit the data. Power consumption can hence be divided into three domains: sensing, aggregation, and communication. In this paper we mostly concentrate on minimizing energy for sensing by using smart monitoring schedules. The energy for data aggregation is practically not affected by monitoring schedules but for distributed, self-organizing schedules it is necessary to take in account associated energy loss due to increase in communication for establishing better schedules [9].

Our model of a sensor network is close to one described in [8, 9]. We assume that sensors are sprayed over the region $R$ which is required to monitor, and each sensor $p_i$ has its own *monitored region* $R_i$ which it *covers*, i.e., $p_i$ can collect the trustful data from $R_i$ without help of any other sensor. Although we assume that the monitored region are convex (for the clarity of presentation we will use disks of the same radius $r$), our algorithms can be generalized to arbitrary region shape. We also assume that each sensor $p_i$ has also a certain initial energy supply $b_i$ which will be measured in time during which $p_i$ can collect information from $R_i$.

We also assume that the number of sensors largely exceeds the amount necessary to monitor the required region $R$. Therefore, it is possible to turn some sensors in the idle mode saving their energy and prolonging the network lifetime. Sensors can interchange their mode multiple times and possible energy loss can be taken in account. The main constraint is that the monitored region $R$ should be completely (or partially with specified portion of $R$) covered at any moment by active sensors. This assumption agrees with [7], where an OS-directed power management technique to improve the power efficiency of sensor nodes is proposed.

Below we give a formal definition of the energy preserving scheduling problem.

A set of sensors $C$ covering $R$ will be called *sensor cover*. Then a *monitoring schedule* is the set of pairs $(C_1, t_1), \ldots, (C_k, t_k)$, where $C_i$ is a sensor cover and $t_i$ is time during which $C_i$ is active.

**Sensor Network Life Problem (SNLP)**. Given a monitored region $R$, a set of sensors $p_1, \ldots, p_n$ and monitored region $R_i$ and energy supply $b_i$ for each sensor, find a monitoring schedule $(C_1, t_1), \ldots, (C_k, t_k)$ with the maximum length $t_1 + \ldots + t_k$, such that for any sensor $p_i$ the total active time does not exceed $b_i$.

Note that this formulation has never been clearly stated in the previous literature. In [8] (see also [1, 5]) the problem has been reduced to so called *disjoint set cover problem*. There it is assumed that any sensor cannot participate in different sensor covers, i.e., it can be only once change its mode from sleeping to active. Then all sensor covers should be disjoint. A different model assumes that the most of energy consumption of wireless networks comes from routing the traffic, rather than monitoring [10] and the task is to find the best traffic flow routes for a given set of traffic demands, e.g., using concurrent flow approaches [4].

The advantage of the introduced formulation is unbounded. Indeed, if one sensor has $n$ units of energy while $n$ other sensors have 1 unit of energy and any 2 sensors can monitor the region, then using the same sensor in different sensor covers increases the lifetime by factor $n$. On the other hand, it is much more complicated to find the maximum possible advantage of multiple mode interchange over disjoint sensor cover when the original energy supply is the same for all sensors. Figure 1.1 gives an example showing advantage of multiple mode interchange in case of the same initial energy supply. In this example there are 3 sensors with disc monitored regions which are supposed to monitor a dark square region $R$ inside. Assume that each sensor has

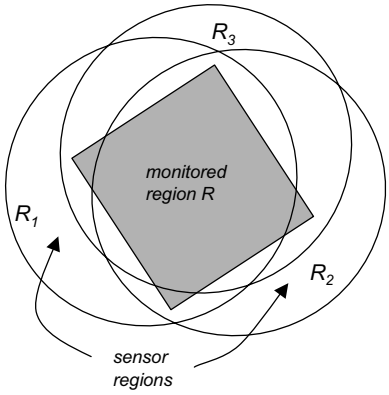


FIG. 1.1. *3 sensors with the disc monitored regions $R_1$, $R_2$ and $R_3$ cover the dark square of the monitored region R. Any two sensors can cover R but any single sensor cannot cover R.*

2 units of energy supply. Then there exists only a single disjoint sensor cover which can last for 2 units of time. On the other hand, the schedule $(\{p_1, p_2\}, 1)$, $(\{p_2, p_3\}, 1)$, $(\{p_3, p_1\}, 1)$ is clearly feasible and lasts for 3 units of time. We conjecture that for the case when the monitored regions are convex this advantage cannot exceed 50%, i.e., Figure 1.1 gives the worst-case example, and, in general case this advantage cannot exceed 100%.[1]

The SNLP can be formulated as a *packing* linear program (see section 3) – one should pack maximum number of schedules into a multidimensional knapsack with dimensions defined by the number of energy units assigned to sensors. For solving SNLP we use the *primal-dual* approach. This approach requires to solve the following dual *covering* problem.

**Minimum Weight Sensor Cover Problem (MWSCP)**. Given a monitored region $R$, a set of sensors $p_1, \ldots, p_n$ and monitored region $R_i$ and the weight $w_i$ for each sensor, find sensor cover with the minimum total weight.

Using approximate solutions for MWSCP we give first provably good approximation algorithms for SNLP based on approximation algorithm for MWCP. Other contributions of our paper include:

- an efficient data structure to represent the monitored area with at most $n^2$ points guaranteeing the full coverage which is superior to the previously used approach based on grid points,
- efficient provably good centralized algorithms for sensor monitoring schedule maximizing the total lifetime for the case when a $q$-portion of the monitored area is required to cover
- efficient provably good approximation algorithm for sensor network lifetime problem which takes in account the (partial) monitoring and communication cost in case when the communication range is at least twice larger than monitoring range,

[1] We have constructed a series of instances with the limit advantage of 100% based on involved combinatorics.

- a family of efficient distributed protocols with trade-off between communication and monitoring power consumption
- extensive experimental study of the proposed algorithms showing significant advantage in quality, scalability and flexibility.

The rest of the paper is organized as follows.

In Section 2 we give an efficient data structure for representation of the monitored area with at most $n^2$ points guaranteeing the full coverage which is superior to the previously used approach based on grid points [8]. This data structure allows efficiently reduce the Minimum Weight Sensor Cover problem to the standard weighted set cover problem. This problem cannot be efficiently solved with guarantee better than $O(\log k)$, where $k$ is the size of the largest set and it is not known to have any better algorithm for the case of planar sets. Anyway, our primal-dual solution is the first algorithm guaranteeing solution which is at most $O(logn)$ worse than the optimum, where $n$ is the number of sensors.

In Section 3 we explain Garg-Könemann algorithm [2] for solving packing linear programs, which allows to solve the Maximum Sensor Network Life problem with almost the same guarantee with which one can solve the dual problem, i.e., Minimum Weight Sensor Cover problem. We also give a practical enhancement of their approach resulted in fast improvement of the quality of obtained solutions.

In many cases, it can be required to cover sufficiently large portion of the monitored region $R$ rather than the entire $R$. In Section 3.1 we formulate the respective problem when only $q \cdot area(R)$ for a given $q \in [0, 1]$. We then give an approximation algorithm which finds the solution at most $(1 + \ln(1 - q)^{-1})$ times heavier than the optimum full coverage. For instance, when $q = 0.9$, i.e., when we wish to cover 90% of the region $R$, this implies an algorithm guaranteeing finding sensor 0.9-cover at most 3.3 times heavier than the the optimum full cover and a monitoring schedule always covering at least 90% of monitored region and at most 3.3 times shorter than the optimal sensor schedule.

In Section 3.2, we show how to take in account communication costs – if the communication radius is at least twice the monitoring radius then we can guarantee constant-factor approximation.

In Section 4, we present a series of distributed algorithms based on our new data structure which are superior to the previous approaches (see [9]) since they account for each node energy supply. We have implemented all the suggested algorithms and present our simulation results in Section 5.

**2. Efficient Data Structure Representing Sensor Coverage.** In [8] a grid data structure is used to express sensor node coverage over an area. They establish a set of grid points that form a $g \times g$-array to discretize the area. Then they determine for each point the subset of covering sensors disks. Next they partition all grid points into *fields*, where a field is defined as a subset of grid points covered by the same set of sensors. Similar idea has been suggested in [3]. The main advantage of this coverage model is ease of implementation if the covering area does not have to be delineated very precisely. If there are few points in the grid, then the area to be covered is not well defined. On the other hand, if the grid is very fine, then the calculation overhead for this model becomes quite large. Another challenging computation task is partitioning into fields since potentially $O(2^{n \times n})$ possible fields may exist only fields which are present should be extracted.

In this paper we present a superior data structure which overcomes the above disadvantages. The monitored region is represented by a planar graph $P = (V, E)$ with

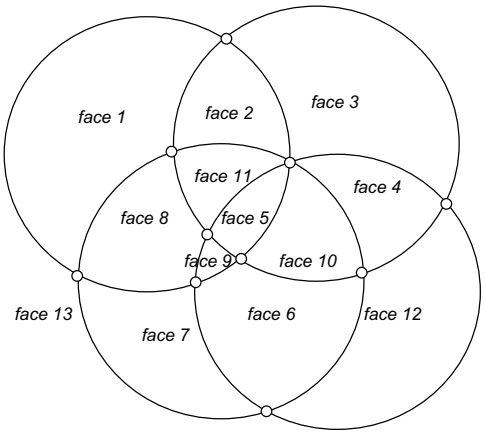


FIG. 2.1. *The data structure for four sensors with disk monitored regions. The planar graph $P = (V, E)$ with vertices corresponding to points of intersections and edges connecting adjacent points. There are 10 vertices, 21 edges and 13 faces.*

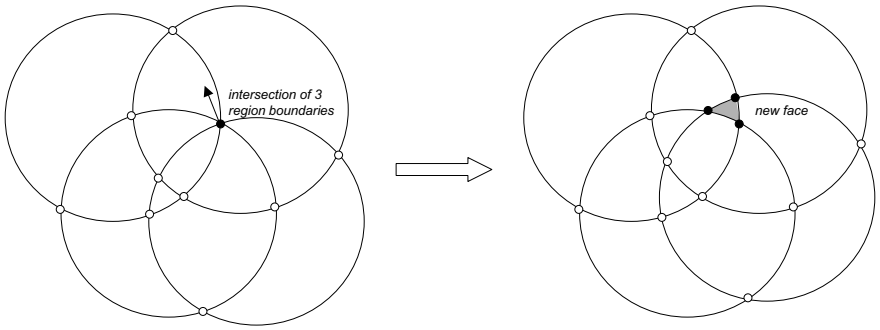


FIG. 2.2. *The left example contains a triple intersection (the vertex is filled). On the right, there is a modified example with one circle slightly moved up. As a result a new dotted face appears.*

vertices $V$ corresponding to the points of intersections of boundaries of all sensor's coverage regions (e.g., circles) and edges $E$ connect pairs of adjacent intersection points along the boundaries of sensor circles (see Figure 2.1).

It is easy to see that the *faces* of the graph $P$, i.e., the parts of the plane bounded by edges, are regions covered by the same set of sensor disks. Therefore, if we would identify all the faces of the graph $P$ we would effectively enumerate all fields and moreover, this will be a complete and accurate representation of *all* fields since we do not use grid points and do not rely on an assumption that we have sufficient amount of grid points to nail each face.

One can efficiently find all faces since the number of such faces is comparatively small. Indeed, the following simple fact bounds the number of faces.

THEOREM 2.1. *Let $m$ be the number of points of intersection of boundaries of the monitored regions of all sensors, then the number of faces in the graph $P$ is at most $m + 2$.*

*Proof.* The proof is based on Euler's formula for planar graphs. Since $P = (V, E)$

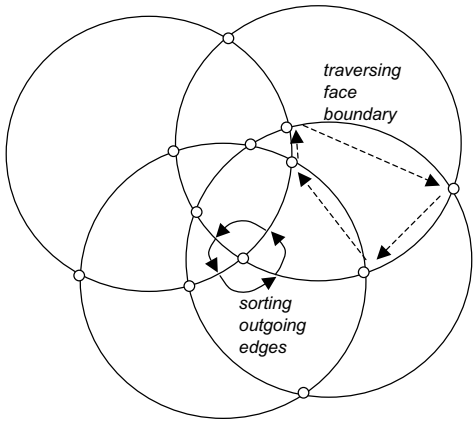


FIG. 2.3. *Solid arcs show how the edges outgoing from the filled vertex are traversed, the dotted arcs show the order in which a face boundary is traversed.*

is a planar graph, $|V| - |E| + |F| = 2$, where $V$, $E$ and $F$ are the sets of vertices, edges and faces of $P$, respectively. If several boundaries intersect at the same point, then by slightly changing boundaries, one can increase the number of faces (see Figure 2.2). Thus, for counting purposes, we can assume that each point can be intersection of boundaries of at most two monitored regions. Then each vertex of the graph $P$ has degree 4. If we sum up degrees of all vertices, then we count each edge twice, therefore, $|E| = 2|V|$. Thus, $|V| - 2|V| + |F| = 2$ and the number of faces equals $|F| = |V| + 2$. □

If the monitored regions are convex, then any two boundaries can intersect at most twice, i.e., there are at most $n(n-1)$ intersection points.

COROLLARY 2.2. *Given $n$ sensors each with convex monitored region, the number of faces of the graph $P$ is at most $n(n-1)+2$.*

Below we describe an efficient implementation of our data structure when each monitored region is a disk (or in general a convex region) (see Figure 2.3). The graph $P$ is considered to be directed with each edge represented by two opposite arcs and each arc belongs to the boundary of exactly one face. For each vertex all outgoing arcs are sorted in a counterclockwise order. The faces are identified by walking along the arcs, if we come to the node using arc $e$, then we should follow the arc next to the next arc which opposite to $e$. Finally, determining which sensors cover which faces is accomplished by measuring the distances of the face vertices to the sensors.

If monitored regions are not convex then we suggest to partition them into convex subregions and apply the planar graph data structure afterwards. This may happened in case of obstacles when collecting visual information (see Figure 2.4). If the monitored regions are given implicitly, then finding intersection points may be a challenging task and we may need to reuse the grid point representation. In this case, finding fields can be done efficiently using hash functions.

**3. Centralized Maximization of Sensor Network Lifetime.** In this section we formulate the Sensor Network Lifetime problem as a packing linear program, give the $(1+\log n)$-approximation algorithm for SNLP, show how to modify our algorithm in case when only partial coverage is required and, finally show how to take in account

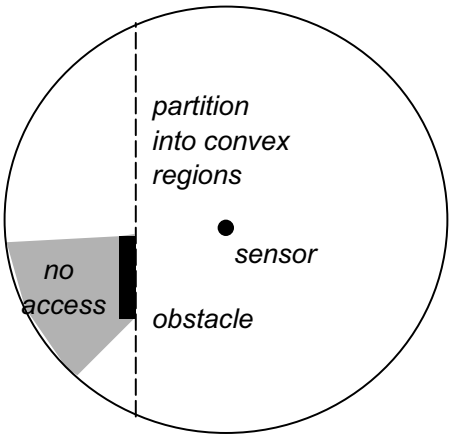


FIG. 2.4. *An example of a sensor with nonconvex monitored region caused by a solid obstacle. The region is partitioned by a dashed line into the convex regions.*

communication cost.

After finding different sensor covers which satisfies sensor network constraint, we have to maximize the network lifetime by assigning the active time stretch for each sensor cover. Formally, for each sensor cover $c_j$ from a family $SC = \{c_1, \ldots c_m\}$, we need to find the time variable $t_j$. We can generate a matrix $C_{ij}$ with rows $i = 1, \ldots, m$ representing each sensor, and columns $j = 1, \ldots, m$ representing each sensor cover. The linear program corresponding to SNLP can be formulated as follows.

$$\text{Maximize :} \qquad \sum_{j=1}^{m} t_j$$

$$\text{Subject to} \qquad \sum_{j=1}^{m} C_{ij} t_j \leq b_i$$

where $b_i$ is energy supply of the sensor $i$ and

$$C_{ij} = \begin{cases} 0 & \text{if sensor i is not in set cover j} \\ 1 & \text{if sensor i is in disk cover j} \end{cases}$$

The linear program above is a packing LP. In general, a packing LP is defined as

$$\max\{c^T x | Ax \leq b, x \geq 0\} \tag{3.1}$$

where $A, b$, and $c$ have positive entries; we denote the dimensions of $A$ as $m$x$n$. In our case the number of columns of $A$ is prohibitively large (exponential in number of sensors) and we will use the $(1+\epsilon)$-approximation Garg-Könemann algorithm [2]. The algorithm assumes that the LP is implicitly given by a vector $b \in R^m$ and an algorithm which finds the column of $A$ minimizing so-called length. The *length* of column $j$ with respect to LP in Equation (3.1) is defined as $length_y(j) = \frac{\Sigma_i A(i,j) y(i)}{c(j)}$ for any

**Input:** A vector $b \in R^m$, $\epsilon > 0$, and an $f$-approximation algorithm $F$ for the problem of finding the minimum length column $A_{q(y)}$ of a packing LP $\{\max\ c^T x | Ax \le b, x \ge 0\}$
**Output:** A set of columns $\{A^j\}_{j=1}^k$ each supplied with the value of the corresponding variable $x^j$, such that $(x^1, \ldots, x^k)$ correspond to all non-zero variables in a near-optimal feasible solution of the packing LP $\{\max\ c^T x | Ax \le b, x \ge 0\}$

(1) Initialize: $\delta = (1+\epsilon)((1+\epsilon)m)^{-1/\epsilon}$, for $i = 1, \ldots, m$ $y(i) \leftarrow \frac{\delta}{b(i)}$, $\quad D \leftarrow m\delta$, $j = 0$
(2) While $D < 1$
Find the column $A_q$ using the $f$-approximation $F$.
Compute $p$, the index of the row with the minimum $\frac{b(i)}{A_q(i)}$
$j \leftarrow j+1$, $\quad x^j \leftarrow \frac{b(p)}{A_q(p)}$, $\quad A^j \leftarrow A_q$
For $i = 1, \ldots, m$, $y(i) \leftarrow y(i)\left(1 + \epsilon \frac{b(p)}{A_q(p)} / \frac{b(i)}{A_q(i)}\right)$, $\quad D \leftarrow b^T y$.
(3) Output $\{(A^j, \frac{x^j}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}})\}_{j=1}^k$

FIG. 3.1. *The Garg-Könemann Algorithm with $f$-approximate minimum length columns*

positive vector $y$. The Garg-Könemann algorithm with $f$-approximate minimum length columns is presented in Figure 3.1.

When applied to our LP (3.1), the Garg-Könemann algorithm can use an (approximation) algorithm $F$ solving the Minimum Weight Sensor Cover problem with weights proportional to the elements of vector $y$, i.e., for each node $i = 1, \ldots, m$, $w(i) = 1/y_i$. This implies the following general result.

THEOREM 3.1. *The Network Lifetime problem can be approximated within a factor of $(1+\epsilon)f$, for any $\epsilon > 0$, using Algorithm on Figure 3.1 with $f$ being the approximation ratio of the algorithm $F$ for the Minimum Weight Sensor Cover problem.*

From section 2 follows that the Minimum Weight Sensor Cover problem is equivalent to the classical Minimum Weight Set Cover problem with $k = O(n^2)$ elements (i.e., faces) to cover. Using standard greedy algorithm for this problem with the approximation ratio $1 + \ln k = 1 + 2\ln n$, we obtain the following

COROLLARY 3.2. *The Network Lifetime problem can be approximated within a factor of $(1+\epsilon)(1+2\ln n)$, for any $\epsilon > 0$, using Algorithm on Figure 3.1 with the algorithm $F$ being the greedy $(1+2\ln n)$-approximation algorithm for the Minimum Weight Sensor Cover problem.*

**3.1. Partial Sensor Coverage .** In this subsection we consider the problem of finding the minimum weighted sensor set partially covering the monitored area.

**Partial $q$-Coverage problem**. Given a constant $q \in [0,1]$, the monitored region $R$ with area $M$ and a set of sensors $Sensors$ find subset $p_1, p_2, \ldots, p_t$ of $Sensors$ such that

$$\text{Minimize :} \qquad \sum w(p_i)$$

$$\text{Subject to} \qquad Area\left(\bigcup_{i=0}^{t} S_i\right) \ge qM,$$

where $M$ is the total monitored area and $S_i$ is the monitored region of sensor $p_i$.

This problem can be reformulated as follows.

**Set $q$-Cover problem**. Given a finite set $X$ of elements each having cost[2] $cost : X \to R^+$ and family of $\mathcal{F}$ of subsets $S_i \subseteq X$, each having weight $w : S_i \to R^+$, and $q \in [0, 1]$, find minimum weight subfamily $\mathcal{C} \subseteq \mathcal{F}$ covering subsets of $X$, such that total cost of elements covered by sets in $\mathcal{C}$ is at least $q \cdot cost(X)$, where $cost(X) = \sum_{x \in X} x$.

The greedy algorithm for Set $q$-Cover problem iteratively chooses the set $S_i \in \mathcal{C}$, with minimum $\left(\frac{w(S_i)}{cost(S_i)}\right)$, where $cost(S_i) = \sum_{x \in S_i} cost(x)$. It stops when the area covered is at least $q \cdot cost(X)$. Due to space limitations we omit the proof of the following

THEOREM 3.3. *The greedy algorithm for Set q-Cover problem finds a cover with weight at most* $1 + \ln \frac{1}{1-q}$ *times the weight of optimal cover of the entire* $X$.

*Proof.* Let $X_i$ be the set of elements of $X$ which are uncovered after selecting set $S_1, S_2, \ldots, S_i$, and cost of $X_i$, $cost : S_i \to R^+$ is total cost of elements of $X_i$. The $cost(X_0)$ is cost of elements of $X$ before selecting any set $S_i$, so that $cost(X_0) = cost(X)$ and,

$$cost(X_i) = cost(X_{i-1}) - cost(S_i) \tag{3.2}$$

Let $S_1^*, S_2^*, \ldots, S_k^*$ be sets selected in optimal solution covering entire $X$, so that,

$$\begin{aligned} \frac{OPT}{cost(S_i)} &\geq \frac{\sum w(S_j^*)}{\sum cost(S_j^*)} \\ &\geq min\left(\frac{w(S_j)}{cost(S_j)}\right) \\ &\geq \frac{w(S_i)}{cost(S_i)} \end{aligned} \tag{3.3}$$

From (3.2),

$$cost(X_i) \leq cost(X_{i-1})\left(1 - \frac{w(S_i)}{OPT}\right)$$

Let m be the maximum number such that remaining cost $cost(X_m)$ is at least $(1 - q)cost(X)$. So that,

$$cost(X_{m+1}) < (1 - q)cost(X) \leq cost(X_m) \tag{3.4}$$

From inequality $\ln(1 + x) \leq x$,

$$\begin{aligned} \ln \frac{cost(X_0)}{cost(X_m)} &\geq -\ln \prod \left(1 - \frac{w(S_i)}{OPT}\right) \\ &= -\sum \ln \left(1 - \frac{w(S_i)}{OPT}\right) \\ &\geq \frac{\sum w(S_i)}{OPT} \end{aligned} \tag{3.5}$$

[2]The cost of an element corresponds to the area of a face.

The cost of solution by Greedy method is equal to $\sum_{i=0}^{m+1} w(S_i)$, so using (3.4),(3.5) approximation ratio is,

$$\begin{aligned} APR &= \frac{\sum_{i=0}^{m} w(S_i) + (w(S_{m+1}))}{OPT} \\ &= \frac{\sum_{i=0}^{m} w(S_i)}{OPT} + \frac{w(S_{m+1})}{OPT} \\ &\leq \ln \frac{cost(X_0)}{cost(X_m)} + 1 \\ &\leq \ln \frac{1}{(1-q)} + 1 \end{aligned}$$

Because, from (3.3),

$$\frac{w(S_{m+1})}{OPT} \leq \frac{cost(S_{i+1})}{cost(X_{i+1})} \leq 1$$

□

COROLLARY 3.4. *When q-portion of the monitored region R is allowed to cover, the SNLP can be approximated within a factor of* $(1+\epsilon)(1+\ln(1-q)^{-1})$*, for any* $\epsilon > 0$*, using Algorithm on Figure 3.1 with the algorithm F being the greedy* $(1+\ln(1-q)^{-1})$*-approximation algorithm for the Minimum Weight Sensor q-Cover problem.*

**3.2. Taking in Account Communication Cost.** In this section we show that our approach for monitoring schedule can be generalized to take in account communication cost (assuming that the communication range of nodes is at least twice the monitoring range).

In this section we adopt a model where each sensor can be in one of the following four states: monitoring, relaying, linking to the base station, and idle. Each state has a different energy consumption per unit of time, which is respectively $C_{monitor}$, $C_{relay}$, $C_{link}$, and 0. In practice, $C_{relay} \leq C_{monitor} \leq C_{link}$, as the sensors necessarily relay their own data, and linking to the satellite uses the most energy.

While in monitoring schedule we need to specify only which nodes are active and which are inactive, here we also need to specify relaying nodes as well as nodes linked to the base. Formally, an *assignment triple* $Q = (M, R, L)$ consists of the following three sets of sensors: monitoring nodes $M = M(Q)$, relaying nodes $R = R(Q)$ and nodes linked to the base $L = L(Q)$. A triple is *feasible* if the set $M$ is a (partial) set cover, and each node in $M$ is either belong to $L$ (i.e., linked to the base) or is connected in the communication graph to at least one node in $L$ via nodes in $R$ (i.e., relaying nodes).

In this context, SNLP becomes the following packing linear program with exponentially many variables $t(Q)$ denoting the time stretch of a feasible assignment triple $Q$:

$$\text{Maximize :} \qquad \sum_{Q} t(Q)$$

$$\text{Subject to} \quad C_{monitor} \sum_{Q \mid i \in M(Q)} t(Q) + C_{relay} \sum_{Q \mid i \in R(Q)} t(Q) + C_{link} \sum_{Q \mid i \in L(Q)} t(Q) \leq b_i$$

where $Q = (M(Q), R(Q), L(Q))$ are feasible assignment triples and $b_i$ is the energy supply of node $i$.

In order to apply the Garg-Könemann algorithm, we need to (approximately) solve the dual problem formulated as follows:

**Weighted Linked Coverage Problem.** Given arbitrary weights $y_i$, find the feasible assignment triple $Q = (M, R, L)$ which minimizes $C_{monitor} \sum_{i \in M} y_i + C_{relay} \sum_{i \in R} y_i + C_{link} \sum_{i \in L} y_i$.

Below we give an approximation algorithm for Weighted Linked Coverage when $M$ is required to be a partial cover. A minor adaptation of the algorithm can be used for the case when $M$ is required to be a complete cover, having approximation ratio $O(\log n)$.

In the first stage, the algorithm computes an approximate partial cover $M$. Using the results from the Subsection 3.1 we deduce that $C_{monitor} \sum_{i \in M} y_i$ is within a constant $c_1$ of the optimum solution for full coverage. In the following, we construct an instance $G = (V, E, c, M')$ of the classical Steiner tree problem which asks for minimum cost tree in the graph $G = (V, E, c)$ which spans all given terminals from $M'$. The set of vertices $V$ coincides with the set of sensors plus one node $s$ corresponding to the base. The subset $M'$ of given terminals is defined as $M' = M \cup \{s\}$. We connect each sensor $i$ and the base $s$ with an edge of cost $y_i C_{link}$. We also connect each pair of sensors $i$ and $j$ which are able to communicate in one hop, the corresponding edge has cost $C_{relay}(y_i + y_j)$. An arbitrary constant-factor approximation algorithm for the Steiner Tree problem (see for example [6]) can be used for obtaining an approximate solution $T$ of this Steiner Tree instance. Then the feasible assignment triple $Q$ consists of $M$, the set of relay nodes $R$ coincides with the set of Steiner nodes in the tree $T$, and, finally, the set $L$ consists of nodes in $T$ adjacent to the base node $s$.

**4. Distributed Algorithms for Lifetime Maximization.** Distributed algorithms for SNLP have been previously considered in [9]. It has been shown that the sensor network lifetime can be substantially increased by using smart self-organizing monitoring schedules. Our approach below has the following advantages: (1) the superior monitored area representation, (2) dynamic accounting for energy supply of each sensor, (3) minimization of the set of active sensors.

We further assume that each sensor $s$ can communicate with all sensors sharing faces with $s$, otherwise, we can, e.g., increase for this purpose communication range. The data structure from Section 2 can be easily built in the distributed manner as follows: (1) each sensor broadcast to its neighbors its id and geographical position, (2) each sensor determines all the faces in its monitored area and associate with each face the id's of all sensors covering this face.

As pointed in [9], there are two main questions which should be answered by any self-organizing monitoring scheduling: (1) What are the rules for each node to decide whether turn itself on or off? and (2) When should nodes make such decision?

We first describe the states of each node and transition rules. In the proposed generic distributed algorithm, each sensor at any moment is in one of the following three states: *active*, i.e., the sensor monitors its monitoring region; *idle*, i.e., the sensor listens to other sensors, but does not monitor its region; and intermediate *vulnerable* state, i.e., the sensor monitors its region but should as soon as possible change its state to either active or idle. Each sensor knows in which state all its neighbors –

any state transition is immediately broadcasted to the neighbors together with the current energy (battery) supply.

The state transitions are described by the following rules:

A. When a sensor is in the vulnerable state, then it should change its state into active state if there is a face which is not covered by any other active or vulnerable sensor and

B. When a sensor is in the vulnerable state, then it should change its state into idle state if all its faces are covered by one of two types of sensors: active or vulnerable sensors with a larger energy supply.

C-D. When a sensor is in an active or idle state, then it should go into the vulnerable state if any neighboring sensor becomes vulnerable.

As soon as any sensor becomes vulnerable, the vulnerable state propagates over the entire network and eventually each sensor settles down in either idle or active state. We call this process a *global reshuffle*.

An extended state diagram corresponding to the proposed distributed algorithm is illustrated on Figure 4.1. Two extra states correspond to the case when a single alive sensor can monitor a region (*permanent* state) and when a sensor exhaust its energy supply (*terminated* state). Once the sensor node becomes active, it will be permanently active till it exhausts all its batteries. When a sensor node near-completely exhausts its energy supply, it will broadcast about that to its neighbors. A minimal subset of idle sensors will become active in order to cover the faces which will be soon abandoned by the exhausted sensor. For the purpose of comparison of our protocol with the protocol from [9], we have implemented the protocol without any reshuffle.

Now we will show that the proposed distributed algorithm is deadlock-free and needs only constant number (per sensor) of broadcasts to the neighbors.

LEMMA 4.1. *The distributed algorithm is deadlock-free, i.e., among vulnerable sensors there is always one that either should become active or idle.*

*Proof.* Toward contradiction, assume that each vulnerable sensor covers at least one face non-covered by any active/permanent sensor and does not have an individual face, i.e. a face which is not covered by any other vulnerable sensor. Then there is always a vulnerable sensor which is not a champion (has a largest energy supply) for any of its faces, i.e., a sensor eligible for transition into the idle state. Indeed, the sensor with the globally largest energy supply among all vulnerable sensors is such a sensor. □

THEOREM 4.2. *Each global reshuffle needs* 2 *broadcasts (to the neighbors) from each sensor and the resulted set of all active sensors form a minimal sensor cover.*

*Proof.* Assume that each sensor knows the coordinates of all its neighbors and, therefore, all the faces. For each reshuffle there is one broadcast informing neighbors that a sensor goes into vulnerable state and also announcing the available energy supply. The second broadcast inform the neighbors that a sensor change it state to one of 4 non-vulnerable states. The resulted set of active/permanent sensors covers all faces since a sensor cannot go to idle state if it has individual face. Finally, the resulted set of active/permannet sensors is minimal since any active sensor has individual face which is not covered by any other active/permannet sensor. □

Now all node states and transition rules are described and we should decide *when* the global reshuffle should happened. Definitely, a near-complete exhaustion of energy supply should be one of such triggering events. A more accurate schedule needs more frequent reshuffles implying increase in communication cost. We suggest to initiate a reshuffle when the energy supply of a sensor drops by a certain predefined threshold

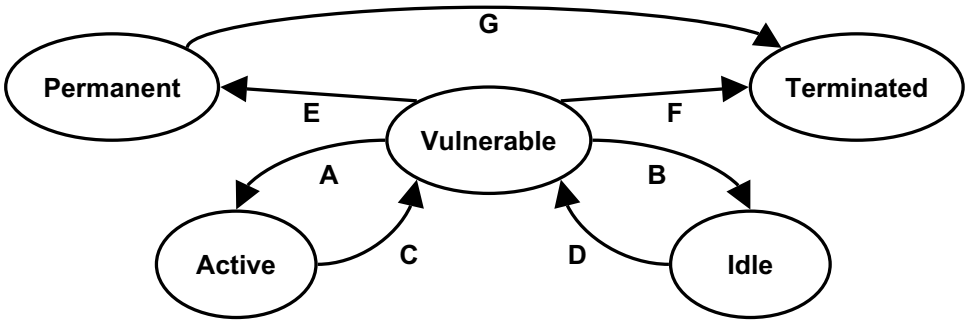


FIG. 4.1. *State diagram of the proposed distributed algorithm for lifetime maximization. A single sensor among vulnerable and active sensors covering a certain face uses transition A, a sensor which does not have maximum energy supply among sensors covering any face uses transition B, transitions C and D are used by neighbors of vulnerable sensors, transition E is a variation of transition A when a sensor is the only alive sensor covering certain face, transitions F and G are used by sensors which exhausted their energy supply.*

value $H$. The smaller reshuffle-triggering threshold $H$, the more frequent reshuffles are performed which will result in a more balanced schedule.

If communicating needs much more energy than sensing, we suggest optional *local* reshuffles with vulnerable state propagation limited to a certain neighborhood (e.g., 3-neighborhood) of the sensor which triggers the reshuffle.

**5. Experimental Study.** All algorithms were implemented in C++. The heuristics were compiled using `gpp` with `-O2` optimization, and run on a Sun workstation Ultra-60. The experiments were run on randomly generated testcases.

For experiments we have taken sensor area as 1000m×1000m, monitored area as 800m×800m. We experimented with 100, 200, 300, 400, 500, and 1000 sensor nodes, sensing range of 100m and 150m, coverage area of 100% ($q$=1) and 90%($q$=0.9). The position of sensor node is randomly distributed in sensor area. We ran our experiments with uniformly assigned batteries (10) to each sensor node and randomly assigned batteries between 10 to 20 to each sensor node. We have taken value of $\epsilon$ equals to 0.1, which decides the quality of Garg-Könemann solution. We primarily report lifetime of the network for Garg-Könemann, Tight, CPLEX and distributed algorithm. We also report the trade-off between communication overhead and lifetime for distributed algorithm with different reshuffle-triggering threshold.

After finding sensor covers from Garg-Könemann solution, we can find the optimal schedule by assigning the best times for each sensor cover by CPLEX, with constraint to satisfy battery requirement of Sensor Nodes and to maximize the total life time. Garg-Könemann's solution does not guarantee that there is a tight energy constraint, i.e., there is a sensor which completely exhausts its energy supply. The "Tight" solution is obtained from Garg-Könemann solution by finding the tightest energy constraint and making it tight by scaling up the timespans for each sensor cover.

Table 5.1 shows test cases and average runtime to find different solution for different number of nodes with sensing range of 100m and 150m. The largest scenario we have considered is with 1000 nodes and 100m sensing range. The number of faces increases as the number of nodes and/or the sensing range increases. The runtime for Tight algorithm is higher than Garg-Könemann, and CPLEX runtime is higher than

| Test Case | Parameter Interval | Maximum |
|---|---|---|
| No. of Nodes | 100-500 | 1000 |
| Sensor Range | 100-150 | 100 |
| No. of Faces | 861-43160 | 81845 |
| Runtimes | | |
| Find Faces | 8.96s - 886s | 1840s |
| Garg-Könemann | 3.55s - 2487.94s | 3605.45s |
| Tight | 5.06s - 2589.94s | 3758.17s |
| CPLEX | 9.75s - 3462.32s | 7887.53s |

TABLE 5.1

*Number of faces and runtime to find faces, Garg-Könemann solution, Tight solution and CPLEX solution for different number of nodes with different sensing range.*

| # of sensors | Reshuffle-Triggering Threshold Value | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 5 | 10 |
| 100 | 31.33 | 30.33 | 29.66 | 28.66 | 26.00 |
| | 3503 | 1732 | 1194 | 735 | 416 |
| 200 | 54.00 | 53.33 | 52.66 | 53.33 | 52.33 |
| | 11220 | 5591 | 3799 | 2396 | 1377 |
| 300 | 120.00 | 116.00 | 112.00 | 111.33 | 105.33 |
| | 35981 | 17431 | 11582 | 7888 | 4991 |
| 400 | 169.66 | 167.33 | 164.66 | 159.0 | 155.0 |
| | 66765 | 33457 | 22551 | 14381.7 | 9393 |
| 500 | **211.43** | 210.33 | 206.33 | 199.66 | 182.33 |
| | **93650** | 52114 | 35285 | 22578.0 | 13040 |

TABLE 5.2

*Lifetime and communication overhead for the distributed algorithm with sensing range 150m, and randomly assigned batteries between 10 and 20. Bold data are based on a single instance.*

for the Tight.

Figures 6.1, 6.2 and 6.3 show the life time of network for CPLEX, Tight, Garg-Könemann and distributed algorithms for different number of nodes. Fig. 6.1 shows the results for, sensor range of 100m, uniformly assigned batteries (10) and 100% coverage of monitored area ($q$=1). Fig. 6.2 shows the results for sensor range of 150m, randomly assigned batteries between 10 and 20, and 100% coverage of monitored area. Fig. 6.3 shows the results for sensor range of 100m, uniformly assigned batteries 10, and 90% coverage of monitored area. The Tight solution is always better than Garg-Könemann's, and CPLEX solution is always better than Tight solution. When we should cover the entire monitored region ($q$=1) (Fig. 6.1, 6.2), distributed solution is very near to centralized solution (GK and REAL), but when constraint is to cover only 90% of monitored area (Fig. 6.3), distributed solution is significantly worse than that of centralized solution.

Table 5.2 shows trade-off between life time and communication overhead for distributed algorithm with different reshuffle-triggering threshold. Sensing range of 100m is used for this experiment. Increasing the reshuffle-triggering threshold reduces the communication overhead but decreases the lifetime.

**6. Conclusions.** In this paper, we have formulated Maximum Sensor Network Life Problem and suggested centralized and distributed algorithms for solving this

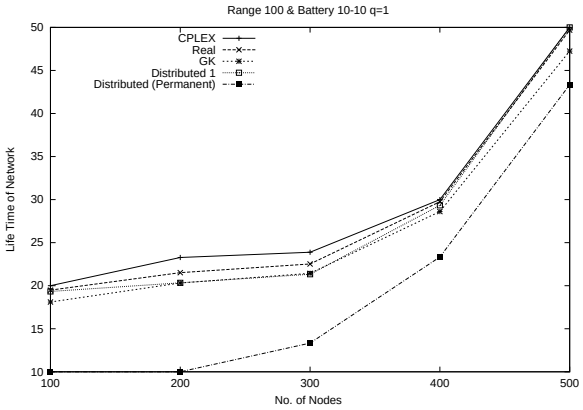


FIG. 6.1. *Lifetime for CPLEX, Tight, GK and the distributed algorithm with reshuffle threshold 1 and Permanent. The sensing range is 100m, battery supply 10, and q=1.*

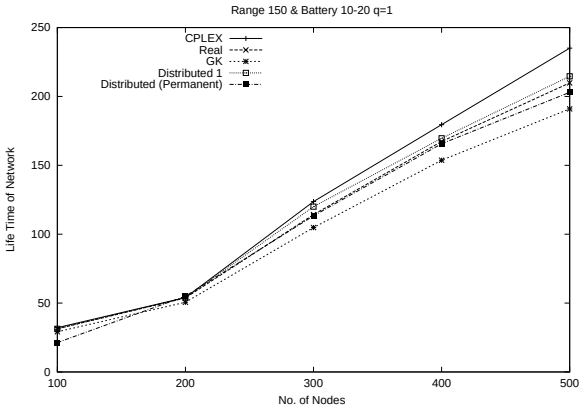


FIG. 6.2. *Lifetime for CPLEX, Tight, GK and the distributed algorithm with reshuffle threshold 1 and Permanent. The sensing range is 150m, battery supply between 10 and 20, and q=1.*

problem. We have explored the case when the monitored area is required to partially covered. We experimentally have checked the quality of our distributed algorithms comparing them with the centralized solution. We are going to simulate our algorithms within the LEACH protocol.

## REFERENCES

[1] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *to appear in ACM Wireless Networks*.

[2] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems", *Proc. 39th Annual Symposium on Foundations of Computer Science*, 1998, pp. 300–309.

[3] R. Hahn and H. Reichl, "Batteries and power supplies for wearable and ubiquitous computing," in *Proc. 3rd Int. Symp. Wearable Computers*, 1999.

[4] F. Leighton and S.Rao, "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms," *Journal of the ACM*, vol. 6, 1989, pp. 787–832.

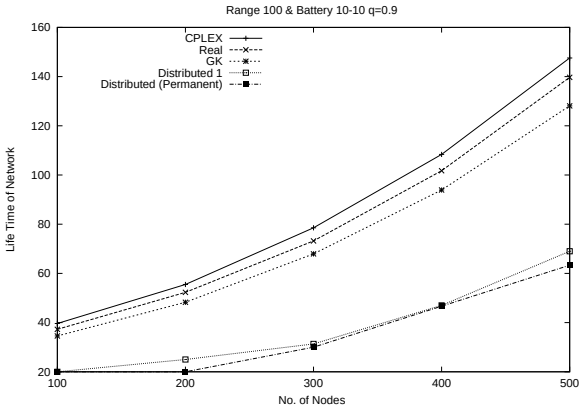


FIG. 6.3. *Lifetime for CPLEX, Tight, GK and the distributed algorithm with reshuffle threshold 1 and Permanent. The sensing range is 100m, the battery supply 10, and q=0.9.*

[5] R. J. Marks II, A. K. Das, M. El-Sharkawi, P. Arabshahi, and A. Gray, "Maximizing lifetime in energy constrained wireless sensor array using team optimization of cooperating systems," in *Proc. IEEE World Conference on Computational Inteligence 2002*, 2002.

[6] G. Robins and A. Zelikovsky, "Improved Steiner tree approximation in graphs," in *Proceedings of the 11th ACM-SIAM Annual Symposium on Discrete Algorithms*, 2000, pp. 770–779.

[7] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless networks," *IEEE Design & Test of Computers*, pp. 62–74, 2001.

[8] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *Proc. IEEE International Conference on Communications (ICC)*, 2001, pp. 472–476.

[9] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proc. ACM WSNA02*, 2002.

[10] G. Zussman and A. Segall, "Energy efficient routing in ad hoc disaster recovery networks," in *Proc. IEEE INFOCOM'03*, 2003.