Maximal Lifetime Scheduling for Sensor Surveillance Systems with K Sensors to One Target

Hai Liu, Pengjun Wan, and Xiaohua Jia

Abstract—This paper addresses the maximal lifetime scheduling for sensor surveillance systems with K sensors to 1 target. Given a set of sensors and targets in an Euclidean plane, a sensor can watch only one target at a time and a target should be watched by k, $k \geq 1$, sensors at any time. Our task is to schedule sensors to watch targets and pass data to the base station, such that the lifetime of the surveillance system is maximized, where the lifetime is the duration up to the time when there exists one target that cannot be watched by k sensors or data cannot be forwarded to the base station due to the depletion of energy of the sensor nodes. We propose an optimal solution to find the target watching schedule for sensors that achieves the maximal lifetime. Our solution consists of three steps: 1) computing the maximal lifetime of the surveillance system and a workload matrix by using linear programming techniques, 2) decomposing the workload matrix into a sequence of schedule matrices that can achieve the maximal lifetime, and 3) determining the sensor surveillance trees based on the above obtained schedule matrices, which specify the active sensors and the routes to pass sensed data to the base station. This is the first time in the literature that this scheduling problem of sensor surveillance systems has been formulated and the optimal solution has been found. We illustrate our optimal method by a numeric example and experiments in the end.

Index Terms—Energy efficiency, lifetime, scheduling, sensor network, surveillance system.

1 Introductions

A wireless sensor network consists of many low-cost and low-powered sensor nodes (called sensors for short) that collaborate with each other to gather, process, and communicate information using wireless communications [4]. Applications of sensor networks include military sensing, traffic surveillance, environment monitoring, building structures monitoring, and so on. One important characteristic of sensor networks is the stringent power budget of wireless sensor nodes, because those nodes are usually powered by batteries that may not be possible to be recharged or replaced after they are deployed in hostile or hazardous environments [15]. The surveillance nature of sensor networks requires a long lifetime. Therefore, it is an important research issue to prolong the lifetime of sensor networks in surveillance services.

In this paper, we discuss a maximal lifetime problem in sensor surveillance systems. Given a set of targets and sensors and a base station (BS) in an area, the sensors are used to watch (or monitor) the targets and collect sensed data to the BS. Each sensor has an initial energy reserve, and a fixed surveillance range and an adjustable transmission range. A sensor can watch at most one target at a time. A target can be inside the surveillance range of several sensors. A typical example is the use of camera to continuously watch some targets, such as cargo containers. In some applications, a target needs to be watched by several sensors at any time for the purpose of viewing the target from several directions, or of locating the geographic position of a target [21]. Without losing generality, we assume that each target should be watched by k, $k \geq 1$, sensors at any time. Since sensors are usually redundantly deployed, our problem is to schedule a subset of sensors to be active at a time to watch the targets and find the routes for the active sensors to send data back to the BS, such that each target should be watched by k sensors at any time and the lifetime of the entire sensor network is maximized. The *lifetime* is the duration up to the time when there exists one target that cannot be watched by k sensors or data can not be forwarded to the BS due to the depletion of energy of the sensor nodes.

The problem consists of two parts: scheduling the sensors to watch targets and routing the sensed data to BS. To our best knowledge, this is the first time in the literature that this problem has been formulated and the optimal solution has been found.

Our problem is different from the connected K-coverage problem as discussed in [23], [24]. The connected K-coverage problem is to select a minimum number of sensor nodes, such that each point in the sensor network is covered by at lease K different sensors, and the induced graph is connected. It assumes that each sensor can cover several points at the same time while each sensor can watch only one target in our problem. Furthermore, the connected K-coverage problem is to minimize the number of sensors to cover points while our problem is not only to schedule sensors to watch targets, but also to find routes to forward data to the BS, such that the lifetime of the surveillance is maximized.

We assume the positions of targets and sensors are static and given. There are many applications that the targets positions are static and given, such as guarding cargo containers, monitoring road traffic, monitoring specific points of buildings, and so on. The location information of

• *The authors are with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong.*
E-mail: {liuhai, pwan, jia}@cs.cityu.edu.hk.

Manuscript received 24 Dec. 2004; revised 27 Dec. 2005; accepted 3 Feb. 2006; published online 25 Oct. 2006.
Recommended for acceptance by C. Raghavendra.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0317-1204.

both sensors and targets can be obtained via a distributed monitoring mechanism [10] or the scanning method [11]. Our method will compute a schedule for sensors to watch targets and the routes to relay data to the BS. This schedule and routes will be disseminated to sensor nodes by the BS at the system initialization stage and all sensor nodes can operate according to the schedule, such as when and for what duration to sleep, watch targets, or relay messages.

The rest of the paper is organized as follows: Section 2 is related work and Section 3 is is the problem definition. Section 4 presents our solution that consists of three parts. Section 4.1 gives a linear programming formulation that is used to compute the maximal lifetime of the surveillance system. In Section 4.2, we show that the maximal lifetime is achievable, and detailed algorithms for finding the schedule are presented. Section 4.3 discusses surveillance trees for routing sensed data to BS. Section 5 presents a numeric example solved by using our method. Simulations are further conducted. We conclude our work in Section 6.

2 Related Work

Extensive research has been done on extending the lifetime of sensor networks. Bhardwaj et al. [12] studied the upper bounds on the lifetime of sensor networks used in data gathering in various scenarios. Both analytical results and extensive simulations showed that the derived upper bounds are tight for some scenarios and near-tight (about 95 percent) for the rest. The authors further proposed a technique to find the bounds of lifetime by partitioning the problem into the subproblems for which the bounds are either already known or easy to derive. A differentiated surveillance service for various target areas in sensor networks was discussed in [15]. The proposed protocol was based on an energy-efficient sensing coverage protocol that makes full coverage to a certain geographic area. It is also guaranteed to achieve a certain degree of coverage for fault tolerance. Simulations showed that a much longer network lifetime and a small communication overhead could be achieved. An energy-efficient surveillance system was designed and implemented in [16]. It was used to detect and track the positions of moving vehicles in a stealthy manner. The system was separated into five phases: system initialization, neighbor discover, sentry selection, report status, power management, and tracking activity. Simulations showed that trade-off between energy-awareness and surveillance performance is adaptable and the extension of network lifetime is achievable.

Another important technique used to prolong the lifetime of sensor networks is the introduction of switch on/off modes for sensor nodes. Recent works on energy efficiency in three aspects, namely area coverage, request spreading and data aggregation, were surveyed in [8]. It pointed out that the best method for conserving energy is to turn off as many sensors as possible, at the same time, however, the system must maintain its functionality. A distributed scheduling algorithm for stationary continuous monitoring sensor networks was investigated in [1]. It assumed that sensor networks are time synchronized. The proposed scheme exploited the time scale difference between network reconfiguration periods and data forwarding periods, to enable sensors to be awake only when necessary. Simulation results showed the network lifetime can be significantly increased. Another node scheduling scheme was developed in [3]. This scheme schedules the nodes to turn on or off without affecting the overall service provided. A node decides to turn off when it discovers that its neighbors can help it to monitor its monitoring area. The scheduling scheme works in a localized fashion where nodes make decisions based on its local information. Similar to [3], the work in [9] defined a criterion for sensor nodes to turn themselves off in surveillance systems. A node can turn itself off if its monitoring area is the smallest among all its neighbors and its neighbors will become responsible for that area. This process continues until the surveillance area of a node is smaller than a given threshold. A deployment of a wireless sensor network in the real world for habitat monitoring was discussed in [13]. A network consisting of 32 nodes was deployed on a small island to monitor the habitat environment. Several energy conservation methods were adopted, including the use of sleep mode, energy efficient communication protocols, and heterogeneous transmission power for different types of nodes.

Different from the above work, we use both of the above mentioned techniques to maximize the network lifetime in our solution: we find the optimal schedule to switch on/off sensors to watch targets in turn and we find the optimal routes to forward data from sensor nodes to the BS. To the best knowledge of the authors, this is the first work so far that addresses how to schedule sensors to watch targets and pass sensed data to the BS, such that the lifetime of the system is maximized.

3 System Model and Problem Statement

We introduce the following notations before getting into details of our method:

B = base station whose energy is unbounded.

S = set of sensors, and $n = |S|$.

T = set of targets, and $m = |T|$.

k = number of sensors that are required to watch a target at any time.

$S(j)$ = set of sensors that are able to watch target j, $j = 1, \ldots, m$.

$T(i)$ = set of targets that are within the surveillance range of sensor i, $I = 1, \ldots, n$.

Notice that $S(i)$ may overlap with $S(j)$ for $i \neq j$, and $T(i)$ may overlap with $T(j)$ for $i \neq j$.

$N(i)$ = set of direct neighbors of sensor i, $i = 1, \ldots, n$.

A direct neighbor of sensor i is a sensor that is within the maximal transmission range of sensor i.

E_i = initial energy reserve of sensor i, $i = 1, \ldots, n$.

d_{ij} = distance between sensor i and j, i, $j = 1, 2, \ldots, n, B$.

R = data rate generated from sensors while watching targets.

e^S = energy required for sensing one unit data.

e^T = energy required for transmitting one unit data.

e^R = energy required for receiving one unit data.

There are two requirements for sensors watching targets:

1. Each sensor can watch at most one target at a time.
2. Each target should be watched by k, $k \geq 1$, sensors at any time.

The problem of our concern is, for given S, T, and k, to find a schedule that meets the above two requirements for sensors watching targets and passing sensed data to the BS, such that the lifetime of the surveillance is maximized.

4 Our Solutions

It is difficult to find an optimal schedule that maximizes the lifetime without knowing the exact value of the maximal lifetime. Therefore, we solve the problem in three steps. First, we compute the upper bound on the maximal lifetime of the system by using linear programming. Second, we find a schedule that achieves the upper bound of the maximal lifetime. Finally, we determine the sensor surveillance trees based on the above optimal solution. We present our optimal solutions step by step.

4.1 Find Maximal Lifetime

We use linear programming (LP) technique to find the maximum lifetime of the system. Let L denote the lifetime of the surveillance system. We introduce two variables:

x_{ij}: total time sensor i watching target j, where $i \in S, j \in T$.
f_{ij}: amount of data transmitted from sensor i to sensor j (the receiver can be BS).

The problem of finding the maximum lifetime for sensors watching targets can be formulated as the following:
Objective: Max L

$$\text{s.t.} \sum_{i \in S(j)} x_{ij} = kL \qquad \forall j \in T; \tag{1}$$

$$\sum_{j \in T(i)} x_{ij} \leq L \qquad \forall i \in S; \tag{2}$$

$$e^s R \sum_{j \in T(i)} x_{ij} + \sum_{j \in N(i) \cup \{B\}} (d_{ij})^{\alpha} f_{ij} + e^R \sum_{j \in N(i)} f_{ji} \leq E_i \quad \forall i \in S; \tag{3}$$

$$R \sum_{j \in T(i)} x_{ij} + \sum_{j \in N(i)} f_{ji} = \sum_{j \in N(i) \cup \{B\}} f_{ij} \qquad \forall i \in S; \tag{4}$$

$$x_{ij} \geq 0, f_{ij} \geq 0. \tag{5}$$

Notice that in the above constrains, topology information that indicates which sensor is connected to which target (or other sensors) is contained in $S(j)$, $T(i)$, and $N(i), i = 1, \ldots, n, j = 1, \ldots, m$.

Equation (1) specifies that for each target j in T, since it requires k sensors to watch target j at anytime, the total time that target j under surveillance is k times of lifetime of the system. That is, each target should be watched by k sensors throughout the lifetime.

Inequality (2) implies that for each sensor i in S, its total working time should not exceed lifetime of the system.

Inequality (3) implies that the total energy cost of a sensor node shall not exceed its initial energy reserve. There are three components of energy cost of a sensor node, which are the cost for sensing data (i.e., watching targets), the cost for transmitting data (which is dependent on the transmission distance), and the cost for receiving data.

Equation (4) is for flow conservation. It implies that for each sensor i in S, the total amount of data sensed and data received should be equal to the amount of data transmitted.

The above formulation is a typical LP formulation, where $x_{ij}, 1 \leq i \leq n$, and $1 \leq j \leq m$, and $f_{ij}, i, j = 1, 2, \ldots, n, B$, are real number variables and the objective is to maximize L. So, the optimal results of x_{ij}, f_{ij}, and L can be computed in polynomial time.

However, L, obtained from computing the above LP formation, is the upper bound on the lifetime, and each x_{ij} specifies only the total time that sensor i should watch target j in order to achieve this upper bound L. Each f_{ij} specifies only the total amount of data transmitted from sensor i to sensor j. What we need is to find a schedule that specifies from what time up to what time which sensor watches which target and through which route to pass the sensed data to the BS. In the next two steps, we will find the schedule and routes that will finally achieve the optimal lifetime L.

The values of $x_{ij}, 1 \leq i \leq n$ and $1 \leq j \leq m$, obtained from the LP, can be represented as a matrix:

$$X_{n \times m} = \begin{bmatrix} x_{11} x_{12} \ldots x_{1m} \\ x_{21} x_{22} \ldots x_{2m} \\ \ldots\ldots \\ x_{n1} x_{n2} \ldots x_{nm} \end{bmatrix}_{n \times m}.$$

We call matrix $X_{n \times m}$ ***workload*** matrix, for it specifies the total length of time that a sensor should watch a target. There are two important features about this workload matrix:

1. the sum of all elements in each column is equal to kL (from (1) in the LP formulation).
2. the sum of all elements in each row is less than or equal to L (from (2) in the LP formulation).

In the next step, we need to find the detailed schedule for sensors to watch targets based on the workload matrix.

4.2 Decompose Workload Matrix

The lifetime of the surveillance system can be divided into a sequence of sessions. In each session, a set of sensors are scheduled to watch their corresponding targets; and in the next session, another set of sensors are scheduled to work (some sensors may work continuously for multiple sessions). Suppose a sensor cannot switch to watch another target within a session. Thus, the schedule of sensors during a session can be represented as a matrix. In this matrix, there are exactly k positive numbers in each column, representing each target should be watched by k sensors; and at most one positive number in each row, representing each sensor can watch at most one target at a time and there is no switching to watch other targets in a session. The rest numbers in the matrix are zeros. Furthermore, all the nonzero elements in this matrix have the same value, which is the time duration of this session. Now, our task becomes to decompose the workload matrix into a sequence of schedule matrices of sessions, represented as:

$$\begin{bmatrix} x_{11} x_{12} \ldots x_{1m} \\ x_{21} x_{22} \ldots x_{2m} \\ x_{31} x_{32} \ldots x_{3m} \\ \ldots\ldots \\ x_{n1} x_{n2} \ldots x_{nm} \end{bmatrix}_{n \times m} = \begin{bmatrix} 0 c_1 0 \ldots 0 \\ c_1 0 0 \ldots 0 \\ 0 0 0 \ldots c_1 \\ \ldots\ldots \\ 0 0 c_1 \ldots 0 \end{bmatrix} + \begin{bmatrix} c_2 0 0 \ldots 0 \\ 0 0 0 \ldots c_2 \\ 0 c_2 0 \ldots 0 \\ \ldots\ldots \\ 0 c_2 0 \ldots 0 \end{bmatrix}$$
$$+ \ldots + \begin{bmatrix} c_t 0 0 \ldots 0 \\ 0 0 c_t \ldots 0 \\ c_t 0 0 \ldots 0 \\ \ldots\ldots \\ 0 c_t 0 \ldots 0 \end{bmatrix} = P_1 + P_2 \ldots + P_t,$$

where $c_i, i = 1, 2, \ldots, t$ is the length of time of session i, and t the total number of sessions. We call this sequence of session schedule matrices P_i, $i = 1, 2, \ldots, t$, the ***schedule*** matrices. Considering the schedule matrix of session i, all elements in it are either "0" or c_i, each column has exactly k nonzero elements, and each row has at most one non-zero element (it could be all "0," indicating the sensor is idle in this session).

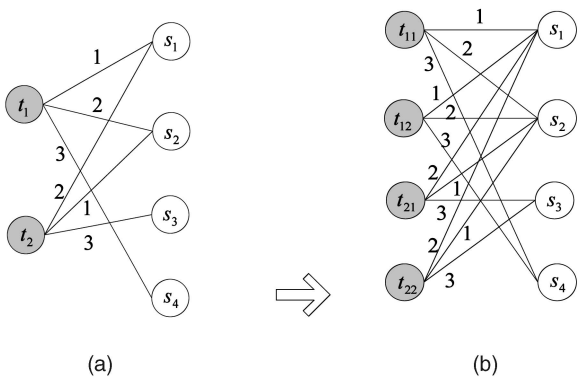

Fig. 1. Convert G to G_k.

Next, we discuss how to decompose the workload matrix into a sequence of schedule matrices. Since each target should be watched by k sensors throughout the lifetime of the system, we have $n \geq km$. We first consider a special case of $n = km$, i.e., the number of sensors is exactly k times of the number of targets in the system. Then, we extend the results to the general cases of $n > km$.

4.2.1 A Special Case $n = km$

We consider the case $n = km$. Let R_i, $i = 1, 2, \ldots, n$, and C_j, $j = 1, 2, \ldots, m$, denote the sum of row i and the sum of column j in the workload matrix, respectively. According to (1) and (2) of the LP formation, we have:

$$C_j = kl, j = 1, 2, \ldots, m. \tag{6}$$

$$R_i \leq L, i = 1, 2, \ldots, n. \tag{7}$$

Notice that the sum of all elements in the workload matrix equals to the sum of R_i, and also equals to the sum of C_j. That is, $\sum_{i=1}^{n} R_i = \sum_{j=1}^{m} C_j = m \times kL$. Since $n = km$, we have:

$$\sum_{i=1}^{n} R_i = n \times L. \tag{8}$$

Combining (7) and (8), we have:

$$R_i = L, i = 1, 2, \ldots, n. \tag{9}$$

Equations (6) and (9) give an important feature of the workload matrix when $n = km$ that the sum of each column is equal to $2L$ and the sum of each row is equal to L. This feature will guarantee the possibility of decomposing the workload matrix into schedule matrices in Theorem 1.

The basic idea of decomposing the workload matrix is to represent $X_{n\times m}$ as a bipartite graph $G(S \cup T, E)$, where one side are sensors $S = (s_1, s_2, \ldots, s_n)$ and the other are targets $T = (t_1, t_2, \ldots, t_m)$, $n = km$. For each nonzero element x_{ij} in $X_{n\times m}$, there is an edge from s_i to t_j and the weight of the edge is x_{ij}.

Considering each schedule matrix of session i, each row has exactly one nonzero element that specifies each sensor should work at session i, each column has exactly k nonzero elements that specify k sensors watching this target at session i. That is, there should be k distinct sensors to watch one target at each session, which can be represented as k sensors matching one target in the bipartite graph G. Thus, the problem of finding a schedule matrix is transformed into the problem of finding ***k*-matchings** in G, i.e., k sensors matching one target.

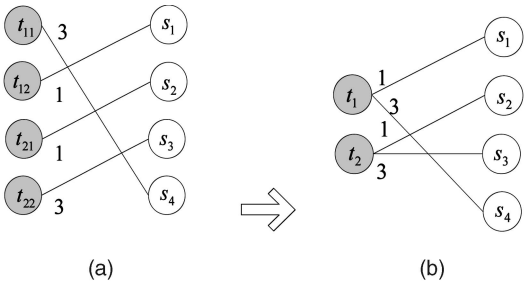

Fig. 2. A perfect matching in G_k and its corresponding k-matching in G.

The k-matching algorithm is not well studied (even though it can be designed by following the idea of the perfect matching algorithm). We make a transformation of the bipartite graph by replacing each target node by k duplicate nodes. The links adjacent to the original target node are adjacent to each of the duplicate nodes. Thus, in the new bipartite graph, one target (duplicate) need match exactly one sensor. The problem of finding a k-matching is transformed to finding a perfect matching in the new bipartite graph. There are many perfect matching algorithms, such as the one presented in [19], can be used to find a perfect matching. Fig. 1 shows an example of replacing target nodes by duplicates. Each target t_j is replaced by k copies, denoted by $t_{j1}, t_{j2}, \ldots, t_{jk}$. An edge from s_i to t_j is replaced by edges from s_i to each of $t_{j1}, t_{j2}, \ldots, t_{jk}$. We denote the resulting bipartite graph by G_k.

The algorithm works as follows: Each time G is converted to G_k, we find a perfect matching on G_k and merge the duplicate targets to obtain a k-matching where there are exactly k sensors match each target. Each k-matching is corresponding to a schedule matrix. Let c_i be the smallest weight of edges in the k-matching. We deduct c_i from the weight of the n edges in the k-matching in G and remove the edges whose weight becomes zero. This operation is repeated until there is no matching can be found in G_k.

For example, suppose we obtain a workload matrix

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 0 & 3 \\ 3 & 0 \end{bmatrix}$$

from the LP $(1) \sim (2)$ for $k = 2$. The matrix is first represented as a bipartite graph $G(S \cup T, E)$, where $T = \{t_1, t_2\}$ and $S = \{s_1, s_2, s_3, s_4\}$ (See Fig. 1a). Then, targets t_1 and t_2 are replaced by duplicates t_{11}, t_{12} and t_{21}, t_{22}, respectively, resulting a new graph G_k (See Fig. 1b). A perfect matching is found in G_k as shown in Fig. 2a and a k-matching is obtained by merging the duplicates targets back as shown in Fig. 2b. The schedule matrix is

$$P_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

based on the k-matching.

The details of the algorithm for finding a k-matching are given below.

***k*-Matching Algorithm**
Input: a workload matrix $X_{n\times m}$.
Output: a schedule matrix P_i.
Begin

Represent $X_{n\times m}$ as a bipartite graph $G(S \cup T, E)$;
Transform G to G_k by replacing each node in T by k duplicate nodes;
Find a perfect matching in G_k;
Merge duplicate targets in the perfect matching and obtain a schedule matrix P_i;

End

Because we try to decompose the workload matrix by using the technique of finding the perfect matchings, we are facing two questions:

1. Does it guarantee that there exists a perfect matching in every round of the decomposition?
2. Does it guarantee that the number of decomposition rounds is bounded?

Theorems 1 and 2 will give answers to the two questions, respectively. To prove Theorem 1, we need the following lemma.

Lemma 1. *For any square matrix $W_{n\times n}$ of nonnegative real numbers, if $R_i = C_j$ for $1 \le i, j \le n$, there exists a perfect matching in the corresponding bipartite graph, where R_i and C_j are the sum of row i and the sum of column j of $W_{n\times n}$, respectively.*

Proof. Let L be the sum of all elements in a row in $W_{n\times n}$, and $A_{n\times n}$ denotes matrix $A_{n\times n} = \frac{1}{L} \times W_{n\times n}$. It is obvious to see that $A_{n\times n}$ is a doubly stochastic matrix [18], where the sum of all elements in any row or column is equal to 1.

Now, we prove the lemma by contradictory. Assuming there does not exist a perfect matching in the corresponding bipartite graph of $A_{n\times n}$, there does not exist n positive entries $A_{n\times n}$ that no two entries in the same column or row. According to the König theorem [19], [20], we can cover all of the positive entries in the matrix with e rows and f columns, such that $e + f < n$. However, since the sum of all lines of $A_{n\times n}$ is equal to 1, it follows $n \le e + f < n$. This contradicts to the assumption. Lemma 1 is proved. □

Theorem 1. *The k-*Matching *algorithm can always find a k-*matching *so long as there are edges in G_k and its time complexity is $O(n^3)$.*

Proof. For any workload matrix $X_{n\times m}$, according to (6) and (9), the sum of each column is equal to kL and the sum of each row is equal to L. In the k-$Matching$ algorithm, we represent $X_{n\times m}$ as a bipartite graph G. Then, G is transformed to G_k by replacing each target by k duplicate nodes. This is equivalent to repeatedly appending matrix $X_{n\times m}$ $(k-1)$ times to the right hand side of the original matrix, resulting a new $n \times n$ workload matrix:

$$W_{n\times n} = \overbrace{[X_{n\times m} X_{n\times m} \ldots X_{n\times m}]}^{k}. \quad (10)$$

$W_{n\times n}$ consists of k matrices of $X_{n\times m}$ as shown in (10) and $n = km$.

Let R'_i and C'_j denote the sum of row i and the sum of column j of $W_{n\times n}$, respectively, and R_i and C_j the sum of row i and the sum of column j of $X_{n\times m}$, respectively. From the construction of $W_{n\times n}$, we can see the sum of each column remains the same as $X_{n\times m}$, i.e., $C'_j = C_j$. From (6), for $X_{n\times m}$, we have:

$$C'_j = kL, \ j = 1, 2, \ldots, n. \quad (11)$$

From (10), we can see the sum of each row of $W_{n\times n}$ is k times of that of $X_{n\times m}$, i.e., $R'_i = k\,R_i$. From (9) for $X_{n\times m}$, we know $R_i = L$. Thus, we have

$$R'_i = kL, \ i = 1, 2, \ldots, n. \quad (12)$$

Therefore, we have $R'_i = C'_j$ for $1 \le i, j \le n$. According to Lemma 1, there exists a perfect matching on the bipartite graph G_k.

Since, in each round i, we deduct c_i from the weight of the n edges in the perfect matching, it is equivalent to deducting the schedule matrix P_i from the workload matrix $W_{n\times n}$. Thus, the resulting matrix $W_{n\times n}$ (after deducting P_i) still holds the condition $R'_i = C'_j$ for $1 \le i, j \le n$. According to Lemma 1, there exists a perfect matching on G_k in every round of the decomposition process. This process stops at the last round where all the remaining edges in G_k make up an exact perfect matching, and they are all removed at this last round.

According to [5], [19], it takes $O(n^3)$ to find a perfect matching if we use depth-first search. Furthermore, it is not difficult to see that the k-$Matching$ algorithm and the algorithm of finding a perfect matching have the same time complexity. Theorem 1 is proved. □

The following theorem states that the number of decomposing of the workload matrix can be bounded by using the k-$Matching$ algorithm.

Theorem 2. *The workload matrix can be exactly decomposed into a sequence of schedule matrices by using the k-*Matching *algorithm and the number of decomposition rounds is bounded by the number of nonzero elements in $X_{n\times m}$.*

Proof. According to theorem 1, a perfect matching can be found in G_k at each round of decomposition until there is no edge left in G_k.

Since each edge in G has k duplicated edges in G_k, at each time of finding a perfect matching, at least k edges in G_k (they are duplicates of the same edge in G) are removed. Therefore, it takes at most $|E|$ number of rounds to remove all edges in G_k, where $|E|$ is the number of edges in G (not G_k), which is the number of nonzero elements in $X_{n\times m}$.

Theorem 2 is proved. □

Thus, the maximal lifetime scheduling problem for sensor surveillance systems with K sensors to 1 target can be solved when $n = km$, where m, n are the number of targets and the number of sensors, respectively. In the next section, we will discuss the general cases of $n > km$ and propose a complete decomposition algorithm.

4.2.2 General Cases $n > km$

When $n > km$, our basic idea is to transform the case to $n = km$ by introducing some dummqy targets and sensors into the system. That is to "fill" the workload matrix $X_{n\times m}$ with some dummy columns and rows, such that the sum of all elements in each row is equal to L and the sum of each column is kL. Let Z denote the dummy matrix. There are two cases.

1. The numbqer of sensors is in multiple of k, i.e., $n\%k = 0$. We append $Z_{n\times(n/k-m)}$ to the right-hand side of $X_{n\times m}$, the resulting matrix, denoted by $W_{n\times n/k}$, is in the form as:

$$W_{(n+n\%k)\times(n+n\%k)/k}=\begin{bmatrix} x_{11}x_{12}\ldots x_{1m} & z_{11} & z_{12} & \cdots & z_{1(n+n\%k)/k-m} \\ x_{21}x_{22}\ldots x_{2m} & z_{21} & z_{22} & \cdots & z_{2(n+n\%k)/k-m} \\ \ldots\ldots & \ldots\ldots & & & \\ x_{n1}x_{n2}\ldots x_{nm} & z_{n1} & z_{n2} & \cdots & z_{n(n+n\%k)/k-m} \\ \ldots\ldots & & & & \\ 0\ \ 0\ldots\ 0 & z_{(n+n\%k)1} & z_{(n+n\%k)2} & \ldots & z_{(n+n\%k)(n+n\%k)/k-m} \end{bmatrix}_{(n+n\%k)\times(n+n\%k)/k}$$

Fig. 3. The resulting matrix for $n\%k>0$.

$$W_{n\times n/k}=\begin{bmatrix} x_{11}x_{12}\ldots x_{1m} & z_{11}z_{12}\ldots z_{1n/k-m} \\ x_{21}x_{22}\ldots x_{2m} & z_{21}z_{22}\ldots z_{2n/k-m} \\ \ldots\ldots & \ldots\ldots \\ x_{n1}x_{n2}\ldots x_{nm} & z_{n1}z_{n2}\ldots z_{nn/k-m} \end{bmatrix}_{n\times n/k}.$$

2. Otherwise, i.e., $n\%k>0$. We first append $n\%k$ dummy rows, which are all $(0,0,\ldots,0)$ vectors, to the bottom of $X_{n\times m}$. It is equivalent to adding $n\%k$ dummy sensors with energy $E_i=0$ to the network. It does not affect our optimal solution. Then, we append $Z_{(n+n\%k)\times((n+n\%k)/k-m)}$ to the right-hand side of $X_{(n+n\%k)\times m}$. The resulting matrix, denoted by $W_{(n+n\%k)\times(n+n\%k)/k}$, is in the form as shown in Fig. 3.

Let $p=n+n\%k$, $q=(n+n\%k)/k$, the dummy matrix can be denote by $Z_{p\times(q-m)}$ and the filled matrix can be denote by $W_{p\times q}$. To make the matrix $W_{p\times q}$ having the features of (6) and (9), i.e., the sum of each column is equal to kL and the sum of each row is equal to L, the dummy matrix $Z_{p\times(q-m)}$ should satisfy the following conditions:

1.

$$R_i'=\sum_{j=1}^{q-m} z_{ij}=L-R_i \text{ for } \forall i=1,2,\ldots,p. \quad (13)$$

2.

$$C_j'=\sum_{i=1}^{p} z_{ij}=L \text{ for } \forall j=1,2,\ldots,q-m. \quad (14)$$

We propose a simple algorithm to compute the dummy matrix $Z_{p\times(q-m)}$. The algorithm starts to assign values to the elements of $Z_{p\times(q-m)}$ from its top-left corner. Let R_i^- and C_j^- record the sum of the remaining undetermined elements of row i and column j, respectively, for $i=1,2,\ldots,p$ and $j=1,2,\ldots,q-m$. Initially, $R_i^-\leftarrow(L-R_i)$ and $C_j^-\leftarrow L$, where R_i and L are computed from matrix $X_{n\times m}$. The strategy of the algorithm is to assign the remaining sum of the row (or column), as much as possible, to an element without violating conditions (13) and (14), and assign the rest elements of the row (or column) to 0. Then, we move down to the next undetermined element from the top-left of the matrix. For example, we start with z_{11}. Now, R_1^- is $(L-R_1)$ and C_1^- is L, i.e., $R_1^-<C_1^-$. Thus, we can assign R_1^- to z_{11}, and assign 0 to the rest of elements of row 1 (so, (13) is met). Then, C_1^- should be updated to $(C_1^- - z_{11})$, because the remaining sum of column 1 now becomes $(C_1^- - z_{11})$ and this value is used to ensure that (14) will be met during the process. Suppose we now come to element z_{ij}, (i.e., elements of z_{kl}, for $k=1,\ldots,i-1$ and $l=1,\ldots,j-1$, are already determined so far). We compare R_i^- with C_j^-. There are three cases:

1. $C_j^->R_i^-$: It means z_{ij} can use up the remaining value the sum of row i, i.e., R_i^-. Thus, $z_{ij}\leftarrow R_i^-$ and the rest elements of this row should be assigned to 0. So, all elements of row i have been assigned and condition (13) is met for row i.
2. $R_i^->C_j^-$: It means z_{ij} can use up the remaining value the sum of column j, i.e., C_j^-. Thus, $z_{ij}\leftarrow C_j^-$ and the rest elements of this column should be assigned to 0, i.e., $z_{kj}=0, k=2,3,\ldots,p$. By doing so, all elements of column j have been assigned and condition (14) is met for column j.
3. $R_i^-=C_j^-$: We can determine elements in both row i and column j by $z_{ij}\leftarrow R_i^-$ and setting the rest elements in row i and in column j to 0. It is easy to see that condition (13) is met for row i and condition (14) is met for column j.

After determining each row (or column), we need to update C_j^- (or R_i^-), before moving to the next row (or column). Each step, we can determine the elements in one row (or column). This process is repeated until all elements in $Z_{p\times(q-m)}$ are determined. The details of the algorithm are given below.

FillMatrix Algorithm
Input: a workload matrix $X_{n\times m}$.
Output: a filled matrix $W_{p\times q}$.
Begin
  $p=n+n\%k$; $q=(n+n\%k)/k$;
  $R_i^-=L-R_i$, for $i=1$ to p;
  $C_j^-=kL$, for $j=1$ to $q-m$;
  $i=1$; $j=1$;
  while $(i\le p)$ && $(j\le q-m)$ **do**
    if $C_j^->R_i^-$ **then**
      //determine elements in row i.
      $z_{ij}=R_i^-$;
      $z_{ik}=0$, for $k=j+1$ to $q-m$;
      // set the rest of row i to 0.
      $C_j^-=C_j^--z_{ij}$;
      $i=i+1$;
    else if $R_i^->C_j^-$
      //determine elements in column j.
      $z_{ij}=C_j^-$;
      $z_{kj}=0$, for $k=i+1$ to p;
      // set the rest of column j to 0.
      $R_i^-=R_i^--z_{ij}$;
      $j=j+1$;
    else

//determine elements in both row i and column j.
$z_{ij} = R_i^-$;
$z_{ik} = 0$, for $k = j+1$ to $q-m$;
$z_{kj} = 0$, for $k = i+1$ to p;
$i = i+1$; $j = j+1$;
endwhile
End

Theorem 3. *For a given workload matrix $X_{n\times m}$,* FillMatrix *Algorithm can compute the filled matrix $W_{p\times q}$, such that the sum of each column and the sum of each row have the features defined in (6) and (9), respectively.*

Proof. At the beginning of the *FillMatrix* Algorithm, row sums and column sums of the dummy matrix are initialized, and then the dummy matrix is worked out step by step to satisfy (13) and (14). So, we can prove a general case: Given row sums R_i' and column sums C_j' of a matrix $Z_{n\times m}$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, the proposed algorithm can compute all elements z_{ij} that satisfy (13) and (14). We use the induction method to prove the theorem.

1. When $n = 1$, $m = 1$, according to the *FillMatrix* algorithm, since $C_1^- = R_1^-$, we have $z_{11} = R_1^- = C_1^- = R_1' = C_1'$. Conditions (13) and (14) are both met.
2. We assume when $n \le p-1$, $m \le q-1$, the proposed algorithm can compute $Z_{n\times m}$, such that (13) and (14) are both met.
3. When $n = p$, $m = q$, according the algorithm, we first compare C_1^- with R_1^-, there are three cases:
   a. If $C_1^- = R_1^-$, then set $z_{11} = R_1^-$, $z_{1k} = 0$, $k = 2, 3, \ldots, m$ and $z_{k1} = 0$, $k = 2, 3, \ldots, n$. For the row 1 and column 1 where z_{ij} have been determined, we have $\sum_{j=1}^{m} z_{1j} = z_{11} = R_1^- = R_1'$ and $\sum_{i=1}^{n} z_{i1} = z_{11} = C_1^- = C_1'$. So, (13) and (14) are both met in row 1 and column 1. The remaining undetermined elements $z_{ij}, i = 2, 3, \ldots, n, j = 2, 3, \ldots, m$, are in the matrix $Z_{(p-1)\times(q-1)}$. According to assumption 2), the remaining matrix $Z_{(p-1)\times(q-1)}$ can be correctly worked out.
   b. If $C_1^- > R_1^-$, then set $z_{11} = R_1^-$, $z_{1k} = 0$, $k = 2, 3, \ldots, m$ and $C_1^- = C_1^- - R_1^-$. For the row 1 where z_{ij} have been determined, we have $\sum_{j=1}^{m} z_{1j} = z_{11} = R_1^- = R_1'$, (13) is met. For column 1 which is updated, we have $C_1^- + z_{11} = C_1'$, it does not violate (14). The remaining undetermined elements z_{ij}, $i = 2, 3, \ldots, n$, $j = 1, 2, 3, \ldots, m$, are in the matrix $Z_{(p-1)\times q}$. We continue run the algorithm to compute the remaining elements in $Z_{(p-1)\times q}$ that satisfies (13) and (14). Note that C_1^- monotonously decreases after each round of assignment and $\sum_{i=2}^{n} R_i^- = \sum_{j=1}^{m} C_j^- > C_1^-$. There must exist $R_l^- \ge C_1^-$ in round l, we set $z_{l1} = C_1^-$, $z_{k1} = 0, k = l+1, l+2, \ldots, n$ and $R_l^- = R_l^- - C_1^-$. Then, the remaining matrix is $Z_{(p-l+1)\times(q-1)}$. According to assumption 2), the remaining matrix $Z_{(p-l+1)\times(q-1)}$ can be correctly worked out.
   c. If $R_1^- > C_1^-$, similar to b, we can prove this case.
4. The proof of cases $n = p$, $m = q-1$ and $n = p-1$, $m = q$ are similar to 3.

Combining 1, 2, and 3 with 4, the proposed algorithm can correctly compute all elements in the matrix $Z_{n\times m}$, such that (13) and (14) are both met.

Theorem 3 is proved. □

Theorem 4. *The time complexity of the* FillMatrix *Algorithm is $O(n^2)$.*

Proof. It is not difficult to see that the time complexity of the proposed algorithm is $O(n^2)$. Theorem 4 is proved. □

Thus, the case of $n > km$ can be smoothly transformed to the case of $n = km$. Integrating together with *FillMatrix* algorithm and k-$Matching$ algorithm, we have the algorithm of decomposing the workload matrix for general cases of $n \ge km$.

DecomposeMatrix Algorithm
Input: a workload matrix $X_{n\times m}$.
Output: a sequence of schedule matrices $P_1, P_2, \ldots, P_t$.
Begin
if $n > km$ **then**
Run *FillMatrix* on $X_{n\times m}$ to obtain $W_{p\times q}$, where $p = n + n\%k$, $q = (n + n\%k)/k$;
Construct a bipartite graph G from $W_{p\times q}$;
while there exist edges in G **do**
Run k-$Matching$ to find a schedule matrix P_i;
Deduct P_i from $W_{p\times q}$ and removeedges with weight 0 in G;
endwhile
Output $W_{p\times q} = P_1 + P_2 + \ldots + P_t$;
End

Theorem 5. *The total time complexity of the* DecomposeMatrix *algorithm is $O(|E| \times n^3)$.*

Proof. Proof is similar to the proof of Theorem 2. □

Given a workload matrix $X_{n\times m}$, using the proposed algorithm, we can fill the matrix to make it a square matrix $W_{p\times q}$ and decompose $W_{p\times q}$ into a sequence of schedule matrices as follows:

$$W_{p\times q} = P_1 + P_2 + \ldots + P_t, \tag{15}$$

where P_i is the schedule matrix which is corresponding to a perfect matching in G_k. According to the Theorem 2, t can be bounded by $p \times q$.

Let P_i' denote the matrix which contains the first n rows and m columns in P_i (i.e., the information for the n valid sensors and m valid targets by dropping the dummy parts), $i = 1, 2, \ldots, t$. By removing the dummy columns and rows in P_i, we have following valid schedule matrices:

$$X_{n\times m} = P_1' + P_2' + \ldots + P_t'. \tag{16}$$

The above discussions conclude that a workload matrix is decomposable to a sequence of schedule matrices such that each value of x_{ij} and f_{ij}, can be actually met. In the next section, we will determine a sensor surveillance tree for each schedule matrix that specifies the routes for active sensors to pass sensed data to BS, such that the maximal lifetime L can be finally achieved.

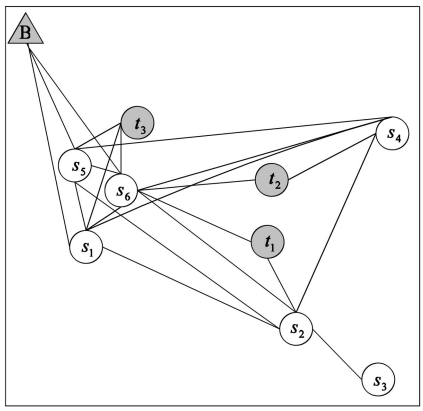

Fig. 4. An example with six sensors and three targets.

4.3 Determine Surveillance Tree

We have obtained a sequence of schedule matrices. Each schedule matrix specifies the active sensors watching targets in this session. That is, the number of sessions is the number of schedule matrices. To allow the active sensors send their sensed data to the BS at each session, we need to construct a sensor surveillance tree whose root is the BS and all leaf nodes are the active sensors. The sensed data flow from active sensors to the BS along the tree. Some active sensors can perform both duties of watching targets and forwarding data for other sensors at the same time.

From computing the LP formulation in Section 4.1, we have obtained a data flow f_{ij} from any sensor node i to sensor node j. To forward data to the BS, each sensor node, say i, needs to follow its outgoing flow f_{ij} in order to achieve the maximal lifetime L. Suppose sensor i has l downstream nodes, denoted by $s_1, s_2, \ldots, s_l$, to forward its data to the BS (i.e., $f_{i1}, f_{i2}, \ldots, f_{il}$ have nonzero values). Since there is no ordering of data flow $f_{i1}, f_{i2}, \ldots, f_{il}$, we simply let sensor i pass its outgoing data first to s_1 until flow f_{i1} is saturated, then it switch to s_2 until the value of f_{i2} is met, $\ldots$, and finally it pass the last flow f_{il} to s_l. The outgoing data of sensor i include its own sensed data and the data it helps others to forward to the BS, as shown in the left-hand side of (4). By following the data flow obtained from the LP formulation in forwarding data to the BS, the optimal routes, in terms of energy efficiency, are used and thus the maximal lifetime L is achieved.

In the sensor surveillance system, after computing the schedule matrices and the data flow, the BS will disseminate this schedule and flow to sensor nodes at the system initialization stage. When the system starts operation, each sensor will watch targets, turn off to sleep, receive, and forward data according to its schedule. Notice that some sensors may work continuously for multiple sessions. There is no need to synchronize sensors to switch target watching at the end of session. Each sensor works according to its own schedule *independently* from the others. A sensor can watch a target continuously until it is time to switch to another target or turn itself off.

Since clock synchronization is now achievable in sensor networks by using some localized method [14] or time synchronization scheme [17], sensors can cooperate correctly for the surveillance work.

5 Experiments and Simulations

5.1 A Numeric Example

We randomly place a BS, six sensors (in clear color in Fig. 4) and three targets (in gray in Fig. 4) in a 10×10 two-dimensional free-space region. For simplicity, we assume that each target requires one sensor to watch at any time, i.e., $k = 1$, and the surveillance range and the maximal transmission range of sensors are set to 0.4×10 and 0.8×10 (our solution can work for systems with nonuniform surveillance ranges and maximal transmission ranges), respectively. Fig. 4 shows neighbors of sensors and the surveillance relationship between sensors and targets. An edge between a sensor and a target represents the target is within the surveillance range of the sensor. An arc from sensor s_i to s_j represents s_j is within the maximal transmission range of s_i (in this example, maximal transmission ranges for all sensors are uniform, so arcs are replaced by edges in Fig. 4). The initial energy reserves of sensors are random numbers generated in the range of [0, 100] with the mean at 50, as shown in Table 1. To simulate the energy consumed on different tasks, we set $e^T = 0.12$ and $e^R = 0.1$. These values are in proportional to the actual power consumption for transmitting and receiving data as pointed out in [22]. The energy consumption ratio for sensing data e^S is set to the same as e^R, i.e., $e^S = 0.1$, because sensing data usually cost similar amount of energy as receiving data [22]. The sensing data rate $R = 1$ and the signal decline factor $\alpha = 2$.

We follow the three steps in our method to find sensor surveillance trees.

First, we use the linear programming, described in Section 4.1, to compute the maximal lifetime L, workload matrix $X_{6\times3}$ and data flows (see Table 2) that achieve L:

$$L = 28.6972,$$

$$X_{6\times3} = \begin{bmatrix} 0 & 0 & 0 \\ 17.2300 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 13.0999 & 0 \\ 0 & 0 & 28.6972 \\ 11.4672 & 15.5973 & 0 \end{bmatrix}.$$

In the workload matrix, we can see target 1 is watched by sensors 2 and 6 for 17.2300 and 11.4672, respectively. The total time for target 1 to be watched is 28.6972, which is the lifetime of the surveillance system.

Second, we run the *FillMatrix* algorithm to append a dummy matrix to the workload matrix to make it a square matrix $W_{6\times6}$, where the sum of each column and the sum of each row are all equal to L:

TABLE 1
The Initial Energy Reserves of Six Sensors

Sensors	1	2	3	4	5	6
E_i	52.6522	90.9600	76.7550	92.3830	92.7103	61.5487

TABLE 2
The Data Flows among Six Sensors and the BS

Date flow	The amount of data flow
$s_1 \rightarrow$BS	13.97655
$s_2 \rightarrow s_1$	13.97655
$s_2 \rightarrow s_5$	3.25345
$s_4 \rightarrow s_5$	13.0999
$s_5 \rightarrow$BS	54.674638
$s_6 \rightarrow s_5$	9.624088
$s_6 \rightarrow$BS	17.440412

$$W_{6\times 6} = \begin{bmatrix} 0 & 0 & 0 & 28.6972 & 0 & 0 \\ 17.2300 & 0 & 0 & 0 & 11.4672 & 0 \\ 0 & 0 & 0 & 0 & 17.2300 & 11.4672 \\ 0 & 13.0999 & 0 & 0 & 0 & 15.5973 \\ 0 & 0 & 28.6972 & 0 & 0 & 0 \\ 11.4672 & 15.5973 & 0 & 0 & 0 & 1.6327 \end{bmatrix}.$$

Then, we run the *DecomposeMatrix* algorithm to decompose $W_{6\times 6}$ into three schedule matrices P_1, P_2, and P_3 (i.e., the decomposition terminates at round 3), such that

$$W_{6\times 6} = P_1 + P_2 + P_3.$$

By removing the dummy columns of the schedule matrices, we have:

$$X_{6\times 3} = \begin{bmatrix} 0 & 0 & 0 \\ 1.6327 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1.6327 & 0 \\ 0 & 0 & 1.6327 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 15.5973 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 15.5973 \\ 0 & 15.5973 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 11.4672 & 0 \\ 0 & 0 & 11.4672 \\ 11.4672 & 0 & 0 \end{bmatrix}.$$

Finally, the surveillance trees based on the above schedule matrices and data flows are determined. The surveillance trees for three sessions are shown as Fig. 5a, Fig. 5b, and Fig. 5c, respectively.

It is easy to see that the surveillance trees in Fig. 5 satisfy both the surveillance requirement and data flow constraints. Thus, the maximal lifetime L is actually achieved.

5.2 Simulations

The simulations are conducted in a 100×100 two-dimensional free-space region. BS, sensors, and targets are randomly distributed inside the region. Again, the surveillance range and the maximal transmission range of all sensors are set to 0.4×100 and 0.8×100, respectively (except the simulations for Fig. 6a and Fig. 6b). We assume that each target should be watched by three sensors at any time, i.e., $k = 3$ (except the simulations for Fig. 6d). The signal decline factor $\alpha = 2$ and the initial energy reserves of sensors are the random numbers in the range of [0, 100], with the mean value of 50. We study the performance against four parameters: surveillance range, maximal transmission range, number of sensors in the network, and number of sensor required by each target during the surveillance. The results presented in the figures are the means of 100 separate runs.

A greedy algorithm is proposed to compare the performance with our optimal solution. The basic idea of the greedy method is as follows: Each time, we find a k-matching in the network graph as a schedule for alive sensors (sensors with nonzero energy reserve) to watch targets. Then, for each sensor scheduled to work in the session, we find the minimal energy cost path from it to the BS. When any node either in watching a target or in the path runs out of energy, it is removed from the network and another k-matching or path is computed. This operation is repeated until the k-matching cannot cover all targets or the path to BS cannot be found. The system lifetime of the greedy method is the minimal surveillance time that each target is watched.

Fig. 6a and Fig. 6b show the lifetime versus the change of surveillance range and the maximal transmission range of sensors, respectively. We set $n = 100$, $m = 10$, and $k = 3$. From the figures, we can see that when the surveillance range (the maximal transmission range) increases, the performance gap between two methods becomes more significant. This is because, with a small surveillance range (maximal transmission range), sensors usually have got a few targets (sensors) within its surveillance range (maximal transmission range). There is hardly any room that our optimization method can take advantages. As the surveillance range (the maximal transmission range) becomes larger, more sensors are able to cover multiple targets (sensors), which gives our method more room to schedule the sensors properly to achieve the maximal lifetime. That is why the performance gap between the two methods becomes more significant as the increase of the surveillance range (the maximal transmission range). Furthermore, we can see that the increase of surveillance range is more effective to extending the system lifetime than the increase of the maximal transmission range. This is because the surveillance range is usually much smaller than the communication range of sensors. It is always the bottleneck of the maximization of system lifetime, and some targets could not be watched by enough sensors often results in quick die of surveillance systems.

Fig. 6c shows how the lifetime is affected by the number of sensors placed in the region (density of sensors). Curves in Fig. 6c exhibit the similar trend as those in Fig. 6a and Fig. 6b. As more sensors deployed in the same region, the density becomes higher. A target can be watched by more sensors and there is a higher chance for a target to be in the surveillance range of multiple sensors. At the same time, a sensor can reach more neighbors and can choose more energy-efficient routes to forward data. Thus, our optimal algorithm takes more advantages by optimizing the schedule and the performance becomes more significant than the greedy method when the density of sensors becomes higher.

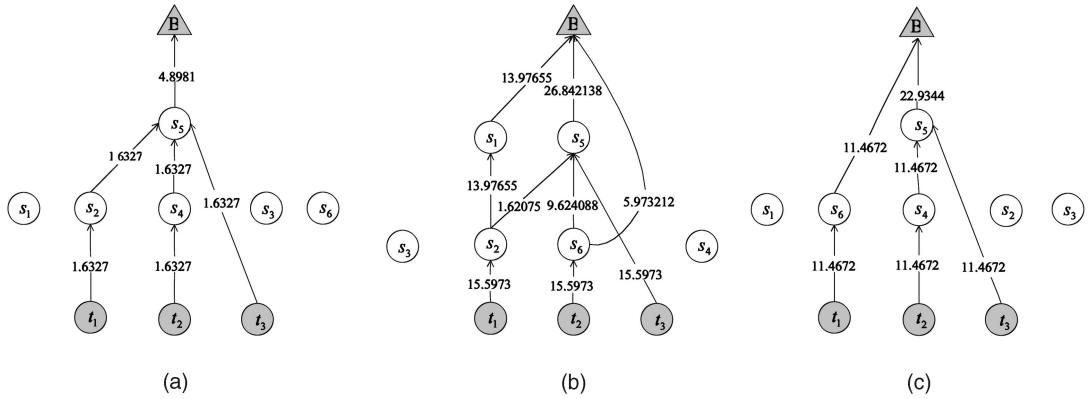

Fig. 5. The surveillance trees for three sessions. (a) Session 1, (b) session 2, and (c) session 3.

Fig. 6d shows how the lifetime is affected by the number of sensors required by each target during the surveillance. We set $n = 100$, $m = 10$, and vary k from 1 to 6. Once again, the performance of our optimal algorithm is much better than that of greedy algorithm. In Fig. 6d, although the performance gap between two algorithms becomes smaller when k increases, the ratio of the lifetime achieved by our algorithm over that of greedy algorithm is about 5 when $k = 1$, while this ratio reaches almost 10 when $k = 6$. It means that our optimal algorithm takes more advantages when k becomes larger. The reason is similar to the analysis we discussed before.

From Figs. 5a, 5b, 5c, 6a, 6b, 6c, and 6d, we have the following conclusions:

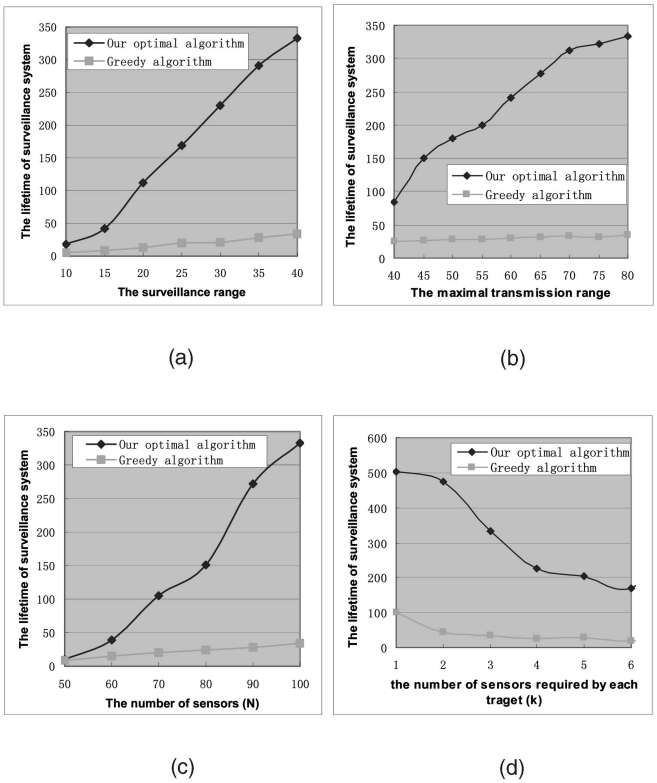

Fig. 6. (a) L versus surveillance range. (b) L versus the maximal transmission range. (c) L versus n when $m = 10$. (d) L versus k.

1. Our optimal algorithm has significantly better performance in the situation where sensors have larger surveillance and communication range, or when sensors are densely deployed and k becomes larger.
2. The increase of surveillance range is more effective to extending the system lifetime than the increase of the maximal transmission range of sensors.

6 Conclusions

This paper addressed the maximal lifetime scheduling for sensor surveillance systems with K sensors to 1 target. This is the first time in the literature that this scheduling problem of sensor surveillance systems was formulated and the optimal solution was presented.

Our solution consists of three steps: 1) compute the maximum lifetime of the system, and find the workload matrix and data flows to BS by using linear programming method, 2) decompose the workload matrix into a sequence of schedule matrices by using perfect matching method. This decomposition can preserve the maximum lifetime. 3) construct a surveillance tree for each session of schedule matrix. These surveillance trees specify how the active sensors forward data flows to BS. It is not difficult to see that our solution is the optimum in the sense that it can find the schedules for sensors watching targets that achieve the maximum lifetime. We illustrated our optimal method by an example in the end. Simulations were further conducted to show that our algorithm has significantly better performance in the situation where sensors have larger surveillance and communication range, or when sensors are densely deployed and k becomes larger.

Acknowledgments

This work was supported by a grant from Research Grants Council of Hong Kong (Project No. CityU 114505).

References

[1] M.L. Sichitiu, "Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2004.
[2] Y. Yu, B. Krishnamachari, and V.K. Prasanna, "Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2004.
[3] D. Tian and N.D. Georganas, "A Coverage-Preserving Node Scheduling Scheme for Large Wireless Sensor Networks," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications,* pp. 32-41, 2002.

[4] C.-Y. Chong and S.P. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proc. IEEE,* pp. 1247-1256, vol. 91, no. 8, Aug. 2003.
[5] R. Gould, *Graph Theory.* The Benjamin/Cummings Publishing Company, Inc., pp. 198-199, 1988.
[6] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks," *Operating Systems Design and Implementation,* Dec. 2002.
[7] R. Diestel, *Graph Theory,* second ed. Springer, pp. 31-33, 2000.
[8] J. Carle and D. Simplot-Ryl, "Energy-Efficient Area Monitoring for Sensor Networks," *Computer,* vol. 37, no. 2, pp. 40-46, Feb. 2004.
[9] L.B. Ruiz et al., "Scheduling Nodes in Wireless Sensor Networks: A Voronoi Approach," *Proc. 28th IEEE Conf. Local Computer Networks (LCN '03),* pp. 423-429, Oct. 2003.
[10] C.-F. Hsin and M. Liu, "A Distributed Monitoring Mechanism for Wireless Sensor Networks," *Proc. Int'l Conf. Mobile Computing and Networking Proc. ACM Workshop Wireless Security,* pp. 57-66, 2002.
[11] Y. Zhao, R. Govindan, and D. Estrin, "Residual Energy Scans for Monitoring Wireless Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf.,* pp. 356-362, 2002.
[12] M. Bhardwaj, T. Garnett, and A. Chandrakasan, "Upper Bounds on the Lifetime of Sensor Networks," *Proc. IEEE Int'l Conf. Comm.,* pp. 785-790, 2001.
[13] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications,* pp. 88-97, Sept. 2002.
[14] Q. Li and D. Rus, "Global Clock Synchronization in Sensor Networks," *Proc. IEEE INFOCOM,* 2004.
[15] T. Yan, T. He, and J.A. Stankovic, "Differentiated Surveillance for Sensor Networks," *Proc. First Int'l Conf. Embedded Networked Sensor Systems,* pp. 51-62, 2003.
[16] T. He et al., "Energy-Efficient Surveillance System Using Wireless Sensor Networks," *Proc. MobiSyS,* June 2004.
[17] K. Römer, "Time Synchronization in Ad Hoc Networks," *Proc. ACM MobiHoc,* 2001.
[18] R.A. Brualdi and H.J. Ryser, *Combinatorial Matrix Theory.* Cambridge Univ. Press, pp. 9-10, 1991.
[19] D.B. West, *Introduction to Graph Theory,* pp. 109-111. Prentice Hall, Inc., 1996.
[20] S. Axler, F.W. Gehring, and K.A. Ribet, *Graph Theory,* second ed. Springer, 2000.
[21] X. Cheng, G. Xue, and D. Chen, "TPS: A Time-Based Positioning Scheme for Outdoor Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2004.
[22] A. Savvides, C.-C. Han, and M. Srivastava, "Dynamic Fine-grained Localization in Ad-Hoc Networks of Sensors," *Proc. MobiCom '01,* pp. 166-179, July 2001.
[23] Z. Zhou, S.R. Das, and H. Gupta, "Connected K-Coverage Problem in Sensor Networks," *Proc. Int'l Conf. Computer Comm. and Networks (ICCCN '04),* pp. 373-378, 2004.
[24] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks," *Proc. ACM SenSys,* 2003.

Hai Liu received the BSc (1999) and MSc (2002) degrees in applied mathematics from the South China University of Technology, China. He is currently a PhD candidate in the Department of Computer Science at the City University of Hong Kong. His research interests include distributed systems, wireless networks, and mobile computing.

Pengjun Wan received the PhD degree (1997) from the University of Minnesota, the MS degree (1993) from the Chinese Academy of Science, and the BS degree (1990) from Tsinghua University. He is currently an associate professor of computer science at the Illinois Institute of Technology. His research interests include wireless networks and optical networks.

Xiaohua Jia received the BSc (1984) and MEng (1987) degrees from the University of Science and Technology of China, and the DSc (1991) degree in information science from the University of Tokyo, Japan. Professor Jia is currently associated with the Department of Computer Science at the City University of Hong Kong, His research interests include distributed systems, computer networks, WDM optical networks, and Internet and mobile computing.