Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 50 (2006) 2839–2854
Computer Networks
www.elsevier.com/locate/comnet

Maximal lifetime scheduling for *K* to 1 sensor–target surveillance networks

Hai Liu [a,*], Pengjun Wan [a,b], Xiaohua Jia [a]

[a] *Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong SAR, Hong Kong*
[b] *Department of Computer Science, Illinois Institute of Technology, United States*

Received 11 October 2004; received in revised form 22 July 2005; accepted 4 November 2005
Available online 9 December 2005

Responsible Editor: R. Sivakumar

Abstract

This paper addresses the maximal lifetime scheduling for k to 1 sensor–target surveillance networks. Given a set of sensors and targets in Euclidean plane, a sensor can watch only one target at a time and a target should be watched by k sensors ($k \geqslant 2$) at anytime. Our task is to schedule sensors to watch targets, such that the lifetime of the surveillance system is maximized, where the lifetime is the duration that all targets are watched. We propose an optimal solution to find the target watching schedule for sensors that achieves the maximal lifetime. 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. 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 by extending the Hall's theory, to achieve the maximal lifetime; (3) obtaining a target watching timetable for each sensor based on the schedule matrices. The time complexity of our optimal method is $O(m^2n^3)$, where m, n are the number of targets and the number of sensors, respectively. We illustrate our optimal method by a numeric example and experiments in the end.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Energy efficiency; Lifetime; Scheduling; Sensor network; Surveillance system

1. Introductions and related work

A wireless sensor network consists of many low-cost and low-powered sensor devices (called sensor nodes) 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 and it may not be possible to recharge or replace the batteries after they are deployed in hostile or hazardous environments [15]. The surveillance nature of sensor networks requires a long lifetime. Therefore, it is an

[*] Corresponding author. Tel.: +852 2788 8690.
E-mail addresses: liuhai@cs.cityu.edu.hk (H. Liu), wan@cs.iit.edu (P. Wan), jia@cs.cityu.edu.hk (X. Jia).

1389-1286/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2005.11.001

important research issue to prolong the lifetime of sensor networks in surveillance services.

In this paper, we discuss the scheduling problem in sensor surveillance networks. Given a set of targets and sensors in an area, the sensors are used to watch (or monitor) the targets. A sensor can watch targets that are within its surveillance range, and a target can be inside several sensors' watching range. Suppose each sensor has a given energy reserve (in terms of the length of time it can operate correctly) and each sensor can watch at most one target at a time. In some applications, it needs several sensors to watch one target at any time. For example, it needs three sensors to precisely determine a GPS location of a target [18]. So we assume that each target should be watched by k sensors at anytime and $k \geqslant 2$. The problem is to find a schedule for sensors to watch the targets, such that all targets should be watched by k sensors at anytime and the lifetime of the surveillance is maximized. The lifetime is the duration up to the time when there exists one target that cannot be watched by k sensors due to the depletion of energy of the sensor nodes. By using this schedule, a sensor can switch off to save energy when it is not its turn to watch a target. We assume the positions of targets and sensors are given and are static. This information can be obtained via a distributed monitoring mechanism [10] or the scanning method [11]. We further assume that a data aggregation tree has been constructed and some energy-efficient data gathering protocol [2,6] is taken to gather the data, since discussing how to collect data to the base-station is beyond the scope of the paper.

Extensive research has been done on extending the lifetime of sensor networks. Authors in [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%) for the rest. The authors further proposed a technique to find the bounds of lifetime by partitioning the problem into the sub-problems 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 tradeoff 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, the paper discusses the maximal lifetime scheduling problem for k to 1 sensor–target surveillance systems. To the best knowledge of the authors, this is the first work so far that addresses how to schedule sensors to watch targets in surveillance systems, such that the lifetime of the system is maximized.

The rest of the paper is organized as follows. Section 2 is the problem definition. We first consider a simple case $k = 2$ in Section 3, and present our solution that consists of three parts. Section 3.1 gives a linear programming formulation that is used to compute the maximal lifetime of the surveillance system. In Section 3.2, we show that the maximal lifetime is achievable, and detailed algorithms for finding the schedule are presented. Section 3.3 discusses the final schedule timetable for sensors. Section 4 extends the optimal results to the general cases $k \geqslant 2$. Section 5 presents a numeric example solved by using our method. Simulations are further conducted. We conclude our work in Section 6.

2. System model and problem statement

We consider a set of targets and a set of sensors that are used to watch targets and collect information. We first introduce the following notations:

S	set of sensors
T	set of targets
$n = \|S\|$	number of sensors
$m = \|T\|$	number of targets
k	number of sensors that are required to watch a target at anytime
$S(j)$	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, $i = 1, \ldots, n$
E_i	initial energy reserve 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 $i \neq j$. There are two requirements for sensors watching targets:

1. Each sensor can watch at most one target at a time.
2. For each target, there will be k sensors to watch it at anytime and $k \geqslant 2$.

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, such that the lifetime of surveillance is maximized. The lifetime of surveillance is defined at the length of time until there exists a target j such that the number of sensors that do not run out their energy in $S(j)$ is less than k.

We tackle the problem in three steps. First, we compute the upper bound on the maximal lifetime of the system and a workload matrix of sensors. Second, to achieve the upper bound, we successfully decompose the workload matrix into a sequence of schedule matrices. Finally, we obtain a target watching timetable for each sensor based on the schedule matrices. To present our solutions, we first consider a simple case $k = 2$, then we extend the results to general cases $k \geqslant 2$.

3. A simple case k = 2

In this section, we consider the case that each target requires two sensors to watch at anytime during the lifetime of surveillance systems. We present our optimal solutions step by step.

3.1. 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, and x_{ij} be the variable denoting the total time sensor i watching target j, where $i \in S$, $j \in T$. The problem of finding the maximum lifetime for sensors watching targets can be formulated as the following:

Objective: Max L

$$\text{s.t.} \quad \sum_{i \in S(j)} x_{ij} = 2L \quad \forall j \in T; \tag{1}$$

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

$$\sum_{j \in T(i)} x_{ij} \leqslant E_i \quad \forall i \in S. \tag{3}$$

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

Inequality (2) implies that for each sensor i in S, the total time that it watches targets should not exceed the lifetime of the system. Any service of the sensors beyond that time becomes useless.

Inequality (3) ensures that for each sensor i in S, its total working time (i.e., the total time it watches targets) should not exceed its battery's lifetime.

The above formulation is a typical LP formulation, where x_{ij}, $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$, are real number variables and the objective is to maximize L. The optimal results of x_{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. Now we have two questions:

1. Is this upper bound of lifetime L achievable? If yes, then
2. How to schedule sensors to watch targets, such that each value of x_{ij}, $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$, can be actually met?

In answering question (2), we need to find a timetable for each sensor that specifies from what time to what time that this sensor should watch which target.

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

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

We call the 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 the workload matrix when $k = 2$:

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

In the next step, we need to show that the upper bound of lifetime L is achievable, and find the detailed schedule for sensors to watch targets based on the workload matrix.

3.2. Decompose workload matrix

The lifetime of the surveillance system can be divided into of 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 will not switch to watch another target within a session. Thus, the schedule of sensors during a session can be represented as a non-negative matrix. In this matrix, there are exactly two positive numbers in each column, representing each target should be watched by two sensors; and at most one positive number in each row, representing each sensor can watch at most one target and there is no switching to watch other targets in a session. Furthermore, all the non-zero 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 session schedule matrices, represented as

$$\begin{bmatrix} x_{11}x_{12}\ldots x_{1m} \\ x_{21}x_{22}\ldots x_{2m} \\ x_{31}x_{32}\ldots x_{3m} \\ \cdots \\ x_{n1}x_{n2}\ldots x_{nm} \end{bmatrix}_{n\times m} = \begin{bmatrix} 0c_10\ldots 0 \\ c_100\ldots 0 \\ c_100\ldots 0 \\ \cdots \\ 00c_1\ldots 0 \end{bmatrix} + \begin{bmatrix} c_200\ldots 0 \\ 000\ldots c_2 \\ 0c_20\ldots 0 \\ \cdots \\ 0c_20\ldots 0 \end{bmatrix} + \cdots + \begin{bmatrix} 000\ldots c_t \\ 00c_t\ldots 0 \\ 00c_t\ldots 0 \\ \cdots \\ 0c_t0\ldots 0 \end{bmatrix},$$

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 the *schedule* matrices. Considering the schedule matrix of session i, all elements in it are either 0 or c_i, each column has exactly two non-zero elements, and each row has at most one non-zero element (it could be all 0, indicating the sensor is idle in this session).

The next, we discuss how to decompose the workload matrix into a sequence of schedule matrices. We first consider a simple special case of $n = 2m$, i.e., the number of sensors is two times of the number of targets in the system. Then, we extend the results to the general cases of $n \geqslant 2m$.

3.2.1. A special case $n = 2m$

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

$$C_j = 2L, \quad j = 1, 2, \ldots, m, \tag{4}$$

$$R_i \leqslant L, \quad i = 1, 2, \ldots, n. \tag{5}$$

Furthermore, since $\sum_{i=1}^{n} R_i = \sum_{j=1}^{m} C_j = m \times 2L$ and $n = 2m$, we have

$$\sum_{i=1}^{n} R_i = n \times L. \tag{6}$$

Combining (5) and (6), we have

$$R_i = L, \quad i = 1, 2, \dots, n. \tag{7}$$

Eqs. (4) and (7) imply that for the workload matrix the sum of each column is equal to $2L$ while the sum of each row is equal to L.

The basic idea of decomposing the workload matrix is to represent the workload matrix as a bipartite graph where one side are sensors and the other are targets. The bipartite graph consists of two set of nodes $S = (s_1, s_2, \dots, s_n)$ and $T = (t_1, t_2, \dots, t_m)$, $n = 2m$, representing sensors and targets respectively. For each non-zero element x_{ij} in the workload matrix, there is an edge from s_i to t_j and the weight of the edge is x_{ij}. Thus, the schedule problem in each session is transformed into the problem of finding matching in the bipartite graph, such that there are exactly two sensors matching one target. We first introduce the definition of the *2-matching*.

Definition 1. 2-matching: Let $G(S \cup T, E)$ be a bipartite graph with bipartition $\{S, T\}$, where $|S| \geqslant 2|T|$. 2-matching is defined as a subset of edges in E, such that

1. For each vertex in S, no two edges share the vertex.
2. For each matched vertex in T, there are exactly two edges share the vertex.

The 2-matching means that 1 vertex in T exactly matches 2 vertexes in S. Applying the 2-matching to our problem, T is the targets set and S is the sensors set. We further define the *perfect 2-matching*.

Definition 2. Perfect 2-matching: If $|S| = 2|T|$ and a 2-matching contains $2|T|$ edges, the largest possible, we call it the perfect 2-matching.

For a schedule matrix of size n by m, there are exactly two non-zero elements in a column and at most one non-zero element in a row, since $n = 2m$, there is exactly one non-zero element in each row. That is, such schedule matrix can be represented as a perfect 2-matching in a bipartite graph. Thus, the decomposing process is as follows. We first transform a workload matrix which is computed from Section 3.1 into a bipartite graph. Then we compute a perfect 2-matching in the bipartite graph, which has exactly n edges. Let c_i be the smallest weight of the n edges. Deduct c_i from the weight of the n edges in the perfect 2-matching and remove the edge whose weight becomes zero. This operation is repeated until there is no perfect 2-matching can be found in the bipartite graph.

Because we try to decompose the workload matrix by using the technique of finding the perfect 2-matchings, the questions we have now are:

Q1. Does it guarantee that there exists a perfect 2-matching in every round of the decomposition process?
Q2. If it exists, how to compute?
Q3. Does it guarantee that the decomposing of the workload matrix can be done in polynomial time? That is, is the number of decomposing rounds bounded?

We will answer the above questions one by one.

To answer Q1, we first extend the Hall's Theory [7]. For $\forall U \subseteq T$, we define $N(U) = \{S(j) | j \in U, S(j) \in S\}$.

Theorem 1. *Let $G(S \cup T, E)$ be a bipartite graph with bipartition $\{S, T\}$, where $|S| = 2|T|$. G contains a perfect 2-matching if and only if $|N(U)| \geqslant 2|U|$ for all $U \subseteq T$.*

Proof. Obviously, the condition is necessary. We will prove this necessary condition is in fact sufficient. We apply induction on $|T|$. For $|T| = 1$, the assertion is true.

We assume that the condition is sufficient when $|T| \leqslant p$. That is, there exists a perfect 2-matching in G if $|N(U)| \geqslant 2|U|$ for all $U \subseteq T$.

When $|T| = p + 1$, we try to construct a perfect 2-matching based on the above assumption. Since $|N(U)| \geqslant 2|U|$ for all $U \subseteq T$, there are three cases: (1) $\exists U \subset T$, $|U| \neq 0$ and $|N(U)| = 2|U|$, (2) $\exists U \subset T$, $|U| \neq 0$ and $|N(U)| = 2|U| + 1$, (3) for $\forall U \subset T$, $|N(U)| \geqslant 2|U| + 2$. A perfect 2-matching can be constructed for all three cases. We prove the theorem case by case.

Case 1: If $\exists U \subset T$, $|U| \neq 0$ and $|N(U)| = 2|U|$, we define $V = N(U)$ and $G'((S - V) \cup (T - U), E')$, where E' is a subset of E after subtracting edges related to U and V. We claim that G' satisfies: $|N_{G'}(U')| \geqslant 2|U'|$ for all $U' \subseteq T - U$. If not, there exist $U' \subseteq T - U$ and $|N_{G'}(U')| < 2|U'|$.

We have $|N_G(U' + U)| \leqslant |N_G(U)| + |N_{G'}(U')| < 2|U| + 2|U'| = 2|U + U'|$, that is $|N_G(U' + U)| < 2|U + U'|$.

It contradicts to our assumption. So, according the induction hypothesis, we know there exists a perfect 2-matching in G' and $G - G'$, respectively. Putting the two matchings together, we obtain a perfect 2-matching of T in G.

Case 2: If $\exists U \subset T$, $|U| \neq 0$ and $|N(U)| = 2|U| + 1$, similar to the above case, we claim that G' satisfies: $|N_{G'}(U')| \geqslant 2|U'|$ for all $U' \subseteq T - U$. If not, there exist $U' \subseteq T - U$ and $|N_{G'}(U')| < 2|U'|$. We have

$$\begin{aligned}|N_G(U'+U)| &\leqslant |N_G(U)| + |N_{G'}(U')|\\ &< 2|U| + 1 + 2|U'| = 2|U+U'| + 1,\\ |N_G(U'+U)| &< 2|U+U'|+1, \text{ that is, } |N_G(U'+U)| \leqslant 2|U+U'|.\end{aligned}$$

Furthermore, we know $|N_G(U'+U)| \geqslant 2|U + U'|$, then $|N_G(U'+U)| = 2|U+U'|$. This is the first case we proved. So we can construct a perfect 2-matching for this case.

Case 3: If for $\forall U \subset T$, $|N(U)| \geqslant 2|U| + 2$, we arbitrarily pick two edges ab_1, ab_2 and consider the graph $G'((S - \{b_1, b_2\}) \cup (T - \{a\}), E - \{ab_1, ab_2\})$. For $\forall U' \subseteq T - \{a\}$,

$$\begin{aligned}|N_{G'}(U')| &\geqslant |N_G(U')| - 2 \geqslant 2|U'| + 2 - 2\\ &= 2|U'|, \text{ that is, } |N_{G'}(U')| \geqslant 2|U'|.\end{aligned}$$

So by the induction hypothesis G' contains a perfect 2-matching of $T - \{a\}$. Together with the edges ab_1 and ab_2, we obtain a perfect 2-matching of T in G.

Theorem is proved. □

Based on Theorem 1, the following theorem gives an answer to Q1.

Theorem 2. *For any workload matrix of size n by m, if n* = 2*m, there exists a perfect* 2*-matching in every round of decomposing process.*

Proof. According to Theorem 1, we need to prove $|N(U)| \geqslant 2|U|$ for all $U \subseteq T$ in every round of decomposing process.

We first consider a workload matrix of size n by m, where each row corresponds to a sensor while each column corresponds to a target. For $\forall U \subseteq T$, let $p = |U|$ and $q = |N(U)|$, they represent p targets (columns) and q sensors (rows), respectively. $|N(U)| = q$ means that we can use q rows to cover all non-zero elements in the corresponding p columns.

For each column of p, since q rows cover all non-zero elements in this column, all non-zero elements in this column are distributed on the positions of q rows. That is, the sum of non-zero elements that are covered by q rows in this column is equal to $2L$. Thus, the total sum of non-zero elements that are both covered by q rows and p columns is equal to $2pL$.

For each row of q, since the sum of all elements in each row is equal to L, the sum of non-zero elements that are covered by p columns in this row is less than or equal to L. Thus, the total sum of non-zero elements that are both covered by q rows and p columns is less than or equal to qL. That is $qL \geqslant 2pL$, thus, $|N(U)| \geqslant 2|U|$. According to Theorem 1, we know that there exists a perfect 2-matching in the workload matrix.

In the following, we will prove that after deducting a schedule matrix from the workload matrix, the remaining matrix is still a workload matrix. So the decomposing process can continue.

Assume that we compute a perfect 2-matching and get its corresponding schedule matrix in round i. Since $n = 2m$, there are exactly two non-zero elements in each column and exactly one non-zero element in each row in the schedule matrix. We denote this non-zero element by c_i. After deducting the schedule matrix from the workload matrix, the sum of all elements in each row in the remaining matrix is equal to $L - c_i$, and the sum of all elements in each column in the remaining matrix is equal to $2L - 2c_i$. We replace $(L - c_i)$ with L, the remaining matrix still satisfies the definition of the workload matrix. That is, there exists a perfect 2-matching in every round of decomposing process.

Theorem 2 is proved. □

With the guarantee of Theorem 2, our next step is to find an algorithm to compute the perfect 2-matching in bipartite graphs. We propose an efficient decomposition algorithm that is extension of the labeling algorithm for perfect matching [5]. Let M denote a set of edges of a 2-matching. We use (s_i, t_j) to denote an edge from S to T and (t_j, s_i) denote an edge from T to S. There is no direction of edges in the graph, but this notation helps to describe the algorithm. Furthermore, we call a target t_j in T is *2-unsaturated* to M if and only if there are less than 2 edges in M that start (end) with t_j. The algorithm starts from any matching in the graph. Each time, it selects the first 2-unsaturated target and tries to find an *M-path* (called augment matching path) staring from this target. An *M-path* is a path in the bipartite graph. It starts with a T node that is 2-unsaturated to M and end with an S node that is not in M, and any edge in the *M-path* from T to S should not be in M and any edge from S to T should be in M. We can see that there are always one more non-M-edges than the M-edges in an *M-path* (an M-edge is an edge in M). Thus, by replacing M-edges in the *M-path* by the non-M-edges, the number of edges in M is incremented by 1. We keep on finding this *M-path* for every 2-unsaturated target and increasing the size of M, until a perfect 2-matching is found. For clarity of notation, in the algorithm, "$s_i \in M$" simply means s_i is an end-node of an edge in M and "$s_i \notin M$" means s_i is not an end-node of an edge in M. The detailed algorithm is given as follows. (The labels assigned to targets (sensors) in the algorithm are used to remember the previous vertex of the targets (sensors) in an *M-path*. The target labeled with "$*$" is the start of an *M-path*.)

```
Perfect-2-Matching Algorithm
Input: a bipartite graph $G = (S \cup T, E)$.
Output: a perfect 2-matching $M$.
Begin
   pick any edge from $E$ and add to $M$;
   while there exist 2-unsaturated targets do
      select the first 2-unsaturated target $t$ and
      label $t$ with $*$;
      for $s_i \in S(t)$ and $(t, s_i) \notin M$
         label $s_i$ with $t$;
      while all newly labeled $s_i \in M$ do //M-path
      is not found so far
         for all newly labeled $s_i$ and unlabeled
         $t_j \in T(s_i)$
            if $(s_i, t_j) \in M$
               label $t_j$ with $s_i$;
         for all newly labeled $t_j$ and unlabeled
         $s_i \in S(t_j)$
            if$(t_j, s_i) \notin M$
               label $s_i$ with $t_j$;
      endwhile
      // have found an M-path which starts with $t$
      and ends with $s_i \notin M$.
      remove M-edges in M-path from $M$ and
      add in non-M-edges to $M$;
   endwhile
End
```

The following theorem claims the correctness of the *Perfect-2-Matching* algorithm.

Theorem 3. *The Perfect-2-Matching algorithm is correct and the time complexity is* $O(mn^2)$.

Proof. It is equal to prove that the *Perfect-2-Matching* algorithm can find the perfect 2-matching if it exists in a bipartite graph. We will prove it using induction method.

When $|T| = 1$, $T = \{t_1\}$. Since the perfect 2-matching exists, there should be at least two edges that star with t_1. The algorithm will find all the *M-paths* of t_1 before it enters the second "while" loop. Each *M-path* contains only one edge. Then the perfect 2-matching is found.

We assume that the algorithm is correct when $|T| = m$. That is, the algorithm can find the perfect 2-matching which contains exactly $2m$ edges.

When $|T| = m + 1$, according to the above assumption, a 2-matching which contains $2m$ edges can be found. That is, only one target is left 2-unsaturated. We consider this 2-unsaturated target t. Since we assume the perfect 2-matching M_p exists, the *M-path* which stars with t exists. Our algorithm is to globally search the *M-path* which stars with t in breadth-first order, so we can find the *M-path* which stars with t. The only concern is that we may find an improper *M-path* which stars with edge $(t, s_i) \notin M_p$, and adding this edge may affect the latter searching for the *M-path* which stars with t. In the following, we prove that if $(t, s_i) \notin M_p$, there must exist another perfect 2-matching M'_p which contains (t, s_i).

We assume that the two edges that star (end) with t in M_p are (t, s_p) and (t, s_q), and initialize $M'_p = M_p - \{(t, s_p), (t, s_q)\}$. Note that current 2-matching M increases by 1 each time after an *M-path* is found. If t is still 2-unsaturated after we find an *M-path* which stars with edge $(t, s_i) \notin M_p$, we first add (t, s_i) to M'_p to make its size $2m + 1$. If $s_i \in M'_p$, it means that there exist $(t_j, s_i) \in M'_p$ and (t_j, s_i) conflicts with (t, s_i), we remove (t_j, s_i) from M'_p and compensate it by adding a new edge (t_j, s_k) which comes from M. If $s_k \in M'_p$ still holds, we repeat the above process to eliminate conflict until $s_k \notin M'_p$ or $s_k \in \{s_p, s_q\}$. If $s_k \notin M'_p$ and $s_k \notin \{s_p, s_q\}$, we arbitrarily pick an edge from (t, s_p) or (t, s_q) and add it to M'_p. If $s_k \in \{s_p, s_q\}$, for example $s_k = s_p$, we add the other edge (t, s_q) to M'_p. Now, M'_p contains $2m + 2$ edges and there are exactly two edges share a common vertex in T. That is, M'_p is a perfect 2-matching which contains edge (t, s_i). The *Perfect*-2-*Matching* algorithm is correct.

In the algorithm, to find a perfect 2-matching which contains $2m$ edges, we need to find $2m$ *M-paths*. Furthermore, since we may search all edges in the bipartite graph to find an *M-path*, it costs time

O(mn). So the total time complexity is O($2m^2n$), i.e., O(mn^2).

Theorem 3 is proved. □

The *Perfect-2-Matching* algorithm and Theorem 3 give an answer to Q2. The answer to Q3 is in the following theorem.

Theorem 4. *The decomposing of a workload matrix can be done in* O(m^2n^3), *where* m, n *are the number of targets and the number of sensors, respectively.*

Proof. For the decomposition of the workload matrix, Theorem 2 guarantees that there exists a perfect 2-matching in every round of decomposing process. Theorem 3 shows that we can find the perfect 2-matching if it does exist.

In each round, we use *Perfect-2-Matching* algorithm to compute a perfect 2-matching in the bipartite graph, which has exactly n edges. Let c_i be the smallest weight of the n edges. Deduct c_i from the weight of the n edges in the perfect 2-matching and remove the edge whose weight becomes zero. It is equal to deduct a schedule matrix from the workload matrix, and there is at least one element in the workload matrix becomes zero. This operation is repeated until all elements in the workload matrix become zero. Since there are $n \times m$ elements in the workload matrix, the decomposition round is bounded by $n \times m$.

Combining with Theorem 3, the total time complexity to decompose a workload matrix is O(m^2n^3).

Theorem 4 is proved. □

After answering Q1, Q2 and Q3, the maximal lifetime scheduling problem can be solved when $n = 2m$, where m, n are the number of targets and the number of sensors, respectively. In the next section, we will discuss the general cases $n > 2m$.

3.2.2. General cases $n > 2m$

When $n > 2m$, our basic idea is to transform the case to $n = 2m$ by adding some dummy targets and sensors. 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 and column are equal to L and $2L$, respectively. Let $Z_{p\times q}$ denote the dummy matrix, we consider two cases.

1. If n is even, we add $Z_{n\times(n/2-m)}$ to the right hand side of $X_{n\times m}$, the resulting matrix, denoted by $W_{n\times n/2}$, is in the form as

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

2. If n is odd, we first add a dummy row $(0,0,\dots,0)$ to the bottom of $X_{n\times m}$. It is equal to add a dummy sensor with energy $E_i = 0$ to the network. That does not affect our optimal solution. Then, we add $Z_{(n+1)\times((n+1)/2-m)}$ to the right hand side of $X_{(n+1)\times m}$, the resulting matrix, denoted by $W_{(n+1)\times(n+1)/2}$, is in the form as

$$W_{(n+1)\times(n+1)/2} = \begin{bmatrix} x_{11}x_{12}\dots x_{1m} & z_{11}z_{12}\dots & z_{1(n+1)/2-m} \\ x_{21}x_{22}\dots x_{2m} & z_{21}z_{22}\dots & z_{2(n+1)/2-m} \\ \dots & \dots & \dots \\ x_{n1}x_{n2}\dots x_{nm} & z_{n1}z_{n2}\dots & z_{n(n+1)/2-m} \\ 00\dots0 & z_{(n+1)1}z_{(n+1)2}\dots & z_{(n+1)(n+1)/2-m} \end{bmatrix}_{(n+1)\times(n+1)/2}.$$

To make matrix W having the feature of (4) and (7), i.e., the sum of each column is two times of the sum of each row, the dummy matrix $Z_{p\times q}$ should satisfy the following conditions:

$$1.\ R'_i = \sum_{j=1}^{q} z_{ij} = L - R_i, \quad \text{for } \forall i = 1,2,\dots,p, \tag{8}$$

$$2.\ C'_j = \sum_{i=1}^{p} z_{ij} = 2L, \quad \text{for } \forall j = 1,2,\dots,q. \tag{9}$$

We propose a simple algorithm to compute the dummy matrix $Z_{p\times q}$. The algorithm starts to assign values to the elements of $Z_{p\times q}$ 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 $i = 1,2,\dots,p$ and $j = 1,2,\dots,q$. Initially, $R_i^- \leftarrow (L - R_i)$ and $C_j^- \leftarrow 2L$, 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 (8) and (9), 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 $2L$, 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 condition (8) 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 condition (9) will be met during the process. Suppose we now come to element z_{ij}, (i.e., elements of z_{kl}, for $k = 1, \dots, i-1$ and $l = 1, \dots, j-1$, 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 (8) 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., $z_{kj} = 0$, $k = 2, 3, \dots, p$. By doing so, all elements of column j have been assigned and condition (9) 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 (8) is met for row i and condition (9) 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 $Z_{p\times q}$ are determined. The details of the algorithm are given below.

```
FillMatrix Algorithm (k = 2)
Input: workload matrix $X_{n\times m}$.
Output: dummy matrix $Z_{p\times q}$.
Begin
  if $(n - 2m)\%2 = 0$
    $p = n$;
  else
    $p = n + 1$;
  $q = p/2 - m$;
  $R_i^- = L - R_i$, for $i = 1$ to $p$;
  $C_j^- = 2L$, for $j = 1$ to $q$;
  $i = 1; j = 1$;
    while$(i \leqslant p)$ && $(j \leqslant q)$ do
      if $C_j^- > R_i^-$ then // determine elements in row $i$.
        $z_{ij} = R_i^-$;
        $z_{ik} = 0$, for $k = j + 1$ to $q$; // 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 $q$;
        $z_{kj} = 0$, for $k = i + 1$ to $p$;
        $i = i + 1; j = j + 1$;
    endwhile
  End
```

The following theorem claims the correctness of the *FillMatrix* algorithm.

Theorem 5. *For a given workload matrix $X_{n\times m}$, FillMatrix algorithm can compute $Z_{p\times q}$, such that the resulting matrix $[X_{p\times m}\, Z_{p\times q}]_{p\times p/2}$ is the workload matrix, where the sum of each column is two times of the sum of each row.*

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 conditions (8) and (9). So we can prove a general case: given row sums R_i' and column sums C_j' of a matrix $Z_{n\times m}$, $i = 1, 2, \dots, n, j = 1, 2, \dots, m$, the proposed algorithm can compute all elements z_{ij} that satisfy conditions (8) and (9). 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 $z_{11} = R_1^- = C_1^- = R_1' = C_1'$. The conditions (8) and (9) are both met.
2. We assume when $n \leqslant p - 1$, $m \leqslant q - 1$, the proposed algorithm can compute $Z_{n\times m}$, such that the conditions (8) and (9) 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 $z_{11} = R_1^-$, $z_{1k} = 0$, $k = 2, 3, \dots, m$ and $z_{k1} = 0$, $k = 2, 3, \dots, n$. 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 the conditions (8) and (9) are both met in row 1 and column 1. The remaining undetermined elements z_{ij}, $i = 2, 3, \dots, n$, $j = 2, 3, \dots, m$, 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 $z_{11} = R_1^-$, $z_{1k} = 0$, $k = 2,3,\dots,m$ and $C_1^- = C_1^- - R_1^-$. For the row 1 where z_{ij} have been determined, we have $\sum_{j=1}^m z_{1j} = z_{11} = R_1^- = R_1'$, condition (8) is met. For the column 1 which is updated, we have $C_1^- + z_{11} = C_1'$, it does not violate condition (9). The remaining undetermined elements z_{ij}, $i = 2,3,\dots,n$, $j = 1,2,3,\dots,m$, 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 the conditions (8) and (9). 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 $R_l^- \geqslant C_1^-$ in round l, we set $z_{l1} = C_1^-$, $z_{k1} = 0$, $k = l+1, l+2,\dots,n$ 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)–(3) with (4), the proposed algorithm can correctly compute all elements in the matrix $Z_{p\times q}$, such that the conditions (8) and (9) are both met.

Theorem 5 is proved. □

Theorem 6. *The time complexity of the FillMatrix algorithm is* $O(n^2)$.

Proof. In *FillMatrix* algorithm, each time we compare R_i^- with C_j^-, and determine elements in row i or column j. Note that the number of R_i^- and C_j^- both are at most $O(n)$. So the total time complexity of the proposed algorithm is $O(n^2)$.

Theorem 6 is proved. □

Integrating together with *FillMatrix* algorithm and *Perfect-2-Matching* algorithm, we have the algorithm of decomposing the workload matrix as follows.

```
DecomposeMatrix Algorithm
Input: the workload matrix $X_{n\times m}$.
Output: a sequence of schedule matrices.
Begin
  if $n > 2m$ then
    Run FillMatrix algorithm to obtain a matrix $W_{p\times p/2}$;
  Construct a bipartite graph $G$ from $W_{p\times p/2}$;
  while there exist edges in $G$ do
    Run Perfect-2-Matching algorithm on $G$ to find a perfect 2-matching $M$;
    Record $P_i$; //$P_i$: the scheduling matrix of $M$;
    Deduct $P_i$ from $W_{p\times p/2}$ and remove edges with weight 0 in $G$;
  endwhile
  Output $W_{p\times p/2} = P_1 + P_2 + \cdots + P_t$;
End
```

Theorem 7. *The total time complexity of the DecomposeMatrix Algorithm is* $O(m^2n^3)$.

Proof. Combining Theorem 4 with Theorem 6, it is proved. □

3.3. Obtain schedule timetable

The above discussion concludes that if each target requires two sensors to watch at anytime, the maximal lifetime scheduling problem can be solved in three steps. First, we compute the upper bound on the maximal lifetime of the system by linear programming technique, and get a workload matrix $X_{n\times m}$. Second, we use the *FillMatrix* algorithm to fill the matrix, such that the resulting matrix $W_{p\times p/2}$ satisfies conditions (4) and (7). Then we use the *DecomposeMatrix* algorithm to decompose the workload matrix $W_{p\times p/2}$ into a sequence of schedule matrices as follows:

$$W_{p\times p/2} = P_1 + P_2 + \cdots + P_t, \tag{10}$$

where P_i is the scheduling matrix which corresponds to a perfect 2-matching. According to Theorem 4, t can be bounded by $n \times m$.

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), $i = 1,2,\dots,t$. We have

$$X_{n\times m} = P_1' + P_2' + \cdots + P_t'. \tag{11}$$

Finally, we have obtained a sequence of schedule matrices by decomposing the workload matrix. Each schedule matrix specifies sensors watching targets for the same period of time (called a session). In fact, there is no need for all sensors to start watching their corresponding targets at the same time, and switch synchronously to other targets (or switch off) at the end of a session. Each sensor's schedule can be independent from the others. That is, sensors can switch on/off and switch to watch other targets asynchronously from each other. To come up with the target-watching timetable for sensor i, we simply take the i-th row of all the schedule matrices, and combine the time of the consecutive sessions that it watches the same target (in this case there is no need for sensor i to switch). Finally, we have an independent timetable for each sensor. Since global clock synchronization is achievable in sensor networks by using some localized method [14] and time synchronization scheme [17], sensors can cooperate correctly according to timetable to achieve the maximal network lifetime.

We will show an example in Section 5 to demonstrate the schedule timetable.

4. General cases $k \geqslant 2$

The maximal lifetime scheduling problem can be solved in general cases $k \geqslant 2$. Similar to the discussion of case $k = 2$, we tackle the problem in three steps.

First, we extend the linear programming formulations. The problem of finding the maximum lifetime for sensors watching targets can be formulated as the following:

$$\begin{aligned} \text{Objective: Max} \quad & L \\ \text{s.t.} \quad & \sum_{i \in S(j)} x_{ij} = kL \quad \forall j \in T; && (12) \\ & \sum_{j \in T(i)} x_{ij} \leqslant L \quad \forall i \in S; && (13) \\ & \sum_{j \in T(i)} x_{ij} \leqslant E_i \quad \forall i \in S, && (14) \end{aligned}$$

where L, x_{ij}, inequalities (13) and (14) all remain the same. Eq. (12) specifies that for each target j in T, since there should be k sensors to watch target j at anytime, the total time that the 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. After solving the above LP formulation, we get L, the upper bound on the lifetime, and a workload matrix.

Second, to decompose the workload matrix and achieve the upper bound, we first extend the definitions of 2-matching, perfect 2-matching and 2-unsaturated to the *k-matching*, *perfect k-matching* and *k-unsaturated*, respectively.

Definition 3. k-matching: Let $G(S \cup T, E)$ be a bipartite graph with bipartition $\{S, T\}$, where $|S| \geqslant k|T|$. k-matching is defined as a subset of edges in E, such that

1. For each vertex in S, no two edges share the vertex.
2. For each matched vertex in T, there are exactly k edges share the vertex.

The k-matching means that 1 vertex in T exactly matches k vertexes in S.

Definition 4. Perfect k-matching: If $|S| = k|T|$, and a k-matching contains $k|T|$ edges, the largest possible, we call it the perfect k-matching.

Definition 5. k-unsaturated: A target t_j is called k-unsaturated to a matching M if and only if there are less than k edges in M that start (end) with t_j.

For the workload matrix computed from (12)–(14), if $n = km$, the sum of each column and the sum of each row are equal to kL and L, respectively. Similar to Theorems 1 and 2, we have following theorems.

Theorem 8. *Let $G(S \cup T, E)$ be a bipartite graph with bipartition $\{S, T\}$, where $|S| = k|T|$. G contains a perfect k-matching if and only if $|N(U)| \geqslant k|U|$ for all $U \subseteq T$.*

Proof. Similar to the proof of Theorem 1, we use induction method. We first consider whether there exist $U \subset T$, $|U| \neq 0$ and $|N(U)| = k|U|$. If it is true, similar to the proof of Theorem 1, we can construct a perfect k-matching of T in G. Otherwise, we consider whether there exist $U \subset T$, $|U| \neq 0$ and $|N(U)| = k|U| + 1$. If it is true, we construct a perfect k-matching. Otherwise, we consider $U \subset T$, $|U| \neq 0$ and $|N(U)| = k|U| + 2$. The above process is continued until (1) we construct a perfect k-matching of T in G; (2) for $\forall U \subset T$, $|N(U)| \geqslant k|U| + k$. We can also construct the perfect k-matching in case two. Theorem is proved. □

Theorem 9. *For any workload matrix, if $n = km$, there exists a perfect k-matching in every round of decomposing process.*

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

Then, we extend the *Perfect-2-Matching* algorithm to the *Perfect-k-Matching* algorithm.

```
Perfect-k-Matching Algorithm
Input: a bipartite graph $G = (S \cup T, E), k$.
Output: a perfect $k$-matching $M$.
Begin
  pick any edge from $E$ and add to $M$;
  while there exist $k$-unsaturated targets do
        select the first $k$-unsaturated target $t$ and label $t$ with $*$;
        for $s_i \in S(t)$ and $(t, s_i) \notin M$
           label $s_i$ with $t$;
        while all newly labeled $s_i \in M$ do // $M$-path is not found so far
           for all newly labeled $s_i$ and unlabeled $t_j \in T(s_i)$
              if $(s_i, t_j) \in M$
                 label $t_j$ with $s_i$;
           for all newly labeled $t_j$ and unlabeled $s_i \in S(t_j)$
              if $(t_j, s_i) \notin M$
                 label $s_i$ with $t_j$;
  endwhile
  // have found an $M$-path which starts with $t$ and ends with $s_i \notin M$.
  remove M-edges in $M$-path from $M$ and add in non-M-edges to $M$;
  endwhile
End
```

Theorem 10. *The Perfect-k-Matching algorithm is correct and the time complexity is* $O(mn^2)$.

Proof. Proof is similar to the proof of Theorem 3. □

If $n > km$, we also need to fill the workload matrix by adding some dummy columns (targets) and rows (sensors). The *FillMatrix* algorithm is extended as follows.

```
FillMatrix Algorithm
Input: workload matrix $X_{n\times m}$, $k$.
Output: dummy matrix $Z_{p\times q}$.
Begin
  if $(n - km)\%k = 0$
     $p = n$;
  else
     $p = n + (k - (n - km)\%k)$;
  $q = p/k - m$;
  $R_i^- = L - R_i$, for $i = 1$ to $p$;
  $C_j^- = 2L$, for $j = 1$ to $q$;
  $i = 1; j = 1$;
  while $(i \leqslant p)$ && $(j \leqslant q)$ do
     if $C_j^- > R_i^-$ then // determine elements in row $i$.
        $z_{ij} = R_i^-$;
        $z_{ik} = 0$, for $k = j + 1$ to $q$; // 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 $q$;
        $z_{kj} = 0$, for $k = i + 1$ to $p$;
        $i = i + 1; j = j + 1$;
     endwhile
  End
```

Integrating together with *FillMatrix* algorithm and *Perfect-k-Matching* algorithm, we have the algorithm of decomposing the workload matrix for general cases as follows.

```
DecomposeMatrix Algorithm
Input: workload matrix $X_{n\times m}$, $k$.
Output: a sequence of schedule matrices.
Begin
  if $n > km$ then
     Run FillMatrix algorithm to obtain a matrix $W_{p\times p/k}$;
  Construct a bipartite graph $G$ from $W_{p\times p/k}$;
  while there exist edges in $G$ do
     Run Perfect-k-Matching algorithm on $G$ to find a perfect $k$-matching $M$;
     Record $P_i$; //$P_i$: the scheduling matrix of $M$;
     Deduct $P_i$ from $W_{p\times p/k}$ and remove edges with weight 0 in $G$;
  endwhile
  Output $W_{p\times p/k} = P_1 + P_2 + \cdots + P_t$;
End
```

Theorem 11. *The time complexity of the DecomposeMatrix algorithm is* $O(m^2n^3)$.

Proof. Proof is similar to the proof of Theorem 4. □

Finally, similar to the case $k = 2$, we obtain a target watching timetable for each sensor based on the schedule matrices.

5. Simulation

5.1. A numeric example

We randomly place 6 sensors (in clear color in Fig. 1) and 2 targets (in grey in Fig. 1) in a 100×100 two-dimensional free-space region. For simplicity, the surveillance range of all sensors is set to 40 (our solution can work for any system with non-uniform surveillance range). Fig. 1 shows the surveillance relationship between sensors and targets, with an edge between a sensor and a target if and only if the target is within the surveillance range of the sensor. The initial energy reserves of sensors, in terms of hours, are random number generated in the range of $[0, 100]$ with the mean at 50, as shown in Table 1. We assume $k = 2$ in this example, that is, each target requires two sensors to watch at anytime.

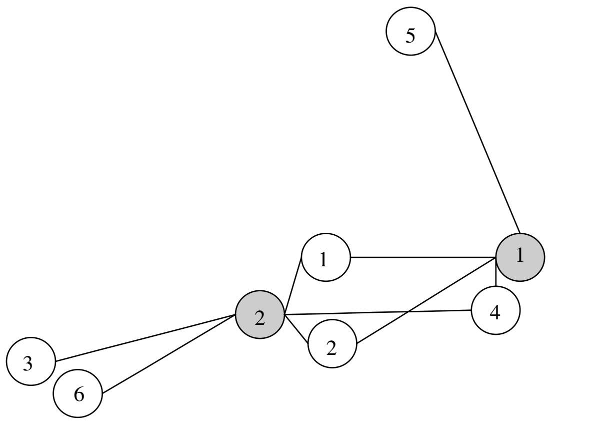

Fig. 1. An example system with 6 sensors and 2 targets ($k = 2$).

Table 1
The initial energy of 6 sensors (h)

	Sensors					
	1	2	3	4	5	6
E_i	68.1335	2.2493	6.9318	3.2419	27.6030	88.9206

We follow the three steps in our method to find the schedule timetable for the sensors.

First, we use the linear programming, described in Section 3.1, to compute the maximum lifetime L and the workload matrix that achieves L:

$$L = 12.4230\ \text{h},$$

$$X_{6\times 2} = \begin{bmatrix} 9.5218 & 2.9012 \\ 1.1751 & 1.0743 \\ 0 & 6.9318 \\ 1.7261 & 1.5157 \\ 12.4230 & 0 \\ 0 & 12.4230 \end{bmatrix}.$$

In the workload matrix, we can see: (1) for each column, the total time for target 1 and target 2 to be watched is 24.8460 h, which is two times of the lifetime of the surveillance system; (2) for each row, the total working time of sensors does not exceed their battery's lifetime.

Second, since $n = 6$, $m = 2$ and $n > 2m$, we run the *FillMatrix* algorithm, proposed in Section 3.2.2, to append a dummy matrix to the workload matrix, such that the sum of each column and the sum of each row are equal to $2L$ and L, respectively:

$$W_{6\times 3} = \begin{bmatrix} 9.5218 & 2.9012 & 0 \\ 1.1751 & 1.0743 & 10.1736 \\ 0 & 6.9318 & 5.4912 \\ 1.7261 & 1.5157 & 9.1812 \\ 12.4230 & 0 & 0 \\ 0 & 12.4230 & 0 \end{bmatrix}.$$

Then we run the *DecomposeMatrix* Algorithm, proposed in Section 3.2.2, to decompose $W_{6\times 3}$ into a sequence of schedule matrices $P_1, P_2, \ldots,$ and P_5 (i.e., the decomposition terminates at round 5), such that

$$W_{6\times 3} = P_1 + P_2 + \cdots + P_5.$$

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

Table 2
The schedule timetable for 6 sensors

Sensors	Watching duty (time duration and watching targets)			
1	0–8.4475 Target 1		8.4475–11.3487 Target 2	11.3487–12.4230 Target 1
2	0–10.1736 Turn off		10.1736–11.3487 Target 1	11.3487–12.4230 Target 2
3	0–1.5157 Turn off	1.5157–8.4475 Target 2	8.4475–12.4230 Turn off	
4	0–1.5157 Target 2	1.5157–8.4475 Turn off	8.4475–10.1736 Target 1	10.1736–12.4230 Turn off
5	0–12.4230 Target 1			
6	0–12.4230 Target 2			

$$X_{6\times 2} = \begin{bmatrix} 1.5157 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1.5157 \\ 1.5157 & 0 \\ 0 & 1.5157 \end{bmatrix} + \begin{bmatrix} 6.9318 & 0 \\ 0 & 0 \\ 0 & 6.9318 \\ 0 & 0 \\ 6.9318 & 0 \\ 0 & 6.9318 \end{bmatrix} + \begin{bmatrix} 0 & 1.7261 \\ 0 & 0 \\ 0 & 0 \\ 1.7261 & 0 \\ 1.7261 & 0 \\ 0 & 1.7261 \end{bmatrix} + \begin{bmatrix} 0 & 1.1751 \\ 1.1751 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1.1751 & 0 \\ 0 & 1.1751 \end{bmatrix} + \begin{bmatrix} 1.0743 & 0 \\ 0 & 1.0743 \\ 0 & 0 \\ 0 & 0 \\ 1.0743 & 0 \\ 0 & 1.0743 \end{bmatrix}.$$

Finally, we obtain target watch timetables for sensors based on the above schedule matrices. The timetable for the 6 sensors is shown in Table 2.

It is easy to see that the timetable in Table 2 satisfies the surveillance conditions that each sensor can watch at most one target at a time and each target is watched by two sensors at anytime.

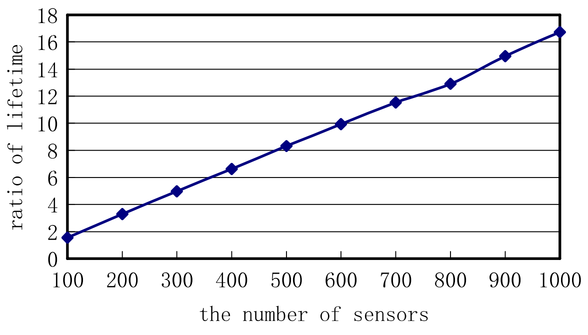

Fig. 2. Ratio of lifetime vs. the number of sensors.

5.2. Experiments

In this section, we study how the network lifetime is affected by varying the number of sensors. We randomly place sensors and targets in a 100×100 two-dimensional free-space region. For simplicity, the surveillance range of all sensors is set to 40. There is an edge between a sensor and a target if and only if the target is within the surveillance range of the sensor. The initial energy reserves of sensors, in terms of hours, are random number generated in the range of $[0, 100]$ with the mean at 50. We assume $k = 3$ in this experiment, that is, each target requires three sensors to watch at anytime. The number of targets is fixed at 20. We vary the number of sensors from 100 to 1000. The results presented in the following figure are the means of 100 separate runs.

We use the metric *ratio of lifetime* to study how the network lifetime is affected by varying the number of sensors. The ratio of lifetime is defined to be the ratio of the maximum network lifetime over the mean lifetime of sensors in the network, such as

$$\text{ratio of lifetime} = \frac{\text{the maximum lifetime}}{\text{the mean lifetime of sensors}}.$$

The simulation results are plotted in Fig. 2.

From Fig. 2, we can conclude that there is a strong linear relationship between network lifetime and the number of sensors in the network. If more sensors are placed in the network, more space is provided to our optimal algorithm to compute longer network lifetime. It means that increasing the number of sensors is one of efficient ways to prolong the network lifetime.

6. Conclusions

This paper addressed the maximal lifetime scheduling problem for k to 1 sensor–target surveillance networks.

Our solution consists of three steps: (1) compute the maximum lifetime of the system and the workload matrix by using linear programming method; (2) extend the Hall's theory and decompose the workload matrix into a sequence of schedule matrices by using perfect *k*-matching method. This decomposition can preserve the maximum lifetime; (3) obtain target watching timetable for sensors. 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. The time complexity of our optimal method is $O(m^2n^3)$, where m, n are the number of targets and the number of sensors, respectively. We illustrated our optimal method by an example in the end. Simulations were further conducted. It showed a strong linear relationship between network lifetime and the number of sensors in the network.

Acknowledgement

This work is supported in part by Research Grants Council of Hong Kong under Grant Number CityU 1149/04E.

References

[1] Mihail L. Sichitiu, Cross-layer scheduling for power efficiency in wireless sensor networks, in: IEEE INFOCOM 2004.
[2] Yang Yu, Bhaskar Krishnamachari, Vikor K. Prasanna, Energy-latency tradeoffs for data gathering in wireless sensor networks, in: IEEE INFOCOM 2004.
[3] D. Tian, N.D. Georganas, A coverage-preserving node scheduling scheme for large wireless sensor networks, in: First ACM International Workshop on Wireless Sensor Networks and Applications, 2002, pp. 32–41.
[4] C.-Y. Chong, S.P. Kumar, Sensor networks: evolution, opportunities, and challenges, Proc. IEEE 91 (8) (2003) 1247–1256.
[5] Ronald Gould, Graph Theory, The Benjamin/Cummings Publishing Company, INC, 1988, pp. 198–199.
[6] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, TAG: a tiny aggregation service for ad-hoc sensor networks, in: Operating Systems Design and Implementation, December 2002.
[7] Reinhard Diestel, Graph Theory, second ed., Springer, 2000, pp. 31–33.
[8] Jean Carle, David Simplot-Ryl, Energy-efficient area monitoring for sensor networks, IEEE Computer 37 (2) (2004) 40–46.
[9] Linnyer B. Ruiz, Luiz Filipe Menezes Vieira, Marcos Augusto Menezes Vieira, et al., Scheduling nodes in wireless sensor networks: a Voronoi approach, in: Proceedings of 28th IEEE Conference on Local Computer Networks (LCN2003), Bonn/Konigswinter, Germany, October 2003, pp. 423–429.
[10] Chih-fan Hsin, Mingyan Liu, A distributed monitoring mechanism for wireless sensor networks, in: International Conference on Mobile Computing and Networking Proceedings of the ACM Workshop on Wireless Security, Atlanta, USA, 2002, pp. 57–66.
[11] Y. Zhao, R. Govindan, D. Estrin, Residual energy scans for monitoring wireless sensor networks, in: IEEE Wireless Communications and Networking Conference, 2002, pp. 356–362.
[12] M. Bhardwaj, T. Garnett, A. Chandrakasan, Upper bounds on the lifetime of sensor networks, in: IEEE International Conference on Communications, 2001, pp. 785–790.
[13] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, John Anderson, Wireless sensor networks for habitat monitoring, in: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, USA, September 2002, pp. 88–97.
[14] Qun Li, Daniela Rus, Global clock synchronization in sensor networks, in: IEEE INFOCOM 2004.
[15] Ting Yan, Tian He, John A. Stankovic, Differentiated surveillance for sensor networks, in: Proceedings of the First International Conference on Embedded Networked Sensor Systems, Los Angeles, USA, 2003, pp. 51–62.
[16] Tian He, Sudha Krishnamurthy, John A. Stankovic, et al., Energy-efficient surveillance system using wireless sensor networks, in: MobiSYS 2004, 6–9 June, Boston, Massachusetts, USA.
[17] K. Römer, Time synchronization in ad hoc networks, in: Proceedings of the ACM Mobihoc, Long Beach, CA, 2001.
[18] Xiuzhen Cheng, Guoliang Xue, Dechang Chen, TPS: a time-based positioning scheme for outdoor wireless sensor networks, in: IEEE INFOCOM 2004.

Hai Liu received his BSc (1999) and MSc (2002) in Applied Mathematics, South China University of Technology, China. He is currently a Ph.D. candidate in Department of Computer Science at City University of Hong Kong. His research interests include distributed systems, wireless networks and mobile computing.

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

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