2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Energy Saving Target Tracking Using Mobile Sensor Networks

Yingying Li and Yun-Hui Liu
Dept. of Mechanical and Automation Engineering
The Chinese University of Hong Kong
Shatin, NT, Hong Kong, China
{yyli,yhliu}@mae.cuhk.edu.hk

Abstract— **This paper addresses the problem of tracking a mobile target using a mobile sensor network while minimizing the energy consumption and maintaining the network connectivity during the tracking process. While minimizing the tracking energy consumption is proved to be NP-complete, an approximately optimal solution named breadth-first leader-follower strategy is presented. Nodes close to the target predicted position find their following nodes based on breadth-first search and lead them to cover the probable region where the target may exist next time instant. Meantime the overall network connectivity can be maintained. We have proved that the energy consumption of the nodes moving under the control of the proposed algorithm is within a scalar factor of the optimal consumption. Simulation has been conducted to demonstrate the performance of the algorithm in different situations. The results show that our algorithm can yield good performance in target tracking while consuming little energy.**

I. Introduction

Due to many attractive characteristics such as wide coverage of the environment, fast response to changes and high reliability for information gathering, sensor networks have been widely used in civil and military applications, such as target tracking, surveillance, environmental control, and health care.

Prior work on target tracking [1][2][3] focused on using static nodes to cover the area where the target moves and determine the target position using the information collected from the nodes in the vicinity of the target. An underlying assumption is that the target is located in the sensing region of at least one node, which is not always correct, especially when the number of nodes is not sufficient to cover the whole area or the environment is unknown. In these scenarios, it is necessary to mount the sensor nodes on mobile robots and make use of their mobility to actively ensure the targets visible to the network all the time.

Compared with static networks, target tracking using mobile networks has received relative little attention. Moreover most researchers have not considered the network connectivity. They assume that there always exists a communication link between any two nodes of the network. Parker [4] combined local virtual forces with high-level behavior-based probabilities to control the node motion. Makarenko [5], Chung [6] and Spletzer [7] got the best next position for the node by optimizing different functions such as fused mutual information gain, position uncertainty matrix, etc. Jung [8] distributed robots according to the target density in different regions. In these papers, the node motion relies on the information collected from all the other nodes. However the complete communication graph can not be obtained in reality due to the limited communication range. Recently Shucker [9] simplified the network into a virtual spring mesh and used the incident edges of every node to generate the control force. Cortes [10] proposed the gradient descent algorithm for a class of utility functions which encode optimal coverage and sensing policies. Here every node only communicates with its neighbors and the network connectivity is guaranteed during the deployment.

However, little research has considered the energy conservation problem in target tracking. A critical issue in the sensor network is power scarcity because of battery size and weight limitations. Inefficient algorithm will lead to a short system lifetime. For mobile sensor networks, node motion is the most important cause for energy dissipation compared with the energy consumed by the communication and sensing components. Therefore the deployment strategy should be designed carefully to minimize the motion energy consumption while achieving the tracking task.

This paper formulates the minimum energy target tracking problem and demonstrates that it is NP-complete. An distributed algorithm called breadth-first leader-follower is proposed to find a near-optimal solution. The algorithm adopts an iterative process. At every iteration, which corresponds to every time instant, the algorithm first predicts a probable region where the target may move to at the next time instant. Second, it detects the nodes close to the probable region, treats them as leaders of the network and optimizes their velocities tracking motion of the target. Then, the motion is propagated to other sensor nodes in the sensor network using the breadth-first search until it finds that no other nodes need to be moved. In this fashion, the network connectivity can be always maintained and the target can be always in the sensing region of the sensor network. In addition, we proved that the energy consumption of the nodes under the control of the proposed method is within a scalar factor of the minimum consumption. Simulations have confirmed the performance of the algorithm. As far as our knowledge is concerned, this

This work is supported in part by Hong Kong RGC under grant 414505 and China NFSC under grant 60334010/60475029. It is also affiliated with the Microsoft-CUHK Joint Laboratory for Human-centric Computing and Interface Technologies.

is the first paper that considered the energy minimization problem of target tracking using mobile sensor networks.

This paper is organized as follows. Section II describes the problem and its mathematical model. Section III presents the breadth-first leader-follower algorithm and Section IV evaluates its performance in different situations. Finally Section V concludes the paper.

II. Problem Definition

We first describe some related graph theory. Then a mathematical model is set up based on logical assumptions.

A. Assumptions

1) The sensing and communication region of the node are modeled as circular discs. Let R_S and R_C denote the radii of the sensing and communication regions respectively.

2) The sensor network is abstracted into an undirected graph $G=(V,E)$. V is the set of nodes. The edge $(n_i, n_j) \in E$ connects node n_i to node n_j if n_i is in the communication region of n_j.

3) The initial network is connected.

4) Every node knows its position accurately.

B. Problem Definition

The moving target tracking [12] can be represented by state-space and observation equations in the following form

$$\begin{aligned} x_t &= f_t(x_{t-1}, w_{t-1}) \\ z_t &= h_t(x_t, v_t) \end{aligned} \quad (1)$$

where z_t is the observation vector, x_t is the state vector, $h_t()$ is the measurement function, $f_t()$ is the system transition function, w_t and v_t are noise vectors, and the subscript t denotes time index. The rule of Bayesian filtering is a two-step process

$$\begin{aligned} p(x_t / z_{1:t-1}) &= \int p(x_t / x_{t-1}) \cdot p(x_{t-1} / z_{1:t-1}) dx_{t-1} \\ p(x_t / z_{1:t}) &= \frac{p(z_t / x_t) \cdot p(x_t / z_{1:t-1})}{p(z_t / z_{1:t-1})} \end{aligned} \quad (2)$$

That the sensing model is ideal means $q_t=0$ and the upper equations are transformed into

$$\begin{aligned} p(x_t / z_{1:t-1}) &= \int p(x_t / x_{t-1}) \delta(x_{t-1} - x^*_{t-1}) dx_{t-1} = p(x_t / x^*_{t-1}) \\ x^*_t &= h_t^{-1}(z_t) \end{aligned} \quad (3)$$

where x^*_t is the position of the target. The probable region is the area where the target may exists at the next time instant, which is decided by $p(x_t / x^*_{t-1})$. This region must be covered by the nodes beforehand to avoid the target escape. Once the target moves into this region, some node will detect it and x^*_t can be computed directly from z_t.

Our object is to minimize the summation of the moving distance of all the nodes to cover the probable region. Its mathematical model is defined as follows

Notations:

$\{pos_i(t)\}_{i=1}^N$: the position of node n_i at time t

$\{vel_i(t)\}_{i=1}^N$: the velocity that node n_i take at time t

M: the probable region of the target at the next time instant.

t: the current time instant.

Δt : the time interval.

$S_i(t)$: the sensing region of node n_i at time t.

Minimize $\sum_{l=t}^{t+\Delta t} \sum_{i=1}^{N} \| vel_i(l) \|$

subject to that the graph connectivity is maintained and the probable region of target is fully covered by the sensing region of the sensor network, i.e. $M \subseteq \bigcup_i S_i(t+\Delta t)$

Fig. 1. Problem definition of target tracking

Here Δt is a constant, so minimizing the moving distance is equal to minimizing the velocity sum of all the nodes.

C. NP-complete Property of the Problem

Next we prove this problem is NP-complete [13]. We consider a special case that the communication range is large enough to maintain network connectivity all along wherever nodes move. Then the node can move directly to any position in the probable region M. We partition M into a large number of small grids X and assume that the nodes can only be located at the centers of the grids. If the partition is fine enough, placing nodes on the grids is the same as the original problem. The sensing circle encompasses a set of grids which form a subset S of X. The elements of S are determined by R_S and the position of the node. Then our problem can be reduced to a minimum distance set cover problem, described as follows:

Minimum Distance Set Cover (MDSC) Problem: Let X be a set of grids g_j $(j=1,\ldots, E)$ to be covered by the sensor network with nodes n_i $(i=1,\ldots,N)$. The distance is denoted by $d_{i,j}$ if n_i moves to grid g_j. Denote the region covered by node n_i at grid g_j by S_j. Let a number $b \in R^+$. Is there a motion strategy $n_i \xrightarrow{d_{i,j}} g_j$ such that $X \subseteq \bigcup_j S_j$ and $\sum d_{i,j} \le b$?

We prove NP-completeness of the *MDSC* problem by reduction from the minimum weight set cover problem (*MWSC*), which is well known to be NP-complete [13].

Minimum Weight Set Cover (MWSC) Problem: U is a finite set and I denotes a family of subsets of U. Any element of U belongs to at least one subset in I. Each subset in I has a weight. Is there a subset $T \subseteq I$ whose members cover U and whose weight sum is less than any number?

Theorem 1. The *MDSC* problem is NP-complete.

Proof: It is easy to see that the *MDSC* problem belongs to the NP class since we can verify in polynomial time whether each element in X is covered by at least one node and whether the sum of the moving distance is $\le b$.

In the optimal solution two nodes can not locate at one grid for that they cover the same subset of X and if one of them is removed, X is still covered and $\sum d_{i,j}$ become smaller, which is contrary to that the solution is optimal. And every node can not move to two grids at the same time. Then we construct a matrix

$$\begin{pmatrix} (n_1, g_1) & (n_1, g_2) & \cdots & (n_1, g_E) \\ (n_2, g_1) & (n_2, g_2) & \cdots & (n_2, g_E) \\ \vdots & \vdots & \ddots & \vdots \\ (n_N, g_1) & \cdots & \cdots & (n_N, g_E) \end{pmatrix}$$

Each element is a subset of X and has a weight $d_{i,j}$. Our object is to find a set of matrix elements so that any two of them exist in different lines and columns, their union includes X and their weight sum is less than b.

Now we show that $MDSC \leq_P MWSC$. Let C denotes the collection of subsets in the *MWSC* problem. Then we construct a matrix A whose elements of every line are the same and correspond to one subset of C. It is easy to see that this can be done in polynomial time.

Suppose *MWSC* has a solution of a set cover C'. By our construction, C' corresponds to a set of lines of matrix A. We choose the diagonal elements of these lines. They must exist in different columns and lines, which satisfy *MDSC*. Now suppose *MDSC* has a solution of a set cover A'. Then the elements of this set cover must exist in different columns and lines of A. Because every line of A corresponds to one subset of C, A' is also a solution of *MWSC*. This concludes the proof. The special case is NP-complete, so the initial problem is also NP-complete and we can not find an optimal solution in polynomial time currently.

Although the problem is NP-complete, in next section we present a distributed strategy named breadth-first leader-follower algorithm which is similar to the greedy algorithm [14]. Every node makes the choice that looks best at the moment.

III. Breadth-First Leader-Follower Algorithm

A. Graph Connectivity and Breadth-First Search

Graph G is connected if there is a path from any node to any other node [11]. Connectivity directly influences the efficiency of information routing and dissemination in the network. It will consume much energy if all the adjacent edges of every node must be maintained during node motion. Here we use tree as the underlying graph to be maintained because it is the sparsest and connected sub-graph of G.

Given a distinguished source node s of G, the distributed breadth-first search (BFS) [14] produces a breadth-first tree with root s that contains all reachable nodes. To keep track of progress, breadth-first search colors each node white, gray, or black. All nodes start out white. If $(u, v) \in E$ and node u is black and node v is white, then node v become gray. Gray nodes may have some adjacent white nodes; they represent the frontier between discovered and undiscovered nodes. After all the gray nodes adjacent to black nodes are discovered, they become black. Then the loop repeats. Gray and black nodes, therefore, have been discovered, but breadth-first search distinguishes between them to ensure that the search proceeds in a breadth-first manner.

```
BFS(G, s)
for each vertex u ∈ V (G) - {s}
        do color[u] ← WHITE
color[s] ← GRAY
Q ← Ø
Enqueue(Q, s)
while Q ≠ Ø
        do u ← Dequeue(Q)
            for each v ∈ Adj[u]
                do if color[v] = WHITE
                        then color[v] ← GRAY
                             Enqueue (Q, v)
            color[u] ← BLACK
```

Here Q denotes a queue, *Enqueue(Q)* and *Dequeue(Q)* denote the first-in and first-out functions respectively.

B. Breadth-First Leader-Follower Algorithm

Given the target position x^*_{t-1}, we use $x_t = f_t(x^*_{t-1}, w_{t-1})$ to predict the probable region of the target next time. Here we assume the distribution of noise w_t is Gaussian and the probable region is considered as a circle with the center pos_T at the mean of $p(x_t / x^*_{t-1})$.

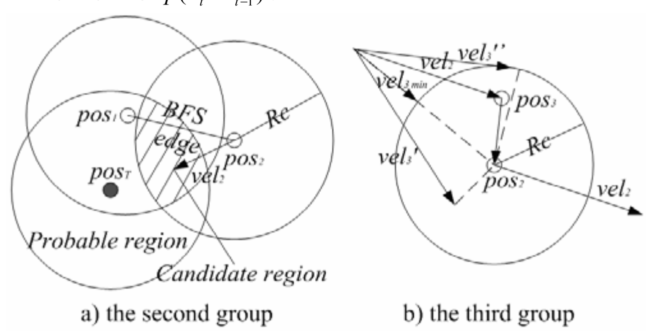

Fig. 2. Breadth-first leader-follower algorithm

Every node has to complete two tasks: maintaining connectivity with its leader and covering the probable region. Our basic idea is that nodes close to the target predicted position lead faraway nodes to move towards the position until the probable region is covered and try to maintain the motion trajectory of every node to be an approximately straight line.

The whole algorithm is an iterative process and in every loop all the nodes are classified into three groups whose motion strategy is described as follows:

1) The nodes that have been leaders in preceding loops do not move. They correspond to the black nodes in BFS.

2) The one-hop neighbors of nodes in the first group, which are the grey nodes of BFS, become new leaders in current loop. They choose their optimal position in the candidate region (the shadow area in Fig. 2a). Here pos_1, pos_2, pos_3 and vel_1, vel_2, vel_3 denote the position and velocity of the nodes in the three groups. The candidate region is the intersection area of the probable region, the communication range of it and the communication range of its first group neighbor. The reason is that for distributed algorithm every node only knows the information of its one-hop neighbors and also has to maintain connectivity with its neighbors.

Fig. 3 shows how to choose the best position for the leader in the candidate region. We partition the candidate region into extremely small grids and each grid is assigned a weight. The nodes can only be located in the center of grids. A set of surrounding grids are covered if the node moves to a certain grid. We sum up the weight of the covered grids as the priority of this grid and choose the grid with highest priority as the node next position. To avoid unnecessary motion, the inner area should have higher priority than the outer area so that the inner area can be covered first. So the principle of grid weight assignment is that the grids closer to pos_T has higher weight (a monotonic decreasing function $f(d)$, d is the distance between the grid and pos_T, $f(0)=1$), and the weight of the grids that have been covered by other nodes are set to zero.

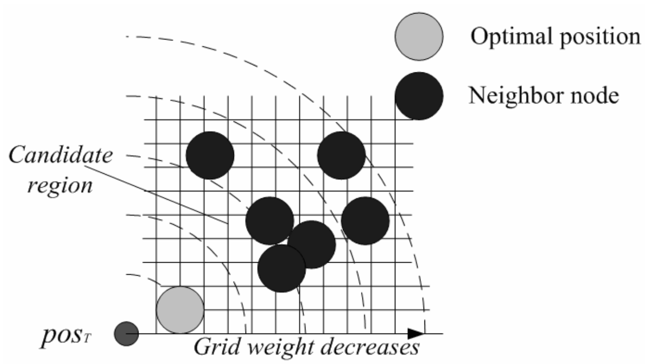

Fig. 3. Partition of the candidate region

For example, the grey circle in Fig. 3 represents the optimal position. If the sensing ranges of nodes are different, the leader should know the sensing range of its neighbors and the sizes of black circles in Fig.3 become different. As the first leader in current loop, it sends its velocity to adjacent nodes which belong to the rest group.

3) The rest nodes which correspond to the white nodes of BFS move to maintain connectivity with their leaders (Fig. 2b). They choose minimum velocities needed to maintain connectivity with their leaders and become new leaders for their following nodes. Here $vel_3^{'}$ and $vel_3^{''}$ denote different values of vel_3. The minimum vel_3 to satisfy $\|pos_2 + vel_2 - (pos_3 + vel_3)\| \le R_C$ is

$$vel_{3\,\min} = (pos_2 - pos_3 + vel_2) * \frac{\|pos_2 - pos_3 + vel_2\| - R_C}{\|pos_2 - pos_3 + vel_2\|} \,. \quad (4)$$

Notations:

$\{g_k\}_{k=1}^{K}$: the position of grid k in candidate region

$\{w_k\}_{k=1}^{K}$: the assigned weight of grid k

At node n_i at every iteration
  Reset vel_i to zero and update its neighbor list
  If n_i is not a leader in preceding loops
    If one of its neighbors is the leader in preceding loops
      vel_i =*genLeaderVel*()
      Send vel_i and pos_i to its other neighbors
    Else if $vel_i = 0$
      Wait to receive velocity from its neighbors
      Compute vel_i as a follower using equation (4)
      (Here vel_l is the first velocity the node receives)
      Send vel_i and pos_i to its neighbors
  Wait for a certain time and move n_i according to vel_i
  Loop again

Function genLeaderVel()
  Find the candidate region and partition it into K grids
  For $k=1...K$
    If $\|g_k - pos_T\|$ >range of the probable region or grid k has been covered by any other node
      $w_k = 0$
    Else
      $w_k = f(\|g_k - pos_T\|)$
  For $k=1...K$
    Sum the weight of the grids covered if the node is located in grid k
  Choose the grid $g_{k\min}$ whose weight summation is maximal
  Return $vel_i = g_{k\min} - pos_i$

Fig. 4. Procedure of the breadth-first leader-follower algorithm

C. Optimal Analysis of the BFLF Algorithm

Definition 1. The *motion expense* of a mobile sensor network is defined as the maximum summation of the moving distance of the network to maintain all the edges if any node moves the distance R_C. Its maximum value is $N*R_C$. It relates to many factors such as the node degree and the node density.

Theorem 2. The BFLF algorithm returns a connected sensor cover with the moving distance of at most $r*|C^*|*H(\max|S|)$, where C^* is the set of optimal sensor cover (not necessary connected), *H(rank)* is the harmonic number $H(rank)=\sum_{i=1}^{rank} 1/i$ [17], S is the set of uncovered grids in the sensing region of every node in C^* before deployment.

Proof: Let C be the set of leader nodes returned by our algorithm and C^* be an optimal set. The probable region is partitioned into small grids X. When a leader is added to C in every loop, we assign a cost to it and spread this cost evenly over the grids covered for the first time. Here the cost is the moving distance of all the nodes driven by this leader.

Let c_x denote the cost allocated to grid $x \in X$, then the cost of the set returned by our algorithm is $\sum_{x\in X} c_x$ and the cost assigned to C^* is $\sum_{S\in C^*}\sum_{x\in S} c_x$. Since each $x \in X$ is in at least one set $S \in C^*$, we have

$$\sum_{S\in C^*}\sum_{x\in S} c_x \ge \sum_{x\in X} c_x \,. \quad (5)$$

Let us consider a node i in C^* and compute the maximum cost accumulated by its sensing region S_i during the entire course of the algorithm. In every loop, some uncovered grids in S_i get covered by the leader node. Let e_j be the number of uncovered grids after the j^{th} loop. The number of uncovered grids in S_i covered during that loop is $e_{j-1} - e_j$.

If D_j is the moving distance of all the nodes in the j^{th} loop and E_j is the number of uncovered grids covered by the leader, after T loops the total cost accumulated by S_i is

$$\sum_{x \in S_i} c_x = \sum_{j=1}^{T} (e_{j-1} - e_j) * D_j / E_j \ . \qquad (6)$$

In every loop, the grid of maximum weight sum is chosen as the next position, so $E_j \geq e_{j-1}$. If the moving distance of the leader is d_j, $d_j \leq R_C$. Then

$$D_j \leq r \Rightarrow D_j / E_j \leq r / e_{j-1} \ . \qquad (7)$$

We now bound this quantity with (6) as follows

$$\sum_{x \in S_i} c_x \leq r * \sum_{j=1}^{T} (e_{j-1} - e_j) / e_{j-1} \leq r * H(|S_i|) \ . \qquad (8)$$

From inequalities (5) and (8), it follows that

$$\sum_{x \in X} c_x \leq r * \sum_{S \in C^*} H(|S|) = r * |C^*| * H(\max |S|) \ . \qquad (9)$$

Therefore, the proof is completed.

IV. Performance Evaluations

We have evaluated the BFLF algorithm from two aspects: the deployment quality which is measured by the coverage percent over the probable region and the deployment energy consumption which is measured by the moving distance.

The algorithm is implemented in *Matlab*. R_S and R_C are set to 1 and 5 respectively. The range of the probable region is 3. Mobile nodes are distributed over a 20*20 field. Fig. 8 shows how the algorithm works. The red circle is the target probable region next time and the little black circle is the sensing region of the node. The red line is the target predicted trajectory and the blue lines are the RNG [13] edges of the network graph.

In the next sections, the simulation results under different situations are presented to demonstrate the performance of the algorithm.

A. Monotonic Function for Grid Weight Assignment

The properties of the monotonic decreasing function adopted in the grid weight assignment directly affect the performance of the algorithm. Fig. 6 shows the coverage percent and the moving distance when the weight function is *f(d)=(1-d/D)^5, 1, 1-d/D, 1-(d/D)^5* respectively. *D* is the range of the probable region.

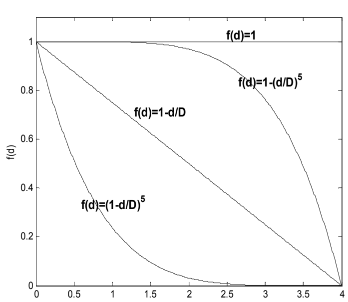

Fig.5. Different monotonic decreasing functions

We can see that large difference between the weight of inner grids and outer grids may lead to overlap of nodes around the center of the probable region. So the coverage percent of the concave function is smaller than that of the convex function after same number of loops. But the limit of the convex function *f(d)=1* is not suitable because it can not drive inner nodes to move towards the center, which leads to slow congregation of nodes. There is no distinct difference in the moving distance.

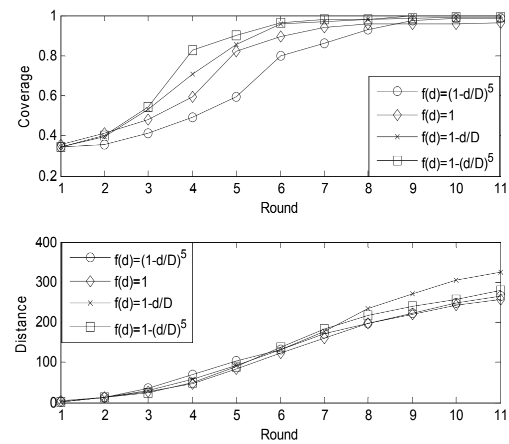

Fig.6. Coverage and distance using different grid weight functions

B. Distance Divisor

It always happens in the BFLF algorithm that different nodes may happen to choose a same grid as their next position in the same loop. To reduce this occurrence probability, we add a distance factor to the grid priority computation. Instead of only using the weight summation value of the covered grids to choose the best, we divide the summation by a monotonic increasing function *q(d)* whose variable is the distance between the grid and the node. It means that the grid closer to the node is preferred when other grids provide similar increase of cover area.

Fig. 7 shows the performance when *q(d)=1, d*0.5, d+1, d*2+1*. Here *q(d)=1* means no distance factor. We can see that the coverage percent increases when the influence of distance is considered.

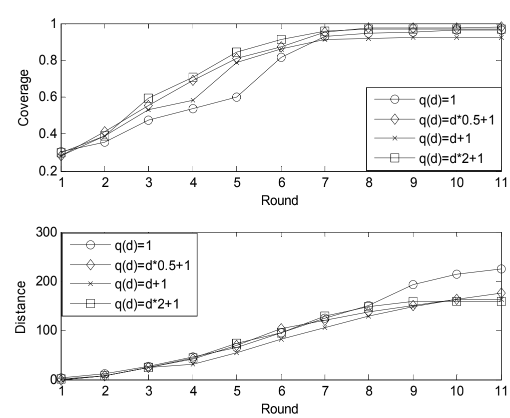

Fig.7. Coverage and distance using different distance functions

C. Grid Size

Fig. 9 shows the performance when the grid size is 1, 1/2, 1/4 and 1/8. Rough partition lead to overlap of nodes and fine partition find more precise cover. This explains why the

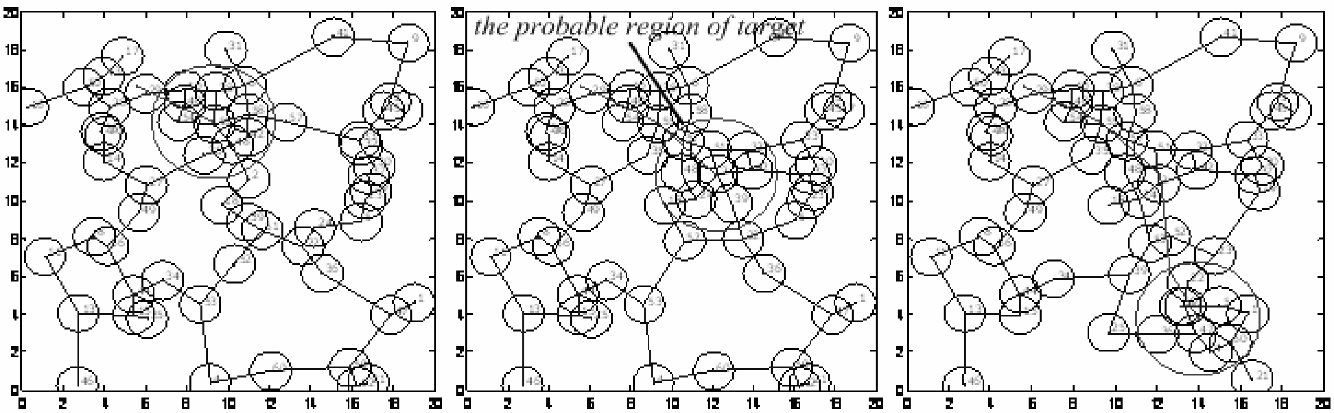

Fig.8. Snapshot of the execution of the breadth-first leader-follower algorithm

coverage percent is higher when the grid is smaller and the moving distance also increases after certain number of rounds. If we do not need precise cover of the probable region, smaller grids will generate better coverage at almost the same cost of moving distance after same number of loops. But this also increases the computation load of system.

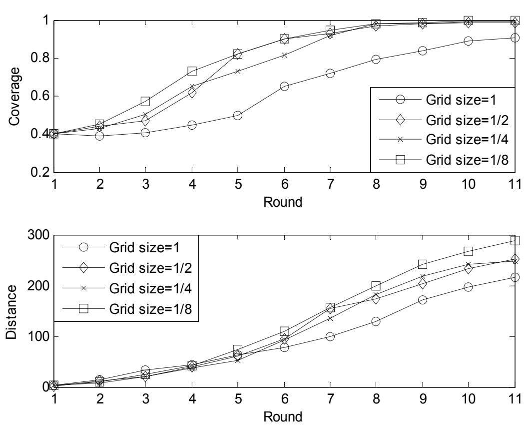

Fig.9. Coverage and distance using different grid sizes

V. CONCLUSION

In this paper, we proposed an approach to trace mobile targets using an active sensor network while saving energy consumption. We first proved that the problem of minimizing the energy consumption of target tracking is NP-complete, and then presented a breadth-first leader-follower algorithm. The algorithm selects the leaders as the nodes which are close to the probable region where the moving target may exist at the next time instant and optimizes their motions to the probable region by minimizing the energy consumption. The motions of the leaders are propagated to other nodes of the sensor network also in an energy saving fashion. The algorithm can always maintain the overall network connectivity. It has been proved that the energy consumption of the network using the proposed approach is within a scalar factor of the minimum energy consumption. Simulation has been conducted to demonstrate the performance of the algorithm.

REFERENCES

[1] F. Zhao and J. Shin, "Information driven dynamic sensor collaboration for tracking applications, " IEEE Signal Processing Magazine, vol. 19, pp. 61–72, 2002.

[2] R. Brooks, C. Griffn, and D. S. Friedlander, "Self-organized distributed sensor network entity tracking," International Journal of High Performance Computer Applications, vol. 16, pp. 207-220, 2002.

[3] W. Zhang and G. Cao, "DCTC: Dynamic convoy tree-based collaboration for target tracking in sensor networks," IEEE Transactions on Wireless Communication, vol. 3, pp. 1689-1701, 2004.

[4] L. E. Parker and B. A. Emmons, "Cooperative multi-robot observation of multiple moving targets, " Proceedings of IEEE International Conference on Robotics and Automation, vol. 3, pp. 2082-2089, 1997.

[5] A. Makerenko, E. Nettleton, B. Grocholsky, S. Sukkarieh, and H. Durrant-Whyte, "Building a decentralized active sensor network," Proceedings of IEEE International Conference on Advanced Robotics, pp. 332-337, 2003.

[6] T. H. Chung, V. Gupta, J. W. Burdick, and R. M. Murray, "On a decentralized active sensing strategy using mobile sensor platforms in a network, " Proceedings of IEEE International Conference on Decision and Control, vol. 2, pp. 1914-1919, 2004.

[7] J. R. Spletzer and C. J. Taylor, "Dynamic sensor planning and control for optimally tracking targets, " International Journal of Robotics Research, vol. 22, pp.7-20, 2003.

[8] B. Jung and G. Sukhatme. "Tracking targets using multiple robots: the effect of environment occlusion, "Journal of Autonomous Robots, vol. 12, pp. 191-205, 2002.

[9] B. Shucker and J. K. Bennett. "Target tracking with distributed robotic macrosensors." Proceedings of MILCOM, 2005.

[10] J. Cortés, S. Martínez and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions, " ESAIM: Control, Optimisation and Calculus of Variations, vol. 11, pp. 691-719, 2005.

[11] R. Diestel, Graph Theory[M]. 2nd edition. Springer-Verlag, 2000.

[12] C. Kreucher, K. Kastella, and A. Hero, "A Bayesian method for integrated multitarget tracking and sensor management," Proceedings of IEEE Inform. Fusion, vol. 1, pp.704-711, 2003.

[13] X. Y. Li, "Algorithmic, geometric and graphs issues in wireless networks," Wireless Communications and Mobile Computing, vol. 3, pp. 119-140, 2003.

[14] M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman and Company, 1979.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms. The MIT Press, 2nd edition, 2001.