A Genetic-Algorithm Based Mobile Sensor Network Deployment Algorithm

EE382C: Embedded Software Systems

Final Report

Yulai Suen

Department of Electrical and Computer Engineering

The University of Texas at Austin

Abstract. This paper describes a genetic-algorithm (GA) based deployment algorithm of mobile sensor network. The algorithm is designed for real-time online deployment for maximum coverage of the environment. The paper presents the details on the algorithm and the implementation, including the major components in our design: recombination, mutation, and the fitness function. The algorithm considers power metrics of the nodes for real-time planning of the next movement. The algorithm was implemented with Java Genetics Algorithm Package [4] and simulated with Network Simulator 2 [5] for performance evaluation. The simulation showed that the algorithm helped the network to avoid local maxima in coverage.

1. INTRODUCTION

A mobile sensor network is composed of a collection of nodes that has sensing, computation, communication, and locomotion capabilities [1]. This paper aims to describe an algorithm for mobile sensor network deployment in an unknown environment by a real-time genetic algorithm (GA). The algorithm allows the network to achieve maximum coverage with minimum energy consumption. It is applicable in both a dynamic environment and a dynamic network topology.

Genetic algorithm was first proposed by Goldberg *et al* in 1989 [3]. It has wide applications in model checking and satisfiability (SAT) problems. The advantage of GA is that the process is completely automatic and avoids local maxima. GA consists of three important components: recombination, mutation, and fitness evaluation. In particular, the fitness function in this algorithm considers the nodes' power metrics, which is the fundamental limitation on both embedded systems and sensor networks. The algorithm is based on the assumption that each node is equipped with a sensor, such as a scanning laser range-finder and omni-camera, which provides the node with the relative distance and bearing of the nearby nodes and obstacles.
The following sections of this paper describe these components in details and the adjustments to general GA for real-time applications. At the end, the paper presents the result of simulations using a discrete-event simulator.

2. RELATED WORK

The deployment problem that this paper addresses is the blanket coverage problem described by Gage [2]. In blanket coverage, the objective is to achieve a static arrangement of nodes that maximize the total detection area. Howard *et al* used potential field techniques and spread the nodes over the environment by driving them with a virtual "force" [2]. The GA approach in this paper achieves a similar repulsive behavior in spreading the nodes and obstacle avoidance. However, this algorithm avoids the local optima problem faced by many other algorithms [6], [7] and [8].

Other researchers also considered using artificial intelligence in sensor network deployment. For example, Haynes compared a few search algorithms including GA and measured their performance in coverage. However, like many other researchers, he assumed that the network has a complete model of the environment and that only offline-planning is required for deployment.

3. CONTRIBUTION

Mobile sensor networks usually face two limitations. First of all, the environment is dynamic. The position of other mobile identities and the geographical features of the environment are usually dynamic. This makes offline planning of deployment by searching through the static map of the environment very inefficient and inaccurate. Second, sensor networks are very sensitive to power consumptions because they are usually designed for applications that run for a long time, e.g. a surveillance system. The embedded systems of the mobile nodes also constrain the available power reservation.

To solve these problems, the algorithm this paper presents contributes in the following ways:

1. Provision of an online GA-based deployment algorithm that is interactive with the dynamics of the environment.
2. Taking into account the power consumption known to a node itself as a consideration when deciding on the locomotion strategy to monitor the power consumption during deployment.
3. Randomization at different stages allows the deployment to converge to a static global optimum in coverage.

4. ALGORITHM

In our algorithm, the network consists of two domains: a server, and clusters of nodes. The server assigns a base-station in each cluster. The base-stations help to monitor the deployment processes from a global point of view. In the following sections, we will describe five important components of the algorithm in detail. Fig 1 is the overview of the algorithm.

A. Initialization

At this stage, the server has to set t_m which serves as an upper bound to the run-time of the algorithm. The users can also specify the target coverage. Then, the direction vector (DV) and the base station set (BSS) are chosen. From the GA point of view, the DV and the BSS are the chromosomes, which carry the genetic information over generations. The assignment to DV and BSS are random.

```
Deployment(MANET)

1. Initialization
   a. direction vector (DV)
   b. base station set (BSS)
   c. $t_m$; default: infinity
   d. $coverage_{target}$; default: min($\sum_i^n cvble[i]$, field_area)
2. Sensor Data Collection
   after $t_m$, each node sends a sensor vector to its router
   sensor_vector = {id, rx_power, energy, coverage}

3. Recombination
   base-stations sort the sensor vector by the fitness function,
   recombines the fittest chromosomes, and send the new chromosomes
   back to the nodes

4. Mutation
   if $\sum_i^n cvg[i][r]$ - $\sum_i^n cvg[i][r-1]$ $\le$ $coverage_{delta}$
      mutate DV and BSS
   decrement of $t_m$ by $t_{delta}$

5. Termination
   if $t_m$ = 0 or $\sum_i^n cvg[i][r]$ $\ge$ $coverage_{target}$
      terminate
   else
      repeat 2 - 5
```

Figure 1. Overview of GA-based mobile sensor network deployment

I. Direction Vector

DV is the probabilistic model of a node moving towards different directions. It has `d` elements, each representing the probability of the movement to a direction `d_i`. The summation of the probabilities of one direction vector is 1.0. The difference in direction between each consecutive element is called direction resolution (DR). DR is determined by the locomotion capability of the nodes, for example, the resolution of the steering servo-motor of the onboard odometer.

II. Base Station Set

The algorithm assumes that all nodes are homogeneous. Therefore, the server randomly assigns the initial base-station in each cluster. BSS is the set of all base-stations in the network. The server also decides on the size of the BSS given the number of nodes in the network.

B. Sensor Data Collection

The nodes collect the data from the network and the environment in the form of sensor vector (SV). The sensor vector (SV) consists of the node's id and energy reserve level, the receiver power from the base-station and the nearby nodes, and the coverage data from the sensor. The coverage data is based on the node's knowledge of nearby nodes. If a node notices that an area is covered by *n* nodes, then the coverage of this node is *coverageArea / n*.

C. Recombination

The DV of the fittest nodes in a cluster are chosen to recombine. The probabilities are recalculated to produce the next generation chromosomes. The fitness function we use will reward and penalize the chromosomes based on the sensor vector.

Fitness Function

The function rewards the increase in coverage and decrease in receiver power while it penalizes the decrease in energy reservation. Note that the function abandons the receiver power when it drops beneath a lower bound below which there would be packet loss. Chromosomes with higher fitness values are chosen to recombine for the next generation. Based on the probability model of the fittest nodes, the bad nodes adjust their DV to produce the next generation.

$$
\text{fitness(i)} = \begin{cases} \text{(coverage[i][r] - coverage[i][r-1])} \\ \text{+ (rx\_power[i][r-1] - rx\_power[i][t])} \\ \text{- (energy[i][r-1] + energy[i][r-1] - energy}_m\text{)} & \text{for rx\_power[i][t]} \geq \text{rx\_power} \\ -\infty & \text{for rx\_power[i][t]} < \text{rx\_power}_m \end{cases}
$$

where coverage is the coverage area,
rx_power is the receiver power,
and energy is the remaining energy.

Figure 2. Fitness function.

D. Mutation

When the deployment reaches a static equilibrium, the nodes would randomly mutate the DV and BSS. Some nodes would randomly vary the probability assignment of the DV. The server will also randomly re-assign some of the base-stations. Through this process we can “shake” the distribution of the nodes over the field to reach a better global optimum. There is also mutation on the probabilistic model at the recombination stage. The degree of mutations is determined by the mutation rate.

E. Termination

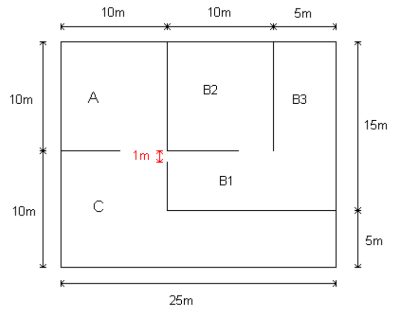
The number of generations of the evolution is bounded by t_m. After each generation, t_m is reduced by t_{delta} so that the nodes would collect sensor vector more frequently as one generation passes the information to the next. This allows a finer adjustment to the deployment in order to reach a static equilibrium. If we allow the algorithm to run for an ideally infinite time, it may achieve a better coverage. However, depending on the situation, the users may like to reserve more energy for tasks after the deployment. The t_m parameter allows the users to put an upper bound on the deployment time. The algorithm terminates after t_m becomes 0 or when the coverage reaches the target coverage.

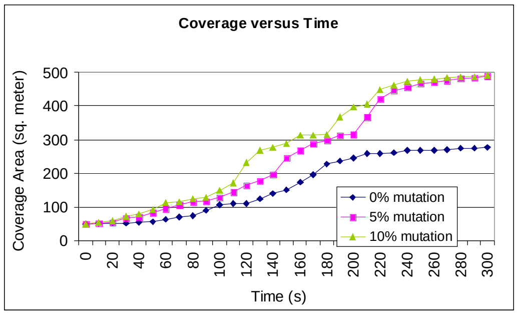
7. IMPLEMENTATION

There are two parts in the implementation: GA, and network simulation. We used Java Genetics Algorithms Package (JPAG) [4] for implementing the GA portion of the deployment, including the fitness function, recombination, and mutation. Our software produces a trace of the node movement and collects the coverage data in the simulation. We transform this data into traffic-pattern and node-movement files readable by Network Simulator 2 (NS2) [5]. NS2 can visualize the node movement in an animator and calculate the energy consumed in the wireless communication.

8. SIMULATION AND RESULTS

The simulation was run in a virtual environment showed in Figure 3a. In this environment, we introduced a possible local maximum by placing a one-meter door gate between area C and area B1 as illustrated. Figure 3b shows the result of the simulation.





a. Simulation environment. b. Simulation result with different mutation rate.
Figure 3. Simulations

A. Coverage

We randomly placed a wired server and 100 wireless nodes into area A. The server assigned 10 clusters each with one base-station. We assumed that each node carried a laser-range finder and an omni-camera with maximum range at four meters. The maximum speed of the nodes was 0.5m/s. The mission of the network was to cover the entire environment with an area of 500m^2. The following table shows the initial values of the parameters.

t_m	60s
t_{delta}	6s
$coverage_{target}$	500m^2

Table 1. Initializations

We used three different mutation rates and ran the simulation for 300s. With 0% mutation, we observed a maximum coverage of 275 m^2. However, the nodes were unable to enter area B as the fitness function penalized them from moving away from their base-stations. The nodes could not explore the new area as the increase in coverage at the gateway tends to be very small. The randomness of 5% mutation permitted the nodes to go through the gateway and obtained a very good coverage. These nodes gained a very high fitness value and were able to be elected to pass their chromosomes to the next generation. The nodes eventually reached coverage of about 490m^2. The simulation for 10% mutation reached the global optimum at about 10s earlier but ended with similar coverage.

We also ran the simulation with a control experiment with 100% randomness in node movement. The network did not reach 50% coverage even after 1200s. The reason was that many nodes overlapped their coverage. Some nodes managed to move through the gateway to area C. However, there were about 5 nodes in area B. The control experiment demonstrated that the fitness function was very effective in guiding the deployment process.

B. Energy Consumption

For the measurement of the energy consumption, we attached constant bit-rate sources to each node. The packet size is set to 512Kb using IEEE 802.11 media access control protocol with 11Mbps bandwidth. In NS2, the energy consumption in communication is measured in terms of the number of packets sent and the packet size. With our GA based approach, there was about 0.02% packet loss. The random control experiment showed about 0.05% packet loss. We also measured the power consumption on locomotion and sensor operation in terms of distance traveled and the number of times the sensor operated. We assumed that the each sensor operation consumed 1J and the motors consume 0.5J per meter. The data is summarized in the following table.

	5% mutation	10% mutation
Energy for communication	3119J	3520J
Sensor Operations (1J/op.)	605J	613J
Total Distance Traveled (0.5J/m)	1007J	1068J
Total	4731J	5201J

Table 2. Energy consumptions.

The results above show that with 10% mutation, the network consumed about 400J more on communication. The nodes also had more sensor operation and traveled 118m more to reach the global optimum. The randomness in base-station assignment increased the frequency of changing in network hierarchy, which in turn created more communication overhead. Also, the randomness in direction vector created extra overhead on locomotion which helped to achieve the global optimum in a short time. The 10% mutation experiment consumed about 10% more energy for the deployment process and gained 10s better performance in time. There is a trade-off between the two performance perspectives for the user when choosing the mutation rate.

9. CONCLUSION

In this paper, we presented an online genetic-algorithm-based mobile sensor network deployment algorithm. To the best of the author's knowledge, this is the first online GA-based algorithm in embedded systems. The paper presents the details on the algorithm and the implementation. The simulation also shows that mutation helps the network avoid local maxima in coverage..

REFERENCES

[1] A. Howard, M. Matari´c, and G. Sukhatme, "Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem," *International Symposium on Distributed Autonomous Robotics Systems* (DARS02), Jun, 2002.

[2] D Gage, "Command control for many-robot systems," *Unmanned Systems Magazine*, 1992, vol. 10, no 4, pp 28-34.

[3] D. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," *The Clearinghouse for Genetic Algorithms* (TCGA), Report 89003, 1989.

[4] Java Genetics Algorithm Package, http://jgap.sourceforge.net/

[5] Network Simulator 2, http://www.isi.edu/nsnam/ns/

[6] T. Yan, T. He, and J. Stankovic, "Coverage: Differentiated surveillance for sensor networks", *Proceedings of the first International Conference on Embedded Networked Sensor Systems*, Nov. 2003, pp. 51-62.

[7] Y. Gao, K. Wu, Fulu Li, "Routing, coverage, and topology control: Analysis on the redundancy of wireless sensor networks," *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications,* Sep. 2003, pp. 104-118.

[8] Y. Zhou, K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Transactions on Embedded Computing Systems* (TECS), Feb. 2004, vol 3, no 1.