Sensor network coverage and data aggregation problem: solutions toward the maximum lifetime

Li Yin

SENSOR NETWORK COVERAGE AND DATA AGGREGATION PROBLEM: SOLUTIONS TOWARD THE MAXIMUM LIFETIME

by

LI YIN

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2007

Approved by

Maggie Cheng, Advisor

Frank Liu

Yinfa Ma

ABSTRACT

Sensor networks have emerged as a premier research topic because of their great long-term economic potential, amazing ability to transform our lives. But as the battery-powered equipment, the sensor has the great limitation in the lifetime, and it is not realistic to replace or recharge the battery for the dead sensor. Usually redundancy sensors are deployed in the monitor area in order to improve the probability of target coverage and for the purpose of backup. This brings new problems in, coverage problem and data aggregations are two of them. In past years, a lot of schemes focused on these problems proposed to prolong the sensor networks' lifetime since lifetime problem is the main constraint on sensor networks' application. First paper introduces solutions to schedule sensors into different sets and set them on and off appropriately to achieve the maximum lifetime while maintaining the required coverage. An optimal solution is provided which could produce the theoretical upper bound on a sensor network's lifetime, and a fast heuristic is implemented with simulation results compared to the optimal solution. The second paper is focus on the problem of energy saving by reducing unnecessary transmission and confliction. A new concept: Balanced Aggregation Tree (BAT) proposed, it could build an efficient aggregation tree whose structure is between Shortest Path Tree and Minimum Spanning Tree and by adjusting a control parameter to achieve the best energy efficiency of a given sensor network, this solution can be used for both aggregate data and non-aggregate data.

ACKNOWLEDGMENTS

The author would like to express his gratitude to his advisor Dr. Maggie Cheng for her guidance, valuable advice and encouragement. Without her kind help and understanding, the author could not have reached this goal. Also thanks to thesis committee members Dr. Liu and Dr. Ma for their advice and participation in the defense; thanks to all the students in the Network Research Lab at University of Missouri Rolla, led by Dr. Cheng, for those numerous instructive discussions and kind support on the research work. The author would also like to take this chance to thank Dr. Cheng for her illuminating explanations in the wireless network area during her classes.

Finally, the author would like to thank his family for their help and encouragement. With their help, the author can study hard and live happily in the peaceful town – Rolla.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

1.1. SENSOR NETWORKS

The sensor is a type of transducer, inexpensive low-power electrical device that could observe the environment features such as temperature, pressure, moisture, light strength, then express these in the man friendly data format. The sensor is usually small in the size, this is the advantage from the aspect of carry and deploy, but it also make the sensor has limited processing speed and storage capacity. Sensors widely using makes people's work and life more easily as sensors could provide users with the critical data in time and most important, they could work in the environment that is hostile to mankind.

Thanks to the exponential growth in the underlying semiconductor technology, the number of transistors on a cost-effective chip and processing or storage capacity on the chip doubles every year. Though it is not good enough to solve sensors' processing and storage ability problem forever, it's good enough to equip sensors with the radio transceiver so that the sensor can communicate with others within its radio range. With this ability, the sensors can be deployed throughout a physical space, providing dense sensing close to physical phenomena, processing and communication the information, and coordination actions with other nodes. Such a set of sensors build a Wireless Sensor Network (WSN).

The development of WSN was originally motivated by military applications such as battlefield surveillance. However WSNs are widely used in many civilian application areas along with scientific research fields. No matter which field it works for, the first step of the procedure is deployment, in order to collect accurate data, sensor nodes are required to sit as close as possible to spots where the information intended to be collected -- namely target. But it is not necessary to let sensors sit right on the target spots because the ability of sensing is not limited with a point but within a certain area, this ability described as Sensing Range, and there is another property named Radio Range, it shows how far the sensors could communication with other sensor nodes.

It is said that a target t is covered by a sensor node s if their Euclidian distance denoted by $|ts|$ is less than the sensing range of s, Rs, i.e., $|ts| < Rs$.

It is not a problem to deploy a sensor network by hand if the target area could be accessed easily, even the exact location of targets and the sensor nodes could be estimated beforehand, so the process of deployment is just put the sensor nodes to the right position. But that is another story if the monitored area is very dangerous or totally unreachable from ground access, under this situation, the sensors have to be deployed in other method, for example: drop from the airplane. Though this deployment method avoids people risking, it brings another problem: the sensors location could not as precise as strategically hand placed, this may cause some of the targets out of surveillance or the whole sensor network disconnected. So redundancy sensors are deployed to compensate this, with a densely sensors, it can be guarantee that all of the targets are covered and the sensor network has a great probability to be connected. At the same time, high dense also brings problems in, for example, if all of the sensors transmit at the same time, conflicting is inevitably. The popular solution is to divide sensors into a hierarchical structure with a cluster header collecting the data within its group and send the aggregated data to its upper layer's cluster header, reach the base station at last. Figure 1.1 shows a typical structure of a wireless sensor network.

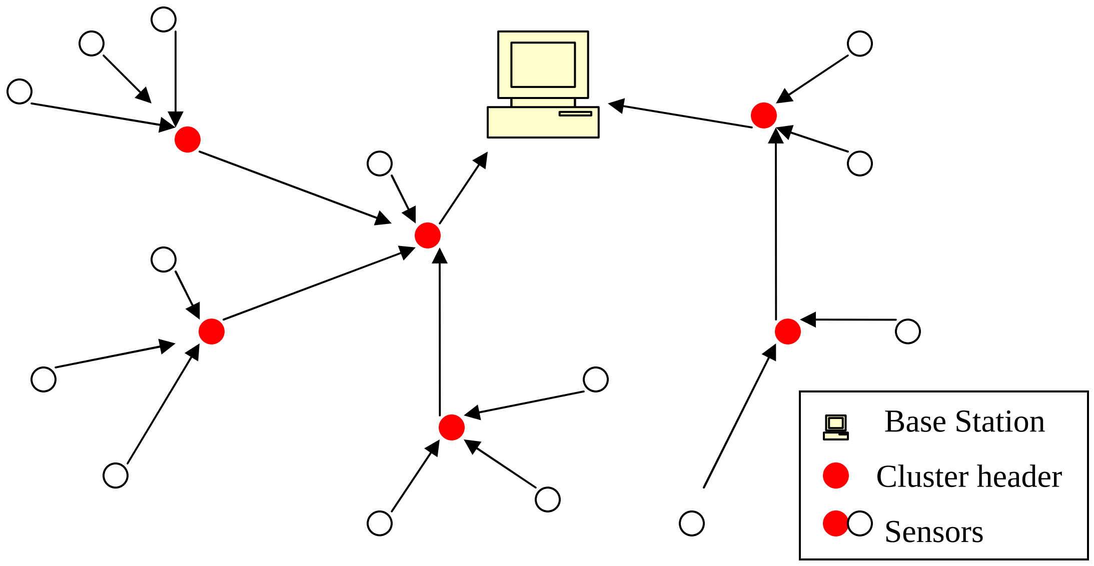

Figure 1.1 Typical WSN structure

Figure 1.1 shows, the sensors may play different roles in the WSN to avoid unnecessary transmission and makes the system high efficient and save the energy. The problem of build a high efficient structure is not involved in the thesis, the thesis focus on the problem of coverage while coverage problem is a sub-problem for that.

1.2. LIFETIME

As introduced before, sensors have the limitation in process speed and storage capacity. Actually there is another more critical constraint for sensors: power supply. The sensor is powered by the battery, once the battery run out, the sensor can do nothing, neither communication nor data collection – this status is called died. If this sensor works at a critical position, its death may cause the whole network disconnected or some targets uncovered, this makes the whole sensor networks paralyze. At the same time, it’s not realistic to replace died sensors’ batteries given that the area sensors deployed may in the hostile area. Also, it is not worth the cost to replace the battery because the price of sensors is very cheap and the number of sensors deployed could be huge. This is another reason to deploy redundancy sensors: for backup, once a sensor died, there always exists another sensor to replace it, this makes the network strong and could last longer. As analyzed before, the redundancy not only brings the coverage problem in, also set up a goal for the set coverage problem: to maximum network lifetime. And the network lifetime is defined as the accumulated functional time.

As a fundamental problem in the field, lifetime problem has been studied for a long time. In the past few years, lots of research works has been done on the problem in making efficient use of battery energy towards a longer network lifetime, including energy aware routing, energy efficient data dissemination and hierarchical aggregation mentioned before, transmission power control and node activity scheduling. These common approaches tried to reduce the unnecessary communication among sensors as much as possible to improve the energy efficiency.

On another side, in addition to satisfy the coverage requirement, the user would wish to organize the sensors in the way that the energy usage could as efficient as possible so the total network lifetime could be maximized, too.

1.3. OVERVIEW OF MAIN CONTRIBUTION

In the coverage problem, an optimal solution is proposed for the maximum lifetime sensor scheduling problem, which could find the upper bound of a sensor network's lifetime. This research reveals the relationship between the degree of redundancy in sensor deployment and achievable extension on network lifetime, which can be a useful guide for practical sensor network design.

The proposed Balanced Aggregation Tree algorithm could reduce the redundancy communication in a partial data aggregation based on the ratio of aggregated data. Which is the result of tradeoff between Shortest Path Tree and Minimum Spanning Tree, and it could achieve the minimum energy use for the specific network, the energy is saved in this way, so the lifetime of the sensor network prolonged.

These two solutions are energy efficient, the schedule could effectively avoid the unnecessarily energy drain, so that to achieve a longer lifetime.

2. SENSOR NETWORK COVERAGE PROBLEM REVISITED: SOLUTIONS TOWARD THE MAXIMUM LIFETIME

2.1. INTRODUCTION TO COVERAGE PROBLEM

2.1.1. Coverage Problem. It is easy to image that in a dense sensor network, each target is covered by more than one sensor. The requirement for the WSN is all of targets should under the surveillance, so for each target, it is good enough as long as there exist a sensor node covers it. At the same time, it is possible that one sensor may cover more then one target in the random deployment process. Actually if the sensing range is big enough, one single sensor node may cover all of the targets. So it is possible to organize sensors into some sets that the sensors in each individual set are enough to cover the whole target area. Obviously, since one set of sensors could cover all of the targets, to let all of the cover sets working all the time is a great waste of energy because of conflicting and redundancy transmission. So the problem is: how to find these cover sets?

In previous literatures, this coverage problem can be classified into the three types [19]: Point Coverage, Barrier Coverage, and Area Coverage.

Point Coverage covers a set of specific points (targets). A lot of works [3], [4], [5], [20] focus in this type, usually they present the scheme to extend a sensor network's operational time by organizing the sensors into a maximal number of disjointed set covers that are activated successively. Obviously, this thesis fells into this category, but with a different way in finding set covers. This will be addressed later.

There is an issue [21] defined the concept of a Barrier Coverage, which derives a theoretical foundation to determine the minimum number of sensors to be deployed so intruders crossing a barrier of sensors will always be detected by at least k active sensors.

Area Coverage is the most discussed coverage problem, where the main objective of the sensor network is to cover an area instead of a point. In the category of area coverage, sensors are used in greater numbers for field operation, and efficient sensors deployment becomes obvious strategies to maintain coverage. Hence, some specific deployment algorithms existing in the literatures try to find out the optimal sensor placement locations in order to maintain sufficient coverage [22], [23].

[9] points out, the coverage concept is a measure of the quality of service (QoS) of the sensing function and is subject to a wide range of interpretations due to a large variety of sensors and applications. The goal is to have each location in the physical space of interest within the sensing range of at least one sensor.

While this thesis is not only cares about the problem of coverage but also energy saving to prolong the network lifetime. And sensor nodes are designed with a switch that could alternate between an active working mode and inactive sleep mode. The motivation for this scheme is not just to turn off redundant sensors to save energy. Research shows that if batteries are given sufficient recovering period between two intensive consumption periods, the actual battery lifetime is extended ([1]). Therefore appropriate scheduling will not only improve sensor network lifetime, but also individual battery's performance.

At the same time, constraints about routing connectivity and network coverage must be considered when design a sensor node activity scheme as long as a connected network desired. However, these two constraints are inherently related. Former research [2] shows, if radio range is twice larger of sensing range, then sensors that fully cover the monitored area will be all connected. This thesis assumes that the radio range of sensors is sufficiently large to maintain routing connectivity. So for each full coverage set, the sensors inside could construct a connected network.

Figure 2.1 shows an example of coverage problem.

Figure 2.1 shows that sensors colored with red could fully coverage the whole field while sensors colored in green could achieve this independently, too.

In the past, majority research works on this topic has focused on organizing sensor nodes into mutually exclusive subsets so that at anytime only one set is active, and the optimization objective becomes to maximize the number of disjoint subsets. The two full covers shown in Figure 2.1 are disjoint set covers. This approach has assumed an unnecessary and overly restrictive requirement that subsets must be disjoint. This restriction significantly limits how much we can extend a sensor network's lifetime because sensor nodes with energy left from indistinct disjoint subsets may construct a new set cover that could cover all of the targets. Previous works [3], [4] and [5] etc. all belongs to this category.

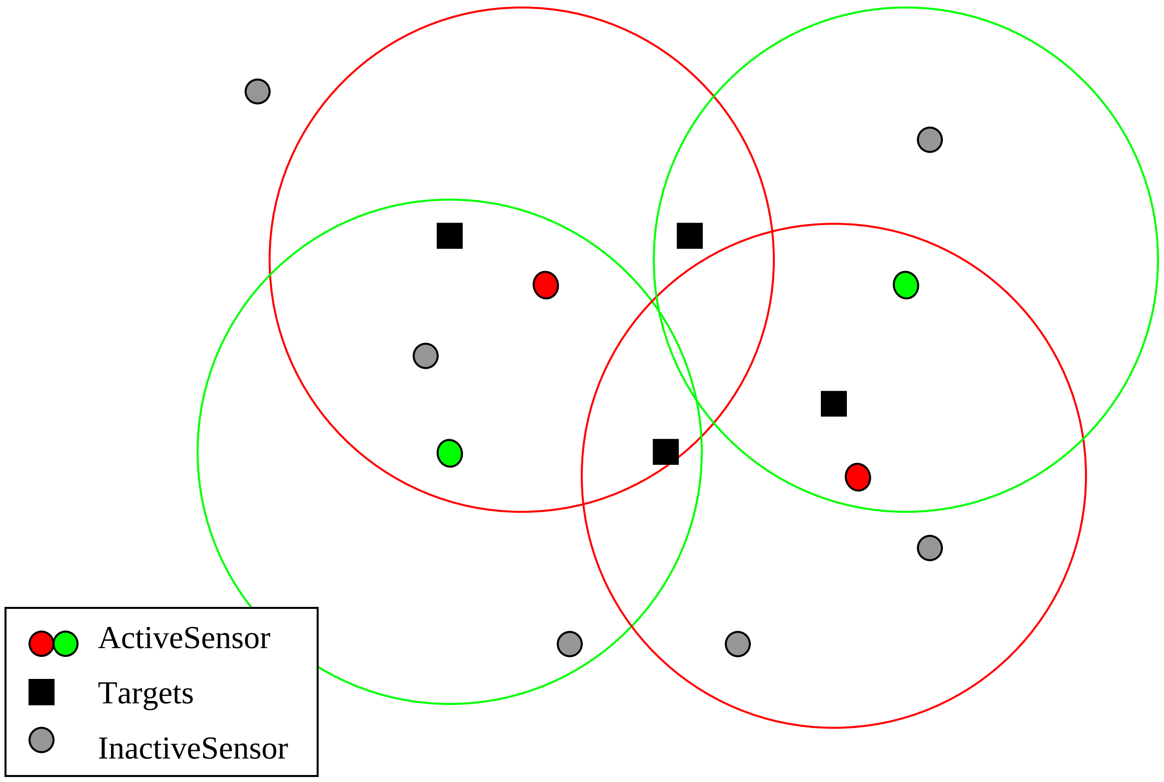

Figure 2.1 Example of Coverage

So the maximum disjoint set covers and maximum lifetime problems are two different problems. Using the approach mentioned before, the number of disjoint set covers does not have direct correspondence with lifetime.Therefore in this project, the solution addresses the problem directly – find the schedule that produces the maximum lifetime, instead of trying to find the maximum number of disjoint set covers. As a result of this scheduling, sensors may join different sets as long as the accumulated energy consumption does not exceed its energy reserve. Each set is called a set cover, and the entire sets are non-redundant set covers, i.e., no one is a subset of another. The maximum lifetime coverage problem in NP-hard, but it is not necessarily computationally expensive for some sensor networks in practice, where the number of non-redundant set covers is within a hundred. It turns out to be practically applicable to use this optimal solution in such small sensor networks. For large sensor networks, a fast heuristic is proposed that selects the most effective set covers as a good representative of the whole set. Less effective set covers are trimmed away to reduce the computation time. The simulation results show that such simplification only slightly reduces network lifetime, and it is still the best of its kind.

2.1.2. Linear Programming. A Linear Programming (LP) problem is a special case of a Mathematical Programming problem. From an analytical perspective, a mathematical program tries to identify an extreme (i.e., minimum or maximum) point of a function -- objective function, which furthermore satisfies a set of linear equality and inequality constraints. In other words, given a polytope and a real-valued function:

$$f(x_1, x_2, ..., x_n) = a_1x_1 + a_2x_2 + ... + a_nx_n + b$$

defined on this polytope, the goal is to find a point in the polytope where this function has the smallest (or largest) value. Such points may not exist, but if they do, searching through the polytope vertices is guaranteed to find at least one of them.

Linear programs are problems that can be expressed in standard form:

Maximize

$$c^T X$$

Subject to

$$Ax \le b$$

Where

$$x \ge 0$$

x represents the vector of variables, while c and b are vectors of coefficients and A is a matrix of coefficients. The expression to be maximized or minimized is called the objective function ($c^T X$ is this case). The equations $Ax \le b$ are the constraints which specify a convex polyhedron over which the objective function is to be optimized. Linear programming can be applied to various fields of study. Most extensively it is used in business and economic situations, but can also be utilized for some engineering problems. Some industries that use linear programming models include transportation, energy, telecommunications, and manufacturing. It has proved useful in modeling diverse types of problems in planning, routing, scheduling, assignment, and design.

Linear programming is an important field of optimization for several reasons. Many practical problems in operations research can be expressed as linear programming problems. Certain special cases of linear programming, such as network flow problems and multicommodity flow problems are considered important enough to have generated much research on specialized algorithms for their solution. A number of algorithms for other types of optimization problems work by solving LP problems as sub-problems.

Under the advent of modern computing technology, for Linear Programming problems, there are few algorithms such as Simplex could solve the function very efficiently. In this thesis, MATLAB is used to solve to LP.

2.2. COVERAGE PROBLEM DEFINITION

Given a set S of N sensor nodes with location information, sensing range and initial energy reserve, and a set T of M targets with location information, assuming each sensor node can switch to an active mode when working, and sleep mode if necessary, find a schedule of sensors can guarantee that at any time, all targets can be covered by active sensor nodes at that time and the accumulated functional time of the network is the maximum.

Based on the location information and sensing range, we can construct a bipartite graph between sensor nodes and targets, where a link (i, j) exist if and only if the distance between sensor i and target j is within the sensing range of sensor i. Moreover, from this bipartite graph, we can build a coverage matrix $A = [a_{ij}]_{N \times M}$, where $a_{ij} = 1$ means sensor i covers target j, otherwise $a_{ij} = 0$. Both of the solutions that will show up in the following sections take this matrix as the input.

Actually, the topic of computing the optimal schedule to achieve the maximum lifetime is NP-hard, because its subclass – where sensors must be put in disjoint sets – is NP-hard. The following section addresses given coverage matrix A, how to find the optimal schedule to achieve the maximum network lifetime.

2.3. SOLUTIONS FOR COVERAGE PROBLEM

2.3.1. An Optimal Solution. The optimal solution can be found by using a two-phase algorithm: In phase I, compute the complete set of non-redundant covers. Each cover is a subset of sensors belongs to set S that completely cover all targets in set T without redundant sensors in it, in other words, removing any sensor from it will leave some target uncovered; In phase II, compute the optimal solution by solving the linear program – assigning the time slice for selected set covers, a schedule for sensors to turn on and off in order to achieve the maximum lifetime

2.3.1.1. Non-Redundant Set Covers. Table 2.1 is the code of the algorithm that compute the complete set of non-redundant covers from the given coverage map $A = [a_{ij}]_{N \times M}$ for N sensors and M targets.

Table 2.1 NRSETCOVERS

```
NRSETCOVERS( $A, N, M$ )
    1.  Initialize $k$ = 0, $S$ = {1,… $N$ }, $T_0$ = {1,… $M$ }
    2.  for $i$ = 1 to $N$
    3.      do set $c[i]$ = 1 and $c[p]$ = 0 for $p \neq i$
    4.          $T = T_0 \setminus \{ j \mid a_{ij} = 1\}$
    5.          $l = i$
    6.          while $T \neq \emptyset$
    7.              do $l$++
    8.                  if $l$ > N
    9.                      then break;
    10.                 if $a_{lj} = 1$ and $j \in T$
    11.                     then $c[l]$ = 1
    12.                         $T = T \setminus \{ j \mid a_{lj} = 1\}$
    13.         if $l > N$
    14.             then $c[i]$= 0
    15.                 continue
    16.             $k$++
    17.             $C_k = c$
```

Table 2.1(Continued) NRSETCOVERS

```
18.     while true
19.       do find the largest $l \in S$ with c[$l$] = 1
20.         if $l = i$
21.             then break
22.           $c[l] = 0$
23.           if $l = N$
24.             then continue
25.           $T = T_0 \setminus \{ j \mid a_{i'j} . c[i'] = 1, \forall i' < l \}$
26.           while $T \neq \emptyset$
27.             do $l{+}{+}$
28.               if $l > N$
29.                 then break
30.               if $a_{lj} = 1$ and $j \in T$
31.                 then $c[l] = 1$
32.                      $T = T \setminus \{ j \mid a_{lj} = 1 \}$
33.            if $T = \emptyset$
34.              then $k{+}{+}$
35.                   $C_k = c$
REMOVEREDUNDANCY ($C$)
```

Table 2.1 shows the greedy algorithm perform the exhaust search over the sensor nodes in sequence, trying to dig out all of the possible non-redundancy full coverage sets. The line 1 --35 start with the first sensor node, check possible full coverage sets combinations, once it find a full coverage set, it will not try other combinations that including this set, in this way, it can avoid part of the redundancy sets, for example, if the algorithm find the set {1, 3, 5} first, it will not take set {1, 3, 5, 7} as another full coverage set because the sensor 7 is unnecessary here. But the result sets generated by it still could contain redundancy in other order, for example, the set of sensor nodes {1, 3, 5} is a full coverage set already, but the algorithm may find another set {1, 2, 3, 5} before set {1, 3, 5} though the sensor 2 is unnecessary here because the algorithm probing sets' element in lexical order. This redundant problem can not be solved between line 1 – 35 without reduce the efficiency. But the redundancy sets need to be removed before running the linear program on it because these sets cause energy waste and will increase programming running time greatly. So another algorithm designed to deal with check and remove the redundancy sets as in Table 2.2:

Table 2.2 REMOVEREDUNDANCY

```
REMOVEREDUNDANCY($C[K']$)
1.  for $p = 1$ to $K'$
2.      do if $C_p$ = $Nil$
3.              then continue
4.          for $q = p + 1$ to $K'$
5.              do if $C_q$ = $Nil$
6.                      then continue
7.                  $C = C_p \& C_q$
8.                      $\triangleright$ $C_p$ bitwise AND $C_q$
9.                  if $C = C_p$ (or $C_q$)
10.                     then $C_q$ = $Nil$ (or $C_p$ = $Nil$ )
```

Let K' denote the number of sets generate by line 1 --35 in **NRS**ET**C**OVERS. Running time for it is $O(K'NM)$ and $O(K'^2N)$ for line 36. This algorithm take K' sets as input, checking if the current set has the including or included relationship with any set after it, if so, remove the one has redundancy nodes. The performance is good if K' is small. But it is possible that K' is a very big number, K' may several even hundred times greater than N or M, in this scenario, the running time for line 36 could be very high. This is the motivation to design an alternative algorithm for **R**EMOVE**R**EDUNDANCY for big $K^{'}$ and small N, which described in Table 2.3. It takes $O(NM)$ to check if a subset of sensors could form a complete cover or not, the alternative algorithm runs at $O(K^{'}N^{2}M)$. So the tradeoff between these two redundancy remove algorithm is, when $K^{'} > N \times M$, the alternative algorithm trigged, otherwise, because the former algorithm is more efficient, so call **R**EMOVE**R**EDUNDANCY.

Table 2.3 REMOVEREDUNDANCY-ALTERNATIVE

```
REMOVEREDUNDANCY-ALTERNATIVE($C[K^{'}]$)
1.  for $p$ = 1 to $K^{'}$
2.      do for each sensor $s$ in cover $C_p$
3.          do if $C_p \backslash \{ s \}$ is still a complete cover
4.              then remove $C_p$
5.                  break
```

The difference between **R**EMOVE**R**EDUNDANCY-**A**LTERNATIVE and **R**EMOVE**R**EDUNDANCY is the alternative algorithm does not check the relationship between sets, it only check inside the sets -- by removing elements in the set one by one to see if there is redundancy exist, the set got removed once a redundancy find. The **NRS**ET**C**OVERS can guarantee that the removed set's corresponding minimum full coverage is in the set and can not be removed.

This is phase I, running time is exponential to N in theory, but could be much less if the link probability is high in the bipartite graph, so the search for covers does not go through every possible combination of sensors -- specially, the while loop in line 6 and line 26 "while $T \neq \varnothing$" break out before every sensor is tested. The actual running time is dependent on $K^{'}$, the number of set covers before the call to **R**EMOVE**R**EDUNDANCY. The simulation section will show how the size of K influences the runtime of the two-phase algorithm.

The following example illustrates the algorithm in details: given coverage matrix $A = [a_{ij}]_{N \times M}$ for $N = 5$ sensors and $M = 4$ targets:

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

Here is a detail description about the process from line 1 to line 35 take node 1 as example: Node 1 try combination with node 2 first, and find out set {1, 2} is a full coverage, so put {1, 2} in the results set and break from node 2; then node 1 continue with node 3, and find out {1, 3} is not a full coverage, so continue probing with {1, 3} and find out {1, 3, 4} is a full coverage, then break {1, 3, 4}, continue with {1, 3} and find out {1, 3, 5} is another full coverage, the probing with 3 is done; next continue with node 4, in the same way, set {1, 4} is not full coverage while {1, 4, 5} is; and {1, 5} is the last feasible full coverage set with node 1. Then start with node 2, probing combination with nodes after it…

At last, before running the **R**EMOVE**R**EDUNDANCY, the **NRS**ET**C**OVERS could get nine full covers sets from A, they are:

$C_1 = 11000$, sensors {1, 2}

$C_2 = 10110$, sensors {1, 3, 4}

$C_3 = 10101$, sensors {1, 3, 5}

$C_4 = 10011$, sensors {1, 4, 5}

$C_5 = 10001$, sensors {1, 5}

$C_6 = 01100$, sensors {2, 3}

$C_7 = 01010$, sensors {2, 4}

$C_8 = 00110$, sensors {3, 4}

$C_9 = 00101$, sensors {3, 5}

By running **R**EMOVE**R**EDUNDANCY, C_2 get removed because $C_8 \in C_2$ but C_8 is already a complete cover, since the goal is to find the maximum lifetime of the whole network, less sensors involves, more energy left and potentially much network lifetime possible. In this way, covers C_2, C_3, C_4 are redundant and get removed. Assuming the initial energy reserve is 1 unit for each sensor node. The optimal solution for the linear program has a lifetime of 2.5 units, which can be achieved by setting as following:

$C_1 = 11000$, sensors {1, 2} $\rightarrow 0.5$ unit

$C_5 = 10001$, sensors {1, 5} $\rightarrow 0.5$ unit

$C_6 = 01100$, sensors {2, 3} $\rightarrow 0$ unit

$C_7 = 01010$, sensors {2, 4} $\rightarrow 0.5$ unit

$C_8 = 00110$, sensors {3, 4} $\rightarrow 0.5$ unit

$C_9 = 00101$, sensors {3, 5} $\rightarrow 0.5$ unit

While the method of disjoint covers could find out only two disjoint covers; for example {1, 2} and {3, 4}, with each cover active for one unit, so the total lifetime is 2 units. Compared with this, the optimal solution achieved 25% longer lifetime.

Here is another example, which has only 1 disjoint set covers: {1, 2} or {1, 3} or {2, 3}, while the optimal solution could easily find 3 non-redundant covers with each working for 0.5 unit time. The lifetime will be improved from 1 unit to 1.5 units, showing 50% increase.

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

$C_1 = \{1,2\} \quad \rightarrow 0.5$

$C_2 = \{1,3\} \quad \rightarrow 0.5$

$C_3 = \{2,3\} \quad \rightarrow 0.5$

These two examples show that, with a reasonable organization on the set covers could last network lifetime.

2.3.1.2. Linear Program Formulation. There are K covers after the redundancy sets removed in Phase I, these sets could be expressed as a matrix $[C_{ik}]_{N\times K}$: let C_{ik} = 1 if sensor i is in cover k , C_{ik} = 0 otherwise. Along with following variables:

t_k :	time for cover k to be active;
t_{ik} :	time for sensor i to be active in cover k .

Obviously, if sensor i is in cover k , $t_{ik} = t_k$; Otherwise, t_{ik} = 0. The maximum lifetime sensor coverage problem can be cast as a linear program as Table 2.4:

Table 2.4 Linear Program Formulation

Maximize	$\sum_{k=1}^{K} t_k$		(1)
Subject to	$(C_{ik}-1)\times t_{ik} = 0$,	$\forall i, \forall k$	(2a)
	$C_{ik}\times(t_{ik}-t_k) = 0$,	$\forall i, \forall k$	(2b)
	$\sum_{k=1}^{K} t_{ik} \le 1$,	$\forall i$	(2c)
	$0 \le t_{ik}, t_k \le 1$,	$\forall i, \forall k$	(2d)

Equalities (2a) and (2b) guarantee that if C_{ik} = 1, $t_{ik} = t_k$; if C_{ik} = 0, t_{ik} =0. Inequality (2c) is for homogeneous sensor networks, where every sensor has the same amount of energy, 1 unit for here. It is easy to extend this to heterogeneous networks if sensors have different initial energy: let $\sum_{k=1}^{K} t_{ik} \le E_i$, where E_i is the normalized lifetime of sensor i based on its initial energy reserve. The lifetime achieved in this way is different from the *first-node-die* lifetime as assumed in most other work -- it is the total time for the network being functional.

Phase II involves the process of solving linear program. The linear program with $n = (N+1) \times K$ variables can be solved in polynomial time $O(n^3)$. The linear programming solver package in MATLAB is used here.

2.3.2. A Fast Heuristic.

2.3.2.1. First K Covers. As mentioned before, the algorithm **NRSETCOVERS** uses exhaustive search to find all non-redundant set covers. It is critical that K , the number of covers set, is small, because the running time of the linear program solvers is proportional to K^3 , this could be a huge consumption on computer resources, for both the memory and CPU usage. Motivated by this, statistics research performed on the result generated by the optimal solution, and it is observed that not all of set covers have equal contribution to network lifetime. The following charts are a statistics on the member numbers inside the sets, the Figure 2.2 shows the distribution of set covers according to sensor number, obviously, most of the sets concentrate with 7 or 8 members

But Figure 2.3 shows another story, it shows though sets with 4, 5, 6 sensor nodes not a majority in the number, but they contribute a lot to the lifetime while the lifetime contribute make by 7 and 8 not as dominate as they in the number. The output shows that each set cover with 7 members only assigned with a very short time slice, this is not feasible in reality because frequently mode changing affect the network's work efficiency.

This means it is important to eliminate some set covers in order to reduce the input size to the phase of solving linear program. The following algorithm is designed to select the first K set covers that have most contribution to network lifetime. To select the first K set covers, a new metric is defined, called *Effective Coverage*, defined as follows.

If sensor i covers p targets and q_k of them have already been covered by other sensor nodes in the set S_k , then the *Effective Coverage* of sensor i in a set S_k is defined as:

$$EC(i,k) = p - q_k$$

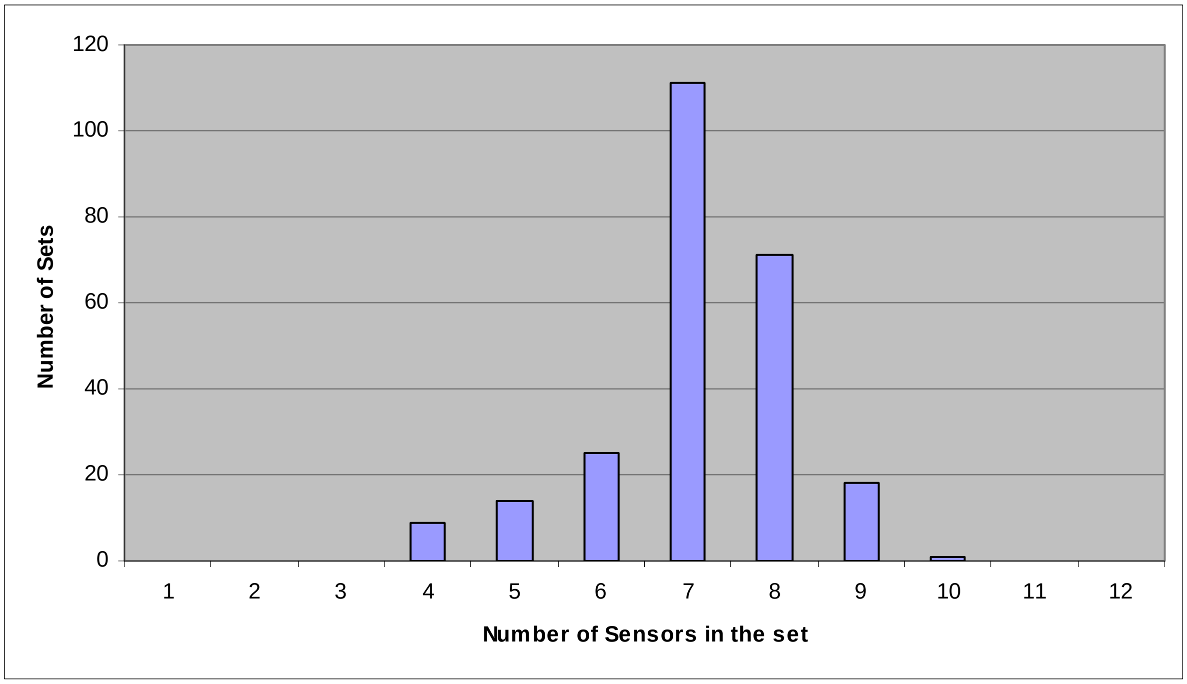

Figure 2.2 Sets Statistic

In other words, *Effective Coverage* of sensor i in set S_k is the number of uncovered targets that it could covers. While p does not change as the algorithm progresses, q_k does change and depend on the order we select sensors. The following algorithm computes the first K covers that put each sensor in the most coverage-effective way.

In this algorithm, initially, S is a $N \times N$ matrix with the elements on diagonal set to 1, and there are two auxiliary matrixes with the same size as S to save value of EC and q_k for the corresponding elements inside S. During the process, the size of S may increase first then decrease, but will around the same magnitude with N.

At each step, the sensor nodes with the greatest effective coverage are selected. Effective coverage is the primary metric in selecting sensors. If several sensors are in a tie, the one that shows more promise toward a complete cover is preferred. This could be determined by comparing the number of targets that would have been covered in this set if sensor i is included, the closer to M the better. This is the secondary metric. If using the primary and secondary metrics still cannot solve a tie case, we use the third metric, p -- the number of targets that a sensor covers, and select the one with smaller p. If all three metrics are tie, include all of them. The algorithm terminates when the best covers all have been selected therefore $S = \varnothing$ or the specified number of covers have been found, whichever comes first. CheckRedundancy() simply does line 2-4 of RemoveRdendancy-Alternative.

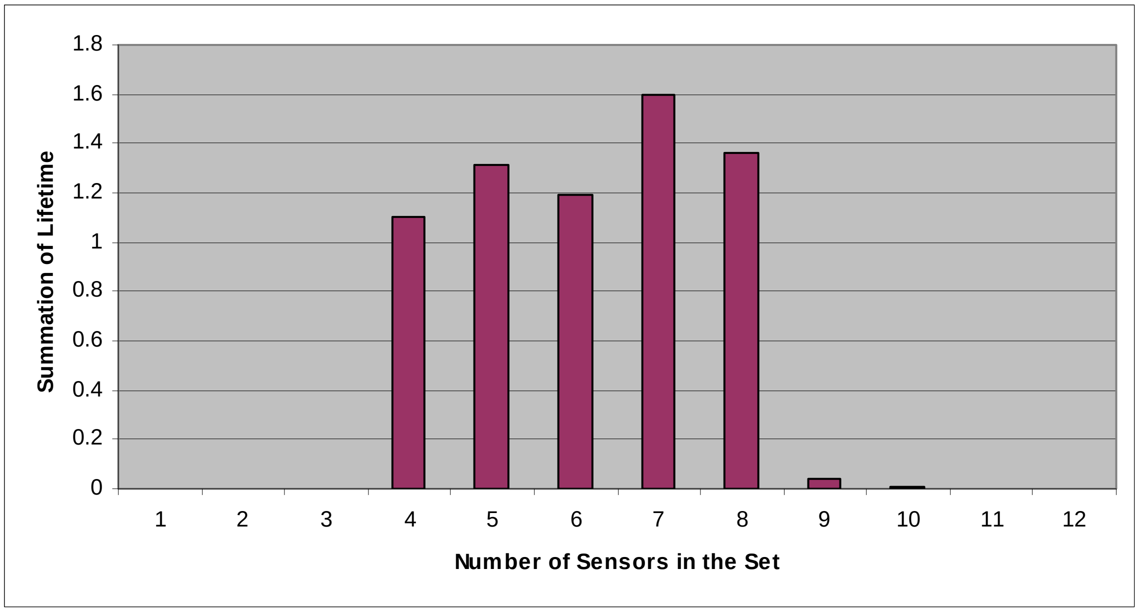

Figure 2.3 Lifetime Statistic

Using this algorithm, selecting the first five set covers in example one will give us the following:

$C_1 = 10001$, sensors $\{1,5\}$ $\rightarrow 0.5$ unit

$C_2 = 11000$, sensors $\{2,1\}$ $\rightarrow 0.5$ unit

$C_3 = 01010$, sensors $\{2,4\}$ $\rightarrow 0\,.5$unit

$C_4 = 00110$, sensors $\{3,4\}$ $\rightarrow 0.5$ unit

$C_5 = 00101$, sensors $\{3,5\}$ $\rightarrow 0.5$ unit

Table 2.5 FIRSTKCOVER

```
FIRSTKCOVER( $A, N, M, count$ )
1.  initialize $num = 0, K = N, S_k = \{k\}$ for $k = 1..K$
2.  $S = \bigcup_{k=1}^{K} S_k$
3.  while $S \neq \emptyset$ and $num < count$
4.      do compute $EC(i,k)$ for each $i$ in each set $S_k$
5.          select the $(i^*, k^*)$ such as:
6.              $EC(i^*, k^*) = \max_{i,k} ( EC(i,k) )$
7.          $S_{k^*} = S_{k^*} \bigcup i^*$
8.          if $S_{k^*}$ is a complete cover
9.              then $S = S \setminus \{ S_{k^*} \}$
10.                 CheckRedundancy( $S_{k^*}$ )
11.                 if $S_{k^*} \neq \emptyset$
12.                     then $C = C \bigcup S_{k^*}$
13.                         $num{+}{+}$
14. return $C$
```

The linear program solver will assign 0.5 unit time for each cover and that gives a total of 2.5 units' lifetime, the same result as the optimal solution.

For the second example, we get {1, 2}, {1, 3}, {2, 3}, with each active for 0.5 unit time, we get a total lifetime of 1.5 units, still same as the optimal solution.

2.3.2.2. An Iterative Heuristic. The result from firstKcover includes the most important group of set covers that dominate the network lifetime. This information is used to solve the linear program in section 2.3.1.2. Then for each sensor, the energy consumption P_i should be deduct from the initial total energy E_i (assumed to be 1.0 initially), so the remaining energy of each sensor i becomes $E_i = E_i - P_i$. After removing the sensors with $E_i = 0$, the algorithm firstKcovers could run on the new input repeatedly until no more complete set cover exist.

After the first iteration, the remaining energy should be considered during the process of selecting sensors. In case that two sensors tie in their effective coverage, the one with more remaining energy is preferred to be select, then the other two metrics introduced in section 2.3.2.1 follow. That is, the order of preference becomes: more effective coverage, more remaining energy, closer to a complete cover, smaller degree in the bipartite graph. This tends to reduce the disparity in energy distribution, make energy consumption health, and therefore potentially increases network lifetime.

2.4. SIMULATION FOR COVERAGE PROBLEM

In order to reveal the performance character of the algorithm, there are two aspects need to be explored:

First, how good the algorithm could be. To show this, experiments conducted with different parameters on sensor density, target density and sensor range. These results reflect the solution’s performance property with different network redundancy.

Second, how better the algorithm could be. A solution is valueless if it could not perform better than the current popular one in the field. Based on this, comparison experiments not only designed between optima algorithm and iteration heuristic algorithm – this is to show how close they are, but also with a well performance algorithm – GREEDY-MSC [6]. These results illustrate the advantage over other algorithms with concrete data.

The simulation is conducted using MATLAB combined with c++ under Unix.

2.4.1. Redundancy And Network Lifetime. Since the objective value of the linear program provides an optimal solution for the maximum network lifetime, it is possible to study how redundancy can affect network lifetime by using this. For all of the simulation experiments, we randomly deploy sensors and targets on a 500×500 plane. The network lifetime is normalized to the battery lifetime. For example, if a battery can last for one unit of time, network lifetime is 13.45 means the network can be operational for 13.45 units of time.

So first focus on how much longer lifetime the algorithm could achieve by increasing redundancy. Redundancy could be realized by either increasing sensor's sensing range, or by deploying more sensors while keeping the same sensing range. Both will increase Δ_t, the average number of sensors covering a target.

Figure 2.4 shows the scenario with fixed number of sensors, 15 here and fixed number of targets, 15 and 30 individually, how the network lifetime be affected with the sensing range increasing.

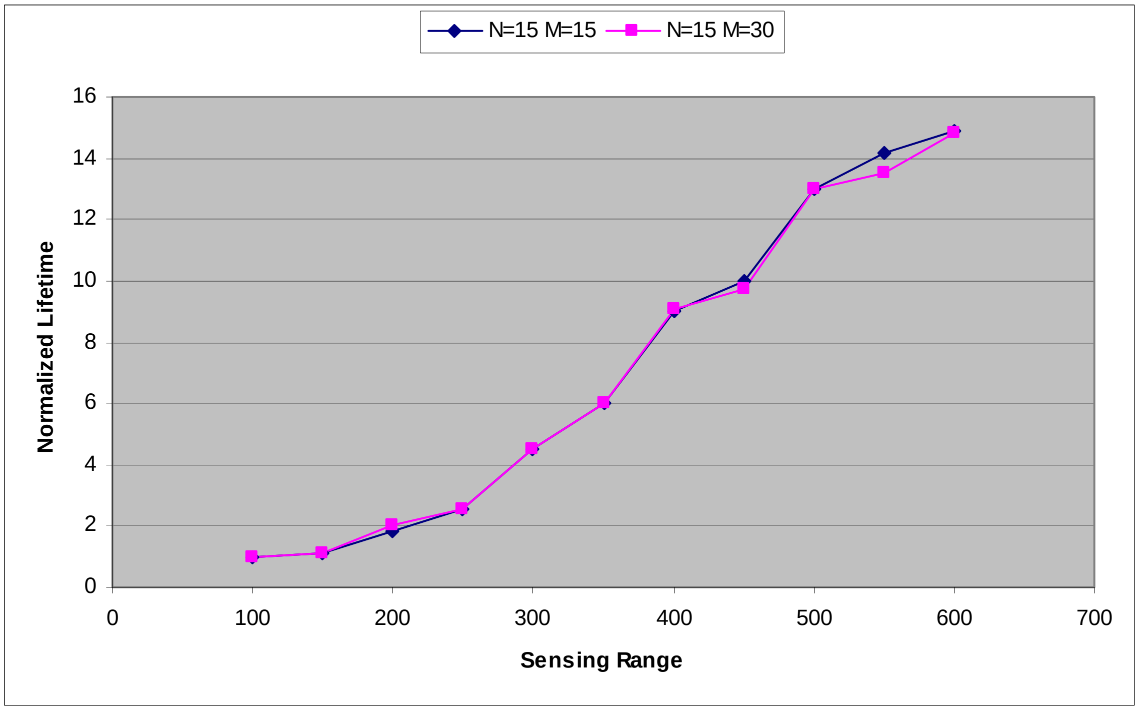

Figure 2.4 OPT with Fixed NM and Increasing Range

Obviously, lifetime increasing with sensor range increasing when sensor number and target number are fixed, also, Figure 2.4 shows the lifetime is not sensitive to the number of targets for the proposed algorithm in the experiment, because targets follow a random uniform distribution. Doubling the number of targets does not always decrease the lifetime.

Figure 2.5 shows the situation of fixed number of targets, 20 and 50 individually and fixed sensing range, 300 and 250 individually, how the network lifetime changing with the increasing number of sensors.

Figure 2.5 shows the network lifetime increasing with sensor number increasing when both the targets number and sensing range fixed. Also, the lifetime noticeably drops when the sensing range decrease to 250. On the curve for M=50 and R=250, the average number of sensors covering each target increases approximately from 3 to 10, and the lifetime appears in the same trend. It can be concluded based on this observation that lifetime is roughly linear to the average number of sensors covering each target.

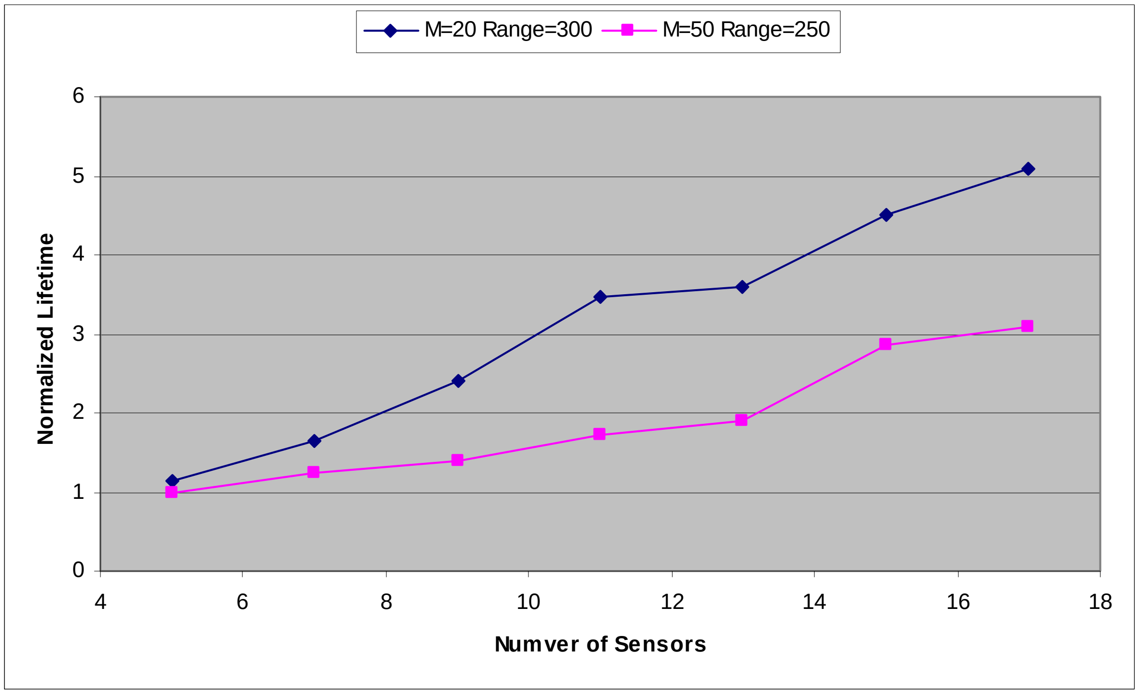

Figure 2.5 OPT with Fixed MR and Increasing N

With optimal solution, K increases dramatically with N increasing, because of it is difficult to get the optimal solution for the network with a large N for the reason of linear program solver runs at the cost of $O(K^3)$, so the heuristic algorithm used here instead of the optimal solution to show the relationship between redundancy and network lifetime (Figure 2.6, 2.7, 2.8). How close the heuristic is to the optimal solution will be exposed in section 2.4.2. Apparently large networks show the same trend as in small networks, and lifetime increases as the number of sensors cover per target increases. It's consistent between Figure 2.4, 2.5 and Figure 2.6, 2.7 even the slopes of the curves.

Similar as Figure 2.4, Figure 2.6 shows the trend of lifetime when sensing range increasing while sensor number and target number are fixed.

In the same way, Figure 2.7 reveals the relationship between lifetime and sensor numbers when target number and sensing range are fixed.

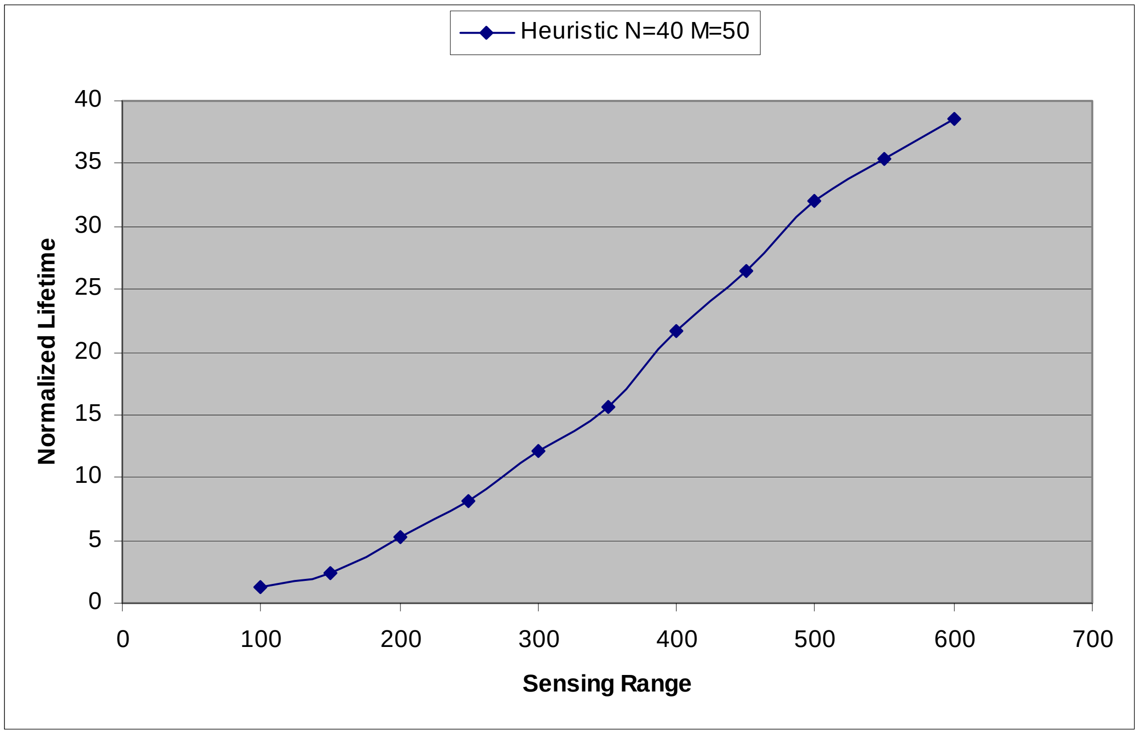

Figure 2.6 HEU with Fixed NM and Increasing Range

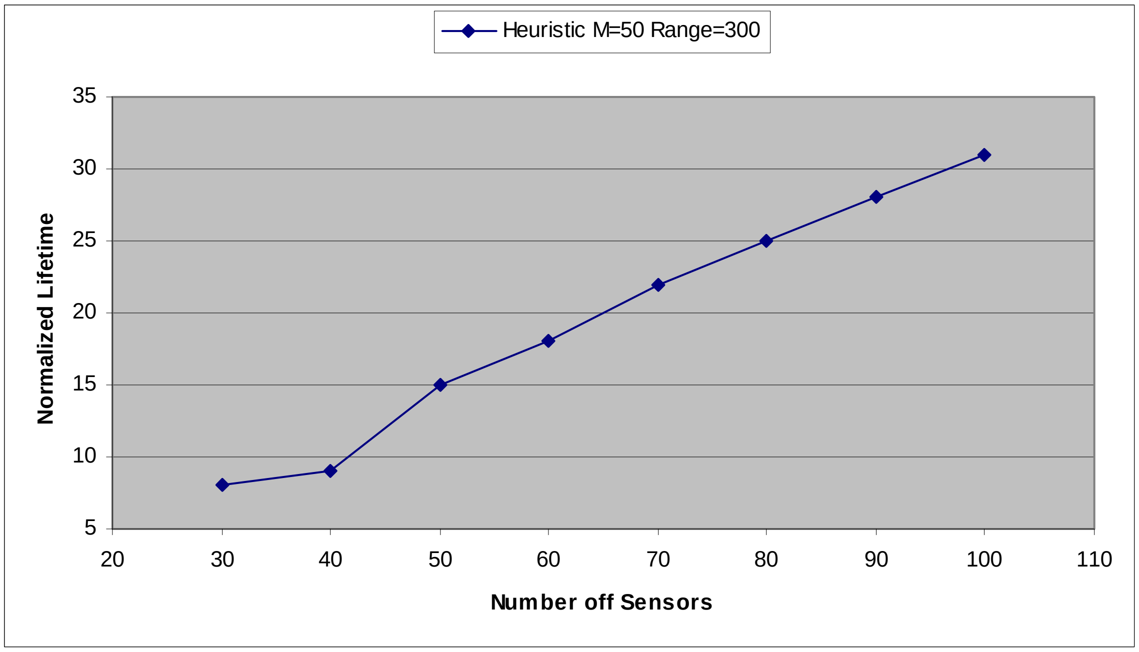

Figure 2.7 HEU with Fixed MR and Increasing N

2.4.2. Performance Comparison.

2.4.2.1. Comparison between Heuristic solution and Optimal solution. In order to know how close the heuristic to optimal solution, so the first merit need to compare is lifetime, computed by them on the same input matrix.

The following table shows the test results for the scenario of 15 sensors and 50 targets, with sensing range increases from 150 to 600. As expected, the heuristic solution finds the optimal solution most of the time, and only for very few cases it misses the optimal solution, but it is still within 3.1% margin of the optimal solution.

Table 2.6 shows the optimal solution results in the lime colored column while the heuristic's in the blue-gray colored column. The rows hold the value to compare: lifetime, the number of set covers input to the linear program solver, for the heuristic, it is the maximum number of set covers among all iterations, and the number of iterations. For the optimal solution, the number of iterations is always 1 since it only uses one pass and does not iterate.

Table 2.6 Performance Comparison between Heuristic and Optimal Solution

Range	**OPT**			**HEU**		
	Lifetime	**Max{K}**	**Iterations**	**Lifetime**	**Max{K}**	**Iterations**
150	1	34.95	1	1	7.55	1
200	1.5	60.7	1	1.49	10.9	1.05
250	2.75	106.45	1	2.66	12.55	1.65
300	4.2	71.15	1	4.13	11.2	1.75
350	5.95	52.35	1	5.91	10.45	1.9
400	7.23	32.55	1	7.12	12.55	1.9
450	9.88	29.8	1	9.61	14.25	2.25
500	11.7	22.05	1	11.49	15.4	1.95
550	13.43	14.5	1	13.43	14.15	1.05
600	14.25	14.4	1	14.25	14.3	1.1

Table 2.6 shows a comparison with all fixed arguments, to further illustrate the difference between these two algorithm under various scenarios, experiments for comparing lifetime, set cover size K and running time with various number of sensor nodes also performed (Figure 2.8-2.10). For sensing range R=300 and M=50, the network size varying from 5 to 25.

Figure 2.8 shows the lifetime computed by the heuristic is exactly the same as the optimal solution, this is the reason why it is feasible to replace optimal solution with heuristic solution does not affect the final results in section 2.4.1.

While Figure 2.9 shows the comparison on the number of set covers they produced for the linear program solver and Figure 2.10 focus at running time of these two algorithms.

Obviously, the number of set covers K increases dramatically with optimal solution while heuristic solution's K keeps a nearly constant, this is a good news for Phase II since solving the linear program is too computationally expensive.

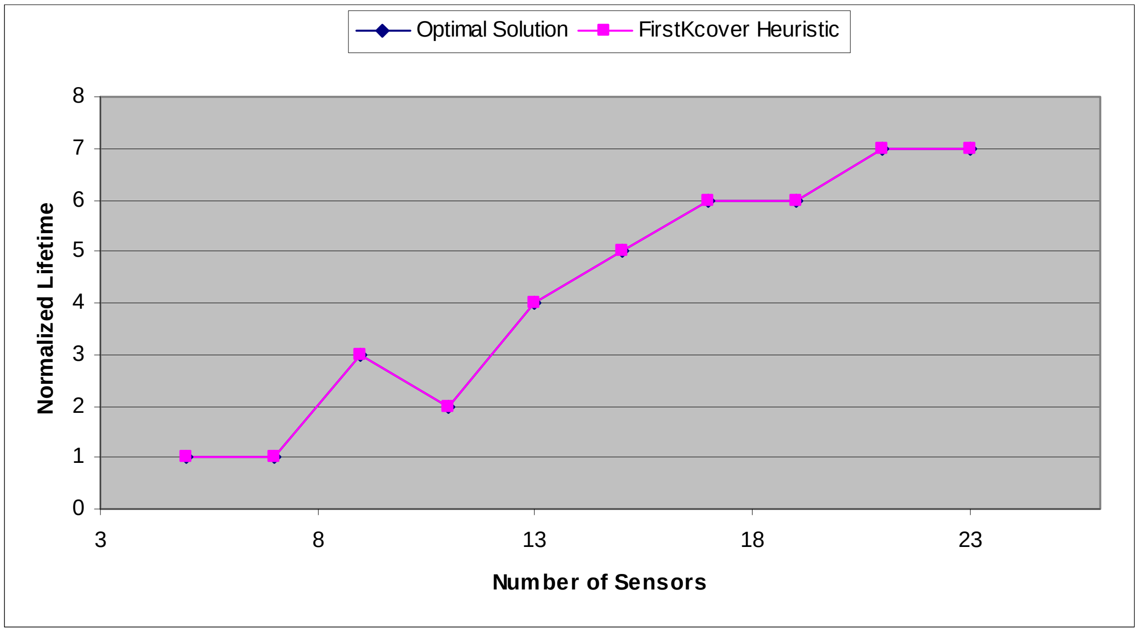

Figure 2.8 Comparison of Lifetime between OPT and HEU with Increasing N

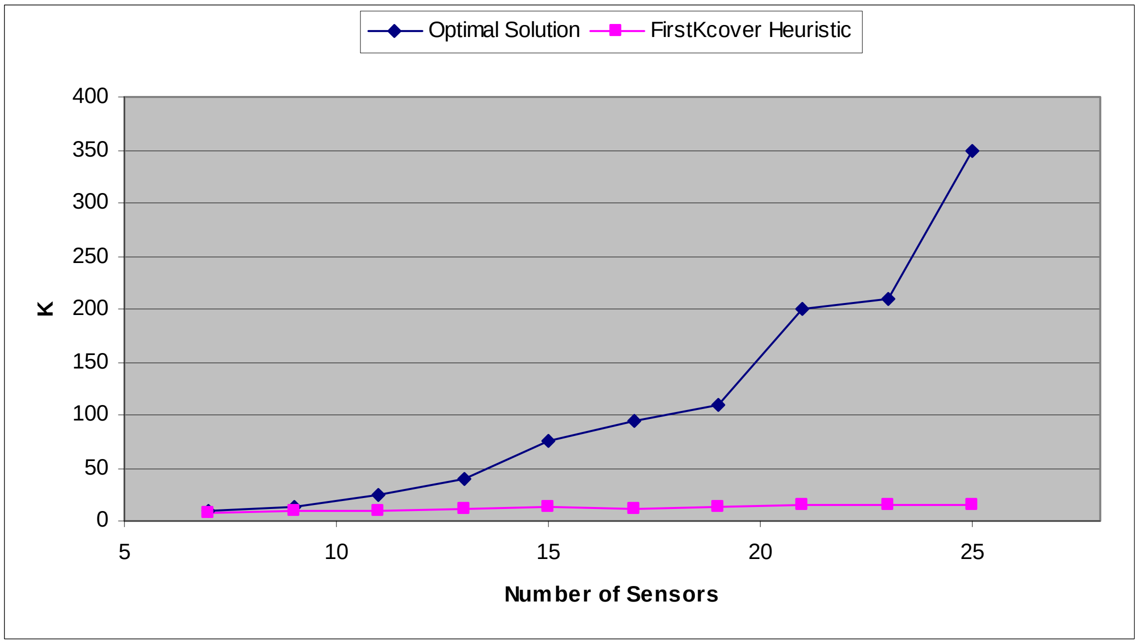

Figure 2.9 Comparison of K between OPT and HEU with Increasing N

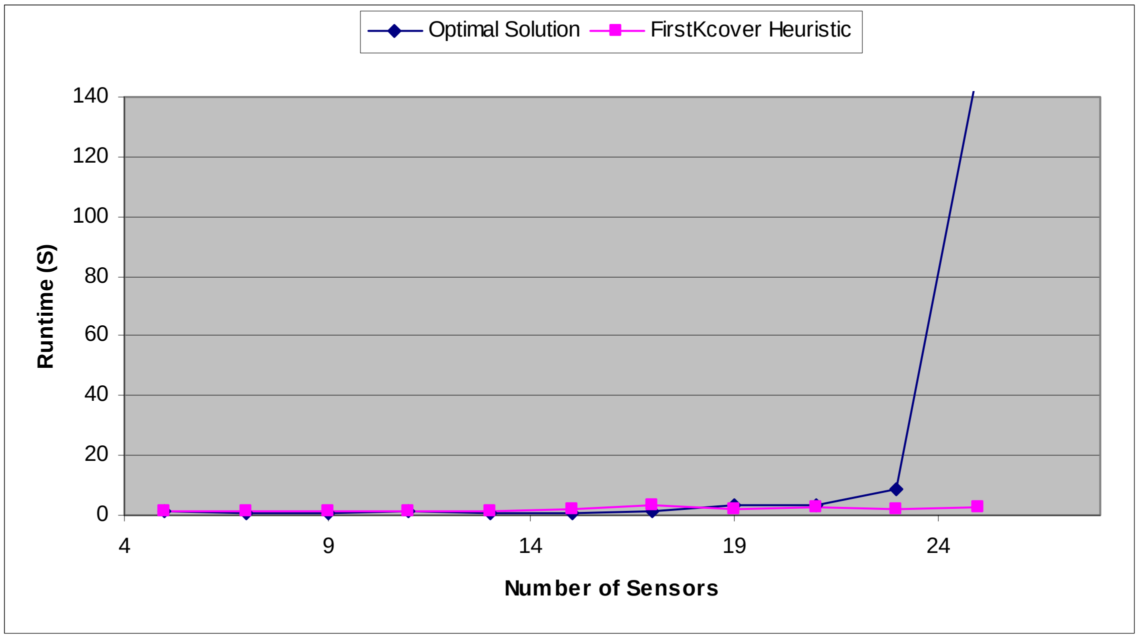

Figure 2.10 Comparison of Runtime between OPT and HEU with Increasing N

This Figure shows the benefit of using the heuristic, it starts to show when the network size increases to 20 sensors – the running time of the optimal solution increases dramatically while heuristic solution keeps at a low time.

2.4.2.2. Comparison between Heuristic and Greedy-MSC. [6] focus on the same topic as this thesis, and there are two algorithms proposed in [6], the winner is a greedy approach: Greedy-MSC.

The comparison between firstKcover heuristic and Greedy-MSC focus on two sides, one is how the sensing range affect network lifetime for both algorithm (Figure 2.11), another is how the number of sensors affect network lifetime (Figure 2.12).

It’s obvious from the plot that firstKcover heuristic always gets more lifetime, especially when the initial energy reserve is non-uniform. The difference between the two curves increases as the number of sensors covering each target increases. In GREEDY-MSC, both the accuracy of results and running time are dependent on the input parameter w -- the time slice size; however in firstKcover, it only depends on the sensor-target coverage map. No matter what initial energy distribution looks like, it can infinitely approach the optimal solution.

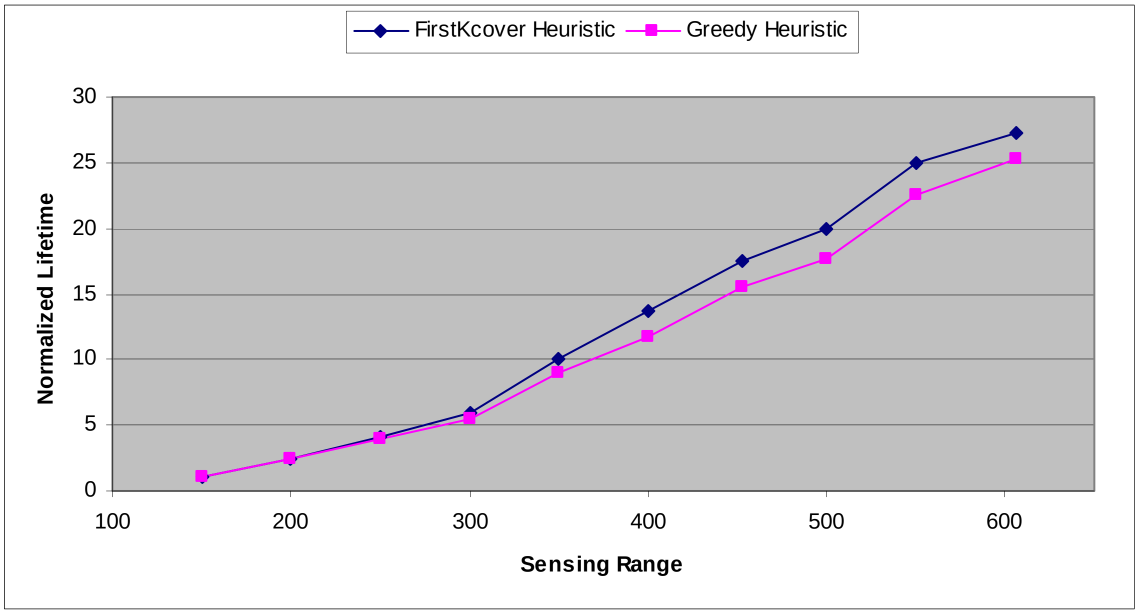

Figure 2.11 Comparison between HEU and GreedyMSC with Increasing Range

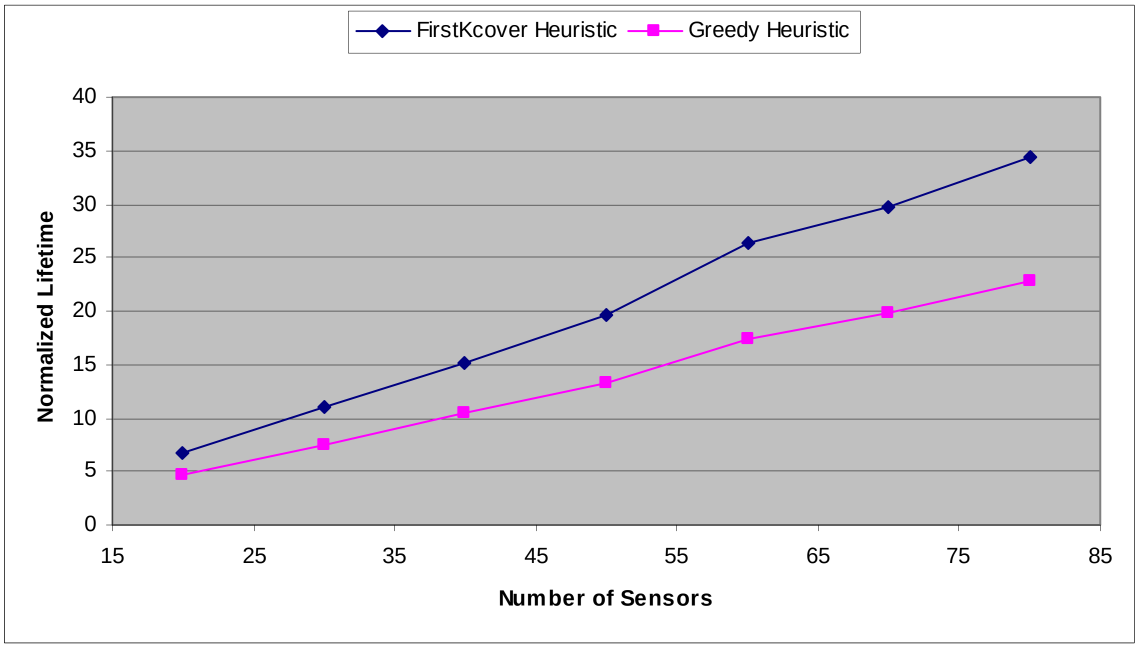

Figure 2.12 Comparison between HEU and GreedyMSC with Increasing N

There is another difference between them: the number of targets could affect Greedy-MSC's performance a lot but as shown in section 2.4.1, targets number is not a big deal for the algorithms proposed in this thesis.

2.5. RELATED WORK

Coverage problem has been addressed using different approaches with various coverage metrics. The most commonly used metric, which is also used in this paper, is the discrete version 0-1 coverage metric, that is, if the target is within the sensing range of the sensor, it is considered covered. With this coverage metric, there is no difference between the target being within 3 feet and 30 feet of the sensor as long as the sensing range $\geq$ 30 feet. Previous work using 0-1 coverage metric includes [3], [4], [6] etc., there is no limit on the size of each subset. This becomes a real problem when the wireless link capacity is limited and all sensors need to ship their sensory data out periodically. [5] addressed how to schedule sensor nodes with bandwidth constraint to achieve minimum breach rate, and based on application needs, three performance metrics for breach rage are introduced.

The continuous version coverage problem is addressed by using a coverage metric based on the signal intensity that the sensor gets from the target, which is a function of the distance between them ([8]). The sum of intensities from multiple sensors measures the likelihood of the target being observer by all sensors. They further defined exposure of a target along a path as the integral of the sum of intensities along the path. While intensity function indicates the sensitivity of a target at a particular point, exposure measures the likelihood of a moving target being detected along a path. Based on this sensing model, they proposed algorithms for calculating the worst-case coverage ([8]) and finding the least-covered path and maximal-support path ([9]). [10] also addresses continuous domain coverage but it uses a different sensing model. [11] further addresses how to find optimal solutions to the best-coverage-least-energy consumption path problem and the best-coverage shortest-path problem. These works do not have maximum lifetime as their optimization objectives, but indirectly can extend lifetime by being energy efficient in all operations.

Distributed approaches that schedules sensors on and off based on local information have also been focus of study in the recent literature. Different from the centralized approach presented in this article, in these works a sensor node switches between an active mode and sleep mode based on the information received from its neighbors and makes its own decision independent from others. These approaches trade optimality for faster and easier implementation. [12] proposed a protocol to minimize the number of active nodes while preserving the original network coverage. In this protocol, a node is scheduled to sleep when its contribution to network coverage is the minimum and removing itself from the network still leaves a fully covered network. Essentially [12] does a density control by using a Cooperative Sensing Model that explores the cooperative exploration of multi-sensors. Other density control approaches are mainly based on Boolean Sensing Model where sensing intensity is based on a continuous model but a threshold value is used to decide if a point is covered or not. [13], [14], and [2] etc. all used this model.

For dense and massive sensor networks, [13] uses probing environment and adaptive sleeping strategies to reduce the number of redundant on-duty nodes. [13] also assumes that faulty nodes exist and node transmission power is adjustable. [13] and many other works use a random uniform distribution method for node deployment that does not guarantee full coverage and connection. [15] uses a different deployment method that not only guarantees coverage but also preserves connectivity, which is similar as used in this thesis. The centralized version of the problems is addressed in [16], where the notion of Connected Sensor Cover is introduced. A connected sensor cover is a subset of sensors that can fully cover the query region and any sensor in the subset can communicate with any other sensor in the subset directly or indirectly through multi-hop communication, and this subset need to be minimized. The connected sensor coverage problem is NP-hard as the less general problem of covering points using line segments is known to be NP-hard [24]. Constructing a minimum connected sensor cover for a query in a sensor network enables the query to be computed by involving a minimum number of sensors without compromising on the accuracy of the query result.

If fault nodes exist in a sensor network, single coverage is not sufficient to satisfy the QoS requirement. [17] addresses the k-coverage problem, i.e., to select a minimal active set of sensor nodes to maintain a complete area k-coverage, which is defined as a minimum set cover problem. It further extends it to address the probabilistic k-coverage problem that requires a point is covered by $\geq k$ sensors at $\geq$ a required probability.

Moving target detection is a different category of coverage problem. [18] defined the worse and best-case coverage problems and proposed polynomial time algorithms to compute them. The coverage calculation here is independent of paths traveled by the target, which is different from [8].

2.6. CONCLUSION ON COVERAGE PROBLEM

To maximize network lifetime under given energy constraints is a fundamental problem in wireless sensor networks, because wireless sensor networks are powered by battery, so the organization with power aware is highly desirable to prolong the network lifetime. Arranging sensors turn on and off at their scheduled time is an efficient method to save the energy, but at the same time, need to guarantee that the active sensors could completely cover all monitored targets. The lifetime metric is the total time during which the sensor network is functional. This thesis provides an optimal solution for the maximum lifetime sensor scheduling problem. The study reveals the relationship between the degree of redundancy in sensor deployment and achievable extension on network lifetime, which can be a useful guide for practical sensor network design.

The proposed algorithm is suitable for small sensor networks. In the future work of this topic, the suboptimal solution for massive sensor networks without increasing computation time dramatically will be addressed, also, distributed and localized algorithms for very large scale networks, and study the tradeoff between computation time and communication overhead in achieving the maximum lifetime need to be further explored. The linear program model can be easily extended to address sensor networks with heterogeneous sensor networks where nodes may have different battery supply. For fault tolerance consideration, the algorithm to find the non-redundant set covers can be modified to make sure each target i is covered by k_i sensors and the linear program can still apply to find the optimal solution.

3. ENERGY EFFICIENT DATA GATHERING ALGORITHM IN SENSOR NETWORKS WITH PARTIAL AGGREGATION

3.1. INTRODUCTION TO DATA AGGREGATION

Wireless sensor networks can potentially be used in habitat monitoring, target tracking, surveillance as well as many other future civil and military applications [26]. Sensors in a network collaboratively accomplish a sensing task, and then relay the information to a specified viewer, often referred to as a sink node or a base station. Sensors are equipped with a sensing unit to gather information, a computing unit for data processing, and a communication unit to communicate with other sensors and the base stations. Due to the bandwidth limitation and the energy limitation, data transmitted through the network should be reduced as much as possible. To this end, in-network data aggregation is desired in many systems [28, 31, 34]. On the other hand, due too the limitations on power supply and computing capability, the large computing task should be avoided at sensor nodes. As a result, some computationally expensive tasks are moved to the base station and raw data is forwarded without in-network processing. A pure aggregation model and a pure non-aggregation model are shown in Figure 3.1 and Figure 3.2.

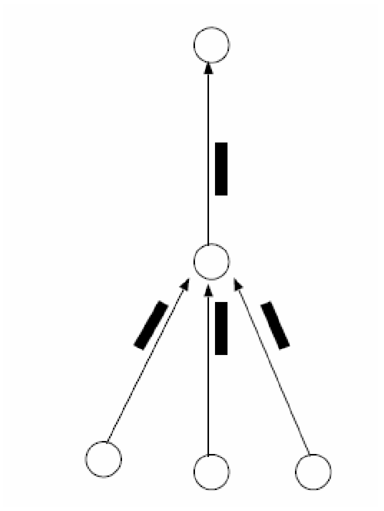

Figure 3.1 Data aggregation model

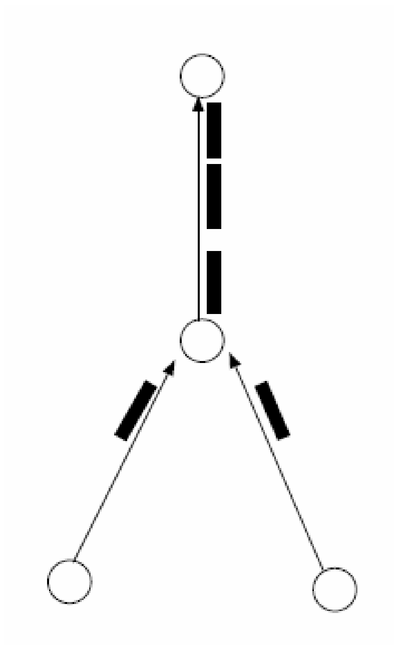

Figure 3.2 Data non-aggregation model

In future sensor networks, data gathering with or without aggregation will co-exist. In particular, some queries and answered with full data aggregation, some with partial aggregation, and some without aggregation at all. Some queries require all sensor nodes to respond, while others only involve a subset of sensors. With flexible in-network data processing, it is possible that for one particular query, some nodes will be aggregators and others are just relay nodes; and the roles of sensors change from query to query. A dynamic topological structure that changes with every query is too expensive to maintain in terms of setup delay and energy consumption. In fact, it is rather infeasible to update the aggregation tree structure if queries are issued frequently. A reasonable assumption is that even though sensors may play different roles for different queries, for a long term each sensor roughly has equal chance to generate raw sensory data. Therefore in this paper we assume a uniform model, in which a fraction of sensory data are fully aggregated and the rest are not aggregated at all. This model does not require specific query information or the source distribution. This fraction is called aggregation ratio, and we assume a uniform ration for every sensor node. Figure 3.3 shows that some data are aggregated while others are not.

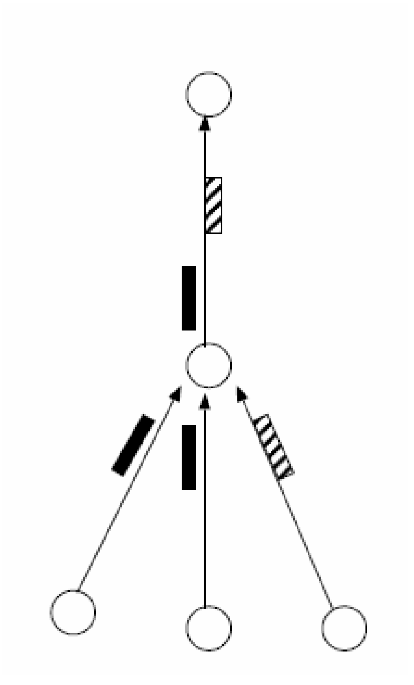

Figure 3.3 Hybrid model

In this paper, we try to find the most energy efficient topological structure for data gathering with a constant aggregation ratio b. In two extreme cases when $b = 0$ (i.e., data are not aggregated at all) and $b = 1$ (i.e., data are fully aggregated), the optimal solutions become the Shortest Path Tree (SPT) and the Minimum Spanning Tree (MST) respectively. In a general case when $0 < b < 1$, to find the minimum energy tree for data gathering is an NP-hard problem. We propose an efficient numerical approach to compute the input parameter α that controls the transition between MST and SPT so that the resulting tree can minimize the energy cost (in Chapter 11). A polynomial time algorithm BAT is proposed to construct such a tree with the given control parameter (in Chapter 10). Through extensive simulations, we show that the proposed algorithm and numerical approach effectively reduce the energy cost of data gathering (in Chapter 12).

3.2. ENERGY COST FOR DATA GATHERING

In this chapter, we formulate the total energy cost associated with data gathering in sensor networks. Since transmission power is the dominant factor among all the activities (transmitting, receiving and local data processing, etc.), we only consider the transmission power. When ignoring the constant factor, the required transmission power

P to send data over a distance d is $P = d^c$, where c is the path loss exponent between 2 and 4.

Let $G(V, E)$ be the graph model of the sensor network, where an edge exists between two sensor nodes if they are within the transmission range of each other. We assign each edge $e \in E$ a weight function $w(e) = |e|^c$. The sub graph that supports data gathering from all sensor nodes to the sink node is a tree rooted at the sink node.

(1). When the aggregation ration $b = 0$, the total energy is the sum of the weights of paths from the source nodes to the sink node r. The total energy is:

$$E_1 = \sum_{v \in V} \sum_{e \in path(v,r)} w(e)$$

In this case, to compute a tree that minimizes E_1 is equivalent to compute the shortest path from each node to the sink node r. Thus the optimal solution is a Shortest Path Tree, which can be found in polynomial time.

(2). When $b = 1$, the total energy is:

$$E_2 = \sum_{e \in tree} w(e)$$

In this case, paths from different sources to the sink node can be shared as much as possible, and the shared paths are only counted once in the sum, therefore to compute a tree that minimize E_2 is equivalent to compute a Minimum Spanning Tree. Thus the optimal solution can be computed in polynomial time. However, if only one subset of nodes is source nodes, it becomes an NP-complete problem.

(3). When $0 < b < 1$, the optimal topological structure is a tree that provides continuous transition between a Minimum Spanning Tree and a Shortest Path Tree. Such a tree has the promise to provide best performance over a long time. In [32] the maximum lifetime data gathering problems are addressed where data are either fully aggregated or not aggregated at all. However, to our knowledge, there is no previous work that has ever addressed the hybrid data aggregation (i.e., $0 < b < 1$), which is more likely to have broader applications that the other two.

In the next chapter, we will show that the Balance Aggregation Tree (BAT) algorithm can be used to construct a tree that is energy efficient for hybrid data aggregation. In the following, we use D_v and $d(v)$ to represent the distance from root r to node v in SPT and the final BAT tree, P_v and $p(v)$ to represent the predecessor of v in SPT and the final BAT tree respectively.

3.3. BAT ALGORITHM

The Balanced Aggregation Tree (BAT) algorithm finds a trade-off between the shortest path property of SPT and the minimum weight property of MST, and provides a smooth transition between the two. A tree is called an α-tree of G if for every node v in the tree, the distance from v to r in G. We will show that BAT algorithm computes an α-tree of G for given $\alpha \geq 1$.

The BAT algorithm is given in the following, where $G(V,E)$ is the graph model of the sensor network, r is the specified root and $\alpha \geq 1$ is the control parameter. In the BAT algorithm, V_B is the confirmed vertex set already on the tree, initialized to include the root r; E_O is the set of edges crossing $\overline{V_B}$ and V_B.

```
BAT ( $G(V,E)$ , $r$ , $\alpha$ )
    Compute the shortest path form $r$ to each node $v \in V$ ;
    Let $D_v$ be the distance from $r$ to $v$ ;
    Let $P_v$ be the predecessor of $v$ on the path.
    **for** each $v \in V$ **do**
            $d(v) = (\alpha + 1)\ D_v$
            $p(v) =$ NULL
    **end for**
    let vertex set $V_B = \{ r \}$, $\overline{V_B} = V \setminus V_B$
    let edge set $E_O$ = all edges connected to $r$
    **While** $\overline{V_B} \neq \phi$ **do**
            find the minimum-weight edge $(u,v) \in E_O$, s.t.
                    $u \in V_B$, $v \in \overline{V_B}$
```

```
        UPDATE ($u$ ,$v$ )
        **if** $d(v) \le \alpha\ D_v$ **then**
            $E_O = E_O \setminus \{edge(x,v) \mid x \in V_B\}$
            $V_B = V_B \bigcup \{v\},\ \overline{V_B} = \overline{V_B} \setminus \{v\}$
            $E_O = E_O \bigcup \{edge(v,w) \mid w \in \overline{V_B}\}$
        **else**
            $E_O = E_O \setminus \{edge(u,v)\}$
        **end if**
    **end while**
    return $T_{BAT} = (V, \{(v, p(v)) \mid v \in V \setminus \{r\}\})$
    END of BAT

    UPDATE ($u,v$ )
        $d(v) = d(u) + w(u,v)$
        $p(v) = u$
    END of UPDATE
```

Theorem 10.1 *Given a graph G with non-negative edge weights, BAT algorithm computes an α -tree of G in O(E+VlogV) time.*

Proof: We first show BAT algorithm terminates within *O(E+VlogV)* time and outputs a single tree, then show that for each node in the tree, $d(v) \le \alpha D_v$.

In the while loop, a vertex $v \in \overline{V_B}$ is added into V_B by an edge (u,v) that straddles V_B and $\overline{V_B}$. It starts from root r and takes $|V|$-1 edges to connect $|V|$-1 non-root nodes onto the tree, so the structure is acyclic and is connected, therefore the resulting structure is indeed a tree.

It can be proved that the algorithm does not have endless loops, because E_O will not become empty before $\overline{V_B}$ becomes empty. This can be proved by contradiction:

Assume there exists a node $v \in \overline{V_B}$ and E_O becomes empty before v is included in V_B. Assume node $u \in V_B$ is the predecessor of v on the shortest path from v to r. Therefore $d(u) \le \alpha D_u$. This leads to $d(u) + w(u,v) \le \alpha D_u + w(u,v)$. Since $\alpha \ge 1$, so $d(u) + w(u,v) \le \alpha(D_u + w(u,v))$. However, $D_u = w(u,v) + D_v$, which leads to $d_u + w(u,v) \le \alpha D_v$. Thus node v could be included in V_B when edge (u,v) is examined. If the predecessor of v is also in $\overline{V_B}$, call it x, then the same proof can lead to that x could be included in V_B before E_O goes empty. Therefore by the end of the while loop, all nodes are included in the tree and all edges $\{(v, p(v))\}$ form a single tree.

The running time of BAT algorithm is *O(E+VlogV)*, because the size of E_O is bounded by $\Delta |V_B| = O(V)$ at any time, where Δ is the maximum node degree, therefore to extract the minimum weight edge from E_O takes $O(\log V)$ time using a priority queue, altogether it is $O(V \log V)$ time; to add edges into E_O and to remove edges from E_O are executed $2|E|$ times altogether, so the total time for BAT is *O(E+VlogV).*

The distance property is direct from the procedure that a node v is added into V_B only if $d(v)$ satisfies $d(v) \le \alpha D_v$.

□

Theorem 10.2 *Given a graph G with non-negative edge weights, to compare a minimum weight α-tree is NP-hard for α >1.*

Proof: In [33], a theorem has been proved that for given $\alpha > 1$ and $1 \le \beta < 1 + \frac{2}{\alpha - 1}$, it is NP-complete to determine whether a given graph G contains a tree that satisfies 1) for every vertex v the distance from u to r in the tree is at most α times the shortest distance from v to r in G ; and 2) the weight of the tree is at most β times the weight of a minimum spanning tree of G. It follows from this theorem that to compute a minimum weight α-tree is NP-hard, because otherwise if we can find the minimum weight α-tree in polynomial time, we can compute its weight W^* in polynomial time, then we can compare W^* with. $\beta \times W_{MST}$: if $W^* \le \beta \times W_{MST}$, then we can determine in polynomial time that G contains a tree that satisfies the two conditions; if $W^* >$

$\beta \times W_{MST}$, then we can conclude in polynomial time that G does not contain a tree that satisfies the two conditions, contradicting the theorem in [33]. □

However, when $\alpha = 1$, the minimum weight α-tree problem becomes to compute a minimum weight Shortest Path Tree. This problem is solvable in polynomial time. While both *the Light Approximate Shortest-path Tree (LAST)* algorithm in [33] and BAT compute an α-tree of the original graph, BAT outperforms LAST in total weight, because the edges of the tree are selected from a larger pool. LAST only uses the edges in MST until a violation on distance occurs. The smallest total weight property is verified through simulation in section 3.5.1 Figure 3.7 ~ Figure 3.10.

3.4. MINIMUM ENERGY TREE STRUCTURE

The transition from a Shortest Path Tree to a Minimum Spanning Tree is controlled by an input parameter α, Increasing α will sacrifice the distance property for better total weight property, and decreasing α will increase the total weight for better distance property. However, how to determine the trade-off in real systems can be a challenging task. In this section we discuss how to choose α to make the resulting tree structure the most energy efficient for a given sensor network.

The lower bound of the optimal solution is achieved by an imaginary tree that behaves like a Shortest Path Tree for non-aggregate data, and behaves like a Minimum Spanning Tree for aggregate data. In a sensor network, if the ratio of non-aggregate data to aggregate data is $a:b$, where $0 \le a,b \le 1$, and $a+b=1$, then the lower bound of the optimal solution is:

$$E_{OPT} = a \times E_{SPT} + b \times E_{MST}$$

Where E_{SPT} is the sum of distances in the Shortest Path Tree and E_{MST} is the total weight of the Minimum Spanning Tree. For any BAT tree, the total energy cost consists of a fraction a of sum of distances and a fraction b of total weight.

$$E_{BAT} = a \times \sum_{v \in V} \sum_{e \in path(v,r)} w(e) + b \times \sum_{e \in T_{BAT}} w(e)$$

By adjusting the control parameter α, we can control the shape of the BAT tree for different a and b, so the resulting total energy E_{BAT} can approach the lower bound. The idea is as follows:

Let y be the ratio of the sum of distances in the Shortest Path Tree to the total weight of the Minimum Spanning Tree. Let $E(v \sim\!\!\rightarrow r)$ be the energy cost along the path from v to root r . The total energy cost is:

$$E = a \times \sum_{v \in V} E_{(v \sim\rightarrow r) \in T_{BAT}} (v \sim\!\!\rightarrow r) + b \times \sum_{e \in T_{BAT}} weight(e)$$

Since the cost along each path is upper bounded by α times that of a Shortest Path Tree, and the total weight of the ($\alpha - \beta$) BAT tree is upper bounded by β times that of a Minimum Spanning Tree. Thus

$$E \le a\alpha E_{SPT} + b\beta E_{MST} \le (a\alpha y + b\beta) E_{MST}$$

To minimize the upper bound of E, we can find the value of α that minimizes $X = a\alpha y + b\beta$. In the worst case, $\beta = 1 + \frac{2}{\alpha - 1}$, the minimum value of X is achieved when $\alpha = 1 + \sqrt{\frac{2b}{ay}}$. Since not every network instance constitutes a worst case scenario, we only use this value as the initial value of α ; the best value for α is to be found by iteration. Therefore we choose α_0 as follows:

$$\alpha_0 = 1 + \sqrt{\frac{2b}{ay}}$$

This allows that when α = 0, α_0 approaches ∞, so there is no limit on the distance to the root, therefore the BAT tree becomes MST; when α = 1, α_0 = 1, so the BAT tree becomes SPT. When 0< α < 1, increasing α or y will get a smaller α_0 , so the tree has smaller distances thus to reduce the energy cost.

Let α_1 = 0.5 $\times$ (α_0 +1). Use α_1 and α_0 as inputs, we construct two BAT trees. If the energy cost $E_{BAT_{\alpha 1}} > E_{BAT_{\alpha 0}}$, let α_2 = 0.5 $\times$ ($\alpha_0 + \alpha_1$), otherwise let α_2 = 0.5 $\times$ (1+ α_1), and so on. The resulting curve of the energy cost will fit in one of the three possibilities:

- Case (a), monotonically decreasing
- Case (b), monotonically increasing

- Case (c), oscillating

Initialize $\alpha_0 = 1 + \sqrt{\frac{2b}{ay}}$, iteratively compute α_1, α_2, and α_3 as shown in Figure 3.4, Figure 3.5, and Figure 3.6. The output from this numerical procedure is α_n . In case (a), the minimum energy is obtained when $\alpha_n = \alpha_0$; in case (b) the minimum energy is obtained when $\alpha_n = 1$; in case (c) , no clear trend is shown, so we take the minimum energy among all computed values resulting from { 1, α_0, α_1, α_2, α_3 }. The above procedure takes at most three iterations. Increasing the number of iterations can definitely get closer to the optimal solution, but since there is no guarantee that it will converge within a finite number of iterations, we restrict it to three iterations only. The energy cost of the BAT tree with $\alpha = \alpha_n$ is compared with the ones that use an arbitrary fixed value such as $\alpha = 2$ and the initial value $\alpha = \alpha_0$. Apparently, α_n gives the lowest total energy cost as verified in the simulation (Figure 3.11)

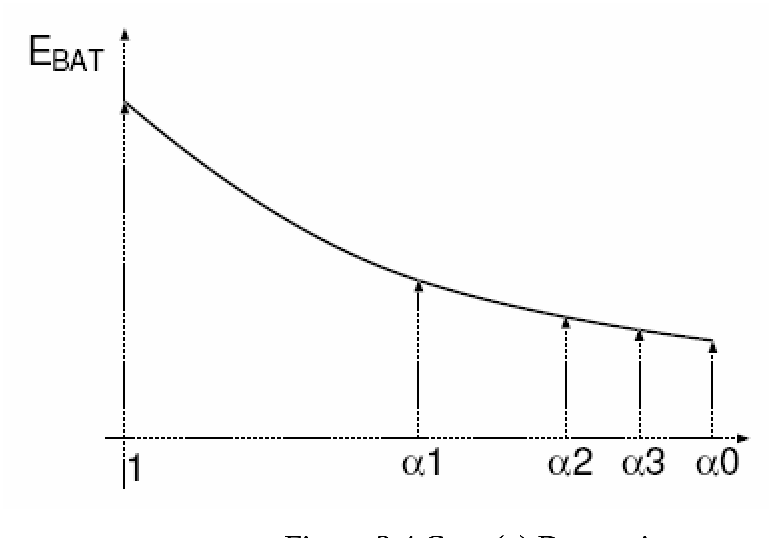

Figure 3.4 Case (a) Decreasing

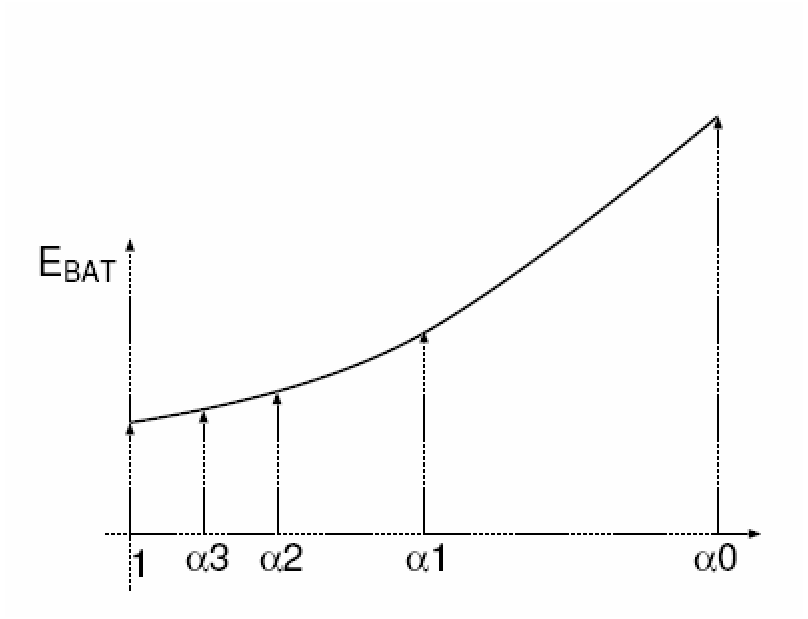

Figure 3.5 Case (b) Increasing

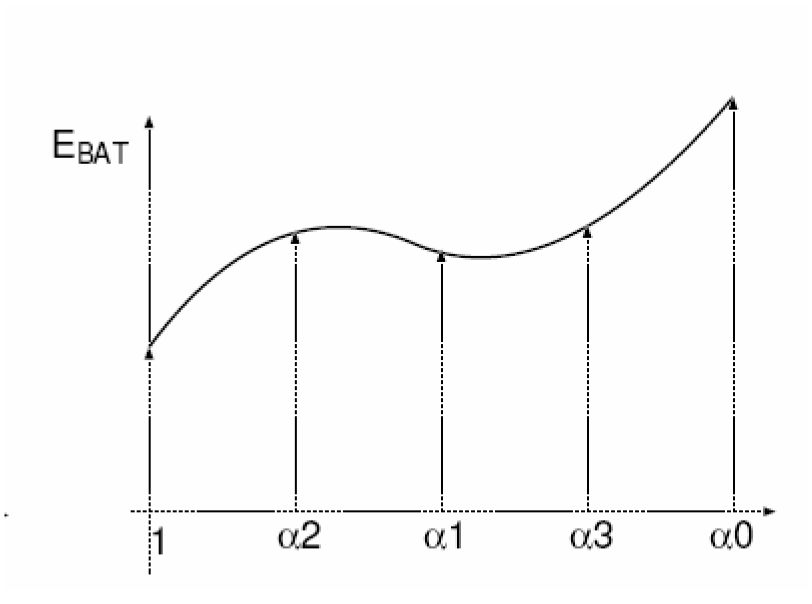

Figure 3.6 Case (c) Oscillating

3.5. SIMULATION OF DATA AGGREGATION

3.5.1. Simulation Setup. For comparison purpose, we use unit-less values, and we consider only the energy cost involved in data transmission and ignore others that are the same for all algorithms.

A sensor network consists of up to 200 nodes, uniformly and randomly scattered around on a 1×1 square. The radio transmission range varies from 0.1 to 0.35. An edge between two nodes exists if they are within the transmission range of each other. We assume a uniform transmission range fro every node, thus all edges are symmetrical. The root node is randomly selected.

3.5.2. Performance Comparison.

3.5.2.1. Compare BAT with other trees. Other trees under consideration are MST, SPT and LAST in [33]. MST has the minimum total weight and SPT has the smallest distance from non-root nodes to the root. However, MST blows off on distances and SPT blows off on total weight. We found by using a small α , BAT can generate a tree that is satisfactory on both total weight and distances. LAST is an efficient algorithm proposed in [33] to compute a trade-off between SPT and MST. In the first simulation (Figure 3.7, Figure 3.8), we study the weight and distance properties of BAT, and use LAST, MST and SPT to compare with. We show the ratios of the total weights from BAT, LAST and SPT to the total weight of the MST, and the ratios of the sum of distances form BAT, LAST and MST to that of the SPT. In 3, a fixed value α_{LAST} = 1.12 is used for LAST and α_{BAT} = 0.9 α_{LAST} is used for BAT. This means BAT needs to satisfy a more restrictive condition on distance. For an arbitrary network, the upper bound of the distance from a non-root node to the root on the BAT tree is at most 90% of that on the LAST tree. The experiments show that with α_{LAST} = 1.12 and α_{BAT} = 0.9 α_{LAST} , BAT always has a smaller total weight and a smaller sum of distance that LAST on the same network. The total weight of SPT could be as high as 190% of that of MST, and the total weight of LAST and BAT are both within 115 ~ 120% of that of MST. BAT, with a more restrictive requirement on individual distance, shows 3 ~ 4% improvement over LAST on total weight. On the distance aspect, the sum of distances in MST could be as high as 146% of that of SPT, the sum of distance in LAST is within 101% that of SPT; and BAT, with such a small , finds the exactly the same distance as SPT.

Following Figures are on a 200-node network, transmission range 0.1-0.35. Compare BAT with other trees on the sum of distances and the total weight, normalized by the sum of distances from SPT and the total weight from MST respectively

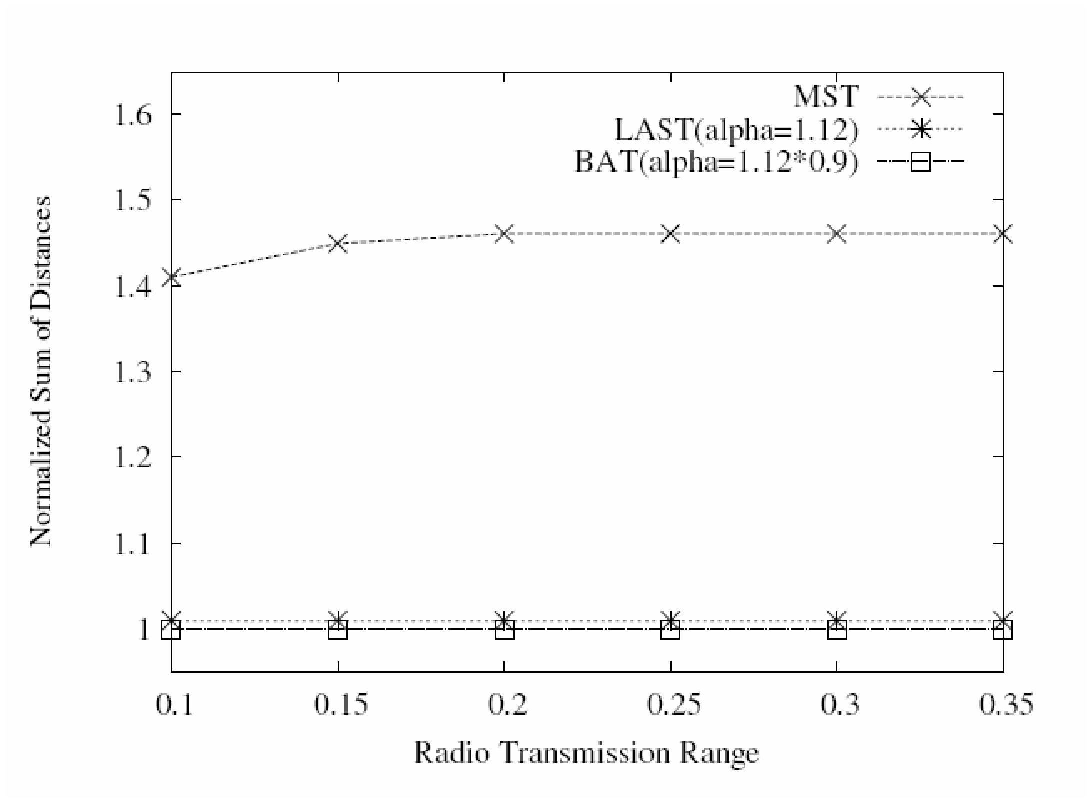

Figure 3.7 Sum of distance

The key parameter to control the tradeoff between the total weight and distances to the root is α . Figure 3.7 and Figure 3.8 shows that BAT, with a proper α value, could do better than LAST on both aspects, but its counterpart LAST couldn't – if LAST can win on distances; it has to lose on total weight. The new challenge is now to find the proper value of α that gives the best performance of BAT, which is provided in Figure 3.9 and Figure 3.10.

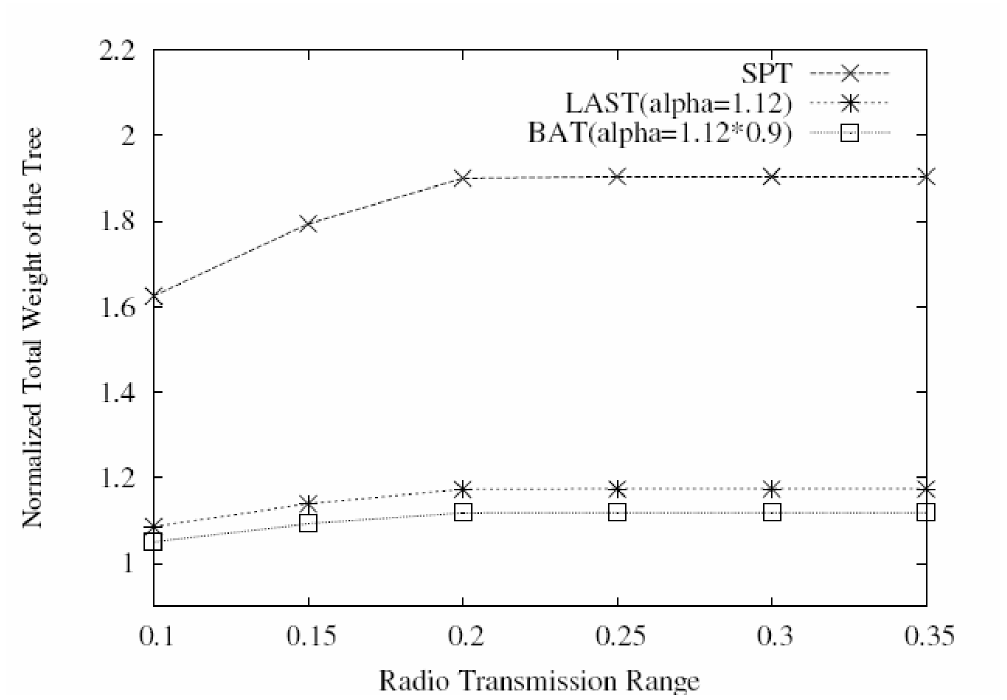

Figure 3.8 Total weight

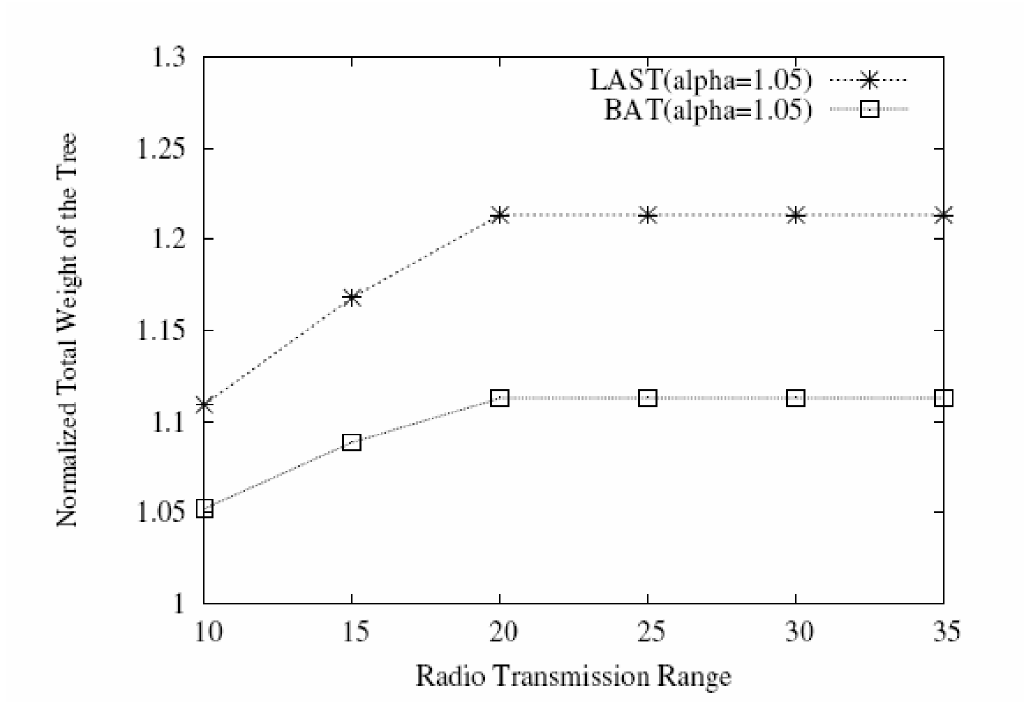

Figure 3.9 With 200 nodes, transmission range = 0.1 – 0.35. Compared BAT with LAST on the total weight. Results normalized by the total weight of MST.

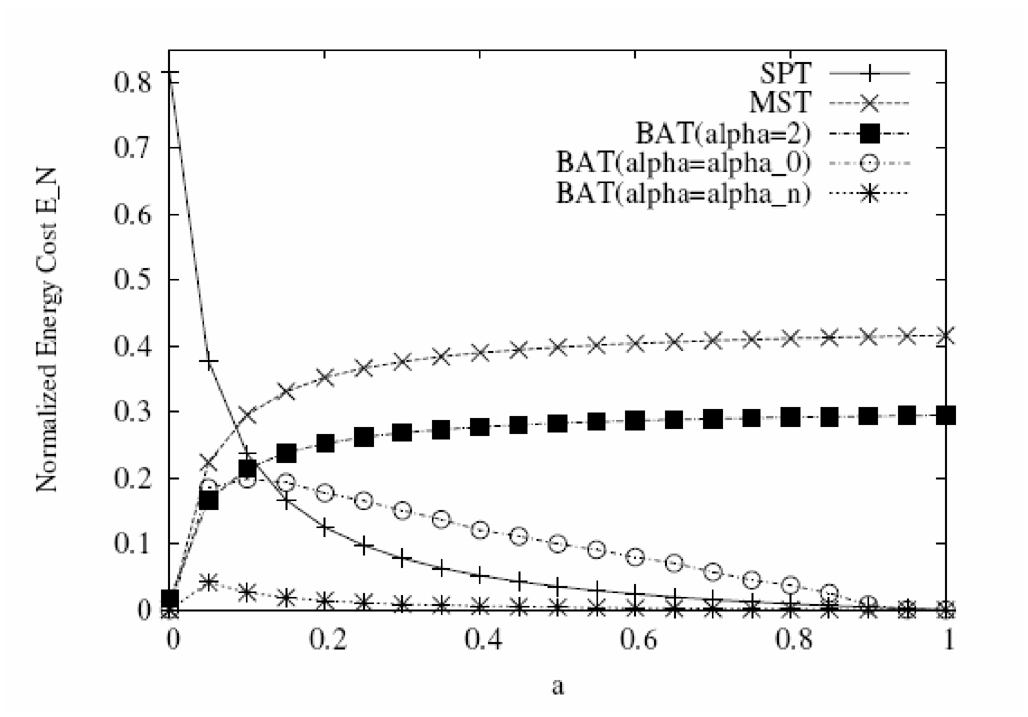

Figure 3.10 Normalized energy costs of SPT, MST and BAT trees (with α =2, $\alpha = \alpha_0$, and $\alpha = \alpha_n$) for 200-node networks, with radio transmission range 0.15

The second experiment is to compare the weight property of BAT and LAST using the same control parameter α . The simulation results in Figure 3.9 show BAT produces trees with 10% less total weight than LAST, with both satisfying the individual distance requirement.

3.5.2.2. Improve Energy Cost by Tuning Control Parameter. The objective of this simulation is to show that the control parameter α can be fine-tuned to improve the energy-efficiency of the data aggregation tree. We compare the energy cost of BAT trees with different values of α for the same network. Let a be the proportion of non-aggregate data, b be the proportion of aggregate data, so $a+b$ =1, and $0 \le a, b \le 1$. We compare BAT trees with $\alpha = \alpha_0$, $\alpha = \alpha_n$, and α = 2 as well as MST and SPT.

The performance metric is the normalized energy cost

$$E_N = \frac{E - E_{OPT}}{E_{OPT}}$$

Where E is the actual energy cost and E_{OPT} is the lower bound of the optimal solution, as defined in section 3.4.

Figure 3.10 shows the average energy cost of each algorithm for 100 instances. The simulation results show that BAT trees with α_0 and α_n coincident with MST when a =0, b =1, and coincident with SPT when a =1, b =0, which are the optimal solutions. The performance of BAT is the best at two ends, when a =1 or a =0. During the transition from MST to SPT, BAT with $\alpha = \alpha_0$ provides and energy cost in between of MST and SPT, but BAT with $\alpha = \alpha_n$ outperforms all the others. It is also observed that the BAT trees with $\alpha = \alpha_0$ and $\alpha = \alpha_n$ both perform better than the one with a fixed value α =2.

In Figure 3.10, the curve for BAT with $\alpha = \alpha_n$ flattens out in most part and approaches the lower bound of the optimal solution, and even the worst case performance is only 4% increase from E_{OPT} . This simulation verifies the scheme described in section 3.4 can effectively find the best value of α that gives the near-optimal energy cost.

3.6. RELATED WORK

In sensor networks, the key challenge in data gathering is energy conservation. A lot of work has been done along this line for energy efficient data gathering [29, 31, 38, 35, 37]. Among many others, data aggregation is the most important approach and has been used in many systems [28, 30, 31, 34, 36, 39]. Data aggregation can reduce the amount of redundant transmissions, thus reduces the energy consumption. [31] proposed Directed Diffused, a localized data-centric scheme, where the data generated by sensor nodes is named by attribute-value pairs and a node (sink) requests data by sending interests for named data. Data matching the interest is then collected and forwarded to the requesting node along the reverse path of the interest propagation. Intermediate nodes can cache, or transform data, and may direct interests based on previously cached data.

In [31, 34] and [39], it is assumed the underlying topological structure of the network is a data aggregation tree, and the internal nodes (non-sink, non-leaf nodes) do the aggregation to reduce the amount of data being transmitted. In [39], to guarantee data aggregation is done within a specified time, Yu et al. used packet scheduling techniques to trade latency for energy. In [39] each sensor node in the tree aggregates the information from its subtree rooted at itself (including all its children and the node itself) and generates a reduced size packet. If the amount of data s each source node generates is known, then the amount of output of source data $s^{'}$ after aggregation is dependent on the number of source nodes d in the subtree and an aggregation factor k, where $k \in [0,1]$ is a control parameter assumed to be the same for all sensor nodes.

Complementary to data aggregation, another possible approach in energy efficient data gathering is to select a subset of sensors fro data transmission instead of using all sensors, and the selected sensors are sufficient to reconstruct the data for the entire sensor networks [27].

3.7. CONCLUSION AND FUTURE WORK

In sensor networks, data gathering is often implemented with certain degree of data aggregation. In this paper, we address the problem of energy-efficient data gathering with various levels of data aggregation, assuming some data will be aggregated and some will be simply forwarded without further processing at forwarding nodes. In order to gather data from source nodes and route data to the sink node, a tree structure is needed as the basic topology. We observed that the Minimum Spanning Tree is the optimal solution if all data is fully aggregated, and the Shortest Path Tree is the optimal solution if no data is aggregated. Between these two extreme cases is the general case, where a certain percent of data is aggregated, for which neither the MST nor the SPT is the optimal solution. We show that we can use the aggregation ration as an input parameter to control the tree structure. Such a tree structure satisfies that the distance from any node to the root is at most α times the shortest distance; such a tree provides a smooth transition from a Shortest Path Tree to a Minimum Spanning Tree. We propose an efficient algorithm BAT to find such a tree. The simulation results demonstrate that BAT algorithm achieves better performance than other tree structures in terms of the energy efficiency of data gathering.

In addition to the consideration of energy, the total weight of the tree also indicates the interference level of the network. The one with the minimum total weight is the best in terms of reducing total interference. Both LAST and BAT provide trees with distances bounded by α times the shortest distance, however, BAT tends to find the one with smaller weight most of the time.

The algorithms proposed in this paper are all centralized. In the future, we will address the implementation of the algorithm in a distributed environment, and study the performance trade-offs if it is implemented locally.

BIBLIOGRAPHY

[1] D. Rakhmatov and S. Vrudhula, "Energy management for battery-powered embedded systems," Trans. on Embedded Computing Sys., vol. 2, no. 3, pp. 277-324, 2003

[2] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," Wireless Ad Hoc and Sensor Networks: An International Journal, vol. 1, no. 1-2, pp. 89-123, January 2005.

[3] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in IEEE International Conference on Communications, Helsinki, Finland, vol. 2, June 2001, pp. 472-476.

[4] M. Cardei, D. MacCallum, M. X. Cheng, M. Min, X. Jia, D. Li, and D. -Z, Du, "Wireless sensor networks with energy efficient organization," Journal of Interconnection Networks, vol.3, no. 3&4, pp. 213-229, 2002.

[5] M. Cheng, L. Ruan, and W. Wu, "Coverage breach problems in bandwidth constrained sensor networks," ACM Transactions on Sensor Networks, June 2007

[6] M. Carder, M. Thai, Y. Li and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in IEEE INFOCOM 2005, 2005, pp.1976-1984

[7] M. Cheng, L. Ruan, and W. Wu, "Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks," in IEEE INOFCOM 2005, 2005, pp. 2638-2645

[8] S. Megerian, F. Koushanfar, G. Qu, G. Veltri, and M. Potkonjak, "Exposure in wireless sensor networks: Theory and practical solutions," Journal of Wireless Nerworks, vol. 8, no. 5, pp. 443-454, September 2002.

[9] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in INFOCOM, vol. 3, April 2001, pp.1380-1387.

[10] X. -Y. Li, P. -J. Wan, and O. Frieder, "Coverage in wireless ad-hoc sensor networks," in IEEE Int'l Conf. on Communications (ICC 2002), New York, April 2002, pp. 3174-3178

[11] X. -Y. Li, P. -J. Wan, and O. Frieder, "Coverage in wireless ad hoc sensor networks," IEEE Transactions on Computers, no. 6, pp. 753-763, June 2003.

[12] B. Yang, H. Yu, H. Li, and H. Hou, "A coverage-preserving density control algorithm based-on cooperation in wireless sensor networks," in WiCOM 2006, 2006, pp. 1-4

[13] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang, "Peas: A robust energy conserving protocol for long-lived sensor networks," in Proc. Of the 23rd International Conference on Distributed Computing Systems (ICDCS03), May 2003, pp. 28-37

[14] D. Tian and N. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in Proc. ACM Workshop on Wireless Sensor Networks and Applications, Atlanta, Oct. 2002., 2002, pp. 32-41 [online]. Available: http://citeseer.ist.psu.edu/tian01coveragepreserving.html

[15] Y. -S. Yen, S. Hong, R. -S. Chang, and H. -C. Chao, "An energy efficient and coverage guaranteed wireless sensor network," in IEEE WCNC 2007, 2007, pp. 2923-2928.

[16] H. Gupta, Z. Zhou, S. R. Das, and Q. Gu, "Connected sensor cover: self-organization of sensor networks for efficient query execution," IEEE/ACM Trans. Netw., vol. 14, no. 1, pp. 55-67, 2006

[17] J. -P. Sheu and H. -F. Lin, "Probabilistic coverage preserving protocol with energy efficiency in wireless sensor networks," in IEEE WCNC 2007, 2007, pp. 2631-2636.

[18] S. Megerian, F. Koushanfa, M. Potkonjak, and M. Srivastava, "Worst and best-case coverage in sensor networks," Mobile Computing, IEEE Transactions on, vol. 4, no.1, pp. 84-92, no. 1, pp. 84-92, 2005.

[19] M. Cardei and J. Wu, "Energy-efficient coverage problems in wireless ad hoc sensor networks," Computer Communications, vol. 29, issue 4, pp. 413-420, Feb.2006

[20] M.Cardei and D. -Z. Du, "Improving wireless sensor network lifetime through power aware organization," Wireless Networks, 11(3), vol. 11, No. 3, pp. 333-340, May, 2005

[21] S. Kumar, Ten H. Lai, and A. Arora, "Barrier coverage with wireless sensors," MobiCom, pp.284-298, Germany, 2005.

[22] S. S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," WCNC, pp. 1609-1614, USA, 2003

[23] Y. Zou and K. Chakrabatry, "Uncertainty-aware and coverage-oriented deployment for sensor networks," journal of Ad Hoc & Sensor Wireless Networks, pp. 89-124, Mar.2005

[24] V. S. A. Kumar, S. Arya, and H. Ramesh, "Hardness of set cover with intersection 1," In Automata, Languages and Programming, 2000.

[25] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, "Introduction to algorithms (second edition)", MIT Press

[26] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (Mar. 2002). “Wireless sensor networks: A survey.,” *Computer Networks (Elsevier) Journal,* 38(4): 393-422

[27] Gupta, H., Navda, V., Das, S. R., and Chowdhary, V. (2005). “Efficient gathering of correlated data in sensor networks,” In *proceedings of the 6th ACM international sysposium on Mobile ad hoc networking and cmputing,* pages 402-413

[28] He., T., Blum, B., Stankovic, J. A., and Abdelzaher, T. F. (2004). “Aida: Adaptive application independent aggregation in sensor networks,” *Special issue on dynamically adaptable embedded systems, ACM Transaction on Embedded Computing System,* 3(2):426-457

[29] Heinzelman, W. R., Kulik, J., and Balakrishnam, H. (1999). “Adaptive protocols for information dissemination in wireless sensor networks,” In *proceedings of the Fifth Annual ACM/IEEB International Conference on Mobile Computing and Networking (MobiCom’ 99),* Seattle, WA, 1999., pages 174-185.

[30] Intanagonwiwat, C., Estrin, D., Govindan, R., and Heidemann, J. (Vienna, Austria, IEEE. July, 2002). “Impact of network density on data aggregation in wireless sensor networks,” In *Proceedings of the 22nd International Conference on Distributed Computing Systems,* pages 457-458.

[31] Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). “Directed diffusion: a scalable and robust communication paradigm for sensor networks,” In *ACM MobiCom,* pages 56-67

[32] Kalpakis, K., Dasgupta, K., and Namjoshi, P.(August 26-29, 2002). “Maximum lifetime data gathering and aggregation in wireless sensor networks,” In *Proceeding of the 2002 IEEE International Conference on Networking (ICN’02),* Atlanta, Georgia, pages 685-696

[33] Khuller, S., Raghavachari, B., and Young, N. (1995). “Balancing minimum spanning trees and shortest-path trees” *Algorithmica,* 14(4):305-322.

[34] Krishnamachari, B., Estrin, D., and Wicher, S. (2002a). “The impact of data aggregation in wireless sensor networks,” In *Proceedings of the 22nd International Conference on Distributed Computing Systems,* pages 575-578

[35] Krishnamachari, B., Estrin, D., and Wicher, S. (2002b). “Modeling data-centric routing in wireless sensor networks,” *In USC, Technical Report CENG* 02-14

[36] Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. (Dec. 2002). “Tag: a tiny aggregation service for ad-hoc sensor networks,” In *USENIX Association 5th Symposium on Operating Systems Design and Implementation (OSDI),* pages 131-146

[37] Sadagopan, N. and Krishnamachari, B. (2004). “Maximizing data extraction in energy-limited sensor networks,” In *INFOCOM 2004,* volume 3, pages 1717-1727

[38] Ye, F., Luo, H., Cheng, J., Lu, S., and Zhang, L. (September 2002). “A two tier data dissemination model for large-scale wireless sensor networks,” In *proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MobiCom 2002),* pages 148-159.

[39] Yu, Y., Krishnamachari, B., and Prasanna, V. (2004). “Energy-latency tradeoffs fro data gathering in wireless sensor networks,” In *INFOCOM 2004. Twenty-third Annul Joint Conference of the IEEE Computer and Communications Societies,* pages 244-255.

VITA

Li Yin was born on July 13, 1981, in Henan, P.R.China. He earned the Bachelor of Science degree at Beijing University of Aeronautics and Astronautics in 2003. The degree of Master of Science in Computer Science will be conferred upon him in Dec, 2007, at the University of Missouri at Rolla.