Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution

Himanshu Gupta, Samir R. Das, Quinyi Gu
Department of Computer Science
State University of New York
Stony Brook, NY 11794
hgupta,samir,quinyigu@cs.sunysb.edu

ABSTRACT

Spatial query execution is an essential functionality of a sensor network, where a query gathers sensor data within a specific geographic region. Redundancy within a sensor network can be exploited to reduce the communication cost incurred in execution of such queries. Any reduction in communication cost would result in an efficient use of the battery energy, which is very limited in sensors. One approach to reduce the communication cost of a query is to self-organize the network, in response to a query, into a topology that involves only a small subset of the sensors sufficient to process the query. The query is then executed using only the sensors in the constructed topology.

In this article, we design and analyze algorithms for such self-organization of a sensor network to reduce energy consumption. In particular, we develop the notion of a *connected sensor cover* and design a centralized approximation algorithm that constructs a topology involving a near-optimal connected sensor cover. We prove that the size of the constructed topology is within an $O(\log n)$ factor of the optimal size, where n is the network size. We also develop a distributed self-organization version of our algorithm, and propose several optimizations to reduce the communication overhead of the algorithm. Finally, we evaluate the distributed algorithm using simulations and show that our approach results in significant communication cost reduction.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms

Algorithms, Performance

Keywords

Sensor networks, sensor coverage, sensor connectivity, query optimization, connected sensor cover

MobiHoc'03, June 1–3, 2003, Annapolis, Maryland, USA.

1. INTRODUCTION

Recent advances in miniature computing with advent of efficient short-range radios have given rise to strong interest in sensor networks [12, 2]. A sensor network consists of sensor nodes with a short-range radio and on-board processing capability. Each sensor can also sense certain physical phenomena like light, temperature, vibrations, or magnetic field around its location. The purpose of a sensor network is to process some high-level sensing tasks in a collaborative fashion, and is periodically queried by an external source to report a summary of the sensed data/tasks. For example, a large number of sensors can be scattered in a battlefield for surveillance purposes to detect certain objects of interest, say tanks. A typical query could be: Report the number of tank sightings at 10 minute intervals for the next 24 hours in a specific region within the battlefield.

Several new design themes have emerged for sensor networks. On one hand, the network must be self-configuring and highly fault-tolerant as the sensors may be deployed in an "ad hoc" fashion. On the other hand, as each sensor has only limited battery energy, the network as a whole must minimize total energy usage in order to enable untethered and unattended operation for an extended time. One technique to optimize energy usage during query execution would be for the network to self-organize, in response to a query, into a logical topology involving a minimum number of sensor nodes that is sufficient to process the query. Only the sensors in the logical topology would participate (communicate with each other) during the query execution. This is a very effective strategy for energy conservation, especially when there are many more sensors in the network than are necessary to process a given query. For example, two sensors in close enough proximity may generate the same or similar sensory data and it may be sufficient to involve only one of the sensors for query processing. The technique of self-organization exploits such redundancy effectively to conserve energy.

In order for the above technique to be of value, the number of control messages used in the self-organization process must be small, so that the overhead of the technique does not offset the expected benefit completely. Note that the overhead is paid only once for a given query, but the benefit is reaped during each execution of the query. Thus, a high overhead for such a technique could still be tolerated for highly redundant networks and/or long running queries.

In this paper, we design and analyze competitive algorithms for the above problem of self-organization of a sensor

network into an optimal logical topology in response to a query. In particular, we design an approximation algorithm that constructs such a topology in response to a query and show that the size of the topology returned by the algorithm is within an $O(\log n)$ factor of the size of an optimal topology, where n is the number of sensors in the network. We also design a distributed version of the proposed approximation algorithm that is run by the sensors in the network and results in a self-organization of the network into a topology involving a near-optimal number of sensors. Through further analysis and experiments, we also show that the communication overhead of the distributed algorithm is reasonably low which makes it very effective over a range of query and network parameters.

The rest of the paper is organized as follows. In Section 2, we provide a formulation of the problem with examples and discuss motivations. In Sections 3 and 4, we present the design and analysis of our proposed centralized approximation and the distributed self-organization algorithms. In Section 5, we present the simulation results depicting the performance of our proposed algorithm. We end with discussions on related work and concluding remarks in Sections 6 and 7, respectively.

2. PROBLEM FORMULATION AND MOTIVATION

In this section, we describe the problem addressed in the article through an example, present motivation, and give a formal definition of the problem. We start with a description of a sensor network model.

A sensor network consists of a large number of sensors distributed randomly in a geographical region. Each sensor has a unique identifier (ID) I and is capable of sensing a well-defined *convex* region S around itself called the *sensing region*. More will be said later about sensing regions. Each sensor also has an a radio interface and can communicate directly with some of the sensors around it. A query in a sensor network asks for a summarization of some sensed data/events over some time window and a geographical region, which is a subset of the overall region covered by the sensing regions of all the sensors in the network. By default, the geographical region associated with a sensor network query is the overall region covered by all the sensors in the network. A query is typically run multiple times, possibly, for different time windows.

Our article addresses the following optimization problem (formally defined later) that arises in sensor networks. Given a query over a sensor network, select a minimum set of sensors, called *connected sensor cover*, such that a) the sensing regions of the selected set of sensors cover the entire geographical region of the query, and b) the selected set of sensors form a connected communication graph where there is an edge between any two sensors that can directly communication with each other. The following example illustrates the problem.

EXAMPLE 1. Consider the sensor network shown in Figure 1. Sensors are represented by small circular dots. We have numbered the relevant sensors with $I_1, I_2, \ldots, I_9$ and shown sensing regions (circular S_i disks) associated with *some* of them (the black nodes/sensors). Let the distance between sensors I_1 and I_2 be equal to t. We assume that any two sensors that are roughly less than t distance apart

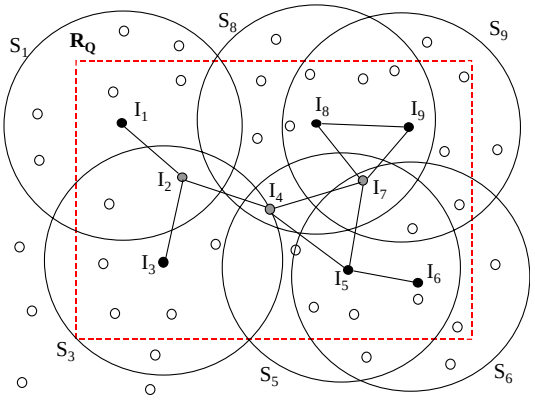


Figure 1: Connected Sensor Coverage Problem.

can directly communicate with each other. Now, let us consider a query over the geographic region represented by the rectangle R in the figure.

We can see that sensing regions associated with the black nodes $(I_1, I_3, I_5, I_6, I_8, I_9)$ are sufficient to cover the query's geographic region – the rectangle R_Q. However, the set of black nodes does not form a connected communication graph, as the sensor nodes I_1 and I_3 cannot communicate. However, if we add the grey sensor nodes (I_2, I_4, I_7) to the black nodes, we get a set of sensors that also forms a connected communication graph as shown in the figure. Thus, the set of nine sensors I_1 to I_9 form a connected sensor cover. Our problem is to find a minimum such cover. □

2.1 Motivation

The following two characteristics of sensor networks lend importance to the connected sensor coverage problem.

1. **Spatial Queries:** Due to the geographical distribution of sensors in a sensor network, each piece of data generated in the sensor network has a geographic location associated with it in addition to a timestamp[4, 13]. Hence, to specify the data of interest over which a query should be answered, each query in a sensor network has a time-window and a geographical region associated with it [4]. By default, the geographical region associated with such spatial queries is the full region covered by sensing regions of all the sensors in the sensor network.

2. **Limited Battery Power:** Sensors are miniscule computing devices with a limited battery power. Also, as evidenced in some recent studies [23], the energy budget for communication is many times more than computation with the available technology. Therefore, minimizing communication cost incurred in answering a query in a sensor network will result in longer lasting sensor networks. Hence, communication-efficient execution of queries in a sensor network is of significant interest.

The motivation for the connected sensor coverage problem addressed in this article comes from the presence of spatial queries in a sensor network and the importance of

executing such queries with minimal energy consumption. Given a query in a sensor network, we wish to select a small number of sensors that are sufficient to answer the query accurately. Also, the selected set of sensors should form a connected communication graph, so that they can form a logical routing topology for data gathering and transmission to the query source. Hence, we wish to select an optimal set of sensors that satisfy the conditions of coverage as well as connectivity, i.e., an optimal connected sensor cover as defined before. Constructing an optimal connected sensor cover for a query enables execution of the query in a very energy-efficient manner, as we need to involve only the sensors in the computed connected sensor cover for processing the query without compromising on its accuracy. Note here that we wish to combine coverage and connectivity in a single algorithm instead of using an alternative approach of treating them as two separate subproblems, as the optimal solution for the combined problem will be always equal or better than the solution obtained by solving for optimal coverage first and then for optimal connectivity. The reason for this is obvious – the sensors selected for mere connectivity in this alternative approach also contribute to coverage. Also, the alternative approach requires two phases and thus incurs possibly higher overheads. Further discussion on the alternative approach appears in Section 3, where a Steiner Tree based approach has been discussed.

The following discussion illustrates the savings in communication achieved by computing a connected sensor cover prior to the execution of a query.

Comparison with the Naive Approach: Given a query over a sensor network, a naive way to run the query will be to simply flood the network with the query. Each sensor node in the network broadcasts the query message exactly once and also remembers the ID of the sensor node it receives the query from. If there are n sensors whose sensing regions intersect with the query's region, then using about n message transmissions, a communication tree spanning the n sensors could be built within the network in a breadth-first manner. Each node in the built tree now responds to the query. The responses propagate upwards in the tree towards the root of the tree (the query source). This again incurs a cost of n message transmissions, assuming that responses are aggregated at each tree node. Thus, the total communication cost incurred in answering q such queries over the same region (possibly with different time windows) is $2qn$ using the above flooding approach.

Now, consider a connected sensor cover of size m sensors. As the connected sensor cover set induces a connected communication subgraph, the total cost incurred in executing q queries over the same region will be $D+2qm$, where D is the communication cost incurred in computing the connected sensor cover. If m is substantially less than n, as would be the case of reasonably dense sensor networks, constructing a connected sensor cover could result in large savings in communication cost even with an overhead D cost.

2.2 Formal Definition of the Problem

We now formally define the connected sensor cover problem addressed in this article. We start with a few definitions.

Definition 1. (**Communication Graph; Communication Distance**) Given a sensor network consisting of a set of sensors $\mathcal{I}$, the *communication graph* for the sensor network is the *undirected*[1] graph CG with $\mathcal{I}$ as the set of vertices and an edge between any two sensors if they can communicate directly with each other. The *communication subgraph induced* by a set of sensors $\mathcal{M}$ is the subgraph of CG involving only the vertices/sensors in $\mathcal{M}$.

An edge in the communication graph is referred to as a *communication edge* between the two given sensors. A path of sensors between I_1 and I_2 in the communication graph is called a *communication path* between the sensors I_1 and I_2. The *communication distance* between two sensors I_1 and I_2 is the length of the shortest path between I_1 and I_2 in the communication graph (which is the number of sensors on the shortest path, including I_1 and I_2). □

Definition 2. (**Connected Sensor Cover; Sensor Cover**) Consider a sensor network consisting of n sensors $I_1, I_2, \ldots, I_n$. Let S_i be the sensing region associated with the sensor I_i. Given a query Q over a region R_Q in the network, a set of sensors $\mathcal{M} = I_{i_1}, I_{i_2}, \ldots, I_{i_m}$ is said to be a *connected sensor cover* for Q if the following two conditions hold:

1. $R_Q \subset (S_{i_1} \cup S_{i_2} \cup \ldots S_{i_m})$
2. the subgraph induced by $\mathcal{M}$ in CG is connected, where CG is the communication graph of the sensor network. In other words, any sensor I_{i_j} in the connected sensor cover can communicate with any other sensor I_{i_k} in the cover, possibly through other sensors in the selected set $\mathcal{M}$.

A set of sensors that satisfies only the first condition is called a *sensor cover* for Q in the network. □

Connected Sensor Coverage Problem: Given a sensor network and a query over the network, the connected sensor coverage problem is to find the smallest connected sensor cover.

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 [16]. 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.

2.3 A Note on Sensing Regions

The sensing region associated with a sensor signifies an area for which the sensor can take the full responsibility for sensing a given physical phenomenon within a desired confidence. The real semantics of a sensing region is application specific. For example, for target detection/tracking applications, the sensing region is a region around the sensor within which the sensor can detect a target with a pre-determined minimum confidence. In such applications, the sensing region for a sensor could be modeled as a circular region of radius d around itself, where d is the distance beyond which a target cannot be detected within a given confidence. In some other applications, sensing regions are defined in terms of the resolution of the application queries or the correlation of the sensed data. For example, consider an application

[1]The algorithms and results in this article also apply to directed communication graphs, but we make the assumption for simplicity.

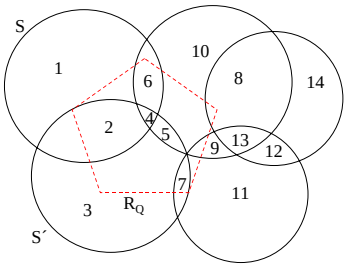


Figure 2: Subelements and Valid Subelements.

that gathers temperature samples in a geographical region monitored by a sensor network. Now, due to the spatial nature of temperature, temperature values at any two points that are less than d distance apart may be highly correlated. In such a case, we can again define sensing regions of circular radius d around each sensor.

As discussed above, typically, we can determine the sensing region for each sensor either as a static approximation of the sensor's location and capabilities, or as a function of query's resolution, or application's confidence requirements. The concept of sensing region similar to ours has been used in recent research, for example, by Slijepcevic and Potkonjak in [25], which addresses a closely related problem, and more recently, by Shakkottai et al. in [24].

If the sensing region is not known a priori, we can solve the connected sensor coverage problem iteratively for increasing sensing regions and pick the minimum solution whose gathered data is sufficiently accurate in comparison with the collective data of all the sensors. Otherwise, without the assumption of sensing regions, the connected sensor coverage problem could be formulated as a problem of selecting a minimum connected set of sensors such that every point in the query region gets a minimum amount of "exposure" from the selected set of sensors. Such a concept of exposure has been defined in [22] albeit in a different context.

In our treatment, the sensing regions can take any convex shape. The convexity assumption is needed to make Observation 1 (defined later). The convexity assumption will be true in practice, unless there are impregnable obstacles in the sensor network region. For ease of presentation, we have shown circular sensing regions in the figures throughout this article.

3. CENTRALIZED APPROXIMATION ALGORITHM

In this section, we present the approximation algorithm for the connected sensor coverage problem. The algorithm runs in polynomial time and guarantees a solution whose size is within $O(r \log n)$ of the optimal, where r is the link radius of the sensor network and is defined later. One of the important features of our algorithm is that it can be easily transformed into a distributed version that has low communication overhead.

Definition 3. (**Subelement; Valid Subelements**) Consider a geographic region with a number of sensing regions. A *subelement* is a set of points. Two points belong to same subelement iff they are covered by the same set of sensing regions. In other words, a *subelement* is a *minimal* region that is formed by an intersection of a number of sensing regions. Given a query region R_Q, a subelement is *valid* if its region intersects with R_Q.

In Figure 2, where R_Q is the given query region, there are fourteen subelements numbered 1 to 14, of which only 1 to 11 are valid subelements. □

Algorithm Description: We designed a greedy algorithm to select a connected sensor cover of near-optimal size. In short, the greedy algorithm works by selecting, at each stage, a path (communication path) of sensors that connects an already selected sensor to a partially covered sensor. The selected path is then added to the already selected sensors at that stage. The algorithm terminates when the selected set of sensors completely cover the given query region.

A more formal and complete description of the designed greedy algorithm is as follows. Let us assume that M is the set of sensors already selected for inclusion in the connected sensor cover by the greedy algorithm at any stage. Initially, M is an empty set. The algorithm starts with including in M an arbitrary sensor that lies within the query's region. At each stage, the greedy algorithm selects a sensor $\mathcal{C}$ (based on a criteria described later) along with a path/sequence of sensors $\mathcal{P}$ that forms a communication path between $\mathcal{C}$ and *some* sensor in M. The selected path of sensors $\mathcal{P}$, which includes $\mathcal{C}$, is then added to M. Thus, at any stage of the algorithm, the communication subgraph induced by M is connected. Also, if at each stage, the selected path of sensors $\mathcal{P}$ covers some yet uncovered (by M) area of the query's region, then the algorithm will eventually terminate with M as the connected sensor cover.

We now describe the criteria used in selection of $\mathcal{C}$ and $\mathcal{P}$ at any given stage of the algorithm. Any sensor $C_i \notin M$ whose sensing region contains a valid subelement, that has been covered by a sensor in M, becomes a *candidate sensor*, i.e., a potential candidate for selection as $\mathcal{C}$. For each such candidate sensor C_i, we construct a *candidate path* P_i of sensors that forms a communication path connecting C_i to some sensor in M. The candidate path P_i that covers the maximum number of uncovered valid subelements per sensor (defined as *benefit* of P_i) is added to M at that stage of the algorithm. We will illustrate the working of the algorithm through an example (Example 2) and describe the algorithm formally in Algorithm 1. First, we define some terms introduced in the above description.

Definition 4. (**Candidate Sensor; Candidate Path**) Let M be the set of sensors already selected by the algorithm. A sensor C is called as a *candidate sensor* if $C \notin M$ and the sensing region of C intersects with the sensing region of some sensor in M. A *candidate path* is a sequence/path of sensors that form a communication path connecting a candidate sensor C with some sensor in M. We use $|P|$ to denote the length of a candidate path P. □

Definition 5. (**Uncovered Valid Subelements; Benefit of a Candidate Path**) An *uncovered valid subelement* is a valid subelement that is not covered by any sensing region of a sensor in M, the set of sensors already selected for inclusion in the connected sensor cover by the algorithm. In Figure 2, if M contains the sensors corresponding to the two left-most sensing disks S and S', then the uncovered valid subelements are 8, 9, 10, and 11.

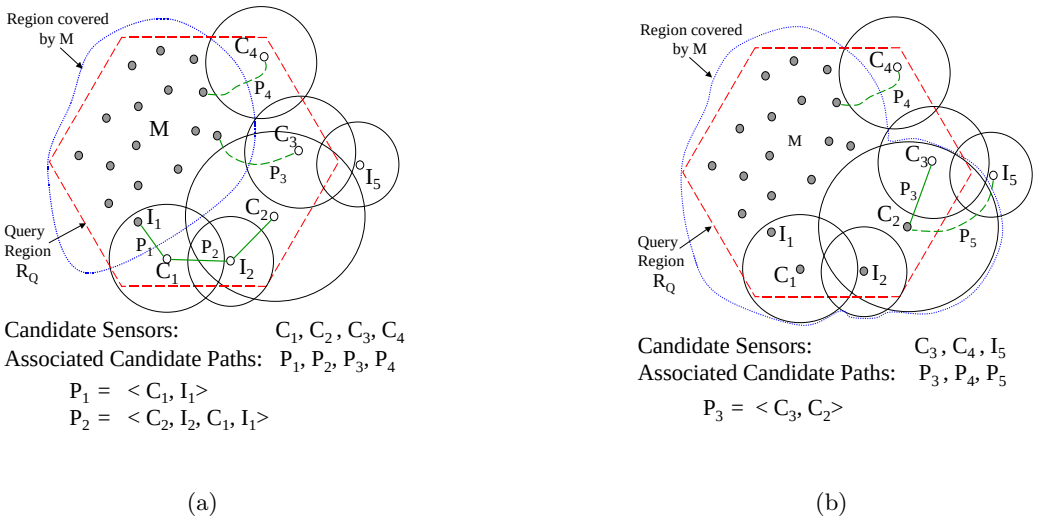


Figure 3: Working of the Centralized Algorithm.

The *benefit* of a candidate path P is the total number of uncovered valid subelements covered by the sensors in P divided by the number of sensors that are in P but not in M. The *most beneficial* candidate path is the candidate path that has the most benefit among the given set of candidate paths. □

EXAMPLE 2. Figure 3 shows a set sensors M (solid circular dots), the region covered by M, query region R_Q, and sensing disks corresponding to some of the sensors not in M (hollow circular dots). Figure 3 (a) and (b) depict two consecutive stages of the algorithm.

Let us consider the stage of the algorithm shown in Figure 3 (a). At this stage, there are at least four candidate sensors viz. C_1, C_2, C_3, C_4, as shown in the figure. The candidate paths associated with the candidate sensors are respectively P_1, P_2, P_3, P_4 as shown. For sake of clarity, we have not shown the set of sensors involved in the candidate paths P_3 and P_4, but the figure shows the actual communication edges and sensors involved in the candidate paths P_1 and P_2. Let us assume that the most beneficial candidate path at this stage is P_2. The algorithm then chooses P_2 as $\mathcal{P}$ and adds the sensors C_1, I_2, and C_2 to the set M.

The addition of sensors C_1, I_2, C_2 to M yields the next stage of the algorithm shown in Figure 3 (b). At this stage, the sensor I_5 becomes a candidate sensor, while C_1 no longer remains a candidate sensor. Also, the figure shows the recalculated and new candidate paths connecting each of the candidate sensors C_3, C_4, I_5 to some sensor in M. Here, we have assumed that the candidate path P_4 doesn't change from the previous stage, while the candidate path P_3 changes to the communication edge (C_3, C_2). Now, at this stage, if P_3 is the most beneficial path at this stage, the algorithm would add the sensor C_3 to M, which yields the next stage (now shown in the figure). Finally, if the algorithm adds to M a candidate path P_4 connecting C_4 to some sensor in M, the set of sensors M would now cover the entire query region and the algorithm returns the new M (obtained by adding C_3, C_4, and other sensors in P_4 to the M shown in Figure 3 (b)) as the connected set cover. □

Algorithm 1. Centralized Algorithm

Input: A sensor network consisting of the set of sensors $\{I_1, I_2, \ldots, I_n\}$. Each sensor I_i has a sensing region S_i associated with it. A query Q over a region R_Q in the network.
Output: A connected sensor cover M for query Q.

BEGIN
Let $\mathcal{Q}$ be the set of sensors whose sensing region intersects with R_Q. Let M denote the set of sensors selected by the algorithm at a given stage.
Let R_M be the region covered by M.
$M := \{I_i\}$, for some $I_i \in \mathcal{Q}$.
while $(R_Q \not\subseteq R_M)$
Let SC be the set of candidate sensors in $\mathcal{Q}$, i.e., the set of sensors in $\mathcal{Q} - M$ whose sensing region intersects with some sensor in M.
Max_Benefit := 0;
for each $C_i \in SC$
Find the most beneficial candidate path P_i for the candidate sensor C_i, i.e., a candidate path P_i with maximum benefit such that $P_i = <I_{i0}, I_{i1}, I_{i2}, \ldots, I_{il}>$ for some l, where $I_{il} \in M$, $I_{i0} = C_i$, and I_{ij} can communicate directly to $I_{i(j-1)}$.
Benefit := (No. of valid subelements covered by the region $((S_{i0} \cup S_{i1} \cup \ldots S_{il}) - R_M))/l$;
if (Benefit > Max_Benefit)
Max_Benefit := Benefit;
$\mathcal{P} := P_i$;
end if;
end for;
$M := M \cup \mathcal{P}$;
end while;
RETURN M;
END. ◇

Observation 1. The maximum number of subelements in a 2-dimensional plane with n disks is $n(n-1)+1$. If we have n *convex* objects, each having l sides, then the maximum number of subelements is $ln(n+1)$. □

From the observation, it is easy to see that Algorithm 1 can be implemented in $O(n^3)$ time, where n is the total number of sensors in the network, by building shortest communication paths for all pairs of sensors in $O(n^3)$ time at the beginning.

Definition 6. (**Link Radius**) The *link radius* of a sensor network is defined as the maximum communication distance between any two sensors whose sensing regions intersect. □

THEOREM 1. *Algorithm 1 returns a connected sensor cover of size at most $(r-1)(1+\log d)|OPT|$, where $|OPT|$ is the size of the optimal sensor cover (not necessarily connected), d is the maximum number of subelements in a sensing region, and r is the link radius of the sensor network. From Observation 1, the connected sensor cover size is within $O(r \log n)$ factor of the optimal.*

PROOF. Whenever, a candidate path of sensors $\mathcal{P}$ is selected for addition to M by Algorithm 1, we charge the uncovered valid subelements covered by $\mathcal{P}$ with $(|\mathcal{P}|-1)$, the number of the unselected (not in M) sensors in $\mathcal{P}$. The charge $(|\mathcal{P}|-1)$ is spread uniformly on each of the uncovered valid subelements covered by $\mathcal{P}$. Hence, each uncovered valid subelement gets charged by $(|\mathcal{P}|-1)/E_{\mathcal{P}}$ units, where $E_{\mathcal{P}}$ is the number of uncovered valid subelements covered by $\mathcal{P}$.

Let OPT be an optimal sensor cover for the given query region in the sensor network. Let us consider a sensor $I_i \in OPT$ and try to compute the maximum charge accumulated by the sensing region S_i of I_i during the entire course of Algorithm 1. At each stage of the algorithm, some uncovered valid subelements in the sensing region S_i get covered by the path of sensors $\mathcal{P}$ selected at that stage. Let e_j be the number of uncovered valid subelements of S_i after the j^{th} iteration of the while loop (j^{th} stage) of the algorithm. Here, e_0 is the total number of valid subelements of S_i. Now, the number of uncovered valid subelements of S_i covered during the j^{th} iteration is $e_{j-1}-e_j$. Note that $e_{j-1}-e_j$ may be zero, if there are no uncovered valid subelements of S_i covered in the j^{th} iteration.

If $\mathcal{P}^j$ is the candidate path selected for addition at the j^{th} iteration and $E_{\mathcal{P}^j}$ is the number of uncovered valid subelements covered by $\mathcal{P}^j$, then the total charge T accumulated by S_i during the entire course of the algorithm is:

$$T = \sum_{j=1}^{j=k} (e_{j-1}-e_j) * (|\mathcal{P}^j|-1)/E_{\mathcal{P}^j},$$

assuming the loop runs for k iterations. As our goal is to only compute an upper bound on T, let us assume, without loss of generality, that all the terms in the above series are non-zero, i.e., for all j, $\mathcal{P}^j$ covers a non-zero number of uncovered valid subelements in S_i.

Now, $E_{\mathcal{P}^1}/(|\mathcal{P}^1|-1) \geq (e_0-e_1)/(r-1)$ and $E_{\mathcal{P}^j}/(|\mathcal{P}^j|-1) \geq e_{j-1}/(r-1)$ for $j \geq 2$, as I_i (along with a candidate path of length at most r and benefit of at least $e_{j-1}/(r-1)$) becomes a candidate sensor eligible for selection as soon as some valid subelement of I_i gets covered by a sensor in M. Thus,

$$T/(r-1) \leq 1 + \sum_{j=2}^{j=k} (e_{j-1}-e_j)/e_{j-1}.$$

Using some algebra ([8], Chapter 35.3), the above gives $T \leq (r-1)(1+\log e_0)$, where $e_0 < d$ is the total number of valid subelements in S_i. As the total charge spread (on the sensing regions of the optimal sensor cover, OPT) during each stage of the algorithm is the number of *new* sensors added to M, the total charge accumulated on the sensing regions of all the sensors in OPT is equal to number of sensors in the solution returned by the algorithm. Thus, the size of the solution M returned by Algorithm 1 is at most $(r-1)(1+\log d)|OPT|$. Note that both r and d are functions of network density. □

Steiner Tree Based Approach: One way to solve the connected sensor cover problem would be to construct a sensor cover and then build a Steiner tree [3] to connect the sensors in the sensor cover. This Steiner tree based approach is conceptually simpler and yields the same theoretical bound ($O(r \log n)$) on the size of the solution returned. However, the distributed implementation of such an approach would require two phases – one to compute the sensor cover and then another to construct the Steiner tree, and thus will possibly incur a higher communication cost than our proposed distributed algorithm in Section 4. Nevertheless, Steiner tree based approach could be used in scenarios, where the sensors selected for coverage and those selected for connectivity incur different costs with the former paying higher, as the activity of sensing and related signal processing may incur energy costs. We are investigating the Steiner tree based algorithm for various possibilities of relative sensing and communication costs, as part of our future work.

3.1 Weighted Version

Algorithm 1 can be generalized to handle the weighted version of the connected sensor coverage problem. In the weighted setting, each sensor has a weight associated and we wish to select a connected sensor cover with the minimum total weight. In practice, we would assign higher weights to sensors that have a lower battery life and/or are critical to the viability of the sensor network so that they have a lesser chance of being selected.

The benefit of a candidate path in the weighted case is defined as the total number of uncovered valid subelements covered by P divided by the total weight of the sensors that are in P, but not M. Thus, to handle the weighted case, the value of `Benefit` in the algorithm is computed as follows:

Benefit = (Number of valid subelements covered by the region $((S_{i0} \cup S_{i1} \cup \ldots S_{il} \cap R_Q) - R_M))/(\sum_{j=0}^{l-1} w_{ij})$, where w_{ij} is the weight of the sensor I_{ij}.

Definition 7. (**Weighted Communication Distance; Weighted Link Radius**) The *weighted communication distance* between two sensors is the weight of the minimum weighted communication path between them.

The *weighted link radius* of a sensor network is defined as the maximum weighted communication distance between any two sensors whose sensing regions intersect. □

THEOREM 2. *For the weighted connected sensor coverage problem, the generalization of Algorithm 1 returns a connected sensor cover of total weight at most $r(1+\log d)|OPT|$, where $|OPT|$ is the total weight of an optimal sensor cover, d is the maximum number of subelements in a sensing region, and r is the weighted link radius of the sensor network.*

PROOF. The proof follows the proof of Theorem 1. However, in the weighted case, the total charge T accumulated by S_i is given by:

$$T = \sum_{j=1}^{j=k} (e_{j-1} - e_j) * (W_{\mathcal{P}^j} - w_{il})/E_{\mathcal{P}^j},$$

where $W_{\mathcal{P}^j}$ is the total weight of the sensors in $\mathcal{P}^j$. Now, $E_{\mathcal{P}^1}/(W_{\mathcal{P}^1} - w_{1l}) \geq (e_0 - e_1)/r$ and $E_{\mathcal{P}^j}/(W_{\mathcal{P}^j} - w_{il}) \geq e_{j-1}/r$ for $j \geq 2$, as I_i (along with a candidate path of weight at most r and benefit of at least e_{j-1}/r) becomes a candidate sensor eligible for selection as soon as some valid subelement of I_i gets covered by a sensor in M. Thus,

$$T/r \leq 1 + \sum_{j=2}^{j=k} (e_{j-1} - e_j)/e_{j-1}.$$

Similar to the previous proof, the above gives $T \leq r(1 + \log e_0)$, where e_0 is the total number of valid subelements in S_i. Note that the total charge accumulated on the sensing regions of all the sensors in OPT is equal to number of sensors in the solution returned by the algorithm. Thus, the total weight of the solution M returned by the algorithm is at most $r(1 + \log d)|OPT|$. □

4. DISTRIBUTED SELF-ORGANIZATION ALGORITHM

In this section, we describe the self-organizing distributed version of Algorithm 1. As stated before, one of the key features of our proposed approximation algorithm is that it lends to a very natural distributed algorithm.

Like the centralized algorithm, the self-organizing distributed algorithm goes through a sequence of stages to build a connected sensor cover within the sensor network for a given query. Throughout the course of the algorithm, the sensor network maintains the following values:

- M, a set of sensors that have already been selected for inclusion in the connected sensor cover by the algorithm. Like the centralized algorithm described in the previous section, the distributed algorithm also increments M by adding a candidate path of sensors to M at each stage.
- SP, a set of candidate paths. Recall that, a candidate path is a sequence of sensors that form a communication path connecting a candidate sensor to some sensor in M, where a candidate sensor is a sensor whose sensing region intersects with some sensing region of a sensor in M. Each candidate sensor has exactly one candidate path associated with it.
- $\mathcal{P}$, the most recently added candidate path, and $\mathcal{C}$, the candidate sensor associated with $\mathcal{P}$.

Each sensor in the network is aware of its membership in M, or $\mathcal{P}$, or in a candidate path in SP. Also, the most recently added candidate sensor $\mathcal{C}$ stores the values M, SP, and $\mathcal{P}$. Each stage of the distributed algorithm consists of the following sequence of transmission phases.

1. **Candidate Path Search:** The most recently added candidate sensor $\mathcal{C}$ broadcasts a *Candidate Path Search (CPS)* message to all sensors within $2r$ communication hops, to select new candidate paths and candidate sensors. Here, r is the link radius of the sensor network. We choose $2r$ (instead of r) so that the CPS message from $\mathcal{C}$ reaches even those candidate sensors whose sensing disks intersect with that of other sensors in $\mathcal{P}$, the most recently added candidate path associated with $\mathcal{C}$.
2. **Candidate Path Response:** Any sensor I that receives a CPS message checks to see if it is a candidate sensor, i.e., if I's sensing region intersects with the sensing region of some sensor in M. If I is a candidate sensor, it unicasts a *Candidate Path Response (CPR)* message to the originating sensor $\mathcal{C}$ of the CPS message. The CPR message contains as candidate path P the sequence of sensors (including I) that the received CPS message passed through since its origination.
3. **Selection of Best Candidate Path/Sensor:** The sensor $\mathcal{C}$, which was the originator of the CPS messages in the current stage, collects all the CPR messages sent to it by the candidate sensors. The candidate path P contained in each received CPR message is added by $\mathcal{C}$, after appropriate truncation, to SP, the set of candidate paths being maintained by the sensor network. After having received all the CPR messages sent to $\mathcal{C}$ during this stage, the sensor $\mathcal{C}$ selects the most beneficial candidate path $\mathcal{P}_{new}$ among all the candidate paths in SP. Let, $\mathcal{C}_{new}$ be the candidate sensor associated with the picked candidate path $\mathcal{P}_{new}$, and let $\mathcal{I}_{new}$ be the set of sensors in the candidate path P_{new}. The sensor $\mathcal{C}$ unicasts a NewC message to $\mathcal{C}_{new}$ with the following updated information: $M = M \cup \mathcal{I}_{new}$; $\mathcal{P} = \mathcal{P}_{new}$; $SP = SP - \mathcal{P}_{new}$. Note that SP has also been augmented with all the candidate paths received in the CPR messages.
4. **Repeat:** The sensor $\mathcal{C}_{new}$ receives the NewC message sent to it by $\mathcal{C}$. After receiving the message, $\mathcal{C}_{new}$ updates the value $\mathcal{C}$ to itself (i.e., $\mathcal{C} = \mathcal{C}_{new}$). That completes the current stage of the algorithm. The above process repeats till the selected set of sensors M cover the entire query region in the sensor network.

The above distributed algorithm guarantees a self-organization of the sensor network into a logical topology that represents a connected sensor cover for the given query. To reduce the *size* of the `CPS` and `NewC` messages, we represent the set M by only the boundary sensors, i.e., the sensors that are on the boundary of the region M covers. On an average, the number of boundary sensors should be the square root of the number of sensors in M.

4.1 Optimizations to Reduce Number of Messages

The following optimization techniques have been used by the distributed algorithm to reduce the number of messages incurred during the self-organization.

1. To reduce the number of messages for coordination, we reuse the candidate paths computed for candidate sensors at later stages of the algorithm. In contrast, the centralized algorithm recomputes the (best) candidate paths for each candidate sensor at each stage and picks the most beneficial candidate path. However, the distributed algorithm does optimize already computed candidate paths by truncating them if some sensor in the candidate path has been newly added to M. Also, the distributed algorithm does recalculate the benefit of each candidate path in SP at each stage.

2. To compute the benefit of a candidate path, the distributed algorithm computes the *area* of the uncovered query region covered by the candidate path instead of the number of uncovered valid subelements covered by the candidate path.

3. To reduce the number of broadcast CPS messages, we stipulate that if a sensor has already been selected in M then it does not broadcast a CPS message received from another sensor. Also, a sensor broadcasts a CPS broadcast message only once during any one stage of the algorithm.

We observed through extensive experiments (as shown later in Figure 4(b)) that the above optimizations do not increase the size of the connected sensor cover constructed. However, they do result in substantial savings in communication cost.

Number of Messages: If the NewC messages are transmitted through an optimal path within M, it is not difficult to show that the total number of messages transmitted during the entire course of the distributed algorithm is $O(k(\log m + b))$ for uniformly distributed sensors, where k is the number of stages the algorithm goes through before terminating, m is the size of connected sensor cover constructed, b is the maximum number of sensors within $2r$ communication hops of any sensor. Here, as in the simulation experiments in the next section, we assume piggybacking of CPR messages at each stage, i.e., during the CPR phase each sensor waits sufficiently long to collect all CPR messages intended to pass through it, and then unicasts all the collected CPR messages to the $\mathcal{C}$ in one message.

5. PERFORMANCE EVALUATION

We have constructed a simulator to evaluate the performance of the distributed self-organization algorithm, and contrast it with the naive flooding-based approach (see Section 2.1). The simulator uses randomly placed sensor nodes in a rectangular region. All sensor nodes have a circular sensing region of radius s associated with them. A communication edge exists between two sensor nodes if they are within a certain distance, called *transmission radius*, from each other. The size of the rectangular region, number of nodes (n), sensing radius (s), and transmission radius (t) are input parameters of the simulator. The link radius (r) is computed in terms of the above parameters and will be described later.

The simulator only models message transmissions. It does not model any link layer protocol or wireless channel characteristics. Thus, all the messages in the simulator are transmitted in an error-free manner.[2] Also, the passage of time is modeled in a time-stepped fashion, wherein during each step, each node receives messages, performs appropriate computations in response to these messages, and then sends out messages as a result of these computations. While such a simulator models an idealized communication subsystem, it is sufficient for our purpose, as we are only interested in counting message transmissions.

As in Section 2.1, let D be the number of messages needed to compute the connected sensor cover and m be the size of the computed connected sensor cover. We assume that the spatial query runs over the entire geographic region with a randomly selected sensor as the query source. The simulator computes D and m, for a given set of input parameters. Let us assume that the query runs q times. We evaluate the threshold value q_θ, such that for $q > q_\theta$ the overall message cost for the query using our distributed self-organization algorithm is lower than the message cost using the naive approach. The number q_θ is obtained by equating $D + 2qm$ to $2qn$ and then solving for q, which gives

$$q_\theta = \frac{D}{2(n-m)}.$$

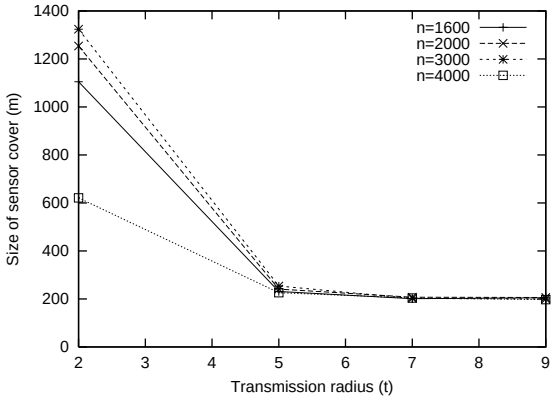
In the simulation results that follow, we have used a 100×100 area. Sensors have a sensing radius $s = 4$. We vary the number of sensors (n) and transmission radius (t); t is varied from 2 to 9, and n from 1600 to 4000. This range of parameters allows us to study performance for very sparse to very dense networks. Our experiments with still lower values of t and n showed that the network was too sparse that a connected sensor cover did not exist. Also, for $t > 8$, the sensors with intersecting sensor disks are reachable within one hop (i.e., $r = 2$). Thus, one set of experiments for $t > 8$ is sufficient.

Note also that only the spatial density of the sensors and the ratio of the sensing and transmission radii affect the performance of the algorithm. Thus, there is no need to vary the size of the area and the sensing radius.

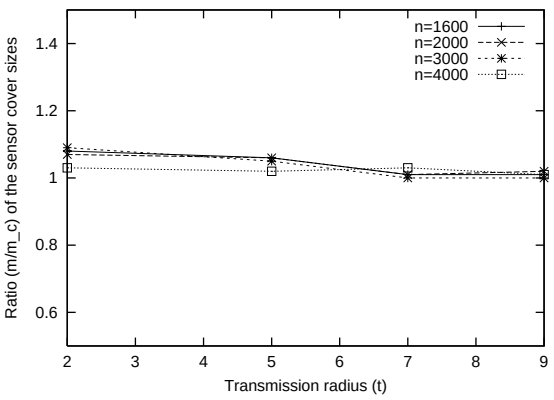
Before we describe the performance plots, we add a note below on computation of the link radius r. While an exact computation is neither necessary nor practical, it is important to have fair idea of the value of r for the given parameters of the network. Too large a value will increase the discovery cost D. Too small a value will decrease cost, but may result in sub-optimal solutions (i.e., large m) or may even fail to reach a solution.

Calculation of the link radius (r): Consider two sensors I_1 and I_2 whose sensing disks touch, i.e., the distance between them is $2s$. Now, if there are enough sensors in between I_1 and I_2 on the straight line connecting I_1 and I_2, there will be a communication path of length $2s/t + 1$ connecting I_1 and I_2. We characterize a network as *dense* if there exist enough $(2s/t)$ number of sensors evenly spread in between (along the line connecting) any two sensors with intersecting sensing disks. The link radius r should be $(2s/t+$

[2]The effect of transmission errors or message losses is a part of our future work. Our algorithms can be extended to use local repair mechanisms to compensate for message losses.



(a) Size of connected sensor cover (m)



(b) Ratio m/m_c

Figure 4: Size of the sensor cover (m) computed by the distributed algorithm and the its ratio with that computed by the centralized algorithm (m/m_c), shown for various transmission radii and number of sensors.

1) for such dense networks.

Without going through a complex probabilistic analysis, it is not possible to accurately calculate the minimum number of random sensors required in a given area for a network to be dense (as defined above) with high probability. We do not feel that such an analysis is warranted as an accurate computation of r is not essential. For our evaluation, we simply consider networks with more than $4s/t$ sensors within a distance $2s$ (i.e., with a linear density of $2/t$) as dense. Since we are using a 100×100 area, a dense network should have at least $(200/t)^2$ sensors. Thus, for dense networks with more than $(200/t)^2$ sensors, we use $r = (2s/t+1)$. For a non-dense network, we simply use a proportionate density factor to "inflate" the value of r, i.e., for a network with n sensors where $n < (200/t)^2$, we use $r = (2s/t+1) * ((200/t)^2/n)$. A fractional value of the computed r is simulated by using a probablity for the last hop forwarding of the CPS message. For example, if $r = 2.3$, the CPS massage on the third hop is forwarded with 30% probablity.

Simulation Results: Figure 4 plots m and ratio m/m_c for various values of n and transmission radius t, where m and m_c are the sizes of the connected sensor covers computed by the distributed algorithm and the centralized algorithm respectively. Note that m and m_c are very small relative to the network size n except for low n and t when the communication graph is very sparse and there is low redundancy in the network. Figure 4(b) depicts excellent performance of the distributed algorithm relative to the centralized version. The ratio m/m_c always remains close to the ideal value of 1. Note here that the distributed algorithm includes optimizations mentioned in Section 4.1. Thus, the optimizations introduced in the distributed algorithm to reduce communication cost do not impact the m/m_c ratio, which remains close to the ideal. Also, the above observation validates our method for computation of r. In fact, lower values of r could be possibly used without impacting the m/m_c ratio significantly, but reducing the communication cost D. Thus, the performance our algorithm could be further improved.

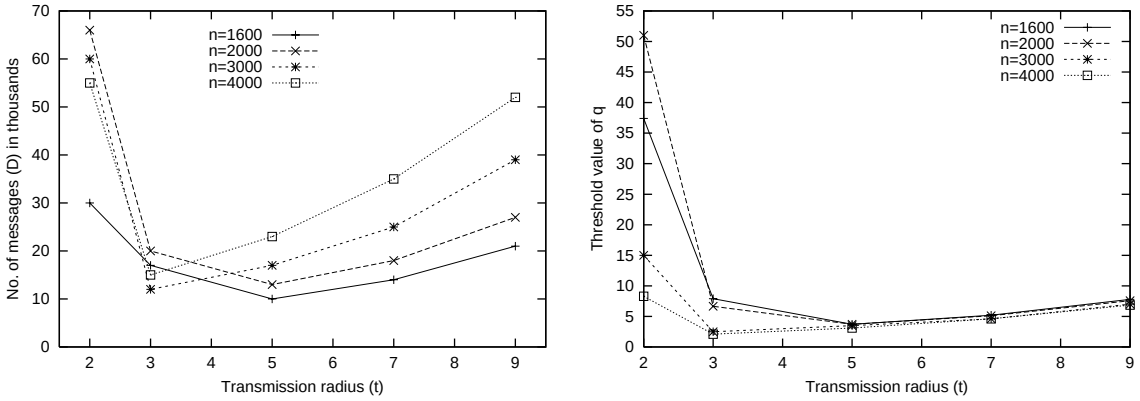
Figure 5(a) depicts how the communication cost D in the distributed algorithm changes with n and t. The explanation for the cusp shape of the D vs. t plot is as follows. From the above discussion, there is a threshold value of t, above which the network becomes dense for a constant n. This threshold is $t_\theta = 200/\sqrt{n}$. Given a network of size n, for $t < t_\theta$ (non-dense networks), the number of neighbors b within $2r$ hops of a sensor is very high, as $b \propto (rt)^2 \propto 1/t^4$ based on our computation of r for non-dense networks. Hence, we see a high number of messages for very low t, which makes sense as the communication graph is sparse for low t.

Now, for $t > t_\theta$, the network is dense and $r = 2s/t + 1$. Thus, b, which is proportional to rt, remains almost constant for $t > t_\theta$. However, with the increase in t, the link radius r decreases which causes an increase in k, the number of stages of the algorithm. With b and m (as seen in Figure 4(a)) being relatively constant, the increase in number of stages causes a slow increase in the number of messages. Also, as $t_\theta \propto 1/\sqrt{n}$, t_θ is larger for smaller n. Hence, we see that the minimum number of messages reached for 1600-2000 number of sensors is at $t = 5$, while for higher number of sensors (n) the minimum is reached at a lower transmission radius ($t = 3$).

Figure 5(b) plots q_θ vs. n for different values of t. This plot is somewhat similar to the plot of D, because of the strong dependency of q_θ on D. Notice that the value of q_θ is fairly small – almost always less than 7 except when the communication graph is very sparse (low n together with low t). This shows that except for very sparse networks, our self-organization technique will always save energy relative to the naive flooding approach, whenever the query runs for more than about 7 times – longer runs giving more energy saving benefits.

6. RELATED WORK

The work most closely related to ours is that by Slijepcsevic and Potkonjak [25], where the authors consider power-efficient organization of sensor networks. They introduce a heuristic that selects mutually exclusive sets of sensors, the members of each of those sets together completely cover the



(a) Communication cost (D)

(b) Threshold value q_θ

Figure 5: The communication cost (D) for the distributed algorithm and the threshold value of q (q_θ) shown for various transmission radii and number of sensors.

monitored/query area. As only one set of sensors need to be active at any time, their technique results in energy savings while preserving coverage. They only present a centralized algorithm which does not extend easily to a distributed algorithm. Moreover, they do not consider the communication cost incurred in the execution of their heuristic or the query. Hence, they do not require the selected sets to be connected. We should note that a repeated execution of our algorithm gives a good heuristic for the problem addressed in their work. In other closely related work, Shakkottai et al. in [24] consider an unreliable sensor grid-network and derive necessary and sufficient conditions for the coverage of the region and connectivity of the network in terms of the transmission radius, sensing radius, and failure rate of the sensors.

In [21], the authors formulate coverage problems to address the quality of service (surveillance) provided by a sensor network. In particular, they address the problem of finding maximal paths of lowest and highest observabilities in a sensor network, which is very different than our connected sensor coverage problem. The work in [19] addresses the problem of selecting a set of k base stations from a given set of n stations to maximize radio coverage. The issue of connectivity, query execution, or energy-efficiency does not arise in the radio coverage setting.

There has been a significant amount of work on developing efficient mechanisms to broadcast a message in a mobile ad hoc network. The idea here is to suppress redundant broadcasts by using only a small number of nodes to broadcast but ensuring that all the nodes in the network receive the broadcast message. The above described problem of selecting a minimum number of nodes such that each node in the network is either selected or a neighbor of a selected node is known as the minimum dominating connected set (MDCS) problem [14]. The work in wireless network research community [9, 18, 26, 1, 7] has primarily focussed on developing energy-efficient distributed algorithms to construct a near-optimal dominating connected set. However, the notion of dominating set is significantly different than that of sensor cover.

Other related problems based on various other notions of coverage are as follows. The Art Gallery Problem [20, 10] considers placement of observers in a room such that each point in the room is "seen" by at least one observer. The Art Gallery Problem was solved optimally in 2D and shown to be NP-hard in 3D case. The essential difference of the Art Gallery and the related problems with our connected sensor coverage problem is that the Art Gallery and related problems require an optimal *placement* of observers, while our problem deals with an optimal *selection* of already placed sensors. From that perspective, the geometric variations [16, 17, 5] of the classic set cover problem are more related to our problem. However, none of the geometric set cover variations addressed in the literature deal with the notion of connectivity. For the disk-cover problem [5], there is a polynomial algorithm that delivers a constant-factor approximation, however, the approximation algorithm does not generalize to other geometric regions (not even rectangles) and due to involved computation required, the straightforward distributed implementation would incur a very large number of messages.

There has been some other work done on optimizing energy consumption in sensor networks. For example, in [6], heuristic techniques have been designed to put sensor network nodes in inactive states based on the observed connectivity in the network and loss of messages. In [15], a randomized clustering algorithm has been devised to select cluster heads in a sensor network so that the energy budget for relaying information to a gateway is distributed evenly among sensor nodes. None of the above work deals with the notions of spatial coverage or connectivity.

7. CONCLUSIONS

We have designed efficient algorithms for self-organization in sensor networks. The self-organization algorithms exploit the redundancy in the sensor network to select a small subset of sensors (called *connected sensor cover*) that is sufficient to process a given query. The motivation is to conserve the overall battery power of the network. We show that the

centralized algorithm computes a near-optimal solution – within logarithmic bound of the optimal. The distributed version uses certain optimizations to reduce message overheads and the simulations show that the size of the solution delivered is of almost the same size as the centralized algorithm.

The sensors that are not selected in the connected sensor cover are not used during query execution, but may be used during the self-organization phase. Since the communication cost incurred in a query execution is proportional to the number of sensors involved, our scheme is able to reduce communication cost substantially. The cost savings are proportional to (i) the density of the network (which reflects on the redundancy) and (ii) the number of times the query is run – longer running queries result in greater cost savings as it reduces the amortized overhead cost of running the self-organization algorithm. We show through simulations that the overhead cost is indeed small enough that running queries for more than a few times (about 7) starts generating cost savings for a wide range of sensor network parameters.

The weighted version algorithm takes into account the remaining battery power in the sensors so that sensors running low on battery has a smaller chance of being selected in the connected sensor cover. This gives a tremendous flexibility for balancing the available energy budget in the network among all sensors, thus providing a longer operational life time. It also worthwhile to note that while we focused primarily on communication cost savings as a method to conserve battery power, our technique can potentially provide further savings depending on the architecture of the sensor. For example, the sensors not in the connected sensor cover can be put to sleep for the duration of time the query is to run (assuming, of course, that they are not used for other concurrent queries). Our technique can also be used to compute multiple disjoint connected sensor covers in a distributed manner. Multiple connected sensor covers can be useful for very long running queries; different covers can be used at different times to balance the battery drain over different sensors.

8. ACKNOWLEDGEMENTS

Samir Das's work has been partially supported by NSF grant ANI-0308631.

9. REFERENCES

[1] K. M. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proc. of the Symp. on Mobile ad hoc networking and computing*, 2002.

[2] B. Badrinath, M. Srivastava, K. Mills, J. Scholtz, and K. Sollins, editors. *Special Issue on Smart Spaces and Environments, IEEE Personal Communications*, 2000.

[3] P. Berman and V. Ramaiyer. Improved approximation algorithms for the steiner tree problem. *J. Algorithms*, 1994.

[4] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *Proc. of Intl. Conf. on Mobile Data Management*, 2001.

[5] H. Bonnimann and M. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete Computational Geometry*, 14, 1995.

[6] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *Proc. of the Conf. on Computer Communications (INFOCOM)*, 2002.

[7] Y. Chen and A. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *Proc. of the Symp. on Mobile ad hoc networking and computing*, 2002.

[8] T. Cormen, C. Lieserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw Hill, 2001.

[9] B. Das, R. Sivakumar, and V. Bhargavan. Routing in ad hoc networks using a spine. In *Proceedings of the Intl. Conf. on Computer Communications and Networks (IC3N)*, 1997.

[10] M. de Berg, M. van Kreveld, M. Overmans, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[11] B. Deb, S. Bhatnagar, and B. Nath. Multiresolution state retrieval in sensor networks. In *Proceedings of International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[12] D. Estrin, R. Govindan, and J. Heidemann, editors. *Special Issue on Embedding the Internet, Communications of the ACM*, volume 43, 2000.

[13] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker. The sensor network as a database. Technical report, University of Southern California, Computer Science Department, 2002.

[14] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4), 1998.

[15] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of Intl. Conf. on System Sciences (HICSS)*, 2000.

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

[17] V. S. A. Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. In *ACM-SIAM Symp. on Theory of Computing*, 1999.

[18] A. Laouiti, A. Qayyum, and L. Viennot. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. In *Proc. of the Hawaii Intl. Conf. on System Sciences (HICSS)*, 2002.

[19] K. Lieska, E. Laitinen, and J. Lahteenmaki. Radio coverage optimization with genetic algorithms. In *Proc. of IEEE Int. Symp. on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 1998.

[20] M. Marengoni, B. Draper, A. Hanson, and R. Sitaraman. System to place observers on a polyhedral terrain in polynomial time. *Image and Visual Computing*, 1996.

[21] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proc. of the Conf. on Computer Communications (INFOCOM)*, 2001.

[22] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, 2001.

[23] G. Pottie and W. Kaiser. Wireless sensor networks. *Communications of the ACM*, 43, 2002.

[24] S. Shakkottai, R. Srikant, and N. Shroff. Unreliable sensor grids: Coverage, connectivity and diameter. In *Proceedings of INFOCOM (to appear)*, 2003.

[25] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *Proc. of IEEE Intl. Conf. on Communications (ICC)*, 2001.

[26] J. Wu and H. Li. A dominating-set-based routing scheme in ad hoc wireless networks. *Telecommunication Systems Journal*, 3, 2001.