Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution[1]

Himanshu Gupta, Zongheng Zhou, Samir R. Das, Quinyi Gu
Department of Computer Science
State University of New York
Stony Brook, NY 11794
hgupta,zzhou,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. The self-organization technique is beneficial for queries that run sufficiently long to amortize the communication cost incurred in self-organization.**

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 develop a distributed self-organization version of the approximation algorithm, and propose several optimizations to reduce the communication overhead of the algorithm. We also design another distributed algorithm based on node priorities that has a further lower communication overhead, but does not provide any guarantee on the size of the connected sensor cover constructed. Finally, we evaluate the distributed algorithms using simulations and show that our approaches results in significant communication cost reductions.

***Keywords*— Sensor networks, sensor coverage, sensor connectivity, query optimization, connected sensor cover**

1 Introduction

Recent advances in miniaturization of computing devices with advent of efficient short-range radios have given rise to strong interest in sensor networks [1, 2]. A sensor network consists of sensor nodes with short range radios 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 design a distributed version of the same 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. We also design another energy-efficient distributed algorithm based on node priorities that incurs a lower communication overhead, but does not provide any guarantee on the solution size constructed. Through further analysis and experiments, we show that the communication overhead of our distributed

[1]Preliminary version appeared in *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003. Author Samir Das's work has been partially supported by NSF grant ANI-0308631.

algorithms is reasonably low which makes them very effective over a wide 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 algorithms. 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. 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.

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 [3, 4]. 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 [3]. 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 [5], 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 query region now responds to the query, and the responses propagate upwards in the tree towards the root of the tree (the query source). The responses also incurs a cost of n message transmissions, assuming that responses are aggregated at each tree node. Thus, to answer q queries asked at different times to gather various snapshots of the query region, the above process of flooding and responding needs to be repeated, and hence, the total communication cost incurred is $2qn$ using the above naive 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*[2] 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. □

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

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 [6]. 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 predetermined 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 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 [7], which addresses a closely related problem, and more recently, by Shakkottai et al. in [8].

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 [9] 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 ob-

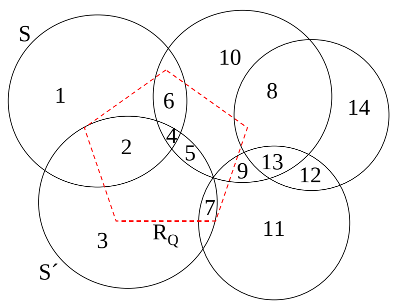


Figure 1: Subelements and Valid Subelements.

stacles in the sensor network region. For ease of presentation, we have shown circular sensing regions in the figures throughout this article. In the absence of the convexity assumption, the techniques presented in this article are still applicable, but the designed *centralized* approximation algorithm may no longer run in polynomial time.

3 Centralized Approximation Algorithm

In this section, we present a centralized 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 1, 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 $\hat{C}$ (based on a criteria described later) along with a path/sequence of sensors $\widehat{P}$ that forms a communication path between $\hat{C}$ and *some* sensor in M. The selected path of sensors $\widehat{P}$, which includes $\hat{C}$, 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 $\widehat{P}$ 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 $\hat{C}$ and $\widehat{P}$ 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 $\hat{C}$. 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 1). First, we define some terms introduced in the above description.

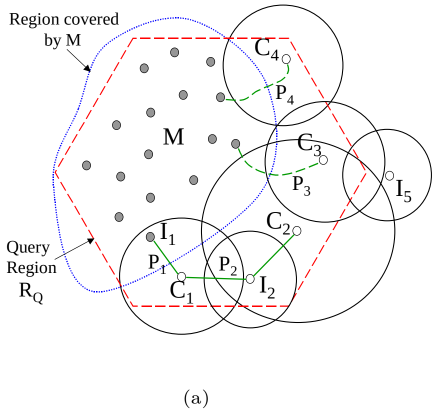
Definition 4 (Candidate Sensor; Candidate Path) Let M be the set of sensors already selected by the algorithm. A sensor C is called as a *candidate sensor* if $C \notin M$ and the sensing region of C intersects with the sensing region of some sensor in M. A *candidate path* is a sequence/path of sensors 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 1, 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.

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 1 Figure 2 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 2(a) and (b) depict two consecutive stages of the algorithm.

Let us consider the stage of the centralized approximation algorithm shown in Figure 2(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



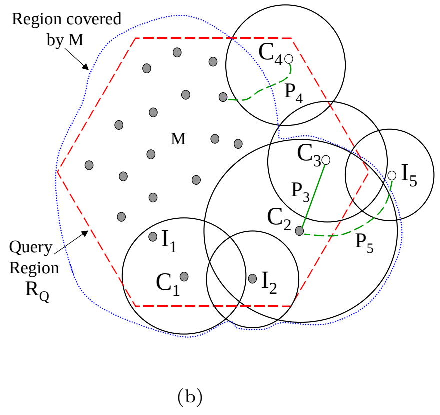


Figure 2: Working of the Centralized Approximation Algorithm. (a) Candidate sensors: C_1, C_2, C_3, C_4; Associated candidate paths: P_1, P_2, P_3, P_4; $P_1 = <C_1, I_1>, P_2 = <C_2, I_2, C_1, I_1>$. (b) Candidate sensors: C_3, C_4, I_5; Associated candidate paths: P_3, P_4, P_5; $P_3 = <C_3, C_2>$.

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 $\widehat{P}$ 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 2(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 2(b)) as the connected set cover solution. □

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 $n^2 l$.* □

From the observation, it is easy to see that the centralized approximation algorithm 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 *The centralized approximation algorithm 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.* □

Steiner Tree Based Approach: One way to solve the connected sensor cover problem in a centralized manner would be to construct a sensor cover and then build a Steiner tree [11] to connect the sensors in the sensor cover. To construct the sensor cover, we can use the greedy algorithm, which will deliver a solution within a $O(\log n)$ factor of the optimal sensor cover.[3] To construct a Steiner tree connecting the constructed sensor cover, we can use the simple 2-approximation algorithm [11] for the Steiner tree problem. This Steiner tree based approach is conceptually simpler, but yields the same theoretical bound ($O(r \log n)$) on the size of the solution returned. In our simulations (discussed in Section 5), we observed that the size of the connected sensor cover delivered by the Steiner tree based approach is larger than that returned by the centralized approximation algorithm. Moreover, the above approach does not yield an efficient distributed implementation due to the global nature of the greedy sensor cover construction.

3.1 Weighted Version

The centralized approximation algorithm 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.

[3]There are even better approximation algorithms [6, 12, 13] for the geometric versions of set cover that could apply to our sensor cover problem if we assume the sensing regions to be circular disks.

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 Lint 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 the centralized approximation algorithm 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.* □

Alternative Cost Models. In our cost model we have tacitly assumed that the only energy cost incurred in the sensor network is for communication. If sensing and associated signal processing costs are also considered, then the sensors that are involved in both sensing and communication will incur a higher cost than those that are involved solely in communication. The centralized approximation algorithm can be easily generalized for such a cost model. A similar $O(\log n)$ bound on the weight of the solution delivered can also be proved because of the observation that each sensor can be looked upon as a logical combination of two sensors: one having a zero transmission range (i.e., having no communication edges) and the original sensing radius, and the other having a zero sensing radius and the original communication edges.

In addition, the centralized approximation algorithm can also be generalized to the "link cost model," wherein the total cost of a connected sensor cover is defined as the sum of the weighted edge costs in a spanning tree of the connected sensor cover. The generalization is achieved by defining the weight of a candidate path as the sum of the weighted edge costs. More generally, our recent work [14] extends the centralized approximation algorithm for the variable radii sensor network model wherein each sensor node can adjust its sensing and transmission radii.

4 Distributed Self-Organization Algorithms

In this section, we present two self-organizing distributed algorithms viz. distributed approximation and distributed Priority algorithms. The distributed approximation algorithm is a distributed version of the centralized approximation algorithm of previous section. The distributed Priority algorithm is based on node priorities and has a lower communication overhead, but does not provide any guarantee on the size of the connected sensor cover constructed.

4.1 Distributed Approximation Algorithm

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

Like the centralized approximation algorithm, the distributed approximation 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.
- $\widehat{P}$, the most recently added candidate path, and $\hat{C}$, the candidate sensor associated with $\widehat{P}$.

We assume that each of the above values keep the sensing region of the contained sensor nodes. Each sensor in the network is aware of its membership in M, or $\widehat{P}$, or in a candidate path in SP. Also, the most recently added candidate sensor $\hat{C}$ stores the values M, SP, and $\widehat{P}$. Each stage of the distributed algorithm consists of the following sequence of transmission phases.

1. **Candidate Path Search:** The most recently added candidate sensor $\hat{C}$ 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 $\hat{C}$ reaches even those candidate sensors whose sensing disks intersect with that of other sensors in $\widehat{P}$, the most recently added candidate path associated with $\hat{C}$. The CPS message carries the most recently added candidate path.
2. **Candidate Path Response:** Any sensor I that receives a CPS message checks to see if it is a new candidate sensor, i.e., if I's sensing region intersects with the sensing region of some sensor in the most recently

added candidate path $\widehat{P}$. If I is a candidate sensor, it unicasts a *Candidate Path Response (CPR)* message to the originating sensor $\hat{C}$ 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 $\hat{C}$, 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 $\hat{C}$, 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 $\hat{C}$ during this stage, the sensor $\hat{C}$ selects the most beneficial candidate path $\widehat{P}_{new}$ among all the candidate paths in SP. Let, $\hat{C}_{new}$ be the candidate sensor associated with the picked candidate path $\widehat{P}_{new}$, and let $\mathcal{I}_{new}$ be the set of sensors in the candidate path P_{new}. The sensor $\hat{C}$ unicasts a *NewC message* "reliably"[4] to $\hat{C}_{new}$ with the following updated information: $M = M \cup \mathcal{I}_{new}$; $\widehat{P} = \widehat{P}_{new}$; $SP = SP - \widehat{P}_{new}$. Note that SP has also been augmented with all the candidate paths received in the CPR messages.
4. **Repeat:** The sensor $\hat{C}_{new}$ receives the NewC message sent to it by $\hat{C}$. After receiving the message, $\hat{C}_{new}$ updates the value $\hat{C}$ to itself (i.e., $\hat{C} = \hat{C}_{new}$). 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 approximation 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.

4.2 Optimizations to Reduce Number of Messages

The following optimization techniques have been used by the distributed approximation 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 reduce the number of broadcast CPS messages, we stipulate that if a sensor is in $(M - I_{new})$, i.e., it has already been selected in previous stages, 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.

[4]Only NewC message needs to be delivered reliably in the algorithm. Loss of some CPS or CPR messages will only affect the performance (solution size), not the correctness, of the algorithm.

In addition, to minimize computation load on the sensor nodes, the distributed algorithm computes the *area* of the uncovered query region covered by the candidate path instead of the number of uncovered valid subelements, to compute the benefit of a candidate path.

We observe through extensive experiments (as shown later in Figure 4(a)) 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 approximation 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 $\hat{C}$ in one message.

4.3 Distributed Priority Algorithm

While the distributed approximation algorithm provides a guarantee on the connected sensor cover size, it needs to carry around a central state (the intermediate solution M) via messages. This could potentially make message sizes large. In this section, we present an alternate distributed approach that uses small and constant size messages. We refer to the alternate approach as the distributed Priority algorithm, as it uses a notion of node priorities. In the distributed Priority algorithm, each sensor uses only local neighborhood information keeping the number and size of messages small. However, there is no guarantee on the size of the connected sensor cover delivered by the distributed Priority algorithm, since the selection of connected sensor cover is based solely on each node's local neighborhood information.

Basic Idea. The distributed Priority algorithm is based on the following idea. A node s is not needed for connectivity if all pairs of neighbors of I have an alternate communication path not involving I. And, a node is not needed for coverage if its sensing region is fully covered by other sensors' sensing regions. Thus, if a node I satisfies both the above conditions, its deletion would preserve connectivity and coverage of the sensor network. Hence, such a node I is marked `deleted`. Node priorities are used to prevent cyclicity of conditions. In particular, only lower priority nodes are used for satisfying conditions of a given node. To determine alternate paths between pairs of neighbors without incurring unreasonable communication cost, we limit ourselves to *k-hop neighborhood information*, i.e., information about communication neighbors of all nodes that are within a communication distance of $k-1$. We chose $k = 3$ for our simulations.

Algorithm Description. The distributed Priority algorithm works as follows. First, each sensor node assigns a random number as a priority to itself.[5] The sensor nodes whose sensing region do not overlap with the query region are *inactivated*, and do not participate in the rest of the algorithm. Now, each active node gathers $l = max(k, r)$-hop information (including node priorities), where k is the constant as described above and r is the link radius of the sensor network. Let $P(I)$ be the priority of a node I. Each active node I periodically tests for the following a set of conditions and marks itself `deleted` if they are satisfied.

C1: In the communication subgraph induced over the set of active nodes, every pair of non-`deleted` active neighbors of I are connected by communication path that lies entirely within the k-hop neighborhood of I and uses *intermediate* nodes having a priority less than $P(I)$. This condition ensures that deletion of I will preserve the connectivity of the communication subgraph induced by the set of active nodes not marked `deleted`.[6]

C2: There is a set of sensor nodes $\mathcal{H}$ such that such that every node in the set $\mathcal{H}$ has a priority less than $P(I)$, and intersection of the query region and the sensing region of I is fully covered by the union of the sensing regions of nodes in $\mathcal{H}$. Note that $\mathcal{H}$ may contain a node that is marked `deleted`.

The `deleted` marking of a node is permanent and in the end, some of the nodes may be left unmarked. We show in Theorem 3 that if the initial set of active nodes form a connected sensor cover, then the set of active nodes that have not been marked `deleted` at any stage forms a connected sensor cover.

Message Communications. Initially, each active node needs to gather $l = \max(k, r)$-hop neighborhood information, where k is the constant in condition C1 and r is the link radius of the sensor network. If n is the total number of active sensor nodes, then l-hop neighborhood information can be gathered using $(l-1)n$ messages using $(l-1)$ phases as follows. In the first phase, each node transmits its neighborhood information to its neighbors. In each of the remaining $(l-2)$ phases, each node collects information transmitted by all its immediate neighbors, and transmits the collected information to its immediate neighbors. At the end of $(l-1)$ phases and $(l-1)n$ total messages, each node would have l-hop neighborhood information. Note that the above process can be implemented in an asynchronous communication paradigm.

After the initial accumulation of l-hop neighborhood information, whenever a node is marked `deleted`, it informs its *communication* neighbors of its `deleted` status, so that they could retest their C1 condition. This can be done using one message transmission for each node that is marked `deleted`. Note that an unsatisfied C1 condition of a node can become `true` only by `deleted` marking of one of its communication neighbors.

Theorem 3 *If the set of active sensor nodes, i.e., the set of sensor nodes whose sensing regions intersects the query region, form a connected sensor cover, then the set of non-*`deleted` *active sensor nodes forms a connected sensor cover at any stage of the distributed Priority algorithm.*

Proof: (sketch) Consider an active node s that is marked `deleted`. Let $F(s)$ be the set of lower-priority active nodes that satisfy the condition C2 for node s. If there is a node $r \in F(s)$ that is marked `deleted`, we update $F(s)$ as $F(s) = F(s) - \{r\} \cup F(r)$. Note that $s \notin F(r)$, and updated $F(s)$ consists of nodes which have a priority less than that of s and whose sensing regions fully cover the sensing region of s. Repeating the above update process for every `deleted` node in $F(s)$, we get a set $F(s)$ that consists of only non-`deleted` lower-priority nodes. Thus, removal of the node s leaves the query region covered by non-`deleted` nodes. Similarly, it can be shown (as in [15]) that once the C1 condition is satisfied for a node I, the deletion of I will always preserve connectivity in the communication subgraph induced by the non-`deleted` active nodes, even if other nodes get marked as `deleted`. It is easy to see that the above proof of correctness does not rely on reliable message transmissions, as long as each active sensor node has accurate information about its immediate (1-hop) neighborhood. ∎

Multiple Concurrent Queries. The above described distributed Priority algorithm can be easily generalized to handle multiple concurrent queries having overlapping query regions. In particular, we need to change the condition C1 by checking for every pair of non-`deleted` active as well as inactive neighbors of I. Then, to handle a new query Q, any previously inactivated sensor node that belongs to the query region of Q is reactivated, and the distributed Priority algorithm executed for the reactivated nodes. The above ensures that the entire set of non-`deleted` active nodes will form a connected sensor cover for the union of the query regions for all the queries in the network. Similarly, the distributed approximation algorithm can also be generalized to handle multiple concurrent queries by inac-

[5]Considering more complicated priority functions based on node degree and/or overlapping area did not result in any performance improvement.

[6]Wu et al. in [15] use a similar condition for computing a connected dominating set.

tivating and reactivating sensor nodes appropriately.

5 Performance Evaluation

We have constructed a simulator to evaluate the performance of the distributed approximation algorithm (see Section 4), and compare its communication overhead and/or solution size with other approaches viz. the naive flooding-based approach (see Section 2.1), centralized approximation algorithm (see Section 3), Steiner-tree based approach (end of Section 3), and the distributed Priority algorithm (Section 4.3).

Simulator. 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. 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.

Simulation Parameters. In the simulation results that follow, we have used a 100×100 area. The query region is the circular region of radius 50 within the rectangle. 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 800 to 6000.[7] 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. The simulator computes D, the number of messages transmitted during the algorithm (a measure of the communication overhead of the algorithm), and m, the size of the connected sensor cover constructed, for a given set of input parameters.

[7]For one experiment, we consider much smaller values of n, and larger values of s and t.

5.1 Performance of Distributed Approximation Algorithm

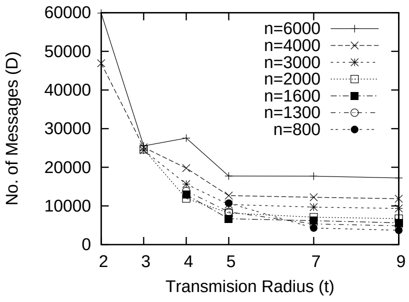
Now, we are ready to describe the performance results of the distributed approximation algorithm and compare them with other approaches. However, before we describe the performance plots for the distributed approximation algorithm, we add a note below on computation of the link radius r. While an exact 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 communication cost. 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). As discussed in the preliminary version [10] of this article, we estimate the link radius r to be $(2s/t + 1)$ for *dense* networks; and $(2s/t + 1) * ((200/t)^2/n)$ for *non dense* networks. Here, we characterize a network as dense if there exist enough $(2s/t)$ number of sensors evenly spread in between (along the link connecting) any two sensors with intersecting sensing disks. In our experiments, we simply consider a network dense if it has at least $(200/t)^2$ sensors.

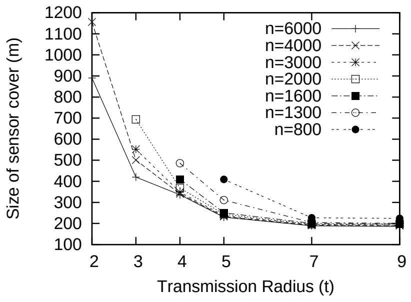
Communication Costs and Solution Size. Let D be the number of messages needed to compute the connected sensor cover and m be the size of the computed connected sensor cover. Query source is a randomly selected sensor. Figure 3 plots D and m for the distributed approximation algorithm for various values of n, the size of the network, and the transmission radius t. Note that m is 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. As a random network for some low values of t and n may not have a connected communication graph with high probability, some data points are missing in the graph plots of this section.

The explanation for the "L" shape of the D vs. t plot in Figure 3 is as follows. From the above discussion on calculation of r, 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 $t > t_\theta$. With b and m (as seen in Figure 3(a)) being relatively constant, the number of messages remain relatively constant for $t > t_\theta$.

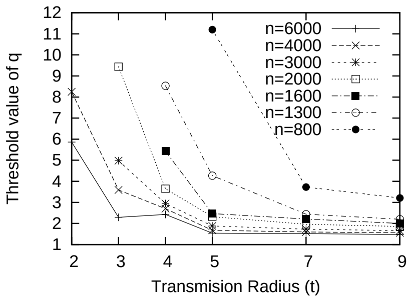
Communication Cost Comparison with the Naive Approach. Let us assume that the given spatial query runs q times. We evaluate the threshold value q_θ, such that for $q > q_\theta$ the overall message cost for the query using our distributed approximation algorithm is lower than the message cost using the naive approach. The number q_θ is



(a) Communication cost (D)

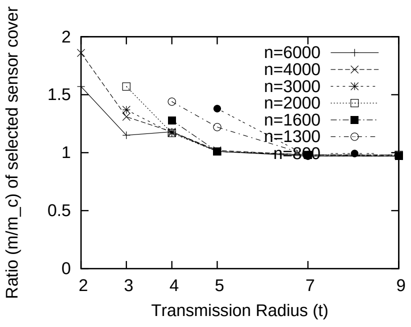


(b) Size of connected sensor cover (m)

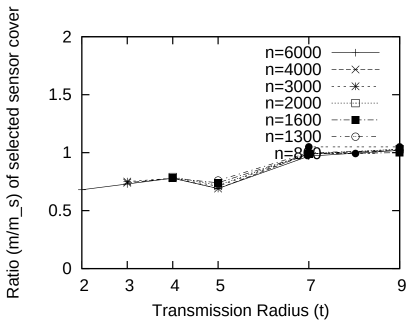


(c) Threshold value q_θ

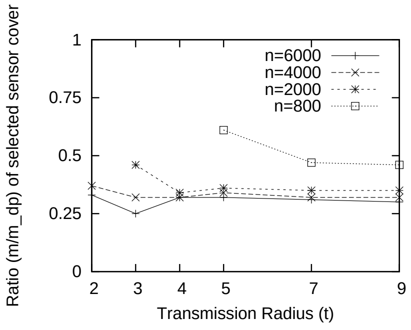
Figure 3: The communication cost (D), size (m) of the connected sensor cover, and threshold value (q_θ of the distributed approximation algorithm, shown for various transmission radii (t) and network size (n).



(a) Ratio m/m_c



(b) Ratio m/m_s



(c) Ratio m/m_{dp}

Figure 4: Ratios (m/m_c), (m/m_s) and (m/m_{dp}), where m, m_c, m_s, and m_{dp} are the connected sensor cover sizes returned by distributed approximation, centralized approximation, Steiner-tree based approach, and distributed priority algorithm respectively.

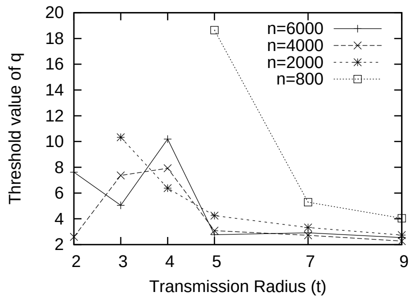
obtained by equating $D + 2qm$ to $2qn$ and then solving for q, which gives

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

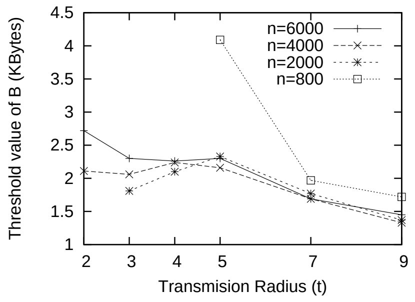
Figure 3(c) plots q_θ vs. n for different values of t. This plot is somewhat similar to the plot of D, because of the strong dependency of q_θ on D. Notice that the value of q_θ is fairly small – almost always less than 8 except when the communication graph is very sparse (low n together with low t). This shows that except for very sparse networks, our distributed approximation algorithm will always save energy relative to the naive flooding approach, whenever the query runs for more than about 8 times – longer runs giving more energy saving benefits. Figure 7(a) plots q_θ vs. n for much smaller values of n. Here, to ensure existence of a connected sensor cover, we choose the sensing radius s to be 12 and vary the transmission radius (t) from 5 to 25. We again observe that the value of q_θ is fairly small.

Solution Size Comparison with the Centralized Approximation Algorithm. Figure 4(a) plots the 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 approximation algorithm and the centralized approximation algorithm respectively. Figure 4(a) depicts excellent performance of the distributed approximation 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 approximation algorithm includes optimizations mentioned in Section 4.2. Thus, the optimizations introduced in the distributed approximation 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.

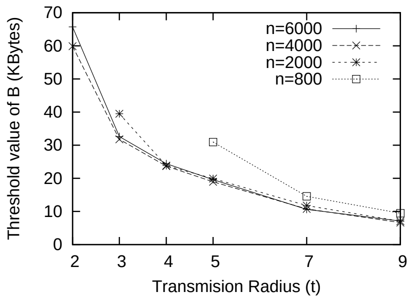
Comparison with Steiner Tree Based Approach. For a given set of network parameters, let m_s be the size of the connected sensor cover returned by the centralized Steiner tree based approach that computes a sensor cover followed by a Steiner tree algorithm to connect the computed sensor cover. Figure 4(b) plots the ratio m/m_s for a wide range of sensor network parameters. Since the ratio is almost always less than 1, we can say that the connected sensor cover returned by the distributed approximation algorithm is almost always smaller than that returned by the Steiner tree based approach. As mentioned before, since the dis-



(a) Threshold value q_θ^{dp}



(b) Threshold value B_θ



(c) Threshold value B_θ^{dp}

Figure 5: Comparison of distributed approximation and distributed Priority algorithms.

tributed implementation of the Steiner tree based approach will also incur much more communication overhead than the distributed approximation algorithm, the Steiner tree based approach does not offer any advantage over our approximation algorithm.

Comparison with Distributed Priority Algorithm. We now compare the performance of the distributed approximation and the distributed Priority algorithms. As before, let D be the number of messages incurred and m be the size of the connected sensor cover returned by the distributed approximation algorithm, and let D_{dp} be the number of messages incurred and m_{dp} be the size of the connected sensor cover returned by the distributed Priority algorithm. As expected, we observe in Figure 4(c) that $m_{dp} > m$, for most values of the network parameters. Figure 5(a) plots the threshold value q_θ^{dp}, where

$$q_\theta^{dp} = \frac{D - D_{dp}}{2(m_{dp} - m)}.$$

If the given spatial query is run q times, then the threshold value q_θ^{dp} is such that for $q > q_\theta^{dp}$ the overall message cost for the query using the distributed approximation algorithm is lower than the message cost using the distributed Priority algorithm. From the Figure 5(a), we observe that q_θ^{dp} is almost always below 10, except for very sparse networks.

Total Number of Bytes Transmitted. In our discussion till now, we have modeled the total communication cost as the number of messages transmitted. In previous experiments, we model the total communication cost as total number of bytes transmitted, and compare the performance of our designed algorithms. Consider a communication tree T of sensor nodes rooted at r. Let us define $f(T)$ as $f(T) = \sum_{i \in T} d(i,r)$, where $d(i,r)$ is the distance of node i from the root r in T. If each sensor node in T generates B bytes of data, then it is easy to see $B \times f(T)$ is the total number of bytes transmitted in gathering data (without any aggregation) from all the sensor nodes to the root r in T. Let T_n, T_{da}, and T_{dp} be the spanning trees constructed by the naive, distributed approximation, and distributed Priority algorithms for their respective connected sensor covers. Also, let D' and D'_{dp} be the total number of bytes transmitted in the execution of the distributed approximation and distributed Priority algorithms respectively. Figure 5(b) plots the threshold value

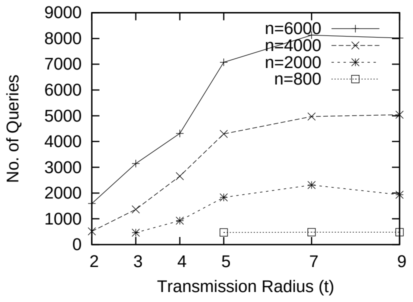
$$B_\theta = \frac{D'}{f(T_{da}) - f(T_n)}.$$

If the given spatial query gathers total B bytes for each sensor node over multiple snapshots, then the threshold value B_θ is such that for $B > B_\theta^{dp}$ the total communication cost using the distributed approximation algorithm is lower than that using the naive approach. We observe that the B_θ value is almost always less than 2.5 KBytes, except for very sparse network graphs. We also plot the threshold value

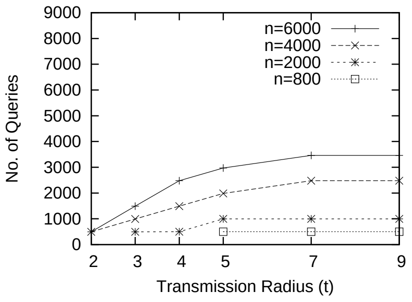
$$B_\theta^{dp} = \frac{D' - D'_{dp}}{f(T_{dp}) - f(T_{da})}$$

in Figure 5(c). The threshold value B_θ^{dp} is such that for $B > B_\theta^{dp}$ the overall communication cost using the distributed approximation algorithm is lower than that using the distributed Priority algorithm. We observe that the value of B_θ^{dp} is almost always below 10 KBytes.

Weighted Version Results. We now present performance results of our distributed approximation and distributed Priority algorithms for the weighted connected sensor cover problem. In particular, through simulations we demonstrate that our application of our designed distributed algorithms can result in significant extension of a sensor network's lifetime. For the performance of the distributed algorithms for the weighted version of the problem, we start with sensor node batteries capable of transmitting 1000 messages each. Whenever needed, we run our weighted-version distributed algorithm to compute a connected sensor cover. Once a connected sensor cover is available, we use the given connected sensor cover to execute the given spatial query as many times as possible till one of the sensors in the sensor cover gets fully depleted of its battery power. Then, we recompute a connected sensor cover using the distributed algorithm. We repeat the process until it is not possible to form a connected sensor cover using even all the sensor nodes still alive. Note that



(a) Network Lifetime using Distributed Approximation Algorithm.



(b) Network Lifetime using Distributed Priority Algorithm.

Figure 6: Lifetime of the sensor network in terms of number of queries, using distributed approximation or distributed Priority algorithm for computing weighted connected sensor covers.

the execution of the distributed algorithm also results in battery power usage of the sensor nodes. Figures 6(a) and (b) plot the total number of times a spatial query covering the entire sensor network region could be run over the sensor network for the given set of network parameters, using the distributed approximation and distributed priority algorithms respectively. Also, if we were to use the naive flooding approach to execute the spatial query, we could run the query only 500 times, as each query execution would require each sensor node to transmit two messages. Thus, Figure 6 shows that our proposed distributed approaches of computing a connected sensor cover results in extending the sensor network life by a factor of upto 15. Also, we see that the distributed approximation algorithm gives better performance than the distributed Priority algorithm.

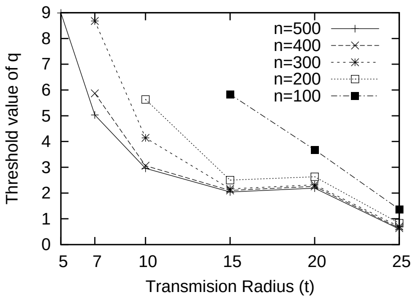
Irregular Sensing Regions and Message Losses. We also simulated the performance of the distributed approximation algorithm for sensor networks with irregular sensing regions. For each sensor node I, we generated an irregular sensing region by randoming choosing six points at a distance of more than 6, but less than 10. The chosen six points are then sorted around the node I, and connected to create a polygonal sensing region. Figure 7(b) plots q_θ vs. n for different values of t for the sensor networks with irregular sensing regions. We observe that the value of q_θ is almost always less than 8. In addition, we also ran simulations to compare the performance of our designed techniques in the presence of unreliable communication links. In Figure 7(c), we plot q_θ against "message success probability" for various values of n and t. We notice that the message losses do not deteriorate the performance of the algorithms. In fact, message losses even lead to improvement in the performance implying the redundancy of the CPS and CPR messages.

6 Related Work

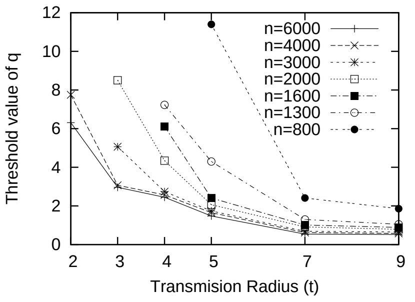
In one of the earliest work related to sensor network coverage [7], the authors introduce a centralized heuristic that selects mutually exclusive sets of sensors, the members of each of those sets together completely cover a geograhical area. As only one set of sensors need to be active at any time, their technique results in energy savings while preserving coverage. However, they only present a centralized algorithm which does not extend easily to a distributed algorithm. Recent work in [16] considers a similar problem of partitioning the sensor network into k "covers" to maximize the sum of the coverage. None of the above works require the sensor covers to form a connected communication graph. We should note that a repeated execution of our algorithm gives a good heuristic for the problem addressed in above works.

In [8], the authors take a probabilistic approach. They consider an unreliable sensor network and derive necessary and sufficient conditions for the coverage of the region and connectivity of the network with high probability in terms of the transmission radius, sensing radius, and failure rate of the sensors. The PEAS protocol [17] considers a probing technique to maintain only a necessary set of sensors in working mode and putting the rest to sleep. Their technique ensures that only one node is active at any time inside any circular disk of probing radius. However, the active nodes form a connected graph with high probability only under certain conditions of network density and relationship between the probing and transmission radius. In [18], the authors investigate linear programming techniques to optimally place a set of sensors on a sensor field (three dimensional grid) for a complete coverage of the field. The k-coverage problem, wherein each point in the query region is required to be covered by at least k sensors, has been recently addressed in [19, 20]. In [21], the authors consider a different definition of coverage. They 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.

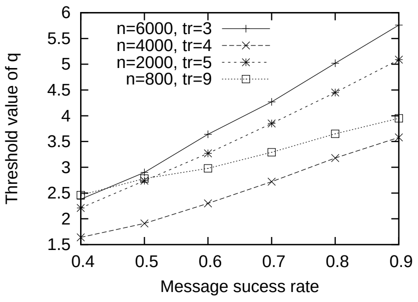
There has been a significant amount of literature outside the sensor network domain that has relationships with the problems we are addressing in this paper. Most relevant is the problem of efficiently broadcast a message in a wireless



(a) Small network sizes. Here, $s = 12$.



(b) Irregular Sensing Regions.



(c) Unreliable Communication Links.

Figure 7: The threshold value q_θ shown for small network sizes, irregular sensing regions, and unreliable communication links

ad hoc network. The idea here is to suppress redundant broadcasts by using only a small number of nodes to propagate broadcast but ensuring that all the nodes in the network receive the broadcast message. The above described problem can be reduced to the minimum dominating connected set (MCDS) problem [22] of selecting a minimum number of nodes such that each node in the network is either selected or a neighbor of a selected node. The work in wireless network research community [23, 24, 25, 26, 27, 15] has primarily focussed on developing energy-efficient distributed algorithms to construct a near-optimal dominating connected set. In a similar vien, distributed algorithms have been developed to identify nodes that are similar in routing perspectives so that other nodes can be turned off to conserve energy. Protcols such as GAF [28], SPAN [29], and ASCENT [30] bear this goal. Also, construction of multiple connected dominating sets for different transmission radii is addressed in [31].

Other related problems based on various other notions of coverage are as follows. The Art Gallery Problem [32, 33] 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 [6, 12, 13] 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 [13], there is a polynomial algorithm that delivers a constant-factor approximation. However, the approximation algorithm for the disk-cover problem does not generalize to other geometric regions (not even rectangles) and a straightforward distributed implementation would incur a very large number of messages, due to the involved computations required.

7 Conclusions

We have designed techniques to exploit the redundancy in the sensor network by selecting a small subset of sensors (called *connected sensor cover*) that is sufficient to process a given query. In particular, we have designed a centralized approximation algorithm which provably delivers a near-optimal solution. In addition, our designed distributed algorithms (distributed approximation and distributed Priority) are also empirically shown to deliver a near-optimal solution. Through extensive simulations, we have shown that our designed techniques result in substantial energy savings in a sensor network. 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.

References

[1] D. Estrin, R. Govindan, and J. Heidemann, eds., *Special Issue on Embedding the Internet,* Communications of the ACM, vol. 43, 2000.

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

[3] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards sensor database systems," in *Proceeding of the International Conference on Mobile Data Management*, 2001.

[4] 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.

[5] G. Pottie and W. Kaiser, "Wireless integrated sensor networks," *Communications of the ACM*, vol. 43, 2000.

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

[7] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *Proceedings of the International Conference on Communications (ICC)*, 2001.

[8] S. Shakkottai, R. Srikant, and N. Shroff, "Unreliable sensor grids: Coverage, connectivity and diameter," in *Proc. of IEEE INFOCOM*, 2003.

[9] 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.

[10] H. Gupta, S. Das, and Q. Gu, "Connected sensor cover: Self-organization of sensor networks for efficient querying," in *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

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

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

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

[14] Z. Zhou, S. Das, and H. Gupta, "Variable radii connected sensor cover in sensor networks," in *Proceedings of the International Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2004.

[15] J. Wu and F. Dai, "Broadcasting in ad hoc networks based on self-pruning," in *Proceedings of IEEE INFOCOM*, 2003.

[16] Z. Abrams, A. Goel, and S. Plotkin, "Set k-cover algorithms for energy efficient monitoring in wireless sensor networks," in *Proceedings of the International Workshop on Information Processing in Sensor Networks (IPSN)*, 2004.

[17] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang, "PEAS: A robust energy conserving protocol for long-lived sensor networks," in *Proceedings of the International Conference on Distributed Computing Systems*, 2003.

[18] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, no. 12, 2002.

[19] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *Proceedings of the International Conference on Wireless Sensor Networks and Applications (WSNA)*, 2003.

[20] Z. Zhou, S. Das, and H. Gupta, "Connected k-coverage problem in sensor networks," in *Proceedings of the International Conference on Computer Communications and Networks (IC3N)*, 2004.

[21] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proc. of IEEE INFOCOM*, 2001.

[22] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, 1998.

[23] 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.

[24] 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*, 2002.

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

[26] K. M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-optimal connected dominating sets in mobile ad hoc networks," in *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2002.

[27] Y. Chen and A. Liestman, "Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks," in *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2002.

[28] Y. Xu, J. S. Heidemann, and D. Estri, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, 2001.

[29] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, 2001.

[30] A. Cerpa and D. Estrin, "Ascent: Adaptive self-configuring sensor networks topologies," in *Proc. of IEEE INFOCOM*, 2002.

[31] B. Deb, S. Bhatnagar, and B. Nath, "Multi-resolution state retrieval in sensor networks," in *Proceedings of Intl. Workshop on Sensor Network Protocols and Applications*, 2003.

[32] 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.

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