Dominating sets-based Self* Minimum Connected Covers of Query Regions in Sensor Networks

Ajoy Datta, Maria Gradinariu, Rajesh Patel

▶ **To cite this version:**

Ajoy Datta, Maria Gradinariu, Rajesh Patel. Dominating sets-based Self* Minimum Connected Covers of Query Regions in Sensor Networks. [Research Report] PI 1803, 2006, pp.22. inria-00069842

HAL Id: inria-00069842
https://inria.hal.science/inria-00069842

Submitted on 19 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1803

DOMINATING SETS-BASED SELF* MINIMUM CONNECTED COVERS OF QUERY REGIONS IN SENSOR NETWORKS

A. K. DATTA M. GRADINARIU R. PATEL

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Dominating sets-based Self* Minimum Connected Covers of Query Regions in Sensor Networks

A. K. Datta* M. Gradinariu R. Patel**

Systèmes communicants
Projet Adept

Publication interne n ° 1803 — may 2006 — 22 pages

Abstract: Sensor networks are mainly used to gather strategic information in various monitored area. Sensors may be deployed in zones where their internal memory or the sensors themselves can be corrupted. Moreover, once deployed sensors cannot be easily replaced, therefore the persistence and the robustness of the network are two main issues that have to be addressed while efficiently deploying large scale sensor networks. Minimum*** Connected Covers of a query region in sensor networks aims at selecting a subset of nodes that entirely covers the monitored area, is strongly connected (i.e. between any two sensors in the selected set there is a communication path) and the final set does not contain a subset with the same properties. Interestingly, the connected query region cover cannot be solved by trivially applying algorithms that compute minimum connected dominating sets designed for adhoc or sensor networks, since these algorithms aim at selecting a connected cover of existing nodes in the network irrespectively of the relationship between their sensing region and the monitored zone. Nevertheless, these algorithms define an efficient connected tilling of the monitored zone that can be further completed to a fully cover. In this paper we propose two novel, fully localized, robust solutions to the minimum connected cover of query regions that can cope with both transient faults (corruptions of the internal memory of sensors) and sensors crash/join. Our algorithms use only 2 bits of memory and follow two different strategies: a greedy strategy (sensors are chosen such that the overlap with sensors already chosen is minimal) and pruning-based approach (sensors are removed from the cover set till the final set of nodes contains only non redundant sensors). We prove the self* (self-configuration, self-stabilization and self-healing) properties of our solutions. Via extended simulations we conclude that our solutions provide better performances in terms of coverage than pre-existing self-stabilizing solutions. Moreover, we observe that the greedy approach performs better than the pruning strategy

Key-words: Connectivity, coverage, minimum connected dominating set, query response system, self-configuring, self-healing, self-stabilizing, self-$*$, sensor networks.

(Résumé : tsvp)

* School of Computer Science, University of Nevada Las Vegas
** School of Computer Science, University of Nevada Las Vegas
*** Selecting a minimal number of connected sensors is an NP hard problem. In our work we address the minimality in terms of inclusion

CNRS

Couvertures minimales et tolerantes aux fautes de zones de surveillance dans les réseaux de capteurs

Résumé : Nous proposons deux infrastructures efficaces pour la couverture d'une zone de surveillance dans les réseaux de capteurs. La construction des infrastructures proposées utilise comme stratégie de construction les ensembles dominants. Les algorithmes proposés sont tolérant aux fautes transitoires ainsi qu'à l'insertion et au départ des noeuds.

Mots clés : réseau de capteurs, ensembles dominats, tolerance aux fautes, auto-stabilisation, couverture connexes

1 Introduction

Recent advances in microprocessor, memory, and wireless communication technology have enabled the production of tiny networked sensors which will revolutionize information gathering and processing in both urban environments and inhospitable terrain. These wireless ad hoc sensor networks consist of a large number of tiny sensing devices with very limited resources that must coordinate amongst themselves to gather, process, and communicate information about their environments. Because these sensors are often densely deployed, in a sensor network there may be some failing sensors or sensors that have merely exhausted their energy supply. However, it may be impossible or infeasible to recharge sensors once they have been deployed, especially if they have been deployed in an inhospitable or physically unreachable terrain. Therefore, since the fundamental constraint on a networked sensor is its energy consumption, only some of the sensors within a particular sensing region, or query region, should be in an active state.

The information to be gathered by a sensor network may concern only a particular sub-area of the monitored area. Therefore, queries should be addressed only to the nodes monitoring this particular area and for energy saving reasons only the queried nodes should reply. Our research is focused on designing a reliable, self-organizing, and self-healing query-response system. A query in sensor networks asks for some data/measurements/events sensed/observed over some period of time at some frequency over a geographical region. Upon receiving a query, the sensors will sense or measure the data and collaborate among themselves to disseminate or fuse the collective data to the sink of the query. Although a query can be initiated in the whole geographical region, typically, a query refers to a subset of the region, called the *query region*. Only sensors inside the query region should be involved in generating the response to the query. Considering the redundancy and our goal of designing an efficient query-response system, all sensors inside the query region should not be actively participating in the protocol to answer the query. To this end, only a subset of sensors, those for which the union of their sensing regions *cover* the query region should be active. However, these sensors should not be selected arbitrarily. That is, in order to make these sensors able to collaborate to detect events, and compute and deliver responses, they must be able to communicate with each other directly or indirectly. So the cover should be strongly connected.

Related Work. The problem of computing minimum connected covers for query regions was first introduced in [11]. Two self-organizing solutions were presented in [11]. Both solutions follow a greedy strategy and none of the solutions is localized. The first solution is centralized - a fixed leader chooses the nodes to be part of the cover. In the second solution, a particular sensor node (not always the same) behaves as the coordinator or the leader. This special node collects all the global information related to the possible new sensors to be added, then decides which ones to choose in the final cover.

The issues of coverage and connectivity, and the relations between them were analyzed in a unified framework in [17]. The CCP protocol [17] can be used to provide different degrees of coverage. It was shown in [17, 23] that if the communication range is at least twice of the sensing range, the complete coverage implies connectivity. When the above condition does not hold, CCP was integrated with SPAN [3] to provide both coverage and connectivity. However, in SPAN, the nodes need to maintain information about two-hop neighborhood. SPAN is a connectivity maintenance protocol where a node volunteers to be a coordinator when it finds that two of its neighbors cannot communicate with each other directly or indirectly. After a node decides to be a coordinator, it announces that with a random delay to reduce the number of redundant coordinators. A similar approach was discussed in ASCENT [2]. ASCENT nodes use the number of active neighbors and message losses to decide if they should be active or passive. However, this protocol does not guarantee complete coverage of the query region.

Probabilistic studies related to coverage and connectivity in unreliable sensor networks were done in [15]. A sensor grid network of unit area was considered. This work includes a necessary and sufficient condition for the network to remain covered and connected in terms of the probability of a node to be active (i.e., not failed) and transmission radius of the nodes. Some optimal conditions for coverage were established

in [23]. An algorithm for coverage was proposed based on those optimal conditions. However, that result is valid only when complete coverage implies connectivity (as discussed above). A coverage protocol using a random delay to announce decision to turn off was proposed in [16]. The issue of connectivity was not addressed in [16]. The GAF protocol [20] uses GPS to reduce the redundant nodes to maintain routing paths in ad-hoc networks. A randomized probing-based density control algorithm was used to maintain coverage under node failures in PEAS protocol [21]. The probing range can be changed to provide different degrees of coverage. All these solutions although efficient in fault free environments are not fault-tolerant neither self-stabilizing. If the coverage set is corrupted than one has to rerun the algorithm in order to repair the overlay. To this end one needs to be informed that the cover was corrupted and to inform each member of the network that it has to re-run its local algorithm.

Very recent solutions to the connected cover problem address the fault-tolerance issues by reinforcing the coverage and connectivity degree. Hence in [24] the authors address the problem of k-coverage. That is, they compute a coverage such that any sensor is covered by k other sensors. This work is further extended in [25] to the k-coverage and k-connectivity problem. The proposed solution involves the computation of a Voronoi diagram for independent sensor nodes. The implementation of local Voronoi diagrams is not addressed, neither the transient faults.

To the best of our knowledge, we proposed in [7, 5] the first, totally decentralized self-stabilizing and fault tolerant algorithms for the minimum connected covers of query regions in sensor networks. The first solution in [7], based on a greedy strategy, needs only one bit per node, however it requests additional knowledge — the distance to the center of the query region. That is, the region is covered in successive waves from outside to inside. The coverage stops once the wave reaches the center of the monitored area. The second solution proposed in the same paper uses the pruning strategy — the elimination of the redundant nodes from the final cover. The removed nodes were chosen such that they do not disconnect their respective neighborhoods and their sensing region is completely covered by their chosen neighbors. Furthermore, in [5] we proposed another pruning-based algorithm that outperformed the solutions proposed in [7]. Nodes were considered redundant if their sensing regions were covered by already chosen nodes and their chosen neighbors were connected through a connection path.

Contributions. In this paper we propose two novel, fully localized, robust solutions to the minimum connected cover of query regions that can cope with both transient faults (corruptions of the internal memory of sensors) and sensors crash/join. That is, our algorithms are self-stabilizing, fault-tolerant and outperforms the solutions in [7, 5]. Our first algorithm follows a greedy strategy that extends the weakly connected maximal independent set algorithm proposed in [10] to the computation of a minimum query region connected cover. That is, first a set of nodes is chosen such that their sensing regions does not overlap and they cover a maximum of the query region. Furthermore, these nodes are connected via a minimum set of bridges. An alterntive for the greedy strategy is the pruning strategy used in our second algorithm. This strategy is similar to pruning used in the computation of connected dominating sets [19, 18, 4, 1, 13, 14, 12]. By definition, a dominating set is a set of vertexes such that every vertex in the graph is either in the dominating set, or adjacent to a vertex in the dominating set. A connected dominating set is a dominating set which is also a connected subgraph. The main difference between the connected dominating sets and the query region connected covers steams in the selection of the dominators. In the former case a node is a dominator of another node if the second node is in the transmission rage of the first node. In the latter case a dominated is a node that communicate with at least one dominator in its neighborhood and which sensing region is totally covered by dominators. Obviously, these problems are not equivalent. However, an efficient connected dominating set is also a good first coverage pattern for the query region that can be extended to a fully coverage. Via extensive simulations we compared the performances of greedy and pruning self-stabilizing algorithms for the computation of connected coverages of query regions. In our experiments greedy-based algorithms produced better coverage sets; however, their stabilization time was slightly weaker.

Outline of the paper. In Section 2, we define the model and specify the connected sensor cover problem. In Section 3, we present a self-stabilizing greedy-based solution to the problem and and the key points of the correctness of our solution. Simulation results and their discussion are included in Section 6. The experiments were conducted with respect to the following metrics: number of nodes in the final cover set, number of query region sensors per cover sensor, and stabilization time. Finally, in Section 7, we present some concluding remarks and propose ideas to extend this research.

2 Preliminaries and Model

Sensor Network. In this research, we consider *sensor networks* [11, 17] consisting of a large number of sensors (also referred to, in this paper, as *sensor nodes* or, simply as *nodes*) which are randomly distributed in a geographical region. We model the sensor network as a *directed* communication graph $G(V, E)$, where each node in V represents a sensor, and each edge $(i, j) \in E$, called *communication edge*, indicates that j is a *neighbor* of i.

For a sensor i, there is a region, called a *sensing region*, which signifies the area in which sensor i can sense a given physical phenomenon at a desired confidence level. The sensing regions are of any convex shape. For the sake of simplicity, especially, for showing examples, the sensing regions are assumed to be circular. The *sensing range* of a sensor i indicates the maximum distance between sensor i and any point p in the sensing region of sensor i. A point p is *covered* (or monitored) by a sensor node i if the Euclidean distance between p and i is less than the sensing range of sensor i.

The *communication region* of sensor i (also called the *transmission region*) defines the area in which sensor i can communicate directly (i.e., in single hop) with other sensor nodes. The maximum distance between node i and any other node j, where j is in the communication region of i, is called the *communication range* of sensor i. Node i can communicate with node j (i.e., i can send a message to j) if the Euclidean distance between them is less than the communication range of i. Then i is called a neighbor of j, and this relation is represented by a directed edge (i, j). The set of neighbors of i is represented by N_i. Two nodes i and j can communicate directly with each other only if $i \in N_j \wedge j \in N_i$, i.e., they are neighbors of each other. If i and j are neighbors of each other, then there are two edges between them: (i, j) and (j, i).

A directed path (sequence) of sensors $i = i_1, i_2, \ldots, i_m = j$, where i_x is a neighbor of i_{x+1} for $1 \leq x \leq m-1$, is called a *communication path* from i to j. The length of the shortest (communication) path (which is the number of sensors on the shortest path) from i to j is called the *communication distance* from sensor i to sensor j.

Program. In this paper, we consider the local shared memory model of communication as used by Dijkstra [8]. The program of every processor consists of a set of *shared variables* (henceforth, referred to as variables) and a finite set of actions. Every processor (or sensor) can only write to its own variables, but can read its own variables and the variables owned by the neighboring nodes.

Each action is of the following form: $< label >:: < guard > \longrightarrow < statement >$. The guard of an action in the program of p is a boolean expression involving the variables of p and its neighbors. The statement of an action of p updates one or more variables of p. An action can be executed only if its guard evaluates to true. We assume a model of *composite atomicity*; i.e., actions are atomically executed, or the evaluation of a guard and the execution of its corresponding statement, if executed, are done in one atomic step.

The *state* of a node is defined by the values of its variables. The *state* of a system is the product of the states of all nodes. We will refer to the state of a node and system as a (*local*) *state* and (*global*) *configuration*, respectively.

Let a distributed protocol $\mathcal{P}$ be a collection of binary transition relations denoted by $\mapsto$, on $\mathcal{C}$, the set of all possible configurations of the system. A *computation* of a protocol $\mathcal{P}$ is a *maximal* sequence of configurations $e = \gamma_0, \gamma_1, ..., \gamma_i, \gamma_{i+1}, ...$, such that for $i \geq 0, \gamma_i \mapsto \gamma_{i+1}$ (a single *computation step*) if γ_{i+1} exists, or γ_i is a terminal configuration. The *Maximality* means that the sequence is either infinite, or it is finite and no action of $\mathcal{P}$ is enabled in the final configuration. All computations considered in this paper are assumed to be maximal. The set of all possible computations of $\mathcal{P}$ in system S is denoted as $\mathcal{E}$. A node p is said to be *enabled* in γ ($\gamma \in \mathcal{C}$) if there exists an action A such that the guard of A is true in γ. Similarly, an action A is said to be enabled (in γ) at p if the guard of A is true at p (in γ).

We assume a *weakly fair and distributed daemon. Weak fairness* means that if a node p is continuously enabled, then p will be eventually chosen by the daemon to execute an action. A *distributed* daemon implies that during a computation step, if one or more nodes are enabled, then the daemon chooses at least one (possibly more) of these enabled nodes to execute an action.

Fault Model. This research deals with the following types of faults: (i) The state or configuration of the system may be arbitrarily corrupted. However, the program (or code) of the algorithm cannot be corrupted. (ii) Nodes may crash. That is, faults can fail-stop nodes. (iii) Nodes may recover or join the network.
The topology of the network may change due to these faults. Faults may occur in any finite number, in any order, at any frequency, and at any time.

Self-stabilization [9]. Let $\mathcal{L}_{\mathcal{A}}$ be a non-empty *legitimacy predicate* of an algorithm $\mathcal{A}$ with respect to a specification predicate *Spec* such that every configuration satisfying $\mathcal{L}_{\mathcal{A}}$ satisfies *Spec*. Algorithm $\mathcal{A}$ is *self-stabilizing* with respect to *Spec* iff the following two conditions hold:
(i) Every computation of $\mathcal{A}$ starting from a configuration satisfying $\mathcal{L}_{\mathcal{A}}$ preserves $\mathcal{L}_{\mathcal{A}}$ (*closure*).
(ii) Every computation of $\mathcal{A}$ starting from an arbitrary configuration contains a configuration that satisfies $\mathcal{L}_{\mathcal{A}}$ (*convergence*).

Problem Specification. In this section we formally define the problem Connected Cover of a Query Region in sensor networks.

Definition 2.1 (Connected Sensor Cover). *Consider a sensor network G consisting of n sensors $I_1, I_2, \ldots, I_n$ Let S_i be the sensing region associated with sensor I_i. Given a query Q over a region R_Q in the sensor network, a set of sensors $\mathcal{SC}_Q = I_{i_1}, I_{i_2}, \ldots, I_{i_m}$ is called a* connected sensor cover *for Q if the following two conditions are satisfied:* **(a) Coverage:** *$R_Q \subseteq (S_{i_1} \cup S_{i_2} \cup \ldots S_{i_m})$.* **(b) Connectivity:** *The subgraph induced by $\mathcal{SC}_Q$ is strongly connected in the sense that any two sensors in this set can communicate with each other directly or indirectly. A cover is considered minimal if it does not include another connected cover.*

Additionally, we require the algorithm (solving the above problem) to be self-organizing, self-stabilizing and self-healing [9, 22]. That is, regardless of the initial state (wrong initialization of the local variables, memory or program counter corruptions) nodes self-configure/self-organize using only local information in order to make the system self-stabilize to a *legitimate state*. The legitimate state is defined with respect to a minimal connected cover formed out of the nodes that can communicate with each other either directly or indirectly. The nodes in this set are the only nodes that remain active. Moreover, under various perturbations, such as node joins, failures (due to crash or energy loss), state corruptions, or weakening of power, the minimal connected cover should be able to self-heal without any external intervention and the impact should be confined within a tightly bounded region around the perturbed area.

The following assumptions are made for our solutions: *(i)*The communication radius equals the sensing radius for the sensors. *(ii)*The sensing radii, and hence the communication radii, of all sensors are equal.

*(iii)*There always exist a sufficient number of sensors in the network with sufficient density to cover the query region if all of them are deployed. *(iv)*There exist a lot of redundant sensors which are either boundary or interior sensors with respect to the query region.

3 Greedy-based Self-$*$ Query Region Connected Cover

Our first solution, Algorithm 3.1, follows a greedy policy. Initially a first tilling of the query region is computed. This first tilling forms a pattern of coverage of the query region that is composed of the union of the sensing radii of sensors chosen such that their sensing regions form a disjoint set. That is, in this first coverage no two sensing disks intersect. The objective of this first phase is to select a maximum set of nodes that verify the above property. Obviously, this first tilling does not cover completely the query region. Therefore a second phase is executed and during this phase the uncovered regions between two sensors already in the cover are covered with new sensors.

The algorithm uses three shared variables, S_i, UID_i, and $Status_i$. S_i represents the sensing region of Sensor i. UID_i is the unique identifier (UID) of Sensor i, which is a positive integer. Finally, $Status_i$ represents the status of a sensor. The status of a sensor may be *unchosen*, *undecided*, or *chosen*. A node with the status chosen is part of the connected cover. In the sequel we assume that only $Status$ and S can be corrupted.

In the following we detail the execution of our algorithm. For the sake of simplicity we decided to present first the execution of the algorithm started in a normal state (all sensors are in the state unchosen) and no faults are detected. Then, we detail the behavior of the algorithm face to transient faults.

3.1 Normal Execution of Algorithm 3.1

The steps of the algorithm are as follows:

1. The algorithm forms an initial pattern of coverage of the query region that is composed of the union of the sensing radii of sensors whose status is *chosen*. These sensing regions also form a disjoint set, in the sense that no two sensing disks within this set intersect. To this end, it changes the status of all *unchosen* sensors whose sensing regions intersect with the query region, and whose sensing regions do not intersect with the sensing region of a *chosen* sensor, to *chosen*. These nodes verify the predicate $MISNode()$. Thus, by applying the rule $\mathcal{A}_3$, an initial pattern of non-overlapping sensing disks, whose sensors are marked as *chosen*, is formed to cover the query region.

2. The uncovered regions between the sensing radii of all *chosen* sensors is then covered as follows:

   (a) If the status of Sensor i is *unchosen*, the sensing disk of Sensor i intersects with some portion of the query region, and Sensor i is not the neighbor of a *chosen* sensor unless it is the neighbor of a *chosen* sensor having the least UID, or if part of the transmission disk of Sensor i is not covered by a *chosen* sensor, then the *unchosen* sensor's status is changed to *undecided*. The reasoning used is that all sensors that lie within the uncovered "gap" regions between the sensing radii of all *chosen* sensors that were marked by $MISNode(i)$, will have part of their sensing disks not covered by the sensing disks of all sensors chosen by $MISNode(i)$. In addition to this, all sensors that have the least UID, within a particular *chosen* node's neighborhood, are needed to ensure connectivity, and also have their status changed to *undecided* (rule $\mathcal{A}_2$).

   (b) To ensure that only the most suitable of these sensors, located within **each** uncovered region, are marked as *undecided*, if any sensor's status is *undecided*, and it has another *undecided* sensor within its transmission (and hence its sensing) disk, whose UID is greater than it's own, or if this

sensor is the neighbor of an *undecided* sensor and does not have the least UID of all neighbors of this *undecided* sensor, then it's status is changed to *unchosen*.

(c) All sensors with an *undecided* status, that do not have another *undecided* sensor with a UID greater than their own, within their transmission (and hence sensing) disks, and that are not the neighbors of an *undecided* sensor and that also have the least UID of all neighbors of this *undecided* sensor, have their status changed to *chosen*.

3. $Redundant_2(i)$ is used to eliminate any redundant *chosen* sensor that has a smaller UID than another *chosen* Sensor j that is within its transmission disk, but that does not have the smallest UID out of all the neighbors of Sensor j.

4. Finally, action $\mathcal{A}_1$ ensures that any redundant sensor or any sensor whose sensing disk does not intersect with the query region, has its status changed to *unchosen*.

5. All *chosen* sensors are in the final query region connected cover.

3.2 Faults and Recovery of Algorithm 3.1

In this section, we focus on the fault handling features of the proposed Algorithm 3.1. The variables that can be corrupted by transient faults are: $Status_i$ and S_i. So, we need to show that our solution can cope with all possible corruptions associated with these two variables. **(1) Wrong initialization of the $Status_i$ variable.** As discussed in the previous subsection, all sensors, if properly initialized, start as *unchosen*. *(a) Sensor i is initialized to* ***undecided*** . If i is not a redundant node, then i remains *undecided*, and subsequently changes to *chosen*. (see Actions $\mathcal{A}_2$ and $\mathcal{A}_3$). That is, no correction is necessary. If i is redundant, then it will satisfy the predicate $Redundant_1(i)$ and will change to *unchosen*. *(b) Sensor i is initialized to* ***chosen*** . If the sensing disk of Sensor i does not intersect with the query region, then, by executing $\mathcal{A}_1$, Sensor i will change to *unchosen*. So, no correction is necessary. If Sensor i is redundant, then it will satisfy the predicate $Redundant_2(i)$, and will change to *unchosen*. If it is non redundant then Sensor i is necessary, either to ensure coverage or connectivity, and should not be unmarked. **(2) Weakening or Failure of sensors, both in terms of communication and sensing ability.** The weakening or failure of sensors will affect their sensing and communication range. In other words, the constant set R_S or R_C will change. Change of R_S or R_C may change the values of $Redundant(i)$, $MISNode(i)$, $BridgeNode(i)$, or $FillNode(i)$. All these changes will be reflected in the change of values of the guards of the corresponding actions. So, eventually, the status of the affected nodes will change due to the execution of these actions. However, these changes will not affect the execution of these actions by the neighbors of the affected nodes. Therefore, any changes in the $Status_i$ variable of the affected nodes will be handled as mentioned earlier.

3.3 Correctness of Algorithm 3.1

Definition 3.1. *The system is considered to be in a legitimate state (i.e., satisfies the legitimacy predicate $\mathcal{L}_{MCSC}$) if the following conditions are true with respect to a query region:*

i) All non-redundant sensors are marked chosen.

ii) All redundant sensors are marked unchosen.

A redundant sensor verifies the predicate $Redundant(i)$ (Algorithm 3.1).

Lemma 3.1 (Coverage). *In any legitimate configuration, the set of sensors chosen by Algorithm 3.1 completely covers the query region R_Q.*

Algorithm 3.1 Query Region Connected Cover Algorithm for Sensor i.

Constants:
R_Q:: Query region;
R_C:: Radius of communication of a sensor in the network;
N_i:: Set of sensors within the communication range of Sensor i;

Shared Variables:
S_i:: Sensing region of Sensor i;
UID_i:: Unique user identification number of Sensor i;
$Status_i \in \{unchosen, undecided, chosen\}$:: Status of Sensor i;

Predicates:

$QryRgnIntrsctn(i) \equiv S_i \cap R_Q \neq \emptyset$;
$\equiv$ sensing disk of Sensor i intersects with some portion of query region;

$NoIntrsctn(i) \equiv Status_i = unchosen \wedge (\forall j \in N_i : Status_j \neq chosen) \wedge (\forall j \in N_i^2 : Status_j = chosen \wedge S_i \cap S_j = \emptyset)$
$\equiv$ status of Sensor i is $unchosen$, there are no $chosen$ sensors within the transmission disk of Sensor i, and sensing disks of Sensor i and Sensor j do not intersect;

$NgbrOfChsn(i) \equiv (\exists j : i \in N_j \wedge Status_j = chosen)$;
$\equiv$ Sensor i is a neighbor of a chosen sensor;

$CoverSensingByChsn(i) \equiv (\exists A \subset N_i, \forall j \in A, Status_j = chosen \wedge S_i \subset \bigcup_{j \in A} S_j)$;
$\equiv$ Sensing region of sensor i is covered by a subset of its chosen neighbors;

$NeighborsConnectivity(i) \equiv (\forall j, t \in N_i, Status_j = Status_t = chosen, \exists k \neq i, Status_k = chosen, j, t \in N_k$;
$\equiv$ Sensing region of sensor i is covered by a subset of its chosen neighbors;

$LeastUIDNgbr(i, x) \equiv i \in N_x \wedge (\forall j \in N_x : j \neq i, UID_i < UID_j)$;
$\equiv$ Sensor i is a neighbor of Sensor x, and is also the neighbor of Sensor x having the least UID;

$LessNotLeastNgbrOfChsn(i) \equiv (\exists j : i \in N_j \wedge Status_j = chosen \wedge UID_i < UID_j \wedge \neg LeastUIDNgbr(i, j))$;
$\equiv$ Sensor i is a neighbor of a $chosen$ sensor whose UID is greater than its own, but Sensor i is not the neighbor of this sensor that has the smallest UID;

$MISNode(i) \equiv QryRgnIntrsctn(i) \wedge NoIntrsctn(i)$;
$\equiv$ status of Sensor i is $unchosen$, and the sensing disk of Sensor i intersects with some portion of the query region, but does not intersect with the sensing disk of a $chosen$ sensor;

$NotOrLeastUIDNgbrOfChsn(i) \equiv \forall j : i \in N_j : (Status_j \neq chosen \vee LeastUIDNgbr(i, j))$;
$\equiv$ Sensor i is not the neighbor of a $chosen$ sensor unless it is the neighbor of a $chosen$ sensor having the least UID;

$BridgeNode(i) \equiv Status_i = unchosen \wedge QryRgnIntrsctn(i) \wedge (NotOrLeastUIDNgbrOfChsn(i) \vee (\exists j \in N_i : \neg NgbrOfChsn(j) \vee \neg CoverSensingByChsn(i)))$;
$\equiv$ status of Sensor i is $unchosen$, sensing disk of Sensor i intersects with some portion of the query region, Sensor i is not the neighbor of a $chosen$ sensor unless it is the neighbor of a $chosen$ sensor having the least UID, or part of the transmission disk of Sensor i is not covered by a $chosen$ sensor;

$FillNode(i) \equiv Status_i = undecided \wedge ((/\exists j \in N_i, (Status_j = undecided \wedge UID_j > UID_i)) \vee (\exists j \in N_i, Status_j = undecided \wedge LeastUIDNgbr(i, j)))$;
$\equiv$ status of Sensor i is $undecided$, and there are no $undecided$ sensors within the transmission disk of Sensor i whose UID is greater than that of Sensor i, or Sensor i is the neighbor of an $undecided$ sensor having the least UID;

$Redundant_1(i) \equiv Status_i = undecided \wedge (\exists j \in N_i : Status_j = undecided \wedge UID_i < UID_j \wedge \neg LeastUIDNgbr(i, j))$;
$\equiv$ status of Sensor i is $undecided$, there is an $undecided$ sensor within the transmission disk of Sensor i whose UID is greater than that of Sensor i, and Sensor i is not the neighbor of this $undecided$ sensor having the least UID;

$Redundant_2(i) \equiv Status_i = chosen \wedge LessNotLeastNgbrOfChsn(i) \wedge CoverSensingByChsn(i) \wedge NeighborsConnectivity(i)$;
$\equiv$ status of Sensor i is $chosen$, Sensor i has a smaller UID than another $chosen$ Sensor j that is within its transmission disk, but Sensor i does not have the smallest UID out of all the neighbors of Sensor j and each pair of chosen neighbors of i are connected though a different node than i.

$Redundant(i) \equiv Redundant_1(i) \vee Redundant_2(i)$;

Actions:
$\mathcal{A}_1 :: \neg QryRgnIntrsctn(i) \vee Redundant(i) \longrightarrow Status_i = unchosen$;

$\mathcal{A}_2 :: BridgeNode(i) \longrightarrow Status_i = undecided$;

$\mathcal{A}_3 :: MISNode(i) \vee FillNode(i) \longrightarrow Status_i = chosen$;

Proof. (sketch) We prove this lemma by contradiction. Suppose the sensing disks of the sensors in the final set chosen by Algorithm 3.1 do not completely cover the query region. It's trivial to prove that the nodes chosen by the $MiSNode()$ predicate form a maximal independent cover. Assume that there exists a region between the sensing disks of the sensors chosen by $MISNode(i)$ that is not covered by the sensing disk(s) of one or more sensors that should've been chosen by the $BridgeNode(i)$ and $FillNode(i)$ predicate.

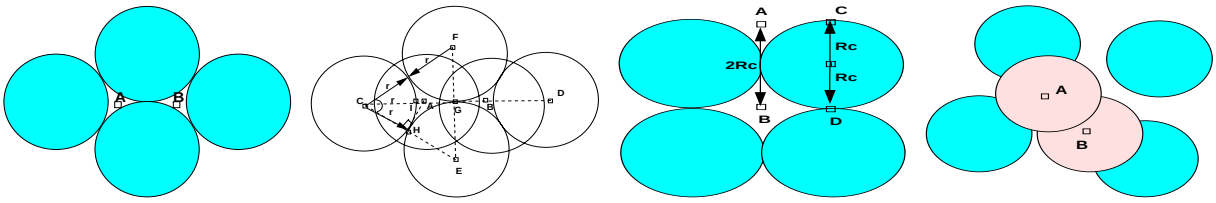
Case 1: The sensors in the maximal independent set chosen by the $MISNode(i)$ predicate formed an initial pattern of coverage in which there are two uncovered regions between the sensing disks of four of these sensors. Figure 3.1(a) is an illustration of this case. Since the graph is densely populated, we can find two sensors in both of these uncovered regions, let's name them Sensor A and Sensor B as shown in Figure 3.1(b), such that $\overline{CA} = \overline{BD}$. By construction, sensors A and B cover the uncovered region. In the following we show that these sensors will have to change their state in "chosen". Assume without restraining the generality that Sensor A has a lesser UID than Sensor B, but Sensor A does not have the least UID of all neighbors of Sensor B, and both Sensor A and Sensor B have no other *undecided* sensors within their transmission disks. Since both nodes are not the neighbors of *chosen* sensors, both nodes must have been marked *undecided* and either one or both nodes were marked *unchosen* by $Redundant_1(i)$ or were not marked *chosen* by $FillNode(i)$. Consequently, sensor A and sensor B are located within each other's communication disk (otherwise $Redundant_1()$ wouldn't have been changed the status of these sensors). So, the distance between Sensor A and Sensor B is less than or equal to the radius of communication. Let $R_C = 1$, in Figure 3.1(b), $\overline{AB} < 1$.

Since $\overline{CE} = \overline{CF} = \overline{EF} = 2$, then $\triangle CEF$ is an equilateral triangle and if we bisect $\angle FCE$, $\triangle CGF$ is a 30-60-90 triangle and $cos\ 30^\circ = \frac{\overline{CG}}{2r} \Rightarrow \frac{\sqrt{3}}{2} = \frac{\overline{CG}}{2r} \Rightarrow 2\overline{CG} = 2\sqrt{3} \Rightarrow \overline{CG} = \sqrt{3} \Rightarrow \overline{CD} = 2\sqrt{3}$.

Similarly, $\triangle CAH$ is a 30-60-90 triangle, and $cos\ 30^\circ = \frac{r}{r+\overline{IA}} \Rightarrow \frac{\sqrt{3}}{2} = \frac{r}{r+\overline{IA}} \Rightarrow \sqrt{3}r + \sqrt{3}\ \overline{IA} = 2r \Rightarrow \sqrt{3}\ \overline{IA} = 2 - \sqrt{3} \Rightarrow \overline{IA} = \frac{2-\sqrt{3}}{\sqrt{3}} \Rightarrow \overline{IA} = \frac{2}{\sqrt{3}} - 1 \Rightarrow \overline{CA} = 1 + (\frac{2}{\sqrt{3}} - 1) \Rightarrow \overline{CA} = \frac{2}{\sqrt{3}} = \overline{CA} = \frac{2\sqrt{3}}{3}$

Since $\overline{BD} = \overline{CA}$, $\Rightarrow \overline{AB} = \overline{CD}$ - $2\overline{CA} = 2\sqrt{3} - 2(\frac{2\sqrt{3}}{3}) = 2\sqrt{3} - \frac{4\sqrt{3}}{3} = \frac{2\sqrt{3}}{3} \Rightarrow \overline{AB} > 1$.

We arrive at a contradiction. Hence, A and B cannot be neighbours so, $Redundant_1()$ cannot mark both of them as unchosen.



(a) Case 1 - Two uncovered gap regions between 4 *chosen* sensors

(b) Case 1 - ($\overline{AB} > R_C$)

(c) Case 2 - (Sensor A and Sensor B are not neighbors)

(d) Case 3 (Sensor A and Sensor B are not neighbors)

Figure 3.1: Suboptimal MIS

Case 2: The sensors in the maximal independent set chosen by the MISNode(i) predicate formed an initial pattern of coverage as shown in Figure 3.1(c). If we let Node A be the sensor in the uncovered region between the four sensors and Node B be a sensor in an uncovered region outside of the four sensors, then by similar reasoning as Case 1, if A and B are not chosen then the distance between Node A and Node B have to be less than or equal to R_C. Assume $R_C = 1$. Since $\overline{AB} = \overline{CD}$, $\overline{AB} = 2R_C = 2 \Rightarrow \overline{AB} > 1$. So nodes A and B are not neighbors, hence they will be chosen.

Case 3: Assume the maximal independent set constructed by the first step of the algorithm as shown in Figure 3.1(c). Since the graph is densely populated, there will be more than one *unchosen* node in each uncovered region. Therefore, there may be more than one *undecided* sensors that are not neighbors of *chosen* sensors, unless they are the "least UID" neighbors of *chosen* sensors, or that have part of their transmission disks not covered by *chosen* sensors. Predicate $FillNode(i)$ and $\mathcal{A}_3$ will mark these nodes as

chosen and $Redundant_1(i)$ will not unmark these nodes. Since the sensing disk of each of these sensors spans a distance of $2R_C$, and yet each sensor will remain *chosen* if it is located at a distance of greater than one R_C from another *chosen* node, each of these uncovered regions will eventually be covered by *chosen* nodes and will remain covered by these *chosen* nodes.

Overall, all the uncovered regions between the sensors chosen by the first step of the algorithm will be eventually covered. □

Lemma 3.2 (Connectivity). *In any legitimate configuration, the set of nodes chosen by Algorithm 3.1 forms a connected graph.*

Theorem 3.1 ($\mathcal{L}_{MCSC}$ satisfies specification). *Any configuration satisfying the legitimacy predicate $\mathcal{L}_{MCSC}$ (per Definition 3.1) satisfies the specification of the connected sensor cover problem (as given by Specification 2).*

Proof. Trivially follows from Lemmas 3.1 and 3.2 □

3.3.1 Proof of Convergence

The goal of this section is to prove that starting from any arbitrary configuration the system converges to a legitimate configuration.

Proof. We formulate this proof by contradiction. Suppose that starting from any arbitrary configuration there exists an execution of the system such that it does not reach a legitimate configuration after an arbitrary finite number of steps. Therefore, there exist a non-redundant sensor that is never marked as chosen or a redundant sensor that is never marked as unchosen.

Case 1: The non-redundant sensor, that may evaluate true Bridge(i) and $Fill(i)$ does not do so and consequently does not execute $\mathcal{A}_3$. A sensor whose sensing disk intersects with the query region and whose sensing disk does not intersect with a *chosen* sensor, or an *undecided* Sensor A that is not the neighbor of any other *undecided* Sensor B whose UID is greater than that of Sensor A, or that is the "least UID" neighbor of Sensor B, is not marked *chosen*. Since any query region sensor that is initially *unchosen*, is non redundant, and whose sensing disk does not intersect with a *chosen* sensor will evaluate $MISNode(i)$ as true, this node will evaluate the guard of $\mathcal{A}_3$ as true. This (non redundant) node will execute $\mathcal{A}_3$ and will change to *chosen*. Hence the contradiction.

Alternatively, since a query region sensor, let's name it Sensor B, that is initially *unchosen* and that is the "least" UID neighbor of a *chosen* (or *undecided*) sensor, and that has no other *undecided* neighbors, will evaluate $BridgeNode(i)$ as true. Sensor B will then execute $\mathcal{A}_2$ and change to *undecided*. Or if Sensor B is initially *undecided*, it will then evaluate $FillNode(i)$ as true and will evaluate the guard of $\mathcal{A}_3$ as true. This (non redundant) sensor will execute $\mathcal{A}_3$ and will change to *chosen*. Hence the contradiction.

Case 2: The non redundant query region sensor is initially marked *chosen*, but executes $Redundant(i)$ and is unmarked. Since any non redundant sensor is one that may be located in an uncovered region and one whose sensing disk is needed to cover the query region, if this sensor is *chosen* and yet is not the neighbor of another *chosen* sensor having a greater UID than its own, then the sensor will evaluate $Redundant_2(i)$ as false and will not become unmarked. Hence the contradiction.

Case 3: If a redundant sensor is marked as *chosen*, $Redundant_1(i)$ or $Redundant_2(i)$ will not unmark this sensor. Since any redundant sensor is one which is not needed to ensure coverage nor connectivity and which is *undecided* and is the "lesser", but not "least UID", neighbor of an *undecided* sensor, or that is *chosen* and is the "lesser", but not "least UID", neighbor of another *chosen* sensor, such a sensor will evaluate $Redundant_1(i)$ or $Redundant_2(i)$ as true and will subsequently execute $\mathcal{A}_1$. So, any such redundant sensor will become unmarked by rule $\mathcal{A}_1$. Hence the contradiction. □

3.3.2 Proof of Self-* Properties

Self-stabilization From the proofs of closure and convergence, it was shown that starting from any initial configuration, Algorithm 3.1 forms a network topology in which all members of the minimum connected sensor cover are connected, and are thus able to communicate with each other, either directly or indirectly. It was also shown that starting from any arbitrary state, the given query region will eventually be completely covered.

Self-healing We formulate this proof by contradiction. Suppose Algorithm 3.1 is not self-healing.

Assume the following faults: (1) sensor crashes and (2) weakening of the sensing capabilities.

Case 1:

Assume by contradiction that if a non redundant node fails, then part of the query region becomes uncovered.

An *unchosen* node in this uncovered region (that is the "least UID" neighbor of all *undecided* nodes within its transmission disk) will evaluate $BridgeNode(i)$ as true, and $FillNode(i)$ as true, this node will execute $\mathcal{A}_2$ and $\mathcal{A}_3$ to become *chosen*. All the unchosen non-redundant nodes in this region will apply the same policy, so the coverage heals. Hence the contradiction.

Case 2: If a part of the query region is covered by a redundant node, then since any node that is *chosen* or *undecided* and that is not the "least UID" neighbor of another *undecided* or *chosen* node, but that has a "lesser" UID than this node, will not evaluate $BridgeNode(i)$ nor $FillNode(i)$ as true, this node will not execute $\mathcal{A}_2$ and change to *undecided*, nor will it execute $\mathcal{A}_3$ and change to *chosen*. Thus, redundant nodes will not cover part of the query region.

Case 3: If there is an arbitrary corruption of the state variables of nodes, including the $Status_i$ variable, then part of the query region may become uncovered, or may be covered by a redundant node. If the $Status_i$ variable for a node is initially *undecided* or *chosen*, then part of the query region may become uncovered, or may be covered by a redundant node.

Since $FillNode(i)$ evaluates to true if a node is *undecided*, and is not the neighbor of any *undecided* sensor having a greater UID than its own, or if it is the "least UID" neighbor of any *undecided* sensor, irregardless of whether it was initially *undecided*, and since a *chosen* node will cover part of the query region, such an arbitrary corruption will still allow a node to execute $\mathcal{A}_3$ and cover the query region. Hence the contradiction.

Alternatively, $Redundant_1(i)$ unmarks a sensor even if it is initially *undecided* and is the neighbor of another *undecided* sensor having a greater UID than its own, but is not the "least UID" neighbor of this sensor. Then no part of the query region will not covered by a redundant nodes. Hence we arrive at a contradiction.

Alternatively, $Redundant_2(i)$ unmarks a *chosen* sensor, irregardless of whether it was initially *chosen*, that is the "lesser UID" neighbor of another *chosen* sensor, but that does not have the least UID out of all the neighbors of this sensor. So no part of the query region will not be covered by a redundant node. Hence we arrive at a contradiction.

Overall, Algorithm 3.1 self-heals.

4 Pruning-based Self* Query Region Connected Cover

4.1 Normal Execution of Algorithm 4.1

In this algorithm, every sensor sends its closed neighbor set (including the value of $Status_i$ of the sensors in this set), to all of its neighbors. The steps of the algorithm are as follows:

1. The algorithm marks all *unchosen* sensors whose sensing regions intersect with some portion of the query region (R_Q), and that are not the neighbors of *chosen* sensors whose UID's are greater than their own, or for which these sensors are not the"least UID" neighbors, as *undecided.*

2. $MCSCNode(i)$ checks if Sensor i is *undecided*, and if a neighbor of Sensor i (i.e., a sensor within Sensor i's transmission disk) is not "dominated" by a *chosen* sensor (i.e., is not within the transmission disk of a *chosen* sensor), or if Sensor i is not the neighbor of a *chosen* sensor whose UID is greater than its own or for which Sensor i is not the "least UID" neighbor. In this case, the sensing disk of Sensor i is needed in the final cover set, and hence Sensor i changes its status to *chosen.*

3. $Redundant_1(i)$ is used to unmark any *undecided* sensor that is the "lesser" neighbor of a *chosen* sensor, but is not the neighbor of this sensor that has the smallest UID, and whose entire transmission disk is covered by *chosen* sensors. In this case, the status of the *undecided* sensor is changed to *unchosen.*

4. $Redundant_2(i)$ removes redundant sensors from the final cover set as follows. If all of the neighbors of Sensor i are within the transmission disk of some *chosen* sensor, and the sensing region of i is totally covered by *chosen* sensors and Sensor i is the "lesser" neighbor of a *chosen* sensor, then Sensor i and all of its neighbors are "dominated" by a *chosen* sensor. In this case, Sensor i should not be in the final set of chosen sensors, and thus changes its status to *unchosen.*

5. Finally, action $\mathcal{A}_1$ ensures that any redundant sensor or any sensor whose sensing disk does not intersect with the query region, has its status changed to *unchosen.*

6. All *chosen* sensors are in the final query region connected cover.

4.2 Faults and Recovery of Algorithm 4.1

In this section, we focus on the fault handling features of the proposed algorithm 4.1. There are three variables used in the solution: S_i, UID_i, and $Status_i$ for a Sensor i. By hypothesis only S_i and $Status_i$ can be corrupted. So, we need to show that our solution can cope with all possible corruptions associated with these two variables. In the following, we will show how they are dealt with in 4.1. **(1) Wrong initialization of the $Status_i$ variable.** All sensors, if properly initialized, start as *unchosen. (a) Sensor i is initialized to* ***undecided*** . Assume that Sensor i is initialized to *undecided.* If i is not a redundant node, then i remains *undecided*, and subsequently changes to *chosen.* (see Actions $\mathcal{A}_2$ and $\mathcal{A}_3$). That is, no correction is necessary. If i is redundant, then it will satisfy the predicate $Redundant_1(i)$ and will change to *unchosen. (b) Sensor i is initialized to* ***chosen*** . If the sensing disk of Sensor i does not intersect with the query region, then, by executing $\mathcal{A}_1$, Sensor i will change to *unchosen.* So, no correction is necessary. If Sensor i is redundant, then then it will satisfy the predicate $Redundant_2(i)$, and will change to *unchosen.* If it is non redundant then Sensor i is necessary, either to ensure coverage or connectivity, and should not be unmarked. **(3) Weakening or Failure of sensors, both in terms of communication and sensing ability.** The weakening or failure of sensors will affect their sensing and communication range. In other words, the constant set R_S or R_C will change. Change of R_S or R_C may change the values of $Redundant(i)$, $SensorCover(i)$, and $MCSCNode(i)$. All these changes will be reflected in the change of values of the guards of the corresponding actions. So, eventually, the status of the affected nodes will change due to the execution of these actions. However, these changes will not affect the execution of these actions by the neighbors of the affected nodes. Therefore, any changes in the $Status_i$ variable of the affected nodes will be handled as mentioned earlier.

Algorithm 4.1 Query Region Connected Sensor Cover Algorithm for Sensor i.

Constants:
R_Q:: Query region;
N_i:: Set of sensors within the communication range of Sensor i;

Shared Variables:
S_i:: Sensing region of Sensor i;
UID_i:: Unique user identification number of Sensor i;
$Status_i \in \{unchosen, undecided, chosen\}$:: Status of Sensor i;

Predicates:
$QryRgnIntrsctn(i) \equiv S_i \cap R_Q \neq \emptyset$;
$\equiv$ sensing disk of Sensor i intersects with some portion of query region;

$NgbrOfChsn(i) \equiv (\exists j : i \in N_j \wedge Status_j = chosen)$;
$\equiv$ Sensor i is a neighbor of a chosen sensor;

$IsLeastUIDNgbr(i, x) \equiv i \in N_x \wedge (\forall j \in N_x : j \neq i \wedge UID_i < UID_j)$;
$\equiv$ Sensor i is a neighbor of Sensor x, and is also the neighbor of Sensor x having the least UID;

$LessNotLeastNgbrOfChsn(i) \equiv (\exists j : i \in N_j : Status_j = chosen \wedge UID_i < UID_j \wedge \neg IsLeastUIDNgbr(i, j))$;
$\equiv$ Sensor i is a neighbor of a $chosen$ sensor whose UID is greater than its own, but Sensor i is not the neighbor of this
sensor that has the smallest UID;

$LessNgbrOfChsn(i) \equiv (\exists j : i \in N_j : Status_j = chosen \wedge UID_i < UID_j$;
$\equiv$ Sensor i is a neighbor of a $chosen$ sensor whose UID is greater than its own

$GrtrOrLeastNgbrOfChsn(i) \equiv (\forall j : i \in N_j : Status_j \neq chosen \vee UID_i > UID_j \vee IsLeastUIDNgbr(i, j))$;
$\equiv$ Sensor i is not the neighbor of a $chosen$ sensor whose UID is greater than its own or for which Sensor i is not the
"least UID" neighbor;

$SensorCover(i) \equiv Status_i = unchosen \wedge QryRgnIntrsctn(i) \wedge GrtrOrLeastNgbrOfChsn(i)$;
$\equiv$ status of Sensor i is $unchosen$, sensing disk of Sensor i intersects with some portion of query region, and Sensor i is not the neighbor
of a $chosen$ sensor whose UID is greater than its own or for which Sensor i is not the "least UID" neighbor;

$MCSCNode(i) \equiv Status_i = undecided \wedge (GrtrOrLeastNgbrOfChsn(i) \vee (\exists j \in N_i : \neg NgbrOfChsn(j)))$;
$\equiv$ Sensor i is an $undecided$ sensor and is not the neighbor of a $chosen$ sensor whose UID is greater than its own or for which Sensor i is
not the "least UID" neighbor, or there is a sensor within the transmission disk of Sensor i that is not the neighbor of a $chosen$ sensor;

$Redundant_1(i) \equiv Status_i = undecided \wedge LessNotLeastNgbrOfChsn(i) \wedge (\forall j \in N_i : NgbrOfChsn(j))$;
$\equiv$ Sensor i is an $undecided$ sensor and is the "lesser" neighbor of a $chosen$ sensor, but is not the neighbor of this sensor that has the
smallest UID, and all sensors within the transmission disk of Sensor i are neighbors of a $chosen$ sensor;

$Redundant_2(i) \equiv Status_i = chosen \wedge LessNgbrOfChsn(i) \wedge NeighborsConnectivity(i) \wedge CoverSensingByChsn(i)$;
$\equiv$ status of Sensor i is $chosen$, Sensor i has a smaller UID than another $chosen$ Sensor j that is within its transmission disk, and the sensing disk of
Sensor i is covered by chosen sensors and chosen neighbors of sensor i are connected through a second path.

$Redundant(i) \equiv Redundant_1(i) \vee Redundant_2(i)$;

Actions:
$\mathcal{A}_1 :: \neg QryRgnIntrsctn(i) \vee Redundant(i) \longrightarrow Status_i = unchosen$;

$\mathcal{A}_2 :: SensorCover(i) \longrightarrow Status_i = undecided$;

$\mathcal{A}_3 :: MCSCNode(i) \longrightarrow Status_i = chosen$;

5 Correctness of Algorithm 4.1

Definition 5.1. *The system is considered to be in a legitimate state (i.e., satisfies the legitimacy predicate $\mathcal{L}_{MCSC}$) if the following conditions are true with respect to a query region:*

i) All non-redundant sensors are marked chosen.

ii) All redundant sensors are marked unchosen.

5.1 Proof of Closure

Lemma 5.1 (Coverage). *In any legitimate configuration, the chosen set computed by Algorithm 4.1 completely covers the query region R_Q.*

Proof. We prove this lemma by contradiction. Suppose the sensing disks of the sensors in the final set chosen by Algorithm 4.1 do not completely cover the query region.
Hence, there is some portion of the query region that is not covered by a *chosen* node.

Since $\mathcal{A}_2$ states that a sensor will change to *undecided* if it is *unchosen*, its sensing disk intersects with some portion of the query region, and if it is not the neighbor of a *chosen* sensor whose UID is greater than its own or for which it is not the "least UID" neighbor, and since the graph is densely populated and all sensors are initially *unchosen*, there will always exist a set of *undecided* nodes, whose sensing disks intersect with the query region, and that will be located at a distance greater than the communication radius, but may also be located less than twice the communication radius from another *chosen* node and from another *undecided* node.

Since an *undecided* node's sensing disk spans a distance of $2R_C$, the union of the sensing disks of all *chosen* nodes and all such *undecided* nodes located at a distance greater than R_C but less than $2R_C$ from any *chosen* or *undecided* node, will completely cover the query region.

Since any *undecided* node will either change to *chosen* by $MCSCNode(i)$ or *unchosen* by $Redundant_1(i)$, and since all such *undecided* nodes are located at a distance greater than R_C from any *chosen* node, each such *undecided* node will evaluate $GrtrOrLeastNgbrOfChsn(i)$ as true and $LessNotLeastNgbrOfChsn(i)$ as false and will change to *chosen* by Rule $\mathcal{A}_3$.

The union of the sensing disks of all nodes that were initially *chosen* and all such *undecided* nodes that changed to *chosen* by executing $\mathcal{A}_3$, completely cover the query region.

Since $Redundant_2(i)$ will only evaluate to true if a node evaluates $LessNotLeastNgbrOfChsn(i)$ as true, and all of its neighbors are covered by a *chosen* node, the $Redundant_2(i)$ predicate will only unmark any of these *chosen* nodes if its entire transmission and sensing disks are completely covered by some other *chosen* node.

The sensing disks of all *chosen* sensors in the final $\mathcal{MCSC}$ completely cover the query region.

Hence we arrive at a contradiction. □

Lemma 5.2 (Connectivity). *In any legitimate configuration, the chosen set computed by Algorithm 4.1 forms a connected graph.*

Proof. We prove this lemma by contradiction. Suppose the sensing disks of the sensors in the final chosen set computed by Algorithm 4.1 do not form a connected subgraph. So, there exists a sensor in the final chosen set, lets name it Sensor A, that is marked *chosen* and is not adjacent to another *chosen* sensor. That is, sensor A is marked *chosen* and is not within the transmission disk of another *chosen* sensor. More precisely, sensor A is marked *chosen* and does not have a *chosen* neighbor.

*Case*1 :

$SensorCover(i)$ and $MCSCNode(i)$ did not mark an *unchosen* sensor that is also a neighbor with a greater UID or the "least UID" neighbor of Sensor A, let's name it Sensor B, as *chosen*, or $Redundant_2(i)$ unmarked this sensor. Since all sensors can have a status of *unchosen*, *undecided*, or *chosen*, and Sensor A has no *chosen* neighbors, all of Sensor A's neighbors must be either *unchosen* or *undecided*. Since Sensor A has no *chosen* neighbors, and since all *undecided* neighbors of Sensor A that evaluate $MCSCNode(i)$ as true will change to *chosen*, all *undecided* neighbors of Sensor A must have evaluated $MCSCNode(i)$ as false. So, all *undecided* neighbors of Sensor A must have evaluated $GrtrOrLeastNgbrOfChsn(i)$ as false, and all neighbors of these *undecided* sensors must have evaluated $NgbrOfChsn(j)$ as true. Since $LessNotLeastNgbrOfChsn(i)$ is the negative of $GrtrOrLeastNgbrOfChsn(i)$, and all neighbors of these *undecided* sensors evaluated $NgbrOfChsn(j)$ as true, all *undecided* neighbors of Sensor A must have changed to *unchosen* after evaluating $Redundant_1(i)$ as true and executing $\mathcal{A}_1$. So, all neighbors of Sensor A are *unchosen*. Therefore, the "least UID" neighbor of Sensor A must be *unchosen*. Hence we arrive at a contradiction.

Case 2:
Sensors A and B are *chosen* neighbors, but Sensor A or Sensor B was unmarked by $Redundant_2(i)$.

As shown in Case 1, since the "least UID" neighbor of Sensor A must be an *unchosen* sensor, let's name it Sensor B, and since Sensor B will change to *chosen* after executing $\mathcal{A}_2$ and $\mathcal{A}_3$, then Sensor B cannot evaluate $LessNotLeastNgbrOfChsn(i)$ as true. So, sensor B cannot evaluate $Redundant_2(i)$ as true. Consequently, sensor B cannot be unmarked by $Redundant_2(i)$. Hence we arrive at a contradiction.

Alternatively, since Sensor A has a greater UID than Sensor B, Sensor A cannot evaluate $LessNotLeastNgbrOf$ as true. So, sensor A cannot be unmarked by $Redundant_2(i)$. Hence we arrive at a contradiction. □

Theorem 5.1 ($\mathcal{L}_{MCSC}$ satisfies specification). *Any system configuration satisfying the legitimacy predicate $\mathcal{L}_{MCSC}$ (per Definition 7.1) satisfies the specification of the connected sensor cover problem (as given by Specification 2.1).*

Proof. The coverage and connectivity properties have been proven in Lemmas 7.1 and 7.2, respectively. The definition of $\mathcal{L}_{MCSC}$ implies that in a legitimate configuration, there exist no redundant *chosen* sensor, meaning that all redundant sensors have been identified and are marked *unchosen*. Therefore, the connected cover set $\mathcal{MCSC}_Q$ computed at this point is the smallest possible by Algorithm 2 $\mathcal{MCSC}$. □

Property 5.1. *The system defined by the legitimacy predicate $\mathcal{L}_{MCSC}$ is silent.*

Proof. In any configuration satisfying $\mathcal{L}_{MCSC}$, all actions of Algorithm 4.1 are disabled. □

Lemma 5.3 (Closure). *The legitimacy predicate $\mathcal{L}_{MCSC}$ is closed.*

Proof. Property 7.1 asserts the closure of $\mathcal{L}_{MCSC}$. □

5.2 Proof of Convergence

The goal of this section is to prove that starting from any arbitrary configuration of the system of sensors, Algorithm 4.1 guarantees that in finite steps, the system will reach a configuration that satisfies the legitimacy predicate $\mathcal{L}_{MCSC}$.

Proof. We formulate this proof by contradiction. Suppose that starting from any arbitrary configuration of the system of sensors, Algorithm 4.1 does not guarantee that in finite steps, the system will reach a configuration that satisfies the legitimacy predicate $\mathcal{L}_{MCSC}$. So, there exists a configuration in which, after any finite number of steps, the system will never reach a configuration that satisfies the legitimacy predicate $\mathcal{L}_{MCSC}$. That is, there exists a configuration in which, after any finite number of steps, the system will never reach a configuration in which all non redundant sensors are marked *chosen* and all redundant sensors are marked *unchosen*.

Case 1: There exists a configuration in which a (non redundant) sensor whose status is *unchosen*, whose sensing disk intersects with some portion of the query region, and that may evaluate $GrtrOrLeastNgbrOfChsn(i)$ and $MCSCNode(i)$ as true, does not do so and does not execute $\mathcal{A}_3$. So, a query region sensor which is *unchosen*, not the neighbor of a *chosen* whose UID is greater than its own or for which it is not the "least UID" neighbor, and that has part of its transmission disk not covered by another *chosen* sensor, is not marked *chosen*. Since any query region sensor that is initially *unchosen*, and is non redundant because it is not the "lesser" neighbor of a *chosen* sensor nor the "least UID" neighbor of this *chosen* sensor, and which has a sensor within its transmission disk that is not the neighbor of a *chosen* sensor, will evaluate $QryRgnIntrsctn(i)$ and $GrtrOrLeastNgbrOfChsn(i)$ and $MCSCNode(i)$ as true, this node will evaluate the guard of $\mathcal{A}_2$ and $\mathcal{A}_3$ as true. So, this (non redundant) sensor will execute $\mathcal{A}_2$, followed by $\mathcal{A}_3$, and will change to *chosen*.

Hence we arrive at a contradiction.

Case 2:

The non redundant query region sensor is initially marked *chosen*, but executes $Redundant(i)$ and is unmarked. Since this sensor executed $Redundant(i)$, it is the neighbor of a *chosen* sensor having a greater UID than itself, but is not the "least UID" neighbor of this *chosen* sensor, and all sensors within its transmission disk are neighbors of a *chosen* sensor. So, this sensor's entire transmission (and sensing) disk is covered by the sensing disks of other *chosen* sensors (i.e. this sensor is redundant).

Hence we arrive at a contradiction.

Case 3:

If a redundant sensor is marked as *chosen* or *undecided*, $Redundant_1(i)$ or $Redundant_2(i)$ will not unmark this sensor. Since a redundant sensor is one whose entire sensing disk is covered by the sensing disks of other *chosen* sensors, and whose removal will not leave part of the query region uncovered, such a redundant sensor having a smaller UID than its *chosen* neighbor, but that is not the "least UID" neighbor of this *chosen* sensor, will evaluate $LessNotLeastNgbrOfChsn(i)$ as true, and will have all of its neighbors evaluate $NgbrOfChsn(j)$ and $ENgbrOfChsn(j,i)$ as true. So, such a (redundant) sensor will evaluate $Redundant_1(i)$ and $Redundant_2(i)$ as true. Consequently, such redundant sensor will execute $\mathcal{A}_1$ and will be unmarked.

Hence we arrive at a contradiction.

□

5.3 Proof of Self-* Properties

Self-stabilization From the proofs of closure and convergence, it was shown that starting from any initial configuration, Algorithm 4.1 forms a network topology in which all members of the minimum connected sensor cover are connected, and are thus able to communicate with each other, either directly or indirectly. It was also shown that starting from any arbitrary state, the given query region will eventually be completely covered. By executing the rules of Algorithm 4.1, network sensors will self-configure to establish a topology that enables communication and sensing coverage under stringent energy constraints. Hence Algorithm 4.1 is self-configuring.

Self-healing We formulate this proof by contradiction. Suppose Algorithm 4.1 is not self-healing. So, if a non redundant node fails, a redundant node joins the network, or if there is an arbitrary corruption of the state variables of nodes, including the $Status_i$ variable, then part of the query region may become uncovered, or may be covered by a redundant node.

Case 1:

If non-redundant node fails, then part of the query region becomes uncovered. Since the graph is densely populated, there is a portion of the graph in which an *unchosen* sensor that is in this uncovered region, does not execute $\mathcal{A}_2$ and $\mathcal{A}_3$ to become *chosen*. But since this *unchosen* sensor is not covered by a *chosen* sensor, and since all *unchosen* sensors will not be the neighbors of any *chosen* sensor, and since this node will also have part of its transmission disk not covered by a *chosen* sensor, it will evaluate the guard of $\mathcal{A}_2$ as true and $MCSCNode(i)$ as true.

This node will execute $\mathcal{A}_2$, followed by $\mathcal{A}_3$, and will become *chosen*.

Hence we arrive at a contradiction.

Case 2:

If part of the query region is covered by a redundant node, then since any node that is the "lesser", but not "least UID", neighbor of a *chosen* node, and whose entire transmission disk is covered by *chosen*

nodes, is redundant and will not evaluate $GrtrOrLeastNgbrOfChsn(i)$ as true, this node will not execute $\mathcal{A}_2$ and change to *undecided*, nor will it execute $\mathcal{A}_3$.

So, this node cannot change to *chosen* to cover the query region.

Hence we arrive at a contradiction.

Case 3:

If there is an arbitrary corruption of one of the state variables of nodes, including the $Status_i$ variable, then part of the query region may become uncovered, or may be covered by a redundant node. If the $Status_i$ variable for a node is initially *undecided* or *chosen*, then part of the query region may become uncovered, or may be covered by a redundant node.

Since $MCSCNode(i)$ evaluates to true if an *undecided* sensor is not the neighbor of a *chosen* sensor having a greater UID than its own or for which it is not the "least UID" neighbor, and if it has part of its transmission disk uncovered, regardless of whether it was initially undecided, and since a *chosen* node will cover part of the query region, such an arbitrary corruption will still allow a node to execute $\mathcal{A}_3$ and cover the query region.

Hence we arrive at a contradiction.

Alternatively, since $Redundant_1(i)$ will unmark a sensor even if it is initially *undecided* and is the "lesser" neighbor of a *chosen* sensor, but is not the neighbor of this sensor that has the smallest UID, and it has all parts of its transmission disk covered by a *chosen* sensor, then part of the query region will not be covered by a redundant node.

Hence we arrive at a contradiction.

Alternatively, since $Redundant_2(i)$ will unmark a *chosen* sensor that is a "lesser", but not "least UID", neighbor of another *chosen* sensor, and whose transmission disk is completely covered by other *chosen* sensors, regardless of whether it was initially *chosen*, part of the query region will not be covered by a redundant node.

Hence we arrive at a contradiction.

6 Simulation and Results

In our simulations, for the first set of experiments, we assumed that nodes are chosen and randomly deployed on a grid of size 500 × 500 (300,000 nodes). Similar to [11, 15, 23] we consider the sensing region associated with a sensor modeled as a circular region around itself. We considered a homogeneous network of 300,000 nodes (i.e. all sensors had the same sensing region — circular of radius 6). We then used varying sizes for a query region, and measured the number of sensors in the final minimum connected cover set, the number of query region sensors (dominated) per MCSC sensor, and the stabilization time for Algorithms 3.1, 4.1[1], Algorithm $MCSC$ 1 [7], Algorithm $MCSC$ 2 [5], and Rule k [4]. The query region used in each simulation varied from 60 × 60 square graph units to 120 × 120 square graph units, in intervals of 10 square graph units. The results of this simulation are summarized in Figures 6.1(a) - (c).

The simulations summarized in Figures 6.1(d)-(i) were performed with a query region of size 90 × 90 square graph units and 100000 sensors. At all query region sizes, Algorithm 3.1 produced fewer nodes in the final cover set than Algorithm 4.1, Algorithm $MCSC$ 2, and Rule k. Algorithm 3.1 produced fewer nodes in the final cover set than Algorithm $MCSC$ 1 when the size of the query region was 80 x 80 or 90 x 90 square graph units. Algorithm 4.1 produced a greater number of nodes in the final cover set than Algorithm 3.1 but fewer nodes in the final cover set than Algorithm $MCSC$ 2 and Rule k, and, when the query region was 80 x 80 square graph units, fewer nodes than Algorithm $MCSC$ 1. Algorithm $MCSC$ 2 produced a greater number of nodes in the final cover set than Algorithm 3.1 and Algorithm 4.1 at all

[1]This algorithm is a self-stabilizing extension of classical pruning strategies for the computation of connected dominating sets. Due to space limitations this algorithm is available as technical report [6] and is proposed in the Appendix section

query region sizes tested. However, it produced a final cover set that was smaller than Rule k's at query region sizes that were less than 90 x 90 square graph units and larger than Rule k's at query region sizes greater than this. Rule k produced the greatest number of nodes in the final cover set at query region sizes that were less than 90 x 90 square graph units, but produced fewer nodes in the final cover set than Algorithm $MCSC$ 2, at query region sizes that were 90 x 90 square graph units or greater. This was due to the fact that Algorithm 3.1 has the strongest redundancy predicate, since it only requires that a Sensor i be the neighbor of a *chosen* sensor and also have a smaller UID than this *chosen* sensor but not the least UID out of all the neighbors of this *chosen* sensor, before it is unmarked. Algorithm $MCSC$ 1 has a redundancy predicate that is weaker than Algorithm 3.1 but stronger than Algorithm 4.1 and Rule k, since it only requires that the sensing disk of Sensor i be covered by the sensing disks of any (black) final cover set nodes before it is unmarked. Algorithms 4.1 and Algorithm $MCSC$ 2 have a redundancy predicate that is weaker than that of Algorithm 3.1 but stronger than that of Rule k, since it requires that a Sensor i be the neighbor of a *chosen* sensor, and also have a smaller UID than this *chosen* sensor but not the least UID out of all the neighbors of this *chosen* sensor, and that all sensors within the transmission disk of Sensor i are also neighbors of a *chosen* sensor, before Sensor i is unmarked. Also, since Algorithm $MCSC$ 2 uses the Connector(i) predicate to ensure connectivity and uses the LessNotLeastNgbrOfChsn(i) predicate as part of its redundancy predicate, in any particular covered area of the query region, only the node with the greatest and the least UID will be marked as *chosen*. Rule k has the weakest redundancy predicate, since it requires that all sensors within the transmission disk of Sensor i be covered by marked sensors and that Sensor i also has the least UID out of all the nodes that cover its transmission disk, before it is unmarked. Also, as shown in Figure 6.1(b)each sensor in the final cover set chosen by Algorithms 3.1 and 4.1 "dominated" a greater number of nodes than Rule k. Final cover set nodes chosen by Algorithm $MCSC$ 2 "dominated" a greater number of nodes than Rule k at query region sizes less than 90 x 90 square graph units, but fewer number of nodes than Rule k at query region sizes greater than this. Thus Algorithms 3.1 and 4.1 did outperform Rule k in the sense that they allowed more nodes to be in an "inactive" state at all the query region sizes tested in our simulation, and Algorithm $MCSC$ 2 outperformed Rule k at query region sizes less than 90 x 90 square graph units. Algorithm 3.1 outperformed Algorithm $MCSC$ 1 at all query region sizes tested since each final cover set sensor chosen by Algorithm 3.1 "dominated" a greater number of nodes than those of Algorithm $MCSC$ 1. Also, Algorithm 4.1 and Algorithm $MCSC$ 1 were very similar in terms of the number of query region nodes "dominated" by each final cover set node, at all query region sizes tested. However, as shown in Figure 6.1(c) Algorithm 3.1 had the highest stabilization time of all the algorithms. This increased stabilization time is attributed to the fact that Algorithm 3.1 has the strongest redundancy predicate, and therefore will incur the greatest time cost when unmarking redundant *chosen* nodes and again producing a sensor cover consisting of nonredundant nodes after re-stabilization. Furthermore, the stabilization time of Algorithm $MCSC$ 2 is greater than Algorithm 4.1 and Rule k. This is due to the fact that Algorithm $MCSC$ 2 has a redundancy predicate that is not weaker than that of both algorithms, and yet sends messages that must travel throughout the network.

Furthermore, the size of the final cover sets produced by Algorithms 3.1 and 4.1 is smaller than that produced by Algorithm $MCSC$ 2 and Rule k. Therefore, both Algorithms 3.1 and 4.1 outperformed Rule k in terms of the size of the final cover set at all sensor densities tested in our simulation. The final cover sets produced by Algorithm $MCSC$ 2 and Rule k were similar in terms of size, when the total number of sensors in the simulation was less than 300,000 nodes. Algorithm 3.1 produced fewer nodes in the final cover set than Algorithm $MCSC$ 1 at sensor densities greater than 200,000 nodes. The final cover set nodes chosen by Algorithm 3.1 "dominated" a greater number of nodes than those chosen by Algorithm $MCSC$ 1 at sensor densities greater than 150,000 nodes. Also, the final cover set nodes chosen by Algorithm 4.1 "dominated" a greater number of nodes than those chosen by Algorithm $MCSC$ 1 at sensor densities greater than 250,000 nodes. Thus Algorithm 3.1 outperformed Algorithm $MCSC$ 1 at

sensor densities greater than 150,000 nodes, and Algorithm 4.1 outperformed Algorithm $MCSC$ 1 at sensor densities greater than 250,000 nodes. However, the number of $MCSC$ sensors for both Algorithms 3.1 and 4.1 did not monotonically increase when the node density was increased, while that of Algorithm $MCSC$ 1 did and that of Rule k did increase sharply when the node density was greater than 300,000 nodes per 500×500 graph units. This may be attributed to the fact that at higher node densities, there may have been a greater number of nodes that covered any particular marked sensor's transmission disk, and thus a less likelihood that a marked sensor had the least UID of all the sensors that covered its transmission disk. Therefore, fewer nodes would have been unmarked at higher node densities by Rule k. Figure 6.1(g) shows that Algorithms 3.1, 4.1, $MCSC$ 1, $MCSC$ 2, and Rule k produced smaller final cover sets as the radius of communication of the sensors was increased. However, Algorithms 3.1 and 4.1 produced smaller cover sets than Rule k at all sizes of the radius of communication that were tested. Also, Algorithm 3.1 produced a smaller final cover set than Algorithm $MCSC$ 1 at all sizes of the radius of communication that were tested. Algorithm 4.1 produced a smaller final cover set when the radius of communication was greater than 7. Also, as shown in Figure 6.1(h), each sensor in the final cover set chosen by Algorithms 3.1 and 4.1 "dominated" a greater number of nodes than Rule k, at all sizes of the radius of communication that were tested. This indicates that Algorithms 3.1 and 4.1 outperformed Rule k, in terms of the size of the final cover set and the number of query region sensors covered by each node in the final cover set, at all sizes of the radius of communication that were tested. Algorithm 3.1 outperformed Algorithm $MCSC$ 1 at all sizes of the radius of communication that were tested, in terms of producing a better cover set. Also, both Algorithms $MCSC$ 2 and Rule k produced a cover set that was very similar in size, when the size of the radius of communication of the sensors was 6 and the size of the query region was 90 x 90 graph units. Also, as the size of the radius of communication was increased, each sensor chosen by Algorithms 3.1, 4.1, and $MCSC$ 2 also "dominated" a greater number of query region sensors. This seems intuitive since the size of the radius of communication is equal to the size of the radius of the sensing disk of sensors in Algorithms 3.1, 4.1, and $MCSC$ 2. Therefore, as the radius of communication was increased in size, there were a greater number of nodes within the transmission disk, and thus within the sensing disk, of chosen sensors in the simulation. Thus, in Algorithms 3.1, 4.1, and $MCSC$ 2, there was a smaller probability of nodes being chosen by $\mathcal{A}_2$ and $\mathcal{A}_3$. Also, since there was an increased likelihood that a node was the neighbor of another *chosen* sensor that had a greater UID than its own but was not the "least UID" neighbor of this *chosen* sensor, a greater number of *chosen* sensors may have been unmarked by the redundancy predicates of both algorithms. Each final cover set node chosen by Algorithm 4.1 "dominated" a greater number of nodes than that of Algorithm $MCSC$ 1 at all sizes of radius of communication that were tested. In this respect, Algorithm 4.1 outperformed Algorithm $MCSC$ 1.

The stabilization times of both Algorithm 4.1 and Rule k were very similar at all sizes of the radius of communication that were tested. Also, despite the fact that Algorithm 3.1 had a higher stabilization time than Algorithm 4.1, Algorithm $MCSC$ 1, Algorithm $MCSC$ 2, and Rule k, Algorithm 3.1 still produced fewer nodes in the final cover set. While Rule k does stabilize faster than Algorithms 3.1, 4.1, and $MCSC$ 2, the slower stabilization times seem justified due to the fact that the latter three algorithms do not compromise connectivity, nor coverage. Also, even though Algorithm 4.1 had a higher stabilization time than Algorithm $MCSC$ 1 and Rule k, it outperformed both of these algorithms in terms of the number of query region sensors "dominated" by each cover set node.

7 Conclusions and Future Work

We presented two local, distributed, scalable, self-$*$ solutions to the minimum query region connected cover problem and showed how this solution is self-organizing and self-healing as well. The algorithms are self*

contained, meaning that after a fault (transient or definitive) occurs in the system, after re-stabilization, only nodes within the locality of the faulty nodes change their status. Note that our solutions can be easily transformed in power-aware algorithms if instead of eliminating nodes based on the value of their identifiers, they will be eliminated based on the state of their batteries. Consequently, only nodes with strong energetic capabilities are chosen in the final cover.

We proved the self* properties of our solutions through analytical analysis. Moreover, we have conducted extended simulations using the following measures: stabilization time, number of nodes in the final cover, number of non-chosen nodes covered by the nodes in the final cover. Our experiments demonstrate that the proposed algorithms performs better than self-stabilizing algorithms we proposed in [7], [5]. Furthermore, we compared the greedy-based solution with pruning based methods. In the case of the query region connected coverage problem the greedy methods offer better results in terms of connected coverage size.

The area of connected coverage still raise a broad class of challenges. The generalization of this problem to k-connected k-coverage problem has been studied in fault-free environments in [24, 25]. An interesting open issue would be to address this problem in fault prone environments and in its generalized form: self-stabilizing k-coverage k-connectivity of query regions.

References

[1] J Carle and D Simplot-Ryl. Energy-efficient area monitoring for sensor networks. *IEEE Press*, pages 40–46, Feb 2004.

[2] A Cerpa and D Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *INFOCOM02 Proceedings of the Conference on Computer Communications*, Jun 2002.

[3] B Chen, K Jamieson, H Balakrishnan, and R Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *MobiCom02 Proceedings of the Seventh Annual Inernational Conference on Mobile Computing and Networking*, pages 85–96, Jul 2001.

[4] F Dai and J Wu. Distributed dominant pruning in ad hoc networks. *Proceedings of ICC'03*, 2003.

[5] AK Datta, M Gradinariu, and R Patel. Distributed self-$*$ minimum connected covering of a query region in sensor networks. In *ISPAN '05*, pages 448–453, 2005.

[6] A.K. Datta, M. Gradinariu, and R. Patel. Dominating-sets based self* connected covers of query regions in sensor networks. Technical report, IRISA, Universite Rennes 1, 2006.

[7] AK Datta, P Linga, M Gradinariu, and P Raipin-Parvedy. Self-* distributed query region covering in sensor networks. In *SRDS05 24th IEEE Symposium on Reliable Distributed Systems*, pages 50–59, Oct 2005.

[8] EW Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the Association of the Computing Machinery*, 17(11):643–644, Nov 1974.

[9] S Dolev. *Self-Stabilization*. MIT Press, 2000.

[10] Roy Friedman, Maria Gradinariu, and Gwendal Simon. Locating cache proxies in manets. In *MobiHoc*, pages 175–186, 2004.

[11] H Gupta, SR Das, and Q Gu. Connected sensor cover: Self-organization of sensor networks for efficient query execution. In *MobiHoc03 Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 189–200, 2003.

[12] F Ingelrest, D Simplot-Ryl, and I Stojmenovic. Smaller connected dominating sets in ad hoc and sensor networks based on coverage by two-hop neighbors. Technical report, Institut National De Recherche En Informatique Et En Automatique, April 2005.

[13] F Kuhn, T Moscibroda, and R Wattenhofer. Initializing newly deployed ad hoc and sensor networks. *MobiCom '04*, 2004.

[14] H Liu, Y Pan, and J Cao. An improved distributed algorithm for connected dominating sets in wireless ad hoc networks. *Proceedings of the ISPA'04*, Dec 2004.

[15] S Shakkottai, R Srikant, and N Shroff. Unreliable sensor grids: Coverage, connectivity and diameter. In *INFOCOM03 Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1073–1083, Apr 2003.

[16] D Tian and ND Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *WSNA02 Proceedings of the First Workshop on Sensor Networks and Applications*, pages 32–41, Sep 2002.

[17] X Wang, G Xing, Y Zhang, C Lu, R Pless, and C Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *ACM SenSys03 Proceedings of the First International Conference on Embedded Networked Sensor Systems*, pages 28–39, Nov 2003.

[18] J Wu. Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):866–881, Sep 2002.

[19] J Wu and H Li. On calculating connected dominating sets for efficient routing in ad hoc wireless networks. *Proceedings of DialM'99*, pages 7–14, 1999.

[20] Y Xu, J Heidemann, and D Estrin. Geography-informed energy conservation for ad hoc routing. In *MobiCom02 Proceedings of the Seventh Annual Inernational Conference on Mobile Computing and Networking*, pages 70–84, 2001.

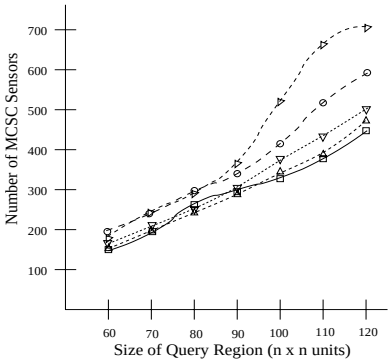
[21] F Ye, G Zhong, J Cheng, S Lu, and L Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *ICDCS03 Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 1–10, 2003.

[22] H Zhang and A Arora. GS3: Scalable self-configuring and self-healing in wireless networks. In *PODC02 Proceedings of the Twentyfirst Annual ACM Symposium on Principles of Distributed Computing*, 2002.

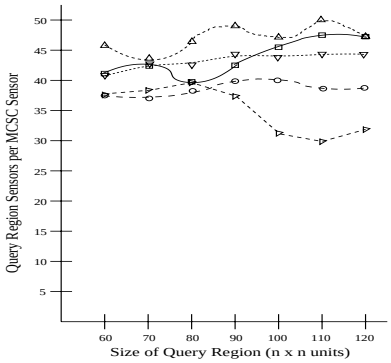
[23] H Zhang and JC Hou. Maintaining sensing coverage and connectivity in large sensor networks. Technical Report UIUCDCS-R-2003-2351, University of Illinois at Urbana Champaign, Jun 2003.

[24] Zongheng Zhou, Samir R. Das, and Himanshu Gupta. Connected k-coverage problem in sensor networks. In *ICCCN*, pages 373–378, 2004.

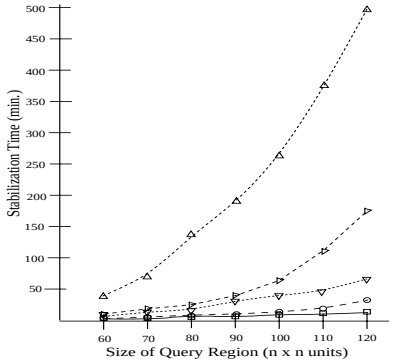
[25] Zongheng Zhou, Samir R. Das, and Himanshu Gupta. Fault tolerant connected sensor cover with variable sensing and transmission ranges. In *SECON*, 2005.



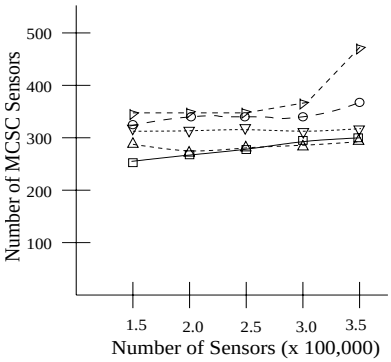
(a) Number of MCSC Sensors for Various Query Region Sizes



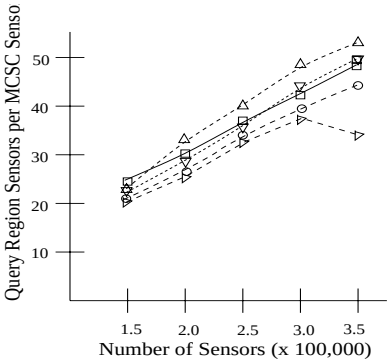
(b) Query Region Sensors per MCSC Sensor for Various Query Region Sizes



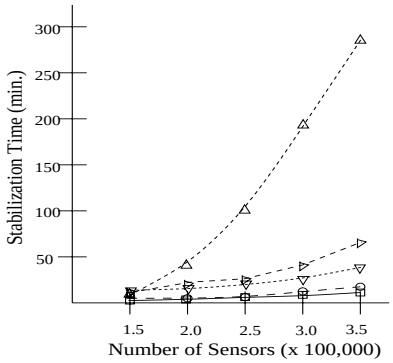
(c) Stabilization Time (minutes) for Various Query Region Sizes



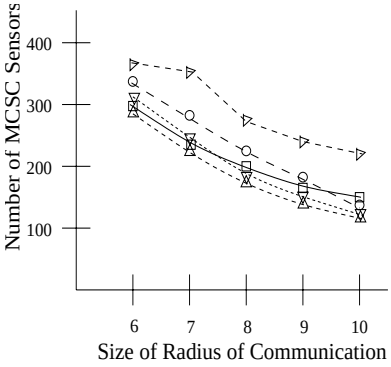
(d) Number of MCSC Sensors for Various Sensor Densities



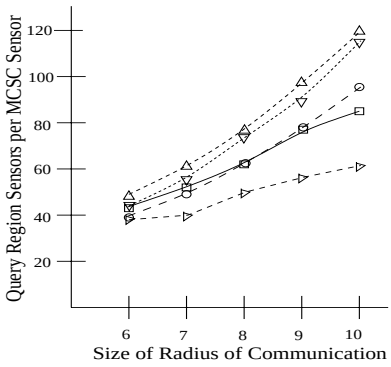
(e) Query Region Sensors per MCSC Sensor for Various Sensor Densities



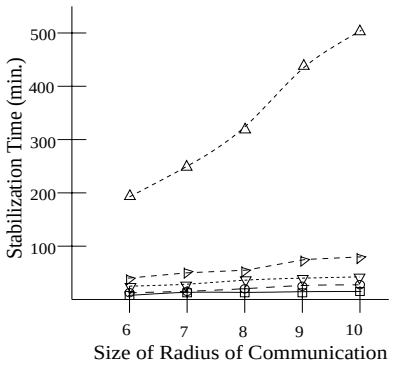
(f) Stabilization Time (minutes) for Various Sensor Densities



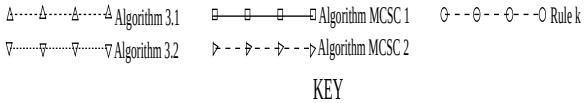
(g) Number of MCSC Sensors for Various Sizes of Radius of Communication



(h) Query Region Sensors per MCSC Sensor for Various Sizes of Radius of Communication



(i) Stabilization Time (minutes) for Various Sizes of Radius of Communication



(j)

Figure 6.1: Experimental Results