A Connectivity Preservation Scheme for Randomly Deployed Wireless Sensor Networks

Nivine Mahmoud Samarji

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
July 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Muhammed Salamah
Supervisor

Examining Committee

1. Assoc. Prof. Dr.Muhammed Salamah
2. Asst. Prof. Dr. Adnan Acan
3. Asst. Prof. Dr. Gurcu Oz

ABSTRACT

A wireless sensor network (WSN) consists of spatially distributed low-power sensors for the purpose of monitoring an area of interest such as battle field or environmental conditions such as weather, earthquakes, pressure, etc. These sensor nodes monitor the field, sense and process the monitored data, then deliver the processed data to the sink in a multi-hop fashion. To achieve communication between nodes, network connectivity should be maintained, which is not always the case especially when sensor nodes are randomly deployed. This will result in the appearance of unreachable nodes or isolated nodes. In most of these cases, network will be partitioned and disconnected. Therefore, connectivity is an essential key factor for determining network quality of service (QoS).

This thesis focuses on achieving high connectivity for randomly deployed wireless sensors by referring to the concept of building a network virtual backbone approach. Although the concept of connected dominating set (CDS) is used as a method to achieve this purpose; however, this method has limitations in presence of isolated or unreachable nodes. Therefore, this thesis contributes to the CDS approach by adding few anchor nodes at calculated distance to gain high network connectivity.

Using MATLAB, extensive simulations have been carried out and the results showed that connectivity has been gained by activating few anchor nodes or spare nodes to random WSNs. Our algorithm had approximately twice the Fiedler value enhancement of Random Addition algorithm.

Keywords: Wireless Sensor Networks, Network Connectivity, Anchor Nodes, QoS.

ÖZ

Kablosuz duyarga ağı (WSN) dağınık ve düşük güç sensörlerden oluşmaktadır. Bu tür ağlar savaş alanı veya hava, deprem ve basınç gibi çevre koşullarının izlenmesinde kullanılmatadır. Bu sensör düğümleri alan duygusu izler, ve izlenen verileri işler, sonra da bir multi-hop metodla işlenmiş verileri belirli bir düğüm istasyonuna iletir. Algılayıcı düğümleri rastgele dağıtıldığı göz önünde bulundurarak, düğümler arasındaki iletişimi sağlamak için, ağ bağlantısı sağlam bir şekilde tutulmalıdır. Ayrıca düğümleri rastgele dağıtıldığından dolayı, bazı düğümler ya da izole edilmiş veya ulaşılamayan duruma gelir. Bu durumların çoğunda, ağ bölümlenmiş ve kopmuş hale gelir. Bu nedenle, bağlantı, ağ hizmeti kalitesinin (QoS) saptanması için önemli bir anahtar faktördür.

Bu tezde sanal ağ omurgası yaklaşımı kavramını kullanarak rastgele dağıtılan kablosuz sensörler için yüksek bağlantı sağlanması üzerinde duruluyor. Bağlı görünen set (CDS) kavramı bu amaca ulaşmak için bir yöntem olarak kullanılmasına rağmen; bu yöntemde, izole edilmiş ya da ulaşılamaz düğüm mevcudiyetinde sınırlamalar vardır. Bu nedenle, bu tez, yüksek ağ bağlantısı elde etmek için hesaplanmış mesafeden birkaç çapa düğümler ekleyerek CDS yaklaşımına katkıda bulunmaktadır.

MATLAB kullanarak, geniş simülasyonlar yapılmıştır ve sonuçlar birkaç çapa düğümler ekleyerek yüksek bağlantı elde edildiğini göstermiştir. Bizim algoritma Rastgele İlavesi Algoritmasının yaklaşık iki Fiedler değer artışı vardır.

Anahtar Kelimeler: Kablosuz Sensör Ağları, Ağ Bağlantısı, Çapa Düğümler, QoS

To My Father's Soul, My Mother, Brothers,
and Husband

ACKNOWLEDGEMENT

First and foremost I would like to show my high gratitude and appreciation to my supervisor, Assoc. Prof. Dr. MUHAMMED SALAMAH for his endless effort and great support along with valuable information that had benefit value on understanding this interesting topic with more details during his supervision of my thesis. I must show my sincere respect to his passion that he had for Wireless Communication Systems, as without his keen support, valuable comments and suggestions, I would not have been able to complete this work.

I would like to express my special thanks to my mother, my brothers, and my lovely husband for their love, trust, and support in all means. I dedicate this study to my father's soul who was and will always be my idol.

Finally, I would like to thank all my friends for supporting and encouraging me during my thesis study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

A	Area
A	Laplacian matrix element
A(G)	Laplacian matrix for graph G
AODV	Ad-hoc On Demand Distance Vector
BS	Base Station
CCOS	Connected Cut Occurred Somewhere
CCPRP	Coverage and Connectivity Preserving Routing Protocol
CDS	Connected Dominating Set
CDSA	Connected Dominating Set with Anchor nodes
CH	Cluster Head
DCD	Distributed Cut Detection Method
DOS	Distributed frOm Source
DSSD	Distributed Source Separation Detection
E	Edge
G	Graph
G(V,E)	Graph consists of vertices and edges
ICCPRP	Improved Coverage and Connectivity Preserving Routing Protocol
MN	Mobile Node
M	Unit in meter
MP-ECCL	Multi-hop-Point Enhancing Coverage/Connectivity with Network Lifetime
N	Number of sensor nodes

QoS	Quality of Service
R	Transmission Range
RFIC	Radio-Frequency Integrated Circuit
SDP	Semi-Definite Protocol
SSBR	Straight Skeleton Construction Based Reconstruction
UDG	Unit Disk Graph
V	Vertex
WSN	Wireless Sensor Network
$\lambda 2$	Second smallest eigenvalue

Chapter 1

INTRODUCTION

1.1 Overview

A wireless sensor network consists of many tiny, cheap, low-power, autonomous microelectronics which are heavily deployed in an area for the purpose of monitoring, sensing and transmitting the sensed data to sink. A sensor is usually made up of an analog-to-digital converter, transceiver, on-board processor, and a battery unit, mobility assistance and location finding system which are optional and depend only on application requirement [4]. The sensor architecture is shown in Figure 1 and Figure 2. The most simulating benefit of using WSN, besides its low-cost, is related to its flexibility and versatility of sensors [1]. Placement of sensor nodes neither requires a special infrastructure to exit, such as network cables for internet connection or electric mains for power supply, nor does it require any human intervention [1] [4]. Instead, sensor nodes adapt themselves to existing environment and use the radio-frequency integrated circuits (RFIC) which is built in sensor nodes for communication. WSNs have been used in numerous areas of interests such as environmental monitoring, battlefield surveillance, healthcare monitoring, transportation, and much more. Besides, its diverse usage and flexibility, there are some serious challenges that face WSN. On first place, deployment of nodes is critical challenge which might cause existence of unconnected partitions when these nodes are randomly deployed. This leads to many negative consequences on network topology as well as network performance.

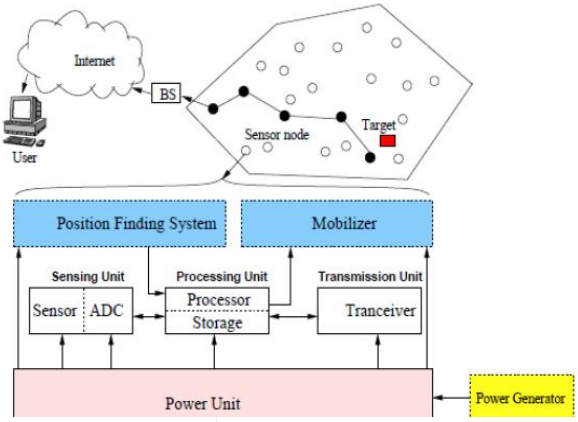

Figure 1: Sensor Architecture [30] [31]

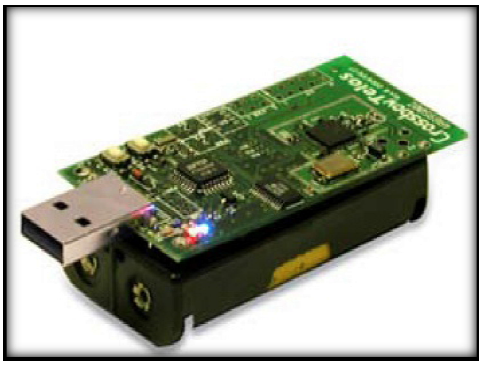

Figure 2: Sensor Node [31]

Therefore, management of network topology is essential to guarantee network connectivity [4]. Second, Energy consumption is considered vital and critical issue to be managed carefully since sensor nodes are supplied with limited battery otherwise nodes will die quickly and data will be lost. The rest of challenges could be node heterogeneity, data aggregation, or fault tolerance. All of the above mentioned challenges have direct impact on the overall network performance and QoS.

1.2 Problem Description

This thesis focuses on achieving high connectivity using different scenarios. First, since sensor nodes are equipped with limited power supply, there should be a managed routing algorithm that reduces energy consumption during transmission. For this purpose, a connected dominating set (*CDS*) [8] will be constructed out of high density nodes who are supposed to act as virtual backbone for network connectivity, and transmit the sensed data to the sink; whereas, the rest of low density nodes who are under same communication range will be set to sleep mode which in this case will save energy and prolong network lifetime. Also, as stated earlier, when wireless sensors are randomly deployed, the probability of having isolated or unreachable nodes is high [7] [8] [12]. Coverage in WSNs is considered a

key factor in QoS. Node coverage is the ratio of number of redundant nodes to the total number of sensor nodes. Area coverage is the ratio of area covered to the total area of the field [6]. Having full connected network is considered ideal in WSNs.

Our study focuses on flat-based network structure illustrated in the construction of connected dominating set (*CDS*) which is known to have positive impact on maximizing network lifetime by using high density connectors as backbone infrastructure and multi-path routing protocol for delivering sensed data between sensor nodes will be done through the *CDS* heads, that are considered to be the network gateway [8].

The work in this thesis will be based on some assumptions such as sensor nodes are homogeneous having same sensing range and energy supply. We assume having Boolean sensing model where sensing task is not affected by strength of emitted signal [22]. The network architecture presented here is modeled as a unit disk graph *(UDG), G= (V, E)* where *V* represents the sensor nodes and *E* represents the link among these sensor nodes. Illustration of *CDS* is shown in Figure 3 where the black nodes represent the *CDS* of graph *G*.

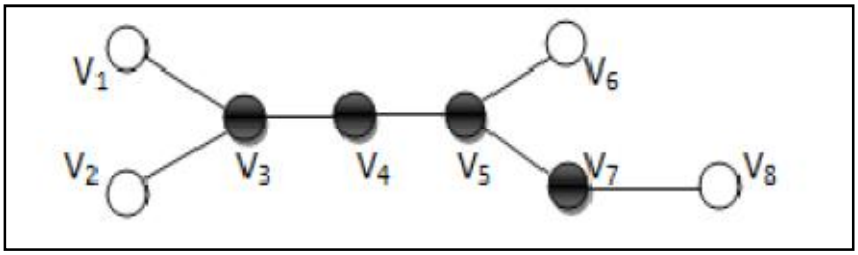

Figure 3: *CDS* of Graph *G*, [8]

This thesis will highlight the problem of network partition, that can caused by different factors such as node failure, and will provide a method to overcome the

network partition and gain high connectivity. Network partition is main reason for splitting network topology into parts, which may lead to disconnecting the communication between source and destination [12]. A survey has been presented [9] that mentioned various partition detection techniques and route discovery methods. Therefore, maintaining a reliable communication between source and destination, partition detection techniques have been used [3] [13] [7] [18].

Simulations have been carried out using MATLAB to analyze the improvement of connectivity and a comparison with different connectivity enhancement algorithms has been made. In particular, the effect of anchor node activation on the network connectivity has been investigated. As well as, the efficiency of our algorithms has been validated by using the network health indicator, the Fiedler value. The findings of this work had a positive impact on the design and implementation of WSNs.

1.3 Related Work

Improving coverage and connectivity of WSNs had simulated many researchers of this field for decades. Network connectivity is considered vital for having successful operations of WSNs. Network connectivity can be maintained either by clustering or virtual backbone [4]. There have been proposed different connectivity schemes for having successful network operations. Different connectivity schemes have been proposed in [4], [15] where cooperative sensing was used to reach a disconnected node by combining the power of emitted signals during the simultaneous transmission. Figure 4 describes the increase in the emission range to reach a destination by summing up the emitted signal power.

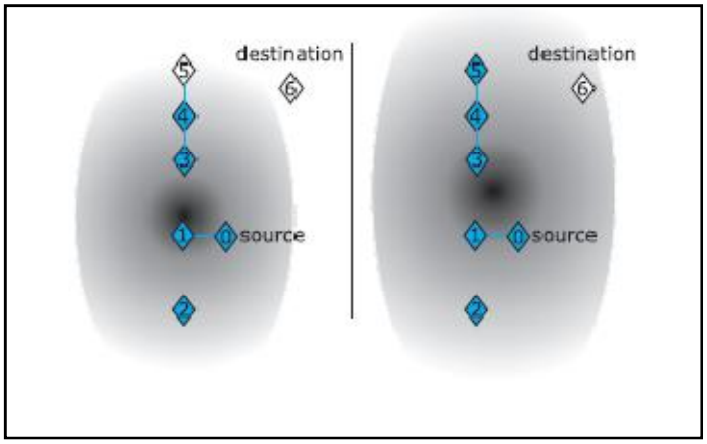

Figure 4: Accumulative Cooperative Transmission, [15]

Sharma et al. [8] proposed a preserving scheme for network connectivity and coverage for randomly deployed sensors. As it is known, WSNs suffer from coverage problem since sensors have limited supplied energy therefore, they have limited sensing range. In their paper, authors focused on maintaining network connectivity with preserved coverage. The approach started with forming the constructed dominating set (*CDS*) which is considered an inward-outward approach that is consisted of high dense nodes forming the virtual backbone for network connectivity. The WSN system can be looked at as inward-outward graph where a graph *G* consisting of vertices *V* as nodes and edges *E* as links, *G (V, E)*. Their algorithm focused at choosing the highest density node and then choosing among the neighbors of this node the highest density node and adding it to virtual backbone and so on. They have simulated their algorithm based on different number of nodes along with different area size, and the results showed that network lifetime had increased since this method had good impact on saving energy by placing the least density nodes to sleep mode. Therefore, connectivity had been maintained where coverage had been preserved. However, their algorithm was affected by the presence of isolated or partitioned network.

Different coverage enhancement schemes in WSNs have been proposed by Dagar and Saroha [25] by using the Delaunay triangulation method and the Grid based coverage. In grid based scheme, the distance between two nodes is given by 2*R where R is sensing range. Whereas the distance measured in Delaunay triangle was computed by Euclidean distance. Results showed that coverage efficiency was maintained using Delaunay triangle. An energy efficiency algorithm has been proposed by Tiwari and Dhoke [1] based on data aggregation in a Cluster head model and Tree based model. They summarized the advantages and disadvantages for these two methods. They stated that both methods aggregate and gather data in energy-efficient manner which maximizes network lifetime. However some issues should be taken into consideration, in cluster head model, number of clusters that form the network should be considered, as well as the cluster head selection procedure and the optimal number of nodes in a cluster. However, for the Tree based model, data loss issue is considered vital since the whole sub-tree under a node that has experienced data loss will also lose the data. Therefore, authors have proposed an adaptive and automated method that will choose which model to use based on given constraints, and which will maximize the lifetime of WSNs. Sakkari and Basavaraju [10] proposed a Multi-hop-Point Enhancing Coverage / Connectivity with Network Lifetime (MP-ECCNL). They focused on maximizing network lifetime by choosing connector nodes with high residual power that will deliver data using a low cost route in a multi-hop fashion to sink. Comparing their results with the original LEACH algorithm, MP-ECCNL had better network lifetime. Network coverage gap has been solved in [14] using the cooperation of mobile robots that are able to move to desired location for network repairing. Idoudi et al. [20] considered cluster based hierarchical scheme and mobile robots able to carry redundant nodes whenever they

are found and use deploy them at position where coverage gap is found. The simulation of this algorithm resulted in network coverage and connectivity improvement by using optimal number of robots.

Nema and Shukla [14] provided a method for network energy saving by considering flat communication network where each sensor node considered a multi-hop fashion for data delivery. They have used mobile anchor nodes which are able to move to specific location in the network where coverage hole exists. Their results had positive impact on overall coverage and connectivity of network as well as maximizing lifetime of network. Tezcan and Wang [2] suggested a distributed algorithm to find an optimal coverage set by eliminating the redundant nodes provided that network connectivity is guaranteed. They extended their study to guarantee network connectivity by finding the minimum number of dominating nodes in the coverage set, where no extra routing protocols will be used as these dominating nodes act as virtual backbone or gateway to carry data to sink. This method provided an effective node scheduling and overall network energy saving. Srivastava and Yadav [6] proposed an improved algorithm to original coverage and connectivity preserving routing protocol (ICCPRP) for the sake to increase network lifetime and providing full connectivity during network operation. They considered heterogeneous nodes having two and three level of heterogeneity, randomly deployed in large area, where an algorithm was performed to determine cluster heads (CHs) based on high density and high energy power. Consequently the non-CH nodes, based on the minimum distance between any CH and themselves, will send join cluster request. As a result, CHs collect data from these non-CH nodes and send the data to sink. This method had shown enhancement in service time as well as 100% sensing coverage ratio has been maintained in comparison with the original CCPRP.

Joshi and Younis [17] presented a novel linear time distributed method to conquer multiple node failure and regain connectivity. They assumed WSN consist of static nodes as well as mobile nodes (MNs) that will help in reconnecting partitioned network. They assumed the field to be modeled as convex polygon for the purpose of finding the minimum path distance that MNs will travel to reconnect with partitioned segments. They named their algorithm Straight Skeleton Construction Based Reconstruction (SSBR). This algorithm allows MNs to move inward along the straight skeleton path of the convex polygon that will be determined once after network initialization. Simulations had resulted in good performance where minimum number of MNs moves along the shortest path to regain connectivity which had good impact on overall network lifetime. Ibrahim et al. [21] have proposed an algorithm that will act as a connectivity precaution by using the Fiedler value [21] that is used as an indicator for the network health and connectivity status. They quantified the network connectivity by the use of Fiedler value, which is the second smallest eigenvalue of the Laplacian matrix representing the network. According to this, the proposed algorithm finds the optimum number of relay nodes that can result in maximum Fiedler value, which will increase the network lifetime. Their results showed an effective increase in Fiedler value by communicating with few relay nodes. In addition, this algorithm showed a less computational time in comparison with other search scheme. However, their algorithm can't guarantee a reconnection of disconnected network by adding relay nodes to disconnected network.

Hole detection technique was inspired by Kleinberg et al. [16] that detected failure in wired network. Their network was modeled as undirected graph of n nodes, and k

edges can be destroyed by an advisory. The failure is detected by some detection agents or sentinels, and they help in pair-wise communication. However, Kelinberg et al method required a large number of detection agents or sentinels in large scale networks, and it suffered from false detections. The idea of using mobile or hybrid sensor nodes was introduced to overcome the coverage problem and have an area covered efficiently [30]. Babaie and Pirahesh [30] used the triangular structure approach for the aim of detecting the location of coverage holes, their sizes, and the movement directions of mobile nodes to repair coverage. They have assumed different coverage hole shape according sensor nodes location, they calculated the coverage hole area for each type of hole structure. After hole calculation, they assure if hole exist based on the area of hole if it is greater than zero. Otherwise, there is no hole in network. Then they allow mobile nodes to move toward the direction of center of either the circumcircle if the area hole is greater than the sensor sensing region or the center of incircle if the area of hole is smaller than the sensor sensing region. The results showed they had exact coverage hole location and size. Another approach to heal coverage hole in WSNS was presented by Sukumaran and Saravanabava [5] where they focused on having a distributed algorithm called QUAD that each node will execute to determine if it is a stuck node or not. They declared that each node should be able to communication with its neighbor in 360°, so each node will execute the QUAD algorithm in each $\pi/4$ partitions, and they assumed a node to be stuck if at least it can't communicate with its neighbor in at least one partition. After detecting a stuck node, this node generates a hole detection packet containing its ID and location and forward it to the boundary node in a right hand direction in which it can receive it back. Using a defined attractive force which will be generated at center of hole , will pull inward nodes toward the center. Their

results came out with effective hole detection and healing process which improved the network coverage. For more hole detection techniques can be found in a survey proposed by Antil and Malik [27]. Sulthana and Ali [3] have proposed a distributed cut detection method (*DCD*) for the purpose of detecting any disconnected portions in wireless sensor networks referred to as "cut". In *DCD* method each sensor node was able to detect its connection status with the source detected by the event arise called disconnected from source (*DOS*) event. Each sensor node keeps track of recent steady state observed which is updated every interval of time. A subset of sensor node is able to experience the connected cut occurred somewhere (*CCOS*) event based on any change of local state and determine the location of cut. The convergence rate of *DCD* method was quite fast and independent of size of network. A low overhead scheme for detecting cuts has been presented by Shrivastava et al. [7], where their method is based on duality transform for detecting linear cut. However, Pimple and Pandey in [13] and Jayashree and Kalaivani in [18], proposed a distributed source separation detection (*DSSD*) method which is not based on linear cut of nodes from source and doesn't require sentinels ; instead, it allows each sensor node to determine if it is connected to a source node , namely sink, by monitoring its local state convergence, as long as there is communication between its neighbors , its local state converges to positive number otherwise it will be zero. *DSSD* method includes only nearest neighbors communication that has good impact on energy efficiency. This algorithm makes use of local state of each node to detect if a re-connection occurs after fixing a failed node.

Different methods have contributed for network recovery. Some of these methods use relay nodes, robot sensors as in [24] [9]. In [24], Dini et al. have presented a low overhead communication method for reconnecting partitioned network by using

mobile nodes. Mobile nodes are sent to the partition once it has been detected. They have two important phases to run while moving in the network. The Monitoring phase allows mobile node to detect its neighbor by checking the communication link with its neighbor. If the communication link is good, it will add this node to its neighboring list. Second phase is the verification phase that will check the connectivity degree with the connected nodes. It will be triggered once the mobile node discovers a decrease in its neighborhood, to check if it has reached an isolated node. Based on the hello message that mobile node will broadcast, it checks the epoch if it is old means reached an isolated node. The mobile node is assumed to carry sensor nodes and able to place them at final position, but this adds complexity to mobile nodes. Simulations have shown both effectiveness in respond to disconnection probability as well as efficiency in terms of communication over head.

1.4 Contributions and Organizations

This thesis has the following contributions:

First, connectivity findings for network with random deployment of homogeneous nodes were tested through simulations. Second, activating the spare or anchor nodes has overcome the problem of isolated nodes found in *CDS*, [8]. Third, activating few anchor nodes at calculated distances has showed 100% connectivity gain of the overall network, in addition to this, analyzing the activation of just minimum number of these few anchor nodes has regained connectivity greater than 90%.

The rest of this thesis is organized as follows. Chapter 2 describes models and methods engaged in this thesis. Chapter 3 describes the proposed CDSA algorithm. Chapter 4 presents different simulations, comparisons, interpretations. Chapter 5 presents the conclusion and future work.

Chapter 2

MODELS AND ASSUMPTIONS

2.1 Assumptions

The work in this thesis will be based on the following assumptions:

Assumption 1: Sensor nodes are homogeneous having same sensing range. We assume having Boolean sensing model where sensing task is not affected by strength of emitted signal [22]. The network architecture presented here is modeled as a unit disk graph *(UDG)*, *G= (V, E)* where *V* represents the sensor nodes and *E* represents the link among these sensor nodes.

Assumption 2: A Base Station knows the exact number of nodes to be spread in LxL m^2 area field plus their position [7]. Base station never fails. All sensor nodes know the sink location and initially a fully connected network was considered [13].

Assumption 3: Assuming the Distributed Source Separation (*DSSD*) is an approach used for detecting existence of isolated nodes in the network [13].

Assumption 4: A time-driven network, where each head of (*CDS*), will send a periodic beacon message containing its status. Since typical epoch durations are in order of seconds or tens of seconds, we assume the epoch length to be 60 seconds [11].

Assumption 5: Assume to have Ad-hoc On Demand Distance Vector (*AODV*) method for routing messages between sensor nodes and sink. This method based on path discovery and maintenance concepts. It allows sensor nodes to discover routes for message delivery. It is known to be one of the most efficient routing protocols due to shortest path discovery characteristic and its lowest power consumption [19].

Assumption 6: Assume to have 10% of the randomly deployed sensor nodes used as spare nodes or anchor nodes that will be activated whenever necessary.

2.1.1 Graph Theory Background

First, a graph is defined as pair $G = (V, E)$ of sets where the elements of V are the vertices (or nodes, or points) of the graph, and the elements of E are its edges. An edge is usually constructed between neighbor vertices that fall under same communication range. A vertex with no neighbors is said to be isolated. A graph is said to be directed if the edges connecting the vertices are one-way; otherwise, the graph is undirected.

A connected dominating set $CDS(G)$ of a graph $G=(V, E)$, is defined as a subset of $V(G)$ where there exists at least one node in $CDS(G)$ which is adjacent to a node in $V(G)$-$CDS(G)$ [8]. *CDS* forms the essential network backbone.

2.2 System Model

We assume having Boolean sensing model, where a node can detect an event if it occurs within its sensing range [22]. This model doesn't take into consideration the environmental effect on the emitted signal strength. System architecture is required to meet the needs of WSNs and to cope with the most difficult constraint which is the power limitation. To make WSN feasible, network architecture should be developed to synthesize the environmental data collection application out of the underlying hardware capabilities. The network architecture presented here is modeled as a unit

disk graph *(UDG), G= (V, E)* where *V* represents the sensor nodes and *E* represents the link among these sensor nodes. Usually *UDG* is used to model network connectivity where nodes discover network topology and estimate the routing strategy as a mean for routing data to collection of points.

Our study focuses on flat-based network structure illustrated in the construction of connected dominating set *(CDS)* which is known to have positive impact on maximizing network lifetime by using high density connectors as backbone infrastructure and multi-path routing protocol for delivering events or sensed data between sensor nodes. Sensor nodes send their events or sensed data to source or sink through their corresponding set head (heads of *CDS*). Using multi-hop routing technique *CDS* heads deliver any info to source too.

2.2.1 Distributed Source Separation Detection Protocol

Distributed Source Separation (*DSSD*) protocol is a distributed approach which allows each sensor node to determine its local status, that is, either separated from source node or still connected. Therefore, this protocol enables each node to monitor its status for isolation detection [13].

In our work, we have assumed the running of *DSSD* protocol. After creating the *CDS* for the sensor network, each *CDS* head will have to detect its scalar status by running the *DSSD* approach and check if it is still connected to the sink or has become isolated. If any set head can't reach the sink, it experiences a Disconnected frOm Source (*DOS*) event [3] [7].

If set head can reach the sink but connected to an isolated *CDS*, it experiences a Connected but a Cut Occurs in the Set (*CCOS*) [3] [7].

This info will be collected at the sink, knowing the location of partitioned or isolated region. At this point, the BS will have to run our proposed algorithm for anchor node activation at the calculated distances.

Chapter 3

THE PROPOSED CONNECTED DOMINATING SET WITH ANCHOR NODES ACTIVATION SCHEME

3.1 Methodology

Our work highlights an essential problem that faces wireless sensor networks especially in case of randomly deployed sensor nodes. Two important issues, one is related to random deployment of nodes which might result in having isolated nodes and unconnected network partitions resulting in network topology destruction. The second issue is related to the restricted battery supply that sensor nodes are encapsulated with, which will degrade with time, causing the sensor node to die and losing network connectivity . Therefore, the above two issues have direct negative impact on network topology and as a result, unconnected partitions will exist resulting in unsatisfied network performance. We have proposed a Connecting Dominating Set with Anchor node activation (*CDSA*) scheme that focuses on the construction of connected dominating set (*CDS*) used in [8] where Sharma et al. have used it for better nodes communication and power supply saving. The *CDS* algorithm used in [8] forms a virtual backbone, consisting of high density nodes that are responsible for data forwarding to the sink. The *CDS* algorithm has proven to be efficient in network energy saving and has a positive impact on network lifetime. However, the *CDS* algorithm suffers from communication deficiency in the presence of isolated nodes, causing the network connectivity to be lost and communication between *CDS* nodes and their neighbors will be missing. Therefore, besides the *CDS*

algorithm that has been used in [8], we have adopted the idea of anchor node deployment used by Nema and Shukla [14]. Anchor nodes used in [14] are mobile nodes in which they are able to move to the desired location for reconnecting purposes. However, in our proposed scheme, we have reduced the complexity of using mobile nodes and considered the activation of static anchor nodes or spare nodes illustrated in Figure 5 and algorithm steps.

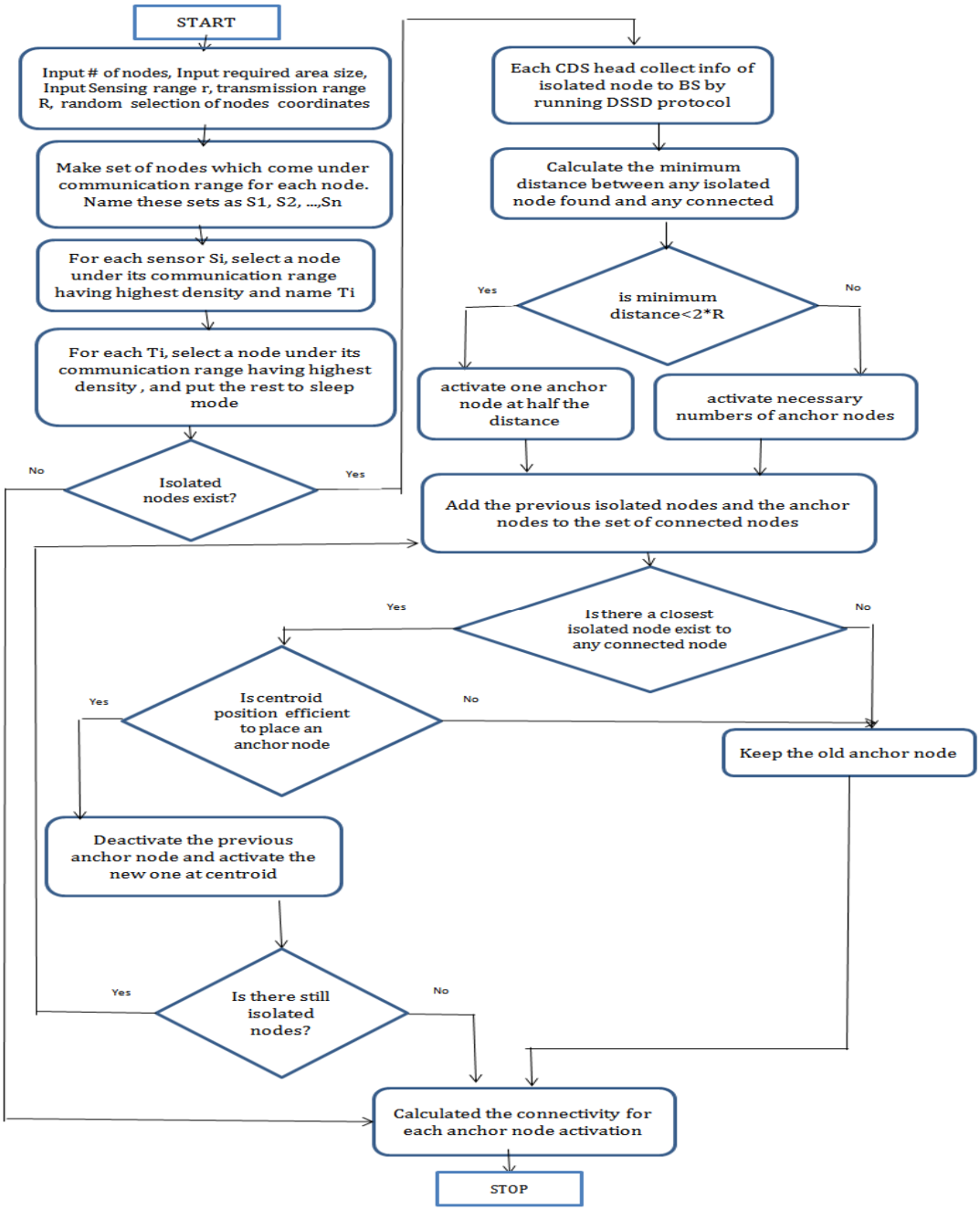

Figure 5: *CDSA* Flowchart

The position of anchor nodes is very important to ensure the communication among respective nodes is maintained. Therefore, we have first considered the minimum distance between any isolated node and connected node. Note that, a connected node is a node having a path to the sink. The minimum distance between any two points is the midpoint; therefore, we have activated the first anchor node at the half of the minimum distance after checking the distance between the midpoint and the isolated node, connected node falls under the communication range. If so, anchor node will be activated at half the minimum distance. However if the distance between midpoint and the isolated node, connected node doesn't fall under communication range, then one anchor node is not sufficient to ensure communication, therefore, necessary number of anchor nodes will be activated at equal distances between isolated node and connected node. After the first activation, the isolated node and anchor node will be added to list of connected nodes. Then, we suggested to find a closest isolated node among the isolated node list if exist, to the previous isolated node, which is now added to list of connected nodes. First we check if the distance between the new isolated node, old isolated node (which is now connected node) falls under the communication range then we consider the position of centroid point of the three nodes: new isolated node, old isolated node which is now connected node, and the previous close node to old isolated node. Also, we check if the distances from centroid position to the three nodes fall under the communication range, then we activate the anchor node at the centroid to ensure communication. If this is the case, we make sure to deactivate the previous anchor node which was activated at half the distance and also delete it from the connected node list. As before, the anchor node and new isolated node will be added to list of connected nodes. Our algorithm continues till no isolated nodes exist in the network.

Explanation of the *CDSA* algorithm is found in the following steps:

Step 1: Inputting the necessary data (number of nodes, sensing range, transmission range, area field, sink location which is in center).

Step 2: Starting from the sink, we find all the nodes that fall under the communication range, and mark them as zero and store them in an array. Then iterating over each neighbor of the sink to find all the nodes that fall under communication range for each neighbor node and store them in same array, and mark all of them zero. Then starting from a non-zero node if available, and continue as before to find its neighbors and mark them as zero, and so on. When finishing from this connected set, store it in another block of same array. We continue creating connected sets till all nodes marked zero. Then, we draw communication line between nodes that are within communication range for each connected set. As a note, having only one block of the array means having one full connected network.

Step 3: We need to create the Connected Dominating Set that consist of nodes having the highest density in each connected set. Therefore, first in each connected set, we need to find the density for each node and sort them in descending order. The first node having highest density will be placed in array called core, and will be the set head. After finding the first set head, we mark it zero as well as its neighbors. Then we select among its neighbor nodes, the one with highest density and store it in core node. Then, we keep selecting the highest density non-zero neighbor and store it in core set and we mark the corresponding neighbors of each core node as zero. This *CDS* algorithm is inward-outward method. We continue in finding the core nodes in each connected set till all nodes are marked zero in each connected set.

Step 4: We find the isolated nodes which are the nodes that are not connected to the sink. For instance, nodes connected to sink are first block of array S{1}. We can now calculate the connectivity of the available network before any anchor node activation.

Step 5: This step is related to connecting isolated nodes with anchor nodes.

First, after knowing each isolated node, we need to find the minimum distance between any isolated node and any connected node with the sink. After finding the minimum distance between isolated node and connected node, we calculate if this distance is less than twice the transmission range. If yes one anchor node is sufficient to be activated at half the minimum distance and to restore the connection of this isolated node. If no then we need to activate the necessary anchor nodes number. Anchor nodes and the isolated nodes that have become connected are to be added to connected node list. After activation of first anchor node (s), then we start iteration, by finding the closest isolated node to previous isolated node that have become connected, and we check if the distance is less than twice the transmission range. If this is the case, we need to make sure if the activation of the anchor node position will be at the centroid (connected node, isolated node that have become connected and the closest isolated node to the previous isolated node), by checking the distance from centroid to each node if it is less than the transmission range, if this is the case then we need to deactivate the first anchor node and place it at the centroid. If not then we keep the first anchor node position at half the minimum distance. We continue with the algorithm till no isolated node exists.

Step 6: We finally calculate the percentage of connectivity after nodes activation. Having optimum number of anchor nodes, high connectivity has been achieved.

Step 7: We have compared our proposed algorithm with different connectivity enhancement techniques as well as with the SDP algorithm which is based on Fiedler value that represents the connectivity.

A demonstration of *CDSA* algorithm is shown in Figure 6.

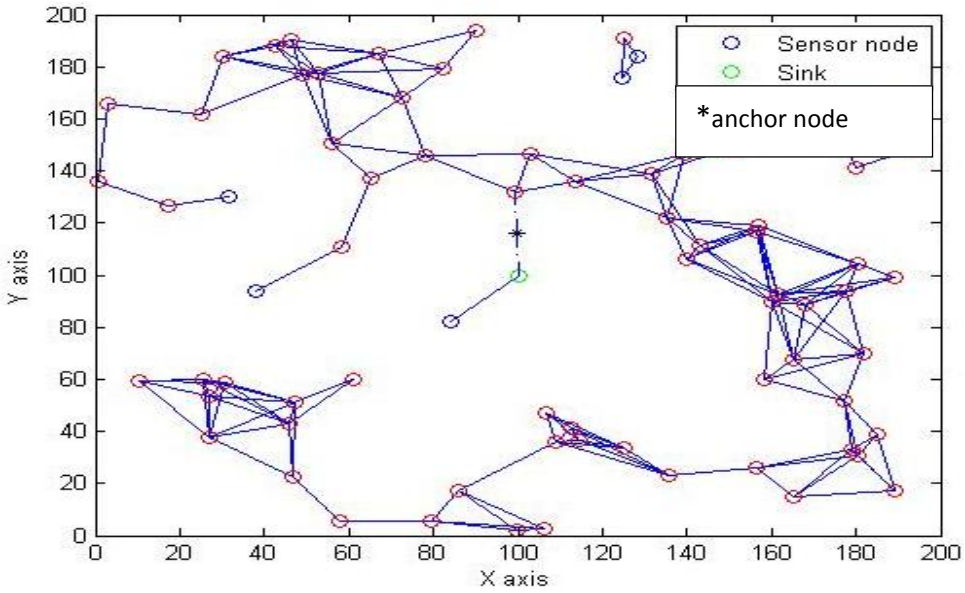

Figure 6: Demonstration of *CDSA* Algorithm

The Fiedler value refers to the algebraic connectivity of graph [23]. Fiedler value refers to the second smallest eigenvalue of matrix representing the network connectivity. Fiedler value measures the network connectivity status. When the network is fully connected the Fiedler value is greater than zero; however, when the network becomes disconnected the Fiedler value is zero. Therefore, based on the Fiedler value of the randomly deployed WSN, we can monitor the status of the network connectivity. Having a positive Fiedler value ensures a connected network which is desired in WSN. More detail on Fielder value is found in Appendix C.

Chapter 4

PERFORMANCE EVALUATION OF CDSA SCHEME

4.1 Simulation Model

Analyzing connectivity of homogeneous WSNs is done through simulations. We have assumed a homogeneous WSN and the simulations are written in MATLAB.

We assume having different number of nodes (25, 50, 75, and 100) placed randomly in LmxLm area field (100mx100m, 200mx200m, and 500mx500m). We have run the proposed algorithm 50 times for each case and calculate the number of isolated nodes and percentage of connectivity by taking the average.

The simulation parameters that were used to obtain the results are listed in Table 1 and Table 2.

Table 1: Simulation parameters in an area 100mx100m

Sink position	(50m, 50m)
Number of nodes	25, 50, 75, 100
Sensing Range	10m
Transmission Range	20m

Table 2: Simulation parameters in an area 200mx200m

Sink position	(100m,100m)
Number of nodes	25, 50, 75, 100
Sensing Range	10m
Transmission Range	20m

Table 3: Simulation parameters in an area 500mx500m

Sink position	(250m,250m)
Number of nodes	25, 50, 75, 100
Sensing Range	10m
Transmission Range	20m

4.2 Results and Interpretation

We have simulated the proposed algorithm based on different network parameters. First, we have considered the above parameters in Table 1, Table 2, and Table 3 for 50 runs and took the average or mean. Connectivity percentage increases with the increases of sensor nodes and decreases when field area to be sensed is getting larger. The respective results are shown in Table 4, Table 5, and Table 6. Figure 7, Figure 8, and Figure 9 illustrate the respectively Table 4, Table 5, and Table 6. For detailed explanation of required number of runs can be found in Appendix B. Also, we have analyzed the change in number of deployed nodes with respect to fixed area size and vice versa. Also, we have analyzed the change in connectivity % for each number of sensors at fixed area size as shown in Figure 14, Figure 15, and Figure 16 and the

change in connectivity % for different area size for fixed number of sensors as shown in Figure 10, Figure 11, and Figure 12.

Table 4: Simulation Results of connectivity % for different N deployed in an area 100mx100m

	Connectivity % for			
Anchor nodes	N=25	N=50	N=75	N=100
0	96.46	98.35	98.59	98.90
1	100	100	100	100
2	100	100	100	100
3	100	100	100	100

Table 5: Simulation results of connectivity % for different N deployed in an area 200mx200m

	Connectivity % for			
Anchor nodes	N=25	N=50	N=75	N=100
0	83.15	88.76	92.39	97.23
1	96.30	97.23	98.38	100
2	99.78	100	100	100
3	100	100	100	100

Table 6: Simulation results of connectivity % for different N deployed in an area 500mx500m

	Connectivity % for			
Anchor nodes	N=25	N=50	N=75	N=100
0	70.25	80.82	86.45	91.22
1	85.81	92.28	95.21	96.61
2	91.23	95.98	98.48	99.28
3	93.89	97.57	99.70	99.98
4	97.71	97.99	100	100
5	100	100	100	100

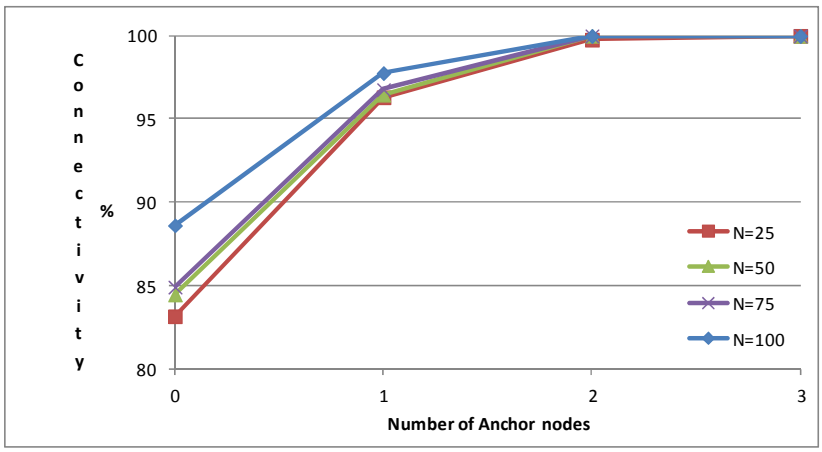

Figure 7: Illustration of Table 4

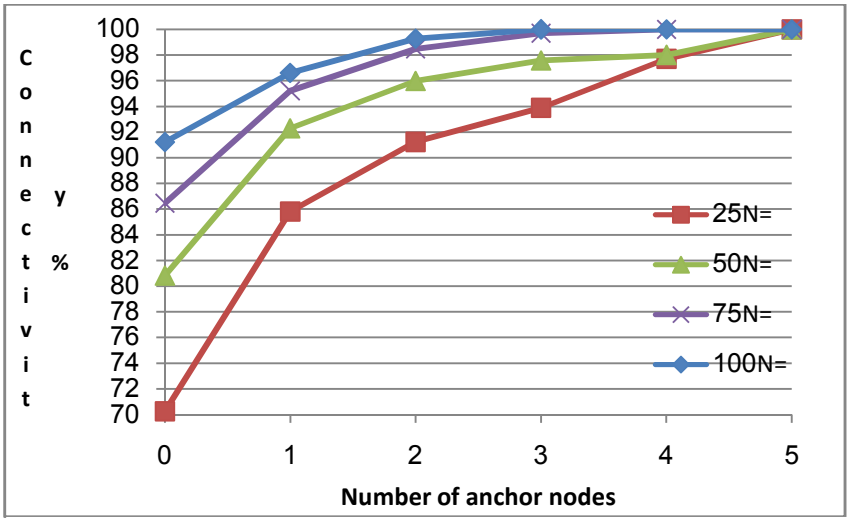

Figure 8: Illustration of Table 5

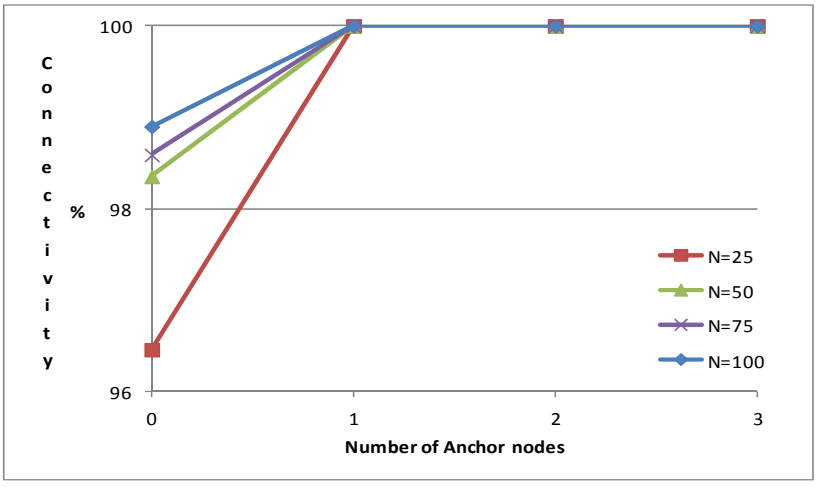

Figure 9: Illustration of Table 6

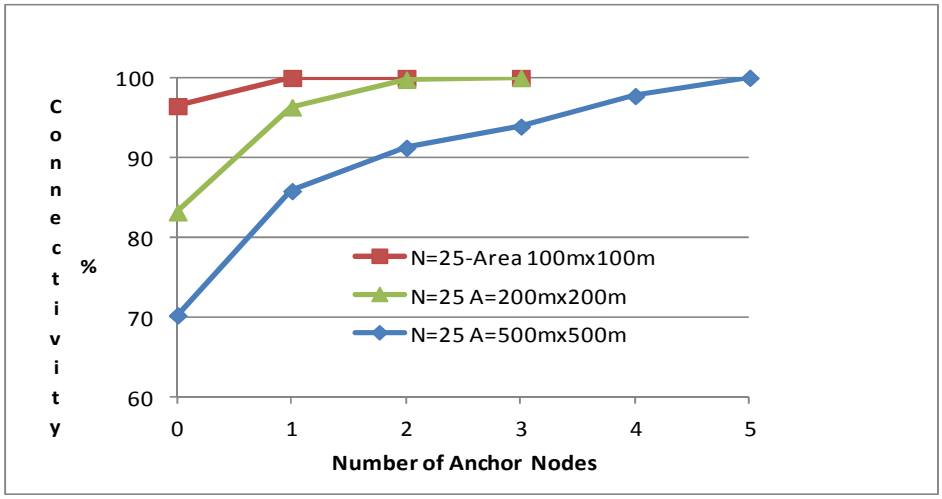

Figure 10: Simulation of connectivity % results for N=25 at different area size

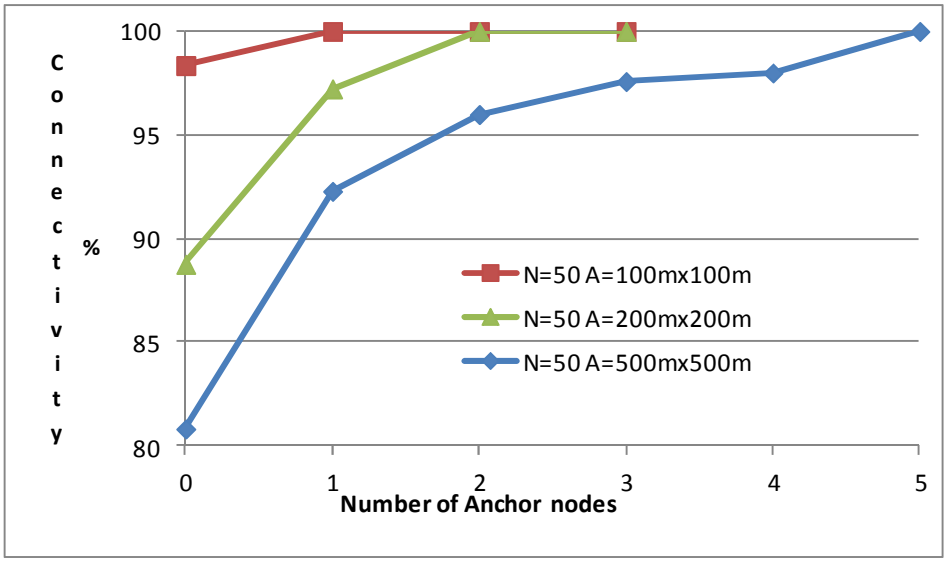

Figure 11: Simulation of connectivity % results for N=50 at different area size

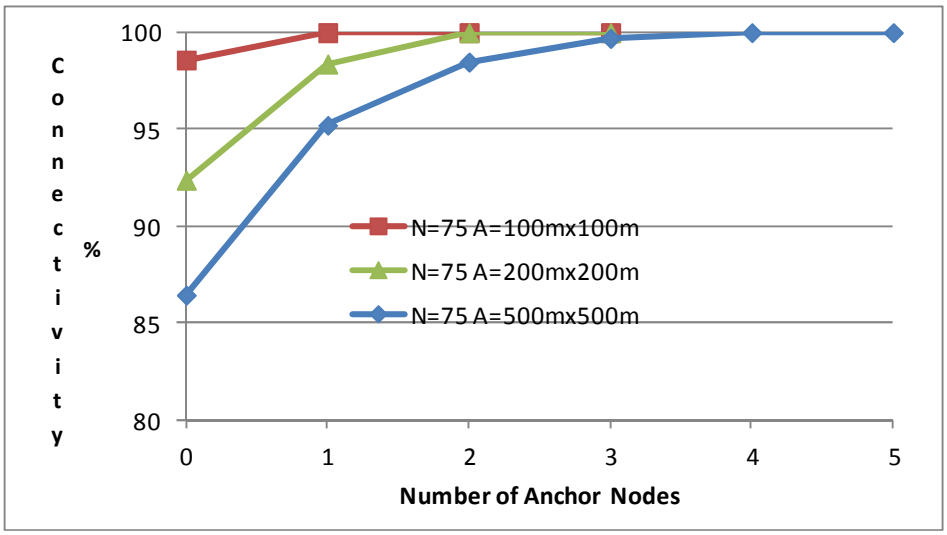

Figure 12: Simulation of connectivity % results for N=75 at different area size

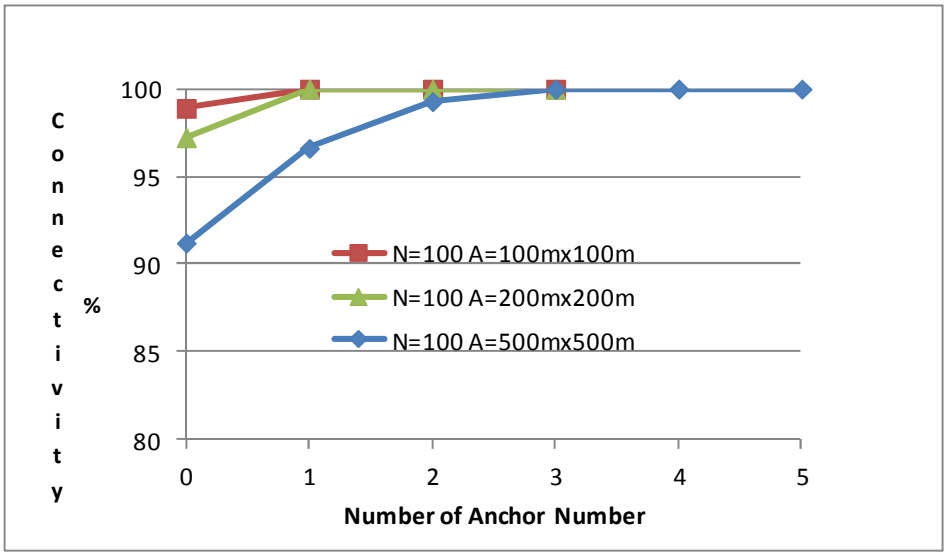

Figure 13: Simulation of connectivity % results for N=100 at different area size

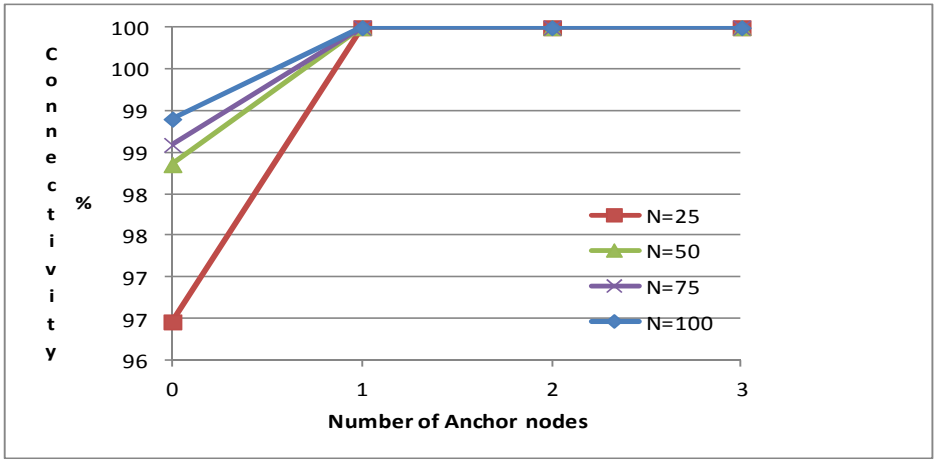

Figure 14: Simulation of connectivity % results for different N at area 100mx100m

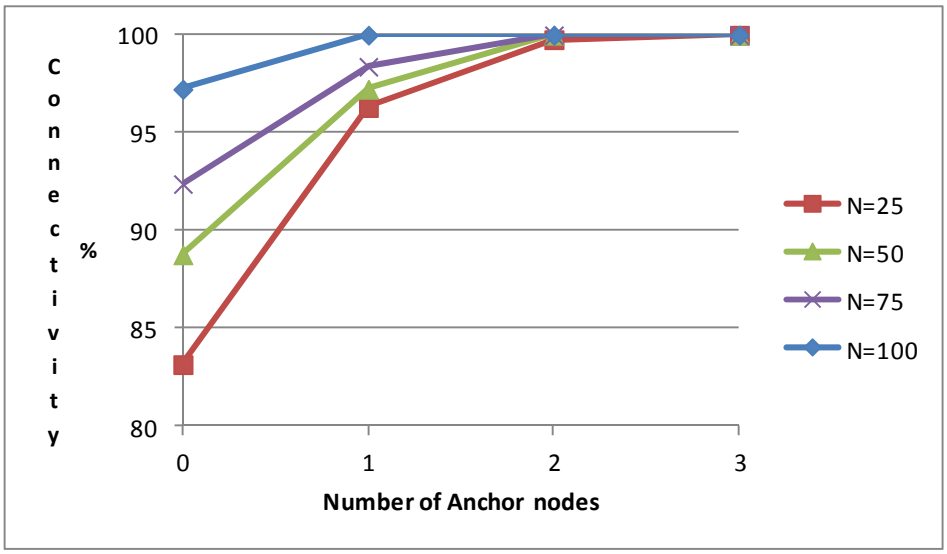

Figure 15: Simulation of connectivity % results for different N at area 200mx200m

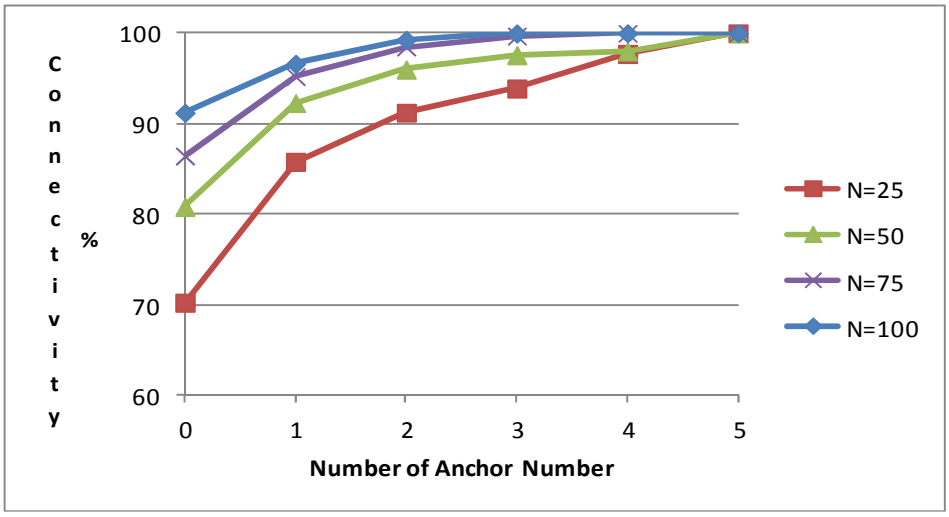

Figure 16: Simulation of connectivity % results for different N at area 500mx500m

As it is shown in Table 5, for 25 and 50 sensor nodes, it is needed to have one anchor node activated to achieve high network connectivity (above 90%). However, to have full network connectivity, 2 anchor nodes are needed to be activated at area 200mx200m and one anchor node needed to be activated at area 100mx100m.

We have also simulated our proposed algorithm for different number of sensor nodes as shown in Table 3. According to results, 2 anchor nodes were needed to be activated for 25 sensors, 1 anchor node for 50 and 75 sensor nodes to have high network connectivity. Results are shown in Table 6. Figure 10, Figure 11, Figure 12, and Figure 13 show that as area size increases, number of anchor nodes needed to be activated to obtain high connectivity % was required. While fixing area size and changing the number of nodes as in Figure 14, Figure 15, and Figure 16 show that as number of nodes increases less anchor nodes were required to be activated to obtain high connectivity %.

4.2.1 Complexity Analysis

Our proposed algorithm is based on distance matrix. We need to find the Euclidean distance between nodes which is fundamental factor for calculating the neighbor density and constructing the CDS method. Since we have N number of nodes, the maximum number of links is N(N-1)/2, Hence if all nodes are connected, there will be maximum of NC^2 links where C is the combination, $NC^2 = N(N-1)/2$. Therefore, our algorithm has complexity of $O(N^2)$. The complexity analysis is validated by Liu et al. [31].

4.2.2 Comparison of CDSA with Different Algorithms

Comparing our proposed CDSA algorithm which is based on using both CDS and anchor node activation with the proposed algorithms for connectivity enhancement in paper [15], our algorithm has proved to be more efficient in terms of having optimum number of nodes needed to have high connectivity; whereas, having high connectivity has been achieved even with low number of nodes as shown in Table 6 where at node number 10, connectivity was 90%, however in other algorithm, high connectivity has been achieved when the number of nodes were 170 for multi-hop , 140 for hybrid 1, and 100 for hybrid 2. Table 7 and Figure 17 show the conducted results of CDS algorithm, CDSA, Multi-hop, Hybrid1 and Hybrid 2 algorithms respectively. It is clear that hybrid 2 algorithm reveals high connectivity (90%) when number of nodes was 100 while the multi-hop and hybrid 1 show less connectivity percentage. However, our proposed algorithm revealed high connectivity (90%) when number of nodes was 10.

Table 7: Connectivity % for different connectivity enhancement algorithms

Area 500mx500m, Sensing range= 25, Transmission range=50. CDS analysis						
Number of Nodes	**CDS% Connectivity [15]**	**Proposed CDSA % Connectivity**	**Multihop % Connectivity [15]**	**Hybrid_1 % Connectivity [15]**	**Hybrid_2 % Connectivity [15]**	**Optimum number of anchor nodes**
10	9.0909	90.1961	2.5	2.5	5	2
20	12.313	90.2439	5	9	9	1
30	15.7557	90.3226	6	11	13	1
40	17.6471	90.4762	6	17	25	0
50	18.1818	90.9091	6	25	38	0
60	27.7152	91.2088	11	38	58	0
70	29.0909	91.358	15	48	72	0
80	34.5679	91.5493	18	58	78	0
90	38.2979	91.6012	28	65	85	0
100	41.7582	93.0693	38	72	90	0
110	62.6718	94.8177	48	78	93	0
120	63.64	96.0397	60	82	95	0
130	71.7557	98.0127	70	86	97	0
140	81.1527	98.082	80	90	97	0
150	85.214	98.3202	81	91	97	0
160	89.313	98.6597	85	92	98	0
170	92.7152	98.7099	88	94	98	0
180	94.4751	99.4733	95	95	98	0
190	98.5075	100	98	96	99	0
200	98.5075	100	98	97	99	0

The results of connectivity percentage in Table 7 of the proposed CDSA algorithm have been selected based on activating the optimum number of anchor nodes that have resulted in high network connectivity (>=90%). Representation of the results found in Table 7 is shown in Figure 17.

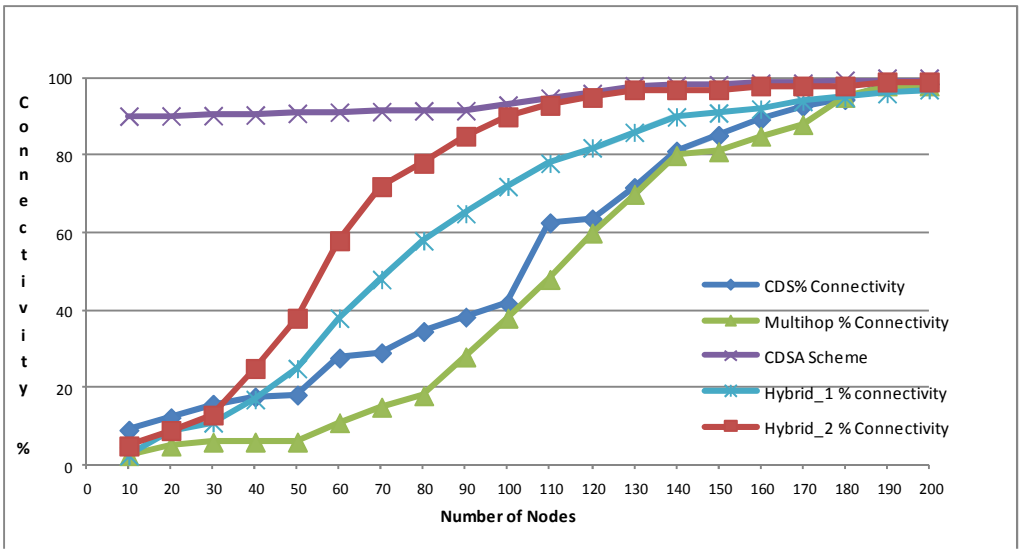

Figure 17: Connectivity % versus number of nodes for different algorithms

4.2.3 Comparison with the semi-definite programming (SDP) optimization problem and random addition algorithm

Ibrahim et al. [21] have proposed a method for connectivity enhancement based on quantifying the network connectivity using the Fiedler value [23]. By definition, the Fiedler value is the algebraic connectivity of a graph [23]. It is considered to be the second smallest eigenvalue of Laplacian matrix whose off-diagonal entries values {0, 1, -1}. Fiedler value is known to be a network health indicator; whenever the Fiedler value greater than zero indicates a connected network and vice versa. Network lifetime is defined as the time needed for the network to become disconnected [21]. Therefore, a direct relation occurs between keeping the network operating for maximum time and maximizing the network lifetime. As stated before, Fiedler value is considered to be the measure of network connectivity. Therefore, Fiedler value is considered to be a measure for network lifetime as well. More details regarding Fiedler value is found in Appendix C. We have taken the average of Fiedler value for 50 runs, for random deployment of number of relays and for the CDS algorithm, then

we compared our results with the ones found in paper [21]. Results are shown in Table 8 and in Figure 18.

Table 8: Fiedler value results

N=100, Area = 10x10, Transmission range is 3		
Taking the average value for 50 runs		
Anchor Nodes	Fiedler Value for CDSA algorithm	Fiedler Value for CDS algorithm
0	2.00	2.00
1	2.58	2.45
2	3.03	2.68
3	3.31	3.01
4	3.56	3.22
5	4.03	3.50
6	4.21	3.72

Comparison of the results in Table 8 with the ones in [21] is shown in Figure 18.

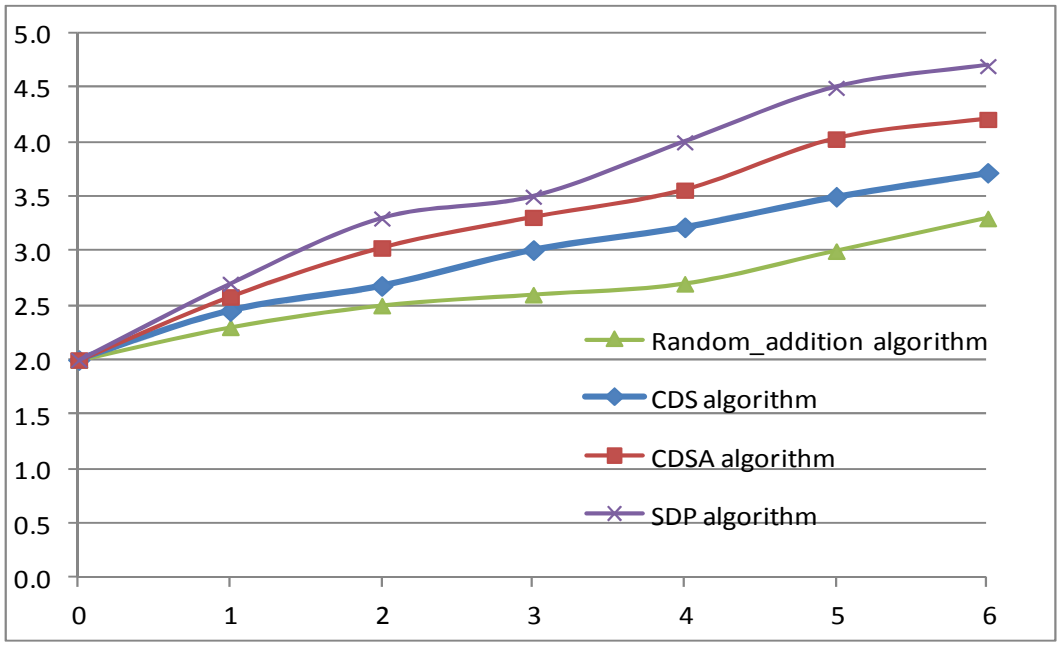

Figure 18: Fiedler value versus number of relays for CDSA scheme and algorithms in [21].

As it is shown, the SDP algorithm has shown better Fiedler values due to many aspects related to uniform distributions of sensor nodes, in addition of using the semi-definite programming package for finding best relays locations. However, our algorithm is based on random deployment of sensor nodes as well as random addition of relays. It is shown that our proposed algorithm has shown better Fiedler value than the CDS algorithm and the random addition algorithm proposed in [21].

The Fiedler value enhancement for CDS algorithm was found to be approximately 20% that is better than the Fiedler value enhancement for random addition algorithm which was found to be 15%, due to the fact that the construction of CDS maximizes the network lifetime and therefore will have higher Fiedler value than the random addition. The Fiedler value enhancement for SDP algorithm was found 35 %; however, the difference isn't too much between Fiedler value enhancement between our algorithm which is found to be 29% and the SDP algorithm which was found 35%. On the other hand, the Fiedler value enhancement for our algorithm is approximately twice the one for Random Addition algorithm which is found to be 15%.

Chapter 5

CONCLUSION AND FUTURE WORK

We have proposed an algorithm to solve the problem of isolated nodes found in randomly deployed wireless sensor networks. We have proposed a CDSA algorithm to solve the problem of isolated nodes found in randomly deployed wireless sensor networks. In this thesis, we have improved the connectivity percentage and network performance for randomly deployed WSN. This algorithm has showed efficiency in terms of connectivity percentage that has reached its maximum by activating few anchor nodes at calculated distances.

The CDSA algorithm has outperformed different connectivity enhancement algorithms. CDSA has reached high connectivity with low number of nodes, at node number 10, connectivity was 90%, however in other algorithm, high connectivity has been achieved when the number of nodes was 170 for multi-hop, 140 for hybrid 1, and 100 for hybrid 2. CDSA algorithm proved to have approximately twice the Fiedler value enhancement (29%) than Random Addition of relays algorithm (15%).

We have simulated our proposed algorithm for small area size of 500mx500m and our Future work will evaluate the network connectivity percentage for randomly deployed sensors in larger area size. We also have assumed having the base station a prior knowledge of sensor nodes locations. In our future work, we will consider a localization algorithm for detecting sensor nodes positions. In addition to evaluating network performance for randomly deployed heterogeneous nodes as well. Finally, a

spanning tree construction algorithm can be used in future work to find the best anchor nodes position for repairing network connectivity purposes in the presence of isolated nodes.

REFERENCES

[1] Tiwari, G., & Dhoke, A. (2015). Adaptive Data Aggregation Method for WSN, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), vol. 5, no.1, 23-24

[2] Tezcan, N., & Wang, W. (2007). Effective Coverage and Connectivity Preserving in Wireless Sensor Network. *IEEE Wireless Communications and Networking Conference(WCNC*), 3388 - 3393.

[3] Sulthana, M., & Ali, S. (2013). A Distributed Cut Detection Method for Wireless Sensor Networks. *International Journal of Modern Engineering Research (IJMER),* vol. 3, no. 4, 2510-2513.

[4] Sultan, A., Merabti, M., Askwith B., & Kifayat, K. (2007). Network Connectivity in Wireless Sensor Network: A Survey. *24th IEEE Instrumentation & Measurement Technology Conferance (IMTC)*.

[5] Sukumaran, V., & Saravanabava, T. P. (2014). Modified Sensor Deployment Algorithm for Hole Detection and Healing Using NS2. *International Journal of Engineering Research and Applications,* vol. 4, no. 4, 43-50.

[6] Srivastava R., & Yadav, A. K. An Efficient Routing Protocol in Heterogeneous Wireless Sensor Network for Coverage & Connectivity Preserving. *International Journal of Communication and Computer Technologies,* vol. 2,

no. 7.

[7] Shrivastava, N., Suri, S., & Toth, C. (2005). Detecting Cuts in Sensor Networks.

[8] Sharma, L., Singh, J., & Agnihotri, S. (2012). Connectivity & Coverage Preserving Schemes for Surveillance Applications in WSN. *Internal Jounal of Computer Applications,* vol. 50.

[9] Shalini, S., Ramalakshmi, K., & Angelin, P. (2013). A Survey on Partition and Recovery Methods of Node Failure in Wireless Sensor Networks. *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 3, no. 12, 932-936.

[10] Sakkari, D. S., & Basavaraju, T. G. (2013). Optimized Coverage & Connectivity for Randomly Deployed Wireless Sensor Network for Lifetime Conservatory. *International Journal of Computer Engineering & Technology (IJCET),* vol. 4, no. 6, 247-255.

[11] Romer, K. (2008). Discovery of Frequent Distributed Event Patterns in Sensor Networks. *5th European Conference, EWSN*, Bologna, Italy.

[12] Ritter, H., Winter R., & Schiller, J. (2004). A Partition Detection System for Mobile Ad-Hoc Networks.

[13] Pimple, J., & Pandey, Y. (2012). Cut Detection in Wireless Sensor Network Using Distributed Source Separation Detection (DSSD) Approach. *International Journal of Scientific and Research Publications,* vol. 2, no. 12.

[14] Nema, S., & Shukla, N. (2014). Calculation on Converage & Connectivity of Random Deployed Wireless Sensor Network Factors Using Heterogeneous Node. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET),* vol. 3, no. 1.

[15] Krohn, A., Beigl, M., Decker, C., Riedel, T., Zimmer, T., & Varona, D. (2006). Increasing Connectivity in Wireless Sensor Network Using Cooperative Transmission.

[16] Kleinberg, J., Sandler, M., & Slivkins, A. (2004). Network Failure Detection and Graph Connectivity. *5th Annual ACM-SIAM*, New Orleans, Louisiana.

[17] Joshi, Y. K., & Younis, M. (2014). Straight Skeleton Based Reconnection in a Wireless Sensor Network. *An Ad Hoc and Sensor Symposium, Globecom.*

[18] Jayashree, R., & Kalaivani, R. (2012). An Algorithm To Detect Separation And Reconnecting Wireless Sensor Network Partitions. *International Journal of Engineering Research & Technology (IJERT),* vol. 1, no. 10.

[19] Jain, R., & Kshirsagar, M. (2012). Study of Different Communication Protocols for Wireless Sensor Networks. *National Conference on Innovative Paradigms in*

Engineering & Technology (NCIPET).

[20] Idoudi, H., Houaidia, C., Saidane, L. A., & Minet, P. (2012). Robots-Assisted Redeployment in Wireless Sensor Networks. *Journal of Network Technology,* vol. 3.

[21] Ibrahim, A., Seddik, K., & Liu, K. (2007). Improving Connectivity via Relays Deployment in Wireless Sensor Networks. *Global Telecommunications Conference*.

[22] Hossain, A., & Mishra, R. (2013). Sensing & Link Model for Wireless Sensor Network: Coverage & Connectivity Analysis. *2nd National Conference EAPE.*

[23] Fiedler, M. (1973). Algebraic Connectivity of Graphs. *Czechoslovac Mathematical Journal.*

[24] Dini, G., Pelagatti, M., & Savino, I. (2008). An Algorithm for Reconnecting Sensor Network Partitions. *EWSN,* 253-267.

[25] Dagar, A., & Saroha, V. (2013). An Efficient Coverage Scheme for Wireless Sensor Network. *International Journal of Advanced Research in Computer Science & Software Engineering,* vol. 3, no. 4.

[26] Banks, J., Carson, J., Nelson, B., & Nicol, D. (2001). Discrete Event System

Simulation. *Prentice Hall, Internal Series in Industrial And System Engineering*.

[27] Antil, P., & Malik, A. (2014). Hole Detection for Quantifying Connectivity in Wireless Sensor Networks: A Survey.

[28] Al-Karaki, J. N., & Kamal, A. E. (2004). Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wireless Communication.*

[29] http://research.ee.port.ac.uk/index.php?page=mobility-optimisation-of-wireless-sensor-networks

[30] Babaie, S., & Pirahesh, S. (2012). Hole Detection for Increasing Coverage in Wireless Sensor Network Using Triangular Structure. *International Journal of Computer Science,* vol. 9, no. 1.

[31] Liu, Z., Wang, B., & Guo, L. (2010). A Survey on Connected Dominating Set Construction Algorithm for Wireless Sensor Network. *Information Technology Journal,* vol. 9.

APPENDICES

Appendix A: Program Code

Appendix A.1: Connectivity % Calculation for CDSA Algorithm

Program Code: wsnsimulation.m

```
clc
clear all
NumberOfNodes=100
n=NumberOfNodes+1      % The number of nodes
L=10       % Dimension of the field
srange=1.5 % sensing range
txrange=3  %transmission range

coord = L*rand(n,2);  % Randomly select the coordinates of the location of n nodes.
coord(1,:) = [L/2, L/2];


for i=1:n
   for j=1:n
      dist(i,j)= sqrt(sum((coord(i,:)-coord(j,:)).^2)); %Finding the euclidean distance
between node i and node j
      dist(j,i)=dist(i,j);% radio channel is reciprocol
   end
end
% dist

count=1;
temp=1:n;

% currentNode=findNewNode(temp, n)
%  temp(currentNode)=0;


%  S{count}(1)=currentNode   % each element in the array S keeps one set
%  of connected nodes.nodes of Si can't communicate with nodes in
%  another set Sj. Splitting the nodes into arrays S{1},S{2},...
while findNewNode(temp, n)<=n
   nextNodeInSet=1;
   S{count}(1)=findNewNode(temp, n);%that is selecting the first node in Si, the
output of findNewNode() is always 1 which is sink
   while length(S{count})>=(nextNodeInSet)
      for k=1:n
         if dist(S{count}(nextNodeInSet), k) < txrange % current node and node k are
in communication range
            if temp(k) ~= 0 %from the set temp, we find the node that is not already
selected in to another set in S,the nodes that are already selected are marked as 0 in
temp
```

```
                S{count}=[S{count} k];%this operation appends the element k at the
end of the vector S{count}
                temp(k)=0; % once a new node is found that is in the transmission range
of one node in the set, that node has to be added to the set
            end
          end
       end
       nextNodeInSet=nextNodeInSet+1;
   end
   S{count}(1)=[];% there was a problem in the loop. The first element was counted
twice.
   %So I had to remove the first element so that the first element is there only once
   count=count+1;
end

S;


figure

plot(coord(:,1), coord(:,2), 'bo') % plot the nodes
hold on
plot(coord(1,1), coord(1,2), 'bo', 'Color', 'g') % plot the sink

legend('Sensor Node', 'Sink')

figure

plot(coord(:,1), coord(:,2), 'bo') % plot the nodes
hold on
plot(coord(1,1), coord(1,2), 'bo', 'Color', 'g') % plot the sink
% draw lines between nodes who are in same communication range
for count=1:length(S)
   for i=1:length(S{count})
      for j=i:length(S{count})
         if dist(S{count}(i),S{count}(j)) < txrange
            plot([coord(S{count}(i), 1) coord(S{count}(j), 1)], [coord(S{count}(i), 2)
coord(S{count}(j), 2)]);
         end
      end
   end
end

% Find the density  of each nodes which is stored in array Sd

density = zeros(1,n);% first the density vector is initialized as zero for all nodes
for count=1:length(S)
   Sd{count}=zeros(1, length(S{count}));% create a vector Sd whose length is same
as S{count} , initially zero
   for i=1:length(S{count})
```

```
      for j=i+1:length(S{count})
         if dist(S{count}(i),S{count}(j)) < txrange %then I find each neighbor for
each node and increment the corresponding
            %value of density vector
           % density(S{count}(i)) = density(S{count}(i))+1;
            %density(S{count}(j)) = density(S{count}(j))+1;
            %Sd{count}(i)=Sd{count}(i)+1;
            Sd{count}(j)=Sd{count}(j)+1;
         end
      end
   end
end

density;

% Finding the nodes with highest density by sorting Sd  and add it to
% dominating set then find its neighbour with highest density . nei
% is vector that stores list of neighbor. Core is array storing DS
% keeping the nodes in Si in a vector temp, and marking all the nodes
% selected as core or neighbor as zero, stopping the iteration when all the nodes are
selected or all values in temp are zeros


for count=1:length(S) %then for each set, I find the core nodes
   if length(S{count}) >1
      nei=[];
      neiDensity=[];
      temp=S{count};
      [Ssorted, idx]= sort(Sd{count}, 'descend');%first, the first core node is found by
finding the highest density node out of all nodes in the set
      Core{count}(1)=S{count}(idx(1));
      temp(idx(1))=0;
      for i=1:length(S{count})
         % for j=i+1:length(S{count})
         if dist(S{count}(i),Core{count}(1)) < txrange
            nei=[nei, S{count}(i)];
            neiDensity=[neiDensity, Sd{count}(i)];
            temp(i)=0;
         end
         %Find the nodes that are to be core nodes
      end
      coreNum=2;
      while nnz(temp)>0 % nnz is inbuilt function in Matlab that returns the number
of non zero elements in a matrix, stopping the iteration when all the nodes are
selected or all values in temp are zeros
         [neiSorted, idxNei]=sort(neiDensity, 'descend'); %then I find core nodes 2
onwards by selecting the highest density nodes out of the neighbours
         Core{count}(coreNum)=nei(idxNei(1));
         temp(temp==idxNei(1))=0;% once a node is either a core node, or a neighbor
of core node, that node is eliminated from the temp vector
```

```
        %we are done when the temp vector
        %is all zero then that vector is
        %completely done that is all the nodes in the set are either core nodes or a
neighbor of core nodes

        neiDensity(nei==Core{count}(coreNum))=[];
        nei(nei==Core{count}(coreNum))=[];
        for i=1:length(S{count})
           % for j=i+1:length(S{count})
           if dist(S{count}(i),Core{count}(coreNum)) < txrange
              if sum(nei==S{count}(i))==0 % this checks whether the i th nodes is a
member of the vector nei
                 if sum(Core{count}==S{count}(i))==0 %checks whether the i th
node in S{count} is a member of the set Core, the sum is zero means the nodes is not
a member of the vector nei
                    nei=[nei, S{count}(i)];                %in other words, whether that
node is already a neighbor of a core node
                    %if the node is in neither set, the node is added to the
                    %set of neighbors of all core nodes
                    neiDensity=[neiDensity, Sd{count}(i)];
                    temp(i)=0;
                 end
              end
           end

           %  end
        end
        coreNum=coreNum+1;
     end
  end
end
%        Core
%   S
length(S)
% Mark the core nodes in red
for i=1:length(Core)
   for j=1:length(Core{i})
      plot(coord(Core{i}(j),1), coord(Core{i}(j),2), 'bo', 'Color', 'r');
   end
end
plot(coord(1,1), coord(1,2), 'bo', 'Color', 'g');
xlabel('X axis');
ylabel('Y axis');

legend('Sensor node', 'Sink', 'CDS node')

isolatedNum=0;
for i=2:length(S)
   if length(S{i})==1
      isolatedNum=isolatedNum+1;
```

```
   end
end

isolatedNum;

%connectivity = (n- isolatedNum)/n;
%S{count}-Core{count} should give the number of nodes that are not in the core
and
%Core{count} gives the number of nodes in the core

%Find the isolated nodes


%    isolatedNodes=[];
%    for count=2:length(S)
%        if length(S{count})==1 % checks if nodes has no neighbors ,or single
%            isolatedNodes=[isolatedNodes, S{count}];%isolatedNodes array of single
isolated nodes
%
%        end
%    end

 NumNodesConnectedtoSink=length(S{1})
NumNodesInBackBoneConnectedToSink=length(Core{1})
NumNodesNotConnectedtoSink=n-length(S{1})
PercentOfNodesinBackBoneConnectedToSink=length(Core{1})/n*100
PecentOfNodesConnectedToSinkBeforeAnchor=length(S{1})/n*100
%connectivitybeforeanchor = NumNodesConnectedtoSink/n*100
FiedlerBeforeAnchor=Fiedler(coord, txrange)

%        isolatedNodes;
%        anch{1}=[];%set of anchor nodes used to connect isolated nodes
%        connectingNode=[];
%
%        % Connect isolated nodes with anchor nodes.
%        isoConnected=[];
%        for i=1: length(isolatedNodes)
%            distIso=dist(isolatedNodes(i),:);% distance between each isolated node and
rest of nodes, but we need to check if closest node is also isolated!
%            distIso(isolatedNodes(i))=inf;
%
%            [minVal, idx] =min(distIso);%idx is the index of the node having minimum
distance from this isolated node
%             minDist=distIso(idx);
%             iter=0;
%             while sum(isoConnected==idx)~=0 && iter < n % here we are checking if
the node that is closest to the isolated node is also another isolated node that is
already connected to another set
%                 distIso(idx)=inf;
%                 [minVal, idx] =min(distIso);
```

```
%             minDist=distIso(idx);
%             iter=iter+1;
%         end
%         connectingNode=[connectingNode, idx];
%         if minDist<2*txrange   % One anchor node is sufficient to connect this
node
%
anch{i}=[(coord(idx,1)+coord(isolatedNodes(i),1))/2,(coord(idx,2)+coord(isolatedN
odes(i),2))/2];% anch is placed at half distance btwn idx and isolatednode
%         else    % if dist is > 2*transmission range, one anchor node is not sufficient.
More than one are required
%             numAnchors=max([1,floor(minDist/txrange)]);  %Find the number of
anchors required,if ceil(60/25 =2.4 )=3 so 3-1 is 2 nodes
%             xinc=(coord(idx,1)-coord(isolatedNodes(i),1))/numAnchors;  %Find the
icrementing required for placing anchor nodes.
%             yinc=(coord(idx,2)-coord(isolatedNodes(i),2))/numAnchors;
SUnconnected={};
count=1;
anch={};
connectedNodes{1}=S{1};  % In the first iteration, only the cluster connected to the
sink is connected.  This is in S{1}
for i=2:length(S)
   SUnconnected{i-1}=S{i};  % All the other sets are unconnected
end
%cc
connTarget=1  %0.90
conncount=1;
   connectivity=1-length(cell2mat(SUnconnected))/(n)

   conn(conncount)=1-length(cell2mat(SUnconnected))/(n)
   FiedlerVal=[];
while (length(SUnconnected)>0 && connectivity < connTarget)
   UCDist=[];
   %if length(SUnconnected)>0
   SvecUC=cell2mat(SUnconnected);  % Convert the arrays into vectors
   connectedNodesVec=cell2mat(connectedNodes); % Convert the arrays into
vectors

   for i=1:length(connectedNodesVec)
      for j=1:length(SvecUC)
         UCDist(i,j)=sqrt(sum((coord(connectedNodesVec(i),:)-
coord(SvecUC(j),:)).^2)); % Find the distance between all connected nodes and all
unconnected nodes
      end
   end

   % in the following code, the minimum of the distance matrix is found.
   if length(connectedNodesVec)>1 %if there is only one row to UCDist, the
following code does not work.  So a special code is written for that case.
```

```
      [minDist, idxC]=min(min(UCDist)); % idxC is the index of the isolated node
that is closest to a connected node
      [minDist, idxR]=min(UCDist(:,idxC)'); % idxR is the index of the connected
not that is closest to an isolated node
   else
      [minDist, idxC]=min(UCDist);
      idxR=1;
   end
   closestConnectedNode=connectedNodesVec(idxR);  % This is the connected node
that is closest to an unconnected node
   closestIsoNode=SvecUC(idxC);  % This the the unconnected node that is closest
to a connected node.
   closestConnectedNodeVec(count)=closestConnectedNode;  % Enter the above
found nodes in array
   closestIsoNodeVec(count)=closestIsoNode; % Enter the above found nodes in
array


   if minDist<2*txrange
      numAnchors=1;
      anch{count}=(((coord(closestConnectedNode,:)+coord(closestIsoNode,:))))/2;
% if the nodes can be connected by one anchor node, find the location of the anchor
node.
      coord(end+1,:)=anch{count};


   else    % if dist is > 2*transmission range, one anchor node is not sufficient.  More
than one are required
      numAnchors=max([1,ceil(minDist/txrange)-1]);  %Find the number of anchors
required,if ceil(60/25 =2.4 )=3 so 3-1 is 2 nodes
      xinc=(coord(closestConnectedNode,1)-
coord(closestIsoNode,1))/(numAnchors+1);  %Find the incrementing required for
placing anchor nodes.
      yinc=(coord(closestConnectedNode,2)-
coord(closestIsoNode,2))/(numAnchors+1);

      for j=1: numAnchors
         anch{count}(j,:)=[coord(closestConnectedNode,1)-j*xinc,
coord(closestConnectedNode,2)-j*yinc];%Finding the location coordinates of the j th
anchor node connecting the ith isolated node
         coord(end+1,:)=anch{count}(j,:);
      end
   end

   for i=1:length(SUnconnected)
      if sum(find(SUnconnected{i}==closestIsoNode))>0  % Find the set that
contains the unconnected node that is now connected by anchor node
         for anchCount=1:length(anch{count}(:,1))
```

```
            SUnconnected{i}(end+1)=length(coord(:,1))-numAnchors +anchCount;  %
add all the newly added anchors to the list of the set that is just connected
%anch{count}(anchCount,:);
         end
         connectedNodes{count+1}=SUnconnected{i}; % and add it to the array of
connected nodes
         if length(SUnconnected)>1
            for k=i:length(SUnconnected)-1
               SUnconnected{k}=SUnconnected{k+1};  % Remove the set connected
using anchor nodes from the array of unconnected nodes.

            end
            SUnconnected(end)=[];
            break
         else
            SUnconnected={};
            break
         end
      end

   end

   if minDist<2*txrange  % try to connect more than one CDS with one anchor node

      SvecUC=cell2mat(SUnconnected);  % Convert the arrays into vectors
      connectedNodesVec=cell2mat(connectedNodes); % Convert the arrays into
vectors

      %for i=closestIsoNode  %1:length(connectedNodesVec)
      UCDist=[];
      for j=1:length(SvecUC)
         UCDist(j)=sqrt(sum((coord(closestIsoNode,:)-coord(SvecUC(j),:)).^2)); %
Find the distance between all connected nodes and all unconnected nodes
      end
      %end

      [minDist, temp]=min(UCDist);
      closestSecondNode=SvecUC(temp);  % This is the closest unconnected node to
the unconnected node that is now conneccted by anchor node

      if minDist<2*txrange

centroid=(((coord(closestConnectedNode,:)+coord(closestIsoNode,:)+coord(closestS
econdNode,:))))/3;

         if sqrt(sum((centroid-coord(closestIsoNode,:)).^2))<txrange &&
sqrt(sum((centroid-coord(closestConnectedNode,:)).^2))<txrange &&
sqrt(sum((centroid-coord(closestSecondNode,:)).^2))<txrange
            %if this condition is satisfied, one anchor node is sufficient
            %to connect all three
```

```
            anch{count}=centroid;
            coord(end+1,:)=anch{count};
            closestIsoNodeVec(count+1)=closestSecondNode;
            anch{count+1}=[];
            for i=1:length(SUnconnected)
               if sum(find(SUnconnected{i}==closestSecondNode))>0  % Find the set
that contains the unconnected node that is now connected by anchor node
                  for anchCount=1:length(anch{count}(:,1))
                     SUnconnected{i}(end)=length(coord(:,1))  ;
%anch{count}(anchCount,:);
                  end
                  connectedNodes{count+2}=SUnconnected{i}; % and add it to the
array of connected nodes
                  if length(SUnconnected)>1
                     for k=i:length(SUnconnected)-1
                        SUnconnected{k}=SUnconnected{k+1};  % Remove the set
connected using anchor nodes from the array of unconnected nodes.

                     end
                     SUnconnected(end)=[];
                     count=count+1;
                     break
                  else
                     SUnconnected={};
                     break
                  end
               end

            end


         end
      end
   end

   connectivity=1-length(cell2mat(SUnconnected))/(n)


   count=count+1;
   conncount=conncount+numAnchors;
   if numAnchors>1
      conn(conncount-numAnchors:conncount)=1-
length(cell2mat(SUnconnected))/(n)
   else
      conn(conncount)=1-length(cell2mat(SUnconnected))/(n)
   end
      if length(FiedlerVal)>0 && numAnchors>1
       FiedlerVal(conncount-numAnchors-1:conncount-1)= FiedlerVal(conncount-
numAnchors-1)
      end
```

```
      FiedlerVal(conncount-1) =Fiedler(coord, txrange)
end
%end

if length(anch)>0

if length(anch{1})>0 % in cases where two isolated nodes can be connected using
one anchor node, I have deleted the other anchor node
   for i=1:length(anch)
      if length(anch{i})~=0
         %  plot([anch{i}(1,1) coord(connectingNode(i),1)], [anch{i}(1,2)
coord(connectingNode(i),2)], '-.');
         for j=1:length(anch{i}(:,1))
            plot((anch{i}(j,1)),(anch{i}(j,2)), '*', 'Color', 'k');
            if j>1
               plot([anch{i}(j,1) anch{i}(j-1,1)], [anch{i}(j,2) anch{i}(j-1,2)], '-.');

            end

            %   plot([anch{i}(j,1) connectingNode(i)], [coord(trialNode, 2)
centroid(2)], '-.')
         end
         plot([anch{i}(j,1) coord(closestIsoNodeVec(i),1)], [anch{i}(j,2)
coord(closestIsoNodeVec(i),2)], '-.');
         plot([anch{i}(j,1) coord(closestConnectedNodeVec(i),1)], [anch{i}(j,2)
coord(closestConnectedNodeVec(i),2)], '-.');

      else
         plot([anch{i-1}(1,1) coord(closestIsoNodeVec(i),1)], [anch{i-1}(1,2)
coord(closestIsoNodeVec(i),2)], '-.');
         plot([anch{i-1}(1,1) coord(closestIsoNodeVec(i-1),1)], [anch{i-1}(1,2)
coord(closestIsoNodeVec(i-1),2)], '-.');
      end
   end
end

legend('Sensor node', 'Sink')
% 1- Number of nodes. 2- Area (L*L). 3- % of nodes in backbone.
% 4- # of nodes connected in backbone (nodes in backbone +nodes connected with
backbone).
% 5- % of nodes not connected to backbone. 6- Min # of anchor nodes deployed. 7-
% of connectivity after anchor deployment.

figure
plot(0:(conncount-1),conn, '-o')
xlabel('Number of anchor nodes')
ylabel('Connectivity')

ax = gca;
%ax.XTick = [0:(conncount-1)];
```

```
set(ax,'XTick',[0:(conncount-1)]);
%grid on
end
anchCount=0;

if length(anch)>0
for i=1:length(anch)
   if length(anch{i})>0
      anchCount=anchCount+length(anch{i}(:,1));
   end
end

end
FiedlerValFinal =Fiedler(coord, txrange);
connectedNodesVec=cell2mat(connectedNodes);

NumberOfNodes=n
disp('Area=')
disp([num2str(L), 'X', num2str(L)])
   %coord=coord;
for i=1:20
      coord(end+1,:)=L*rand(1,2);
   FiedlerVal(end+1)=Fiedler(coord, txrange);

end

 figure
plot(0:length(FiedlerVal),[FiedlerBeforeAnchor, FiedlerVal])
xlabel('Number of Anchor nodes')
ylabel('Fiedler value')
ax = gca;
%ax.XTick = [0:(conncount-1)];
set(ax,'XTick',[0:length(FiedlerVal)]);

%PercentOfNodesInBackbone=
NumberOfAnchorNodes=anchCount
PercentofConnectivity=(length(connectedNodesVec)-anchCount)*100/n
FiedlerBeforeAnchor
 FiedlerVal
FiedlerValFinal
```

Appendix A.2: Finding the Next Node in the Set

Program Code: FindNewNode.m

```
function nextNode=findNewNode(temp, n) % find a node that is not chosen in any
sets ,it takes out the next node that is not marked as zero

for i=1:n+1
   if i<=n
      n;
      length(temp);
   if temp(i)~=0
      break
   end
   end
end

nextNode=i; % if all the nodes are 0, the nextNodes is n+1.  That means
```

Appendix A.3: Calculating the Fiedler Value for the Network

Program Code: FiedlerValue.m

```
function fvalue=Fiedler(coord, txrange)
% Find Fiedler value  of a network

% clc
% clear all
% NumberOfNodes=10
% n=NumberOfNodes+1      % The number of nodes
% L=100       % Dimension of the field
% srange=25   % sensing range
% txrange=25  %transmission range
%
% coord = L*rand(n,2);  % Randomly select the coordinates of the location of n
nodes.
% coord(1,:) = [L/2, L/2];
n=length(coord(:,1))
for i=1:n
   for j=1:n
      dist(i,j)= sqrt(sum((coord(i,:)-coord(j,:)).^2)); %Finding the euclidean distance
between node i and node j
      dist(j,i)=dist(i,j);% radio channel is reciprocol
   end
end

conn=dist<txrange;
laplace=-conn;
for i=1:n
   laplace(i,i)=sum(conn(i,:))-1;
end


laplace;

e=eig(laplace);
fvalue=round(e(2)*1e6)/1e6;
fvalue;
```

Appendix B: Number of Runs for System Simulation

Based on the Discrete Event System Simulation by Banks et al. [26], we have chosen a confidence interval: 100 (1-α) % to be 95 %, (α=0.05).

The required sample size R or number of runs should satisfy equation(1):

$R >= (t_{\alpha/2,R-1}S0/\epsilon)^2$ which is based on t distribution as long as R <50 otherwise R should satisfy equation(2): $R >= (z_{\alpha/2}S0/\epsilon)^2$ which is based on z distribution.

In our algorithm, we have reached number of runs 50 which satisfies equation (2) where variance is 0.0325 and $z_{\alpha/2}$ is z0.025=1.96.

Appendix C: Fiedler Value

In 1973, Miroslav Fiedler, has studied the algebraic connectivity of a graph G [23]. Assuming graph G(V,E) be unidirectional finite graph with fixed order w_1, w_2, …, w_n of set of vertices V and edges E. Considering a Laplacian matrix A(G) which is symmetric, having sum of rows 0, and positive semi-definite ($A(G)=UU^T$ where U is the (0,1,-1) vertex-edge adjacency matrix of graph G). This matrix A(G) has its off-diagonal elements $a_{ik}= a_{ki} = -1$ if $(w_i,w_k) \epsilon E$. That is if there exist a path between node i and node k, then the Laplacian entry value $L_{i,k} = -1$ and $L_{i,i}$ is the degree of connectivity for node i. The eigenvalues of the Laplacian matrix A(G):

$0 <= \lambda1 <= \lambda2 <= a(G) <= \lambda3 <= \ldots. <= \lambda n$. It follows that a(G) is 0 when graph G is disconnected. Therefore, the second smallest eigenvalue $\lambda2=a(G)$ shall be the algebraic connectivity of graph G and its corresponding eigenvector is referred to Fiedler vector. Fiedler vector is usually used to improve the algebraic connectivity of a network topology. The algebraic connectivity has important information on network connectivity of a graph, such as, if the network is disconnected, then the multiplicity of zero reveals number of disconnected sub-graphs.

The author in [23] considered the network field to be equally divided into n sub-regions. They deployed a number of relays at specific positions and then they found which region having the maximum Fiedler value. They then divide the region with maximum Fiedler value into n equally sub-regions after deploying certain amount of relays at specific positions, then they found which region had maximum Fiedler value and keep on repeating same procedure so that no more improvement in Fiedler value.