Computer Networks 123 (2017) 137–152

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Distributed construction of minimum Connected Dominating Set in wireless sensor network using two-hop information

Jasaswi Prasad Mohanty[a,*], Chittaranjan Mandal[a], Chris Reade[b]

[a] Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India
[b] Kingston University, London, UK

ARTICLE INFO

Article history:
Received 4 November 2016
Revised 14 May 2017
Accepted 15 May 2017
Available online 17 May 2017

MSC:
00-01
99-00

Keywords:
Connected Dominating Set (CDS)
Maximal Independent Set (MIS)
Virtual Backbone
Unit Disk Graph (UDG)
Steiner Tree

ABSTRACT

In a Wireless Sensor Network (WSN), neither there is any fixed infrastructure nor any centralized control. Therefore, for efficient routing, some of the nodes are selected to form a virtual backbone. Minimum Connected Dominating Set (MCDS) can be used as a virtual backbone. However, MCDS construction is an *NP-Hard* problem. In this paper, we propose a novel distributed greedy approximation algorithm for CDS construction which reduces the CDS size effectively. The proposed method constructs the CDSs of smaller sizes with lower construction cost in comparison to existing CDS construction algorithms for both uniform and random distribution of nodes. The performance ratio of the proposed algorithm, which is the best at the current moment, is $(4.8+\ln 5)|opt|+1.2$, where $|opt|$ is the size of an optimal CDS of the network. Its time complexity is $O(D)$, where D is the diameter of the network. Its message complexity is $O(nR)$ which is linear, where n is the network size and R is the maximum between number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes. Our simulation shows that ours is the most size optimal distributed CDS construction algorithm.

1. Introduction

A Wireless Sensor Network (WSN) is formed by the wireless links of the sensor nodes deployed in an area. The sensor nodes are spatially distributed to monitor physical or environmental conditions such as temperature, pressure, sound etc. and send their data to the base station cooperatively. Generally, WSNs are used in monitoring of patients, environment, industry, food, agriculture etc. It is also used in earth sensing, search and rescue, disaster control [1] etc. Each node of a WSN has typically several parts: a radio transceiver, a microcontroller, a battery. The network may contain only static nodes, only mobile nodes or a mixture of both depending on the application. Each node of the network helps in routing by forwarding data of other nodes and which node will forward the data is decided dynamically based on the state of the network. WSN is ad-hoc because it does not depend on any existing infrastructure such as router in wired network. In WSN, nodes can communicate efficiently through the use of a virtual backbone. A virtual backbone is a connected subset of nodes deployed in the entire network which helps in routing. A pair of nodes in the network can communicate each other through the backbone nodes. As there is no fixed infrastructure and centralized control in a WSN, a Connected Dominating Set (CDS) can work as a virtual backbone for efficient routing and connectivity [2].

A Dominating Set (DS) of a network is formed by any subset of nodes of the entire network such that, each node either belong to the subset or neighbour of some element of that subset. If the nodes of a DS are connected, then they form a CDS. The CDS is responsible for transmitting messages from any node to any other node. A source node which does not belong to the CDS, sends its message to the destination node by first sending it to one of its neighbouring CDS nodes. If the destination node belongs to the CDS, it gets the message directly, otherwise it gets the message from one of its neighbours which belongs to the CDS. During routing, a CDS node forwards the message to its CDS neighbours only. So, these CDS nodes only maintain the routing information. Therefore, reduction of CDS size can save the storage space and also makes the routing easier and faster. Also by using a smaller sized CDS as virtual backbone the total energy consumption of the network can be reduced, if the non-CDS nodes switch off their radio when they don't have any data to send. Therefore, construction of Minimum Connected Dominating Set (MCDS) of the network is desirable. However, MCDS construction is an NP-complete problem [3]. For this reason researchers are interested for polynomial time approximation algorithms for CDS construction. As there is no centralized management in WSN, distributed algorithms can be useful

* Corresponding author.
E-mail addresses: jasaswiprasad@gmail.com, jasaswiprasad@rediffmail.com (J.P. Mohanty).

http://dx.doi.org/10.1016/j.comnet.2017.05.017
1389-1286/

for finding the MCDS. Energy is vital in WSN because the nodes can't be recharged. Therefore, the distributed approximation algorithms should construct smaller CDSs with low computation and communication costs. The quality of CDS is measured by its performance ratio, which is the ratio of the size of the constructed CDS (by the proposed algorithm) to the size of MCDS. The construction cost is also measured by the overall message and time complexities. To extend the lifetime of the network, in spite of relying on a single CDS, the network should switch between disjoint CDSs [4,5]. Therefore, to switch between CDSs quickly the computation time of CDS construction algorithm should be small enough. In this article, our focus is on constructing size optimal CDS as a virtual backbone of the WSN.

We can construct a CDS either in centralized or distributed manner. Although centralized algorithms provide more accurate information than distributed algorithms, they suffer from scalability problem and hence not feasible for large size WSNs. In centralized algorithms, the reliability of the information accumulated at a centralized processor is low because of the losses involved in multihop transmission. Distributed algorithms are difficult to design. They require only local information exchange between neighbouring nodes. For any WSN in which the average number of hops from any node to central processor is greater than the number of iterations required to perform a task, distributed algorithms are more energy efficient than centralized algorithms [6]. In this paper, we propose a new distributed degree-based greedy approximation algorithm which we name as **D**istributed **C**onstruction of **M**inimum **C**onnected **D**ominating **S**et (DCMCDS) to construct smaller CDSs.

The proposed scheme DCMCDS works in three phases and constructs the CDS using 2-hop information only. In the first phase, it constructs Maximal Independent Set (MIS) in a distributed manner. The MIS is designated as a Pseudo Dominating Set (PDS) because some of the elements may be omitted in the final dominating set. In the second phase, the algorithm constructs a Steiner Tree by adding some more nodes to the PDS, which are needed to interconnect the PDS nodes. In the last phase, the algorithm drops some of the selected PDS nodes to reduce the CDS size further without any loss in coverage or connectivity. Simulation results show that DCMCDS is better than existing CDS construction algorithms in terms of CDS size and construction costs. The performance ratio of the proposed algorithm, which is the best at the current moment, is $(4.8+\ln 5)|opt|+1.2$, where $|opt|$ is the size of an optimal CDS of the network. Its time complexity is $O(D)$, where D is the diameter of the network. It has a linear message complexity of $O(nR)$, where n is the network size and R is the maximum between number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes.

The remaining of the article is organized as follows. In Section 2, we provide some basic definitions which we use in the entire paper. Section 3, provides a review of the works on CDS construction. In the next section (Section 4), we discuss the motivation behind our work and our major contributions. In Section 5, we discuss the centralized version of our proposed scheme in brief. Section 6 discusses the distributed CDS construction algorithm in detail. The analysis of our proposed distributed algorithm is discussed in Section 7. Supporting simulation results are given in Section 8. Finally we presented the conclusion in Section 9.

2. Background

In this section, we discuss some of the fundamental concepts that are useful to understand our work.

Definition 2.1 (DOMINATING SET)**.** In graph theory, a dominating set (DS) for a graph $G(V, E)$ is a subset $V'\subseteq V$ such that for each node $v\in V-V'$, $Adj[v]\bigcap V'\neq\phi$, where $Adj[v]$ denotes set of adjacent nodes of v. The nodes in the dominating set, V' are called **dominators**.

Definition 2.2 (CONNECTED DOMINATING SET)**.** A dominating set which forms a connected subgraph is a Connected Dominating Set (CDS). So, a CDS of a graph is a set of vertices with the following properties:

1. Every vertex of the graph is either belongs to the CDS or is adjacent to atleast one vertex of the CDS.
2. We can reach from any node in the CDS to any other node in CDS by a path which stays entirely within CDS.

The nodes which does not belongs to the CDS are called as **dominatees**.

Definition 2.3 (INDEPENDENT SET)**.** In a graph, a set of vertices in which none of two are adjacent, is called as an independent set or stable set.

Definition 2.4 (MAXIMAL INDEPENDENT SET)**.** An independent set to which by adding any vertex outside the independent set disturbs the property of independent set is called as Maximal Independent Set (MIS) or maximal stable set. In other words, a maximal independent set cannot be a sub set of any other independent set.

Definition 2.5 (UNIT DISK GRAPH)**.** A Unit Disk Graph (UDG) is the intersection of unit disks (of unit radii) in the Euclidean plane. The centre of each disks is a node. So the disk represent the communication range of the node which is same for all nodes. Two nodes are connected by an edge if the Eucledian distance between the two nodes is less than one unit.

Definition 2.6 (STEINER TREE)**.** In a graph $G=(V,E)$, for a given subset of vertices $I\subseteq V$, a Steiner Tree is a tree which interconnects the nodes in I using a set of nodes (known as Steiner nodes) not in I.

3. Related work

For connectivity and coverage in wireless network, CDS can be used as virtual backbone. In 1987, Ephermides first proposed this idea [7]. Since then the research on CDS has never been interrupted. Many researchers proposed different algorithms to construct the CDS. The CDS construction approaches found in the literature can be broadly classified as centralized, distributed and localized algorithms.

In a centralized CDS construction algorithm, the topology information of the entire network is needed at a particular node where the CDS construction algorithm runs. Guha and co-workers [2,8] first proposed two polynomial time centralized algorithms in 1998 . The approximation ratio and time complexities of both these algorithms were $O(\ln\Delta)$ and $O(n^2)$ respectively, where Δ is the maximum node degree and n is the network size. Later on in 2005, Adjih et al. [9] proposed a localized algorithm for CDS construction which was based on multipoint relays (MPR).

In WSN getting the entire topology information at one node is not easy. Therefore, distributed algorithms are very useful for CDS construction. In 1999, Wu and Li [10] proposed the first distributed CDS construction algorithm and Alzoubi et al. [11] reported its approximation ratio as $O(n)$. Later on Stojmenovic et al. [12] and Das et al. [2] proposed different distributed algorithms of approximation ratio $O(n)$ and $O(\log n)$ respectively. However, the time and message complexities of these algorithms are quite high.

Most of the distributed CDS construction algorithms first construct the MIS and then connect these MIS nodes to form a CDS. In 2002, for a UDG, Wan et al. [13] proposed a two phase distributed leader initiated CDS construction algorithm of performance ratio

$8|\mathrm{opt}|+1$. It has the time complexity of $O(n)$, and message complexity of $O(n \log n)$, where $|\mathrm{opt}|$ is the size of an optimal CDS. Later on, Cardei et al. [14] improved the approximation factor to $8|\mathrm{opt}|$. The time and message complexity of Cardei's algorithm is $O(\Delta n)$ and $O(n)$ respectively. Cardei's algorithm grows from a single leader and uses 1-hop neighbours' information for identifying Steiner nodes. The first two phase multiple leaders based distributed algorithm of approximation ratio $192|\mathrm{opt}|+48$ was proposed by Alzoubi et al. [11] in 2002. Zhu and Du in [15] proposed a similar distributed algorithm with a better approximation ratio of 172 in 2004. Among all the distributed approximation algorithms found in the literature, the best approximation ratio of $(4.8+\ln 5)|\mathrm{opt}|+1.2$ is achieved by Li's S-MIS algorithm [16], Das's PSCASTS [17] and Misra's collaborative cover heuristic [18]. All these algorithms first construct an MIS and then connect these MIS nodes to form a Steiner Tree, by using some non-MIS nodes as connectors. The collaborative cover heuristic which uses effective coverage as a metric for MIS construction, constructs CDS of smaller sizes in comparison to other distributed CDS construction algorithm. However the algorithm has a high message and time complexities of $O(n\Delta^2)$ and $O(n)$ respectively. More recently, we find another two phase distributed algorithm for UDG by Jallu et al. [19] with time complexity $O(\Delta)$ and message complexity $O(n)$. Although the algorithm outperforms the existing algorithms in terms of running time however its approximation ratio $104|\mathrm{opt}|+52$ is quite high.

In 2006, Neiberg and Hurink [20] proposed a localized algorithm in which each vertex decides itself whether to be a part of the dominating set or not depending on the vertices which are a constant number of hops away from it. The proposed polynomial time approximation scheme (PTAS) computes the dominating set with $(1+\epsilon)$ approximation $(\epsilon > 0)$. The processing time is upper bounded by the number of vertices present in the radius to be explored. In that work, we find a concept of 2-separated collection which emphasizes that the size of the dominating set can be reduced if the topology can be divided into local 2-separated collections. Hence, we tried to develop this idea by investigating the balance needed between the performance ratio and the locality that needs to be surveyed.

Currently people are also working on k-connected m-dominating set problem. As the nodes may fail due to energy depletion or external damage, k-connected m-dominating sets are really useful for their extended life time. In [21], Zhang et al. proposed an algorithm for minimum 3-connected m-dominating set $(m \geq 3)$ problem and in [22] Wang et al. constructed a minimum 4-connected m-dominating set in UDG. Ding et al. [23] defined a special type of CDS named as *α minimum routing cost connected CDS* (α-MOC-CDS) in which between any pair of nodes there is at least one path in which all intermediate nodes belong to α-MOC-CDS and the number of the intermediate nodes is smaller than α times of that on the shortest path in the original network. Liu et al. [24] constructed a special Strongly Connected Bidirectional Dominating Set which forms a virtual backbone of the network in which the nodes have different transmission ranges. Shi et al. [25] defined a unique problem named as Energy Harvest CDS (EH-CDS) to discover the maximum number of CDSs in a static network which takes the advantages of rechargeable nodes to become energy harvest networks. Also in the literature we find a biology-inspired algorithm for Steiner Tree construction by Liu et al. [26]. Although the algorithm has a lower performance ratio, its running time is quite high (quadratic) and no message complexity is reported by the authors.

Besides using CDS, another way of achieving connectivity and coverage in wireless networks at minimal cost is by using geographic routing [27] which depends on geographic location of each node to forward packets greedily. In case greedy forwarding fails in geographic routing schemes, routing can also be possible through coverage tree based routing. A coverage tree can be formed either in top-down or bottom-up fashion. The coverage tree based routing is a NP-Hard problem. In [28], Das et al. presented the approximation algorithms for Coverage Tree based routing in wireless networks. They proposed both centralized and distributed coverage tree based approximation which incorporates an additional strategy to solve the minimum distance topology (MDT) problem in wireless network having approximation ratio $1+\ln m$, where m is the number of elements. Although we can achieve connectivity and coverage of WSNs at minimal cost by using geographic routing, it requires that each node know their location information. Therefore, researchers have proposed virtual coordinate systems where each node is assigned with some coordinate which is not the actual location or geographic coordinates but in order to serve as a basis of routing, they reflect the underlying connectivity. Virtual coordinates helps in building network overlays (geographic hash tables) to support P2P routing protocols. Virtual coordinate system is useful in low density networks. It can achieve higher success rate and lower hop count when virtual coordinate assignment using dominating set algorithm is used with geographic routing. In [29], Shukla et al. proposed a virtual coordinate assignment protocol which uses dominating set algorithm, to assign virtual coordinates to nodes that have no geographic information having approximation ratio $(4.8+\ln 5)|\mathrm{opt}|+1.2$, where opt is the minimum size dominating set.

In this work, we proposed a distributive CDS construction algorithm which minimizes the CDS size. Out of the various CDS construction algorithms discussed in this section, we compare the performance of our proposed algorithm with the general CDS construction algorithms which use CDS size as their performance metric.

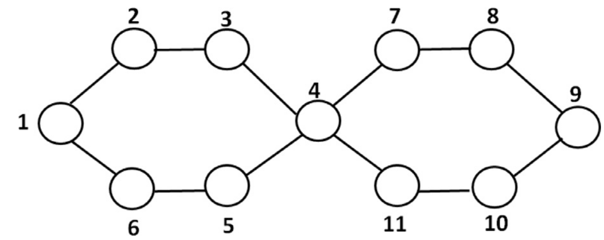


Fig. 1. Network showing an alternative MIS construction technique.

4. Motivations and contributions

Most of the distributive CDS construction algorithms like [13,14,16,18] to achieve a good performance ratio, construct MIS with a property mentioned in [16]. The property says that distance between a MIS and its complement is exactly two hops. This property helps in interconnecting the nodes present in the MIS to form the CDS. However, we observe that in any MIS, each node is maximum three hops away from its nearest MIS node. Therefore, by selecting MIS nodes with three hop separation we can reduce the MIS size. Also reduction in MIS size may reduce the CDS size and hence also improve the ratio of number connectors to number of independent nodes. This ratio has a high impact on the network life time. To demonstrate the above point, let us consider the graph shown in Fig. 1. If we select the nodes to form the MIS such that every node is separated from its nearest neighbour in the MIS by two hops then 1, 3, 5, 7, 9, 11 can form the MIS. However, the MIS can also be formed by the nodes 1, 4, 9. In the latter case each MIS node is separated from its nearest MIS neighbour by three hops. Although the MIS size is smaller in this case, but to construct this type of MIS in a distributed fashion each node needs to know its 2-

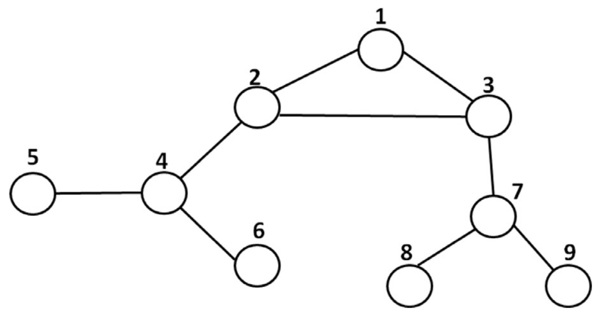


Fig. 2. Network illustrating how PDS can improve CDS size over MIS.

hop neighbours connectivity information. Furthermore, in the later stage to construct the Steiner Tree [16,30] more message exchanges are needed which increases the overhead.

Further, we also observed that after the construction of Steiner Tree, some of the dominators from the MIS can be downgraded to dominatees without any coverage or connectivity loss. A MIS node can be downgraded to a dominatee if all of its neighbours (if any) can be covered either by some Steiner nodes or by some other MIS nodes. To demonstrate this point let us consider a graph shown in Fig. 2. In this network, the minimum size MIS is formed by the nodes 1, 4 and 7. Then to connect these MIS nodes we need nodes 2 and 3 as connectors. Thus, nodes 1, 2, 3, 4 and 7 form the CDS. Now, we can observe that, node 1 in the CDS is redundant, because nodes 2, 3, 4 and 7 can also form a CDS. For this reason, we call this dominating set as pseudo dominating set, since in the later stage some of the MIS nodes may be removed from the CDS for reduction of CDS size further. Motivated with the above discussed issue we tried to reduce the CDS size with minimum number of message exchanges during the CDS construction. We designed a new distributed CDS construction algorithm DCMCDS, which improves the CDS size further over previous approximation algorithms.

The major contributions of our work are as follows:

- A greedy distributed approximation algorithm for Minimum Connected Dominating Set problem is proposed where there is no specific initiating node.
- Smaller size MISs are identified using PDS constructed in a distributed manner
- Steiner Tree is constructed to connect the PDS nodes in a distributed manner. In the later stage, the algorithm selectively removes some nodes of the Steiner Tree to minimize the CDS size.
- The proposed distributed algorithm DCMCDS has the linear time complexity of $O(D)$, where D is the diameter of the network.
- DCMCDS identifies non-trivial CDSs of smaller sizes for both uniform and random distribution of nodes.

5. Centralized CDS construction by DCMCDS

In this section we briefly discuss DCMCDS, a centralized approach to MCDS formation [31] to motivate our distributed MCDS construction described in the next section. DCMCDS works in the following three phases:

A. Pseudo-dominating set construction
B. Improved Steiner Tree construction
C. Removal of redundant dominators

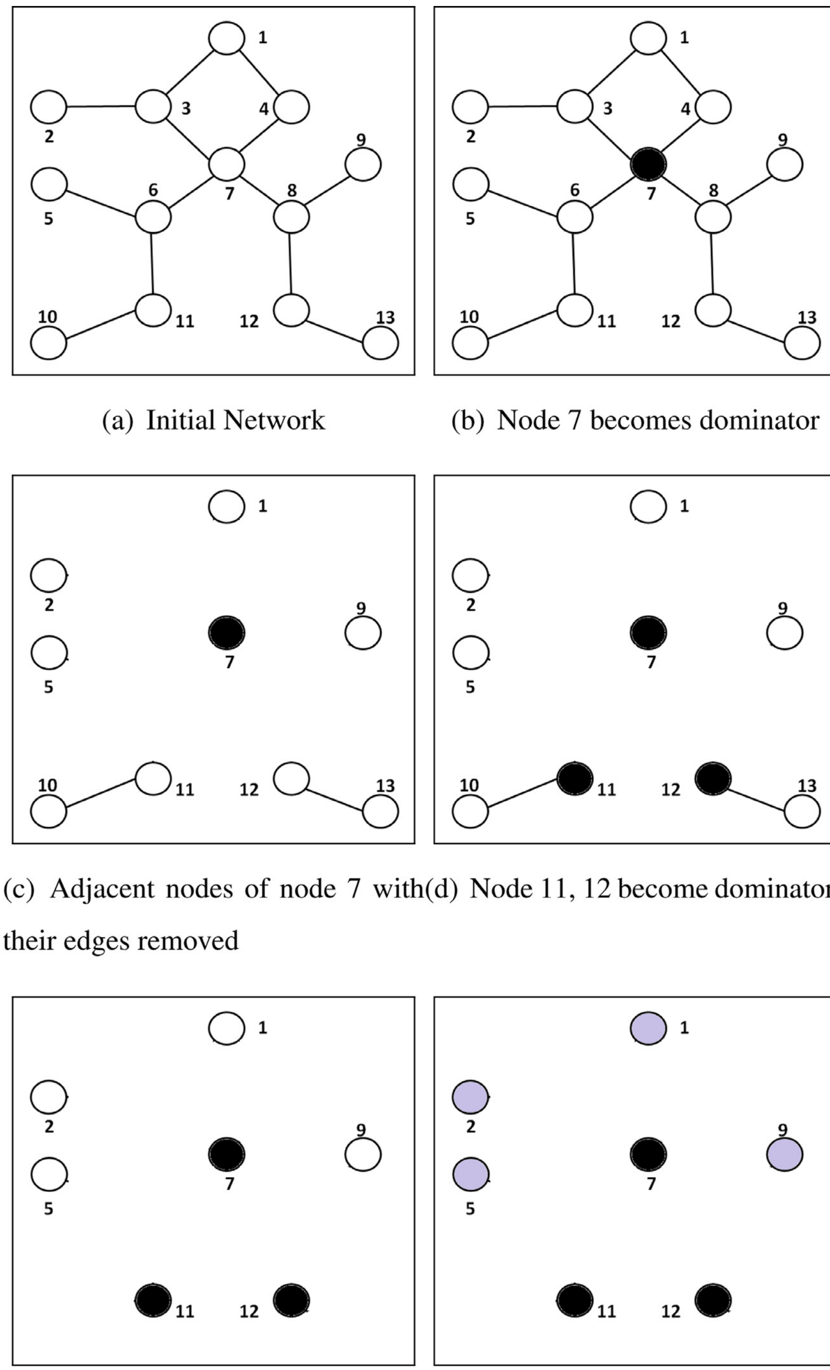
5.1. PDS Construction

In the first phase of our algorithm, we construct a PDS as an MIS in a greedy manner. The construction of the PDS is through a simple degree based algorithm which uses 1-hop and 2-hop neighbours' information of each node. As discussed in the previous section, an MIS node can be separated from its nearest MIS node by at most three hops, the algorithm checks only the 1-hop and 2-hop neighbours' information of each node. Before the start of the algorithm, all the nodes are coloured white. The algorithm finds the dominators and virtual dominators and colours them black and grey respectively. In each round, the algorithm chooses a node u as the dominator if u has a degree higher than its 1-hop and 2-hop neighbours. In case of tie, first the original degree and then the node ID is considered to break the tie. The algorithm can select multiple nodes as dominators in a particular round. The colour of the nodes selected as dominators become black. In a particular round, after the selection of dominators, all the adjacent nodes of the dominators become dominatees and these dominatees with their incident edges are deleted from the topology. The degree (effective degree) of the remaining nodes are updated at the end of each round. The algorithm repeats the above procedure round after round until there is no white node with degree greater than zero. At the end, the nodes with degree zero are considered as virtual dominators and are coloured grey.

We illustrate the PDS construction process with a distribution of nodes shown in Fig. 3(a), in which the nodes 1, 2, 5, 7, 9, 11 and 12 form the PDS. Fig. 3(b)–(f) shows the construction of the PDS in step wise. In the first round, node 7 is selected as dominator as it is having highest degree among all its 1-hop neighbours (3, 4, 6, 8) and 2-hop neighbours (1, 2, 5, 9, 11, 12). Nodes 10 and 13 are not selected as dominators because they have degree lesser than their 1-hop neighbours 11 and 12 respectively. After the selection of node 7 as dominator, all its 1-hop neighbours with their incident edges are removed from the network. In the next round, among the nodes with effective degree greater than zero (10, 11, 12, 13), node 11 and 12 are selected as dominators. Although node 11 has the effective degree same as that of node 10, node 11 is selected because of its higher original degree. Similarly, node 12 is chosen over 13. After the selection of dominator (7, 11, 12), the remaining white nodes 1, 2, 5, 9 are with effective degree zero. So, these nodes are selected as virtual dominators and are coloured grey. We also perform a simple experiment to substantiate that our PDS has smaller cardinality than the MIS selected from other CDS construction schemes. To show that the size of the PDS constructed by our proposed is smaller than the MIS selected from other CDS construction schemes we conducted an experiment. We compare the size of the PDS constructed by our algorithm with the sizes of MISs obtained from collaborative cover heuristic [18] for various sizes of connected networks. Note that collaborative cover heuristic [18] produces smaller MISs than previous MIS selection techniques [13,14]. We run each of the approaches for 100 times and the average result is shown in Fig. 4 which shows that our PDS has smaller sizes in comparison to collaborative cover heuristic [18].

5.2. Improved Steiner Tree construction

In the second phase, to connect all dominators and virtual-dominators, the algorithm selects the Steiner nodes from the dominatees in a greedy manner. The main objective of this phase is to select a minimum number of dominatees as Steiner nodes to construct the CDS. At the beginning of this phase, each of the nodes present in the CDS (dominators and virtual-dominators), forms separate components. For each dominatee, the algorithm calculates *connection-load* which is the current count of number of components it is connected with. In each round, the algorithm selects a dominatee as the connector if the dominatee has a *connection-load* higher than its rival dominatees. In case of a tie, first the node degree and then the node ID is considered to break the tie. The selected dominatees become connectors by changing their colour to



(a) Initial Network
(b) Node 7 becomes dominator
(c) Adjacent nodes of node 7 with their edges removed
(d) Node 11, 12 become dominators
(e) Adjacent nodes of node 11, 12 with their edges removed
(f) Nodes 1, 2, 5, 9 becomes virtual dominator

Fig. 3. Example showing PDS construction (Phase 1 of DCMCDS).

blue. It forms a new component by combining the components it is connecting with itself. After each round, the algorithm updates the *connection-load* of each remaining dominatees. The procedure is repeated until all the dominators and virtual-dominators do not form a single component.

We illustrate the Steiner Tree construction phase through the PDS computed in Phase 1 of this algorithm. Post PDS construction is shown in Fig. 5. Fig. 5(a) shows the PDS in which we find the nodes 7, 11, 12 are dominators and nodes 1, 2, 5, 9 are virtual-dominators. The remaining nodes 3, 4, 6, 8, 10, 13 are dominatees. The dominators and virtual-dominators form individual components. The *connection-load* of the dominatees 3, 4, 6, 8, 10, 13 are 3, 2, 3, 3, 1, 1 respectively. In the first round, node 3 is chosen as the connector although its rival dominatees 6 and 8 have same connection-load (3) and degree (3), however node 3 is with least node-ID. Node 3 forms a new component 1-2-3-7. Now the connection-load of the remaining dominatees 4, 6, 8, 10 becomes 1, 3, 3, 1, 1 respectively. So, among nodes 6 and 8, node 6 is chosen

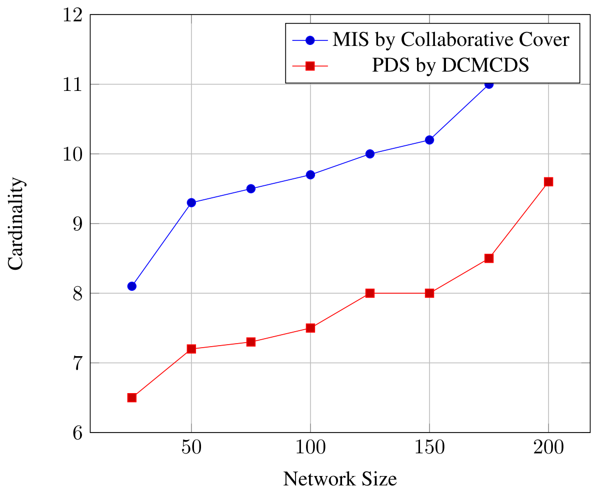


Fig. 4. Performance comparison of PDS construction phase with MIS selection scheme.

as the connector in the second round since it has smaller node-ID (connection-load and degree are the same as node 8). Node 6 forms a new component 1-2-3-6-7-11. In a similar way, in the next round, node 8 is chosen as the connector and forms a single component consisting of all the dominators and virtual-dominators.

5.3. Removal of redundant dominators

This phase of DCMCDS reduces the CDS size by removing redundant dominators and virtual-dominators (if any). A node in the PDS (dominator or virtual-dominator) is redundant if after removing it from the CDS, the resultant CDS is still connected and dominates all other non-CDS nodes. A virtual-dominator is downgraded to a dominatee in two cases: (1) it is connected to the CDS through only one connector or (2) it is connected to the CDS through two connectors and they are adjacent. In all other cases, it is upgraded to a dominator. If the dominatees of a dominator x are adjacent to some other dominators or connectors, then x can be downgraded or not according to: If x is connected to the CDS by one connector or if it is connected to the CDS by two connectors and they are adjacent, then the dominator is downgraded to a dominatee. Otherwise, the status of the node x remains as it is.

After getting the initial CDS as shown in Fig. 5(d), we can reduce the size of it by downgrading the virtual-dominators 1, 2, 5 and 9 as they are connected to the CDS by one connector. The final CDS is shown in Fig. 6.

6. Distributed DCMCDS scheme

In this section, we discuss the details of the distributed algorithm of DCMCDS scheme. During the execution of the algorithm, each node of the network u, maintains the following variables:

- **colour** (u_{colour}): This variable shows the current status of the node. The initial colour of each node is white. The nodes change their colours either to black, grey, yellow or blue when their status changes to either dominator, virtual-dominator, dominatee or connector respectively.
- **nodeID** (u_{ID}): An ID, which is unique for each node.
- **originalDegree** ($u_{odegree}$): This variable stores the initial degree of the node in the graph.

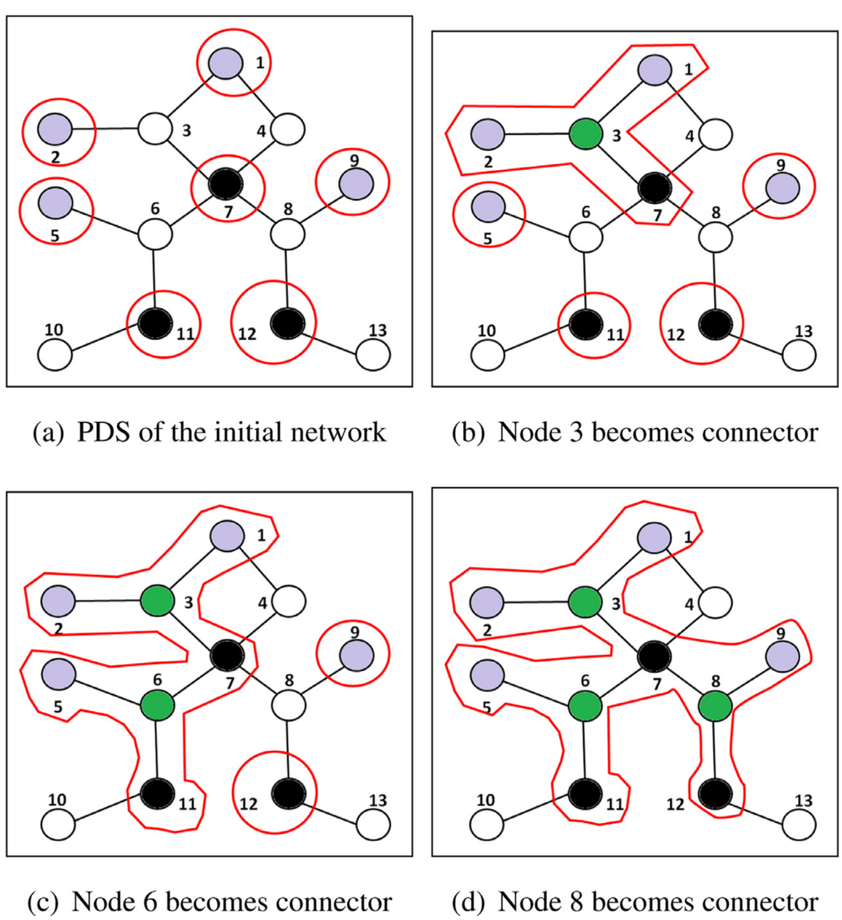


Fig. 5. Example showing Steiner Tree construction (Phase 2 of DCMCDS).

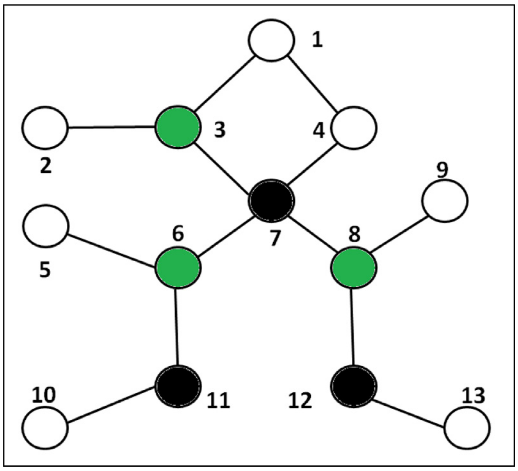


Fig. 6. Final CDS after removing redundant nodes.

- **effectiveDegree** ($u_{edegree}$)**:** This variable stores the effective degree of a node u in the graph. Effecive degree of a node varies from time to time. Effective degree of a node at a particular moment is the number of white nodes adjacent to that node at that particular moment.
- **componentID** (u_{cID})**:** An ID to demarcate nodes belonging to different components. All nodes in the same component have the same *componentID*, which is the least *nodeID* of all the dominators / connectors forming the component.
- **1HopNebsTable** ($N_1(u)$)**:** A table stored at node u which records the *nodeID*, *colour*, *originalDegree* and *effectiveDegree* of all its adjacent nodes.
- **2HopNebsTable** ($N_2(u)$)**:** A table stored at node u which records the *nodeID*, *colour*, *originalDegree*, *effectiveDegree*, *mutualNeighbour*, *mnColour* for its distance-2 neighbours (excluding itself). $N_2(u)$ contains even those 2-hop neighbours of u which are also adjacent to u. The multi-valued attribute *mutualNeighbour* in $N_2(u)$, corresponding to a 2-hop neighbour v, contains the *nodeIDs* of all the nodes that are adjacent to both u and v. The multi-valued attribute *mnColour* stores the colour of the corresponding *mutualNeighbour*.
- **cdsList** ($u_{cdsList}$)**:** This list contains the *nodeIDs* of the members (dominators / virtual-dominators / connectors) of the component, to which node u belongs.
- **connectionCount** (u_{ccnt})**:** This variable records the number of independent components adjacent to u.
- **rivalList** ($u_{rivalList}$)**:** This list contains the *nodeIDs* of the dominatees which are adjacent to the same component, to which node u is adjacent.

In the following sub-sections, first we discuss each of the phases of our distributed DCMCDS scheme in detail. At the end of this section, we discuss the phase transition of the proposed distributed algorithm.

6.1. Node initialization and neighbourhood table creation

In this phase of the distributed DCMCDS, each of the nodes initialize their variables and neighbourhood tables by sending and receiving the following messages:

- HELLO: Each node broadcasts this message to inform about its presence to its neighbours.
- OWN_INFO: Through this message a node informs its *originalDegree* to its neighbours.
- NEB_INFO: This message is sent by a node, to pass on its detailed neighbour information, to all of its neighbours.

Algorithm 1describes the detail initialization procedure.

Algorithm 1 Node initialization.

1: Each node u, initializes its variables as $\langle u_{colour} \leftarrow white\rangle$, $\langle u_{cID} \leftarrow nil\rangle$, $\langle u_{odegree} \leftarrow 0\rangle$, $\langle u_{edegree} \leftarrow 0\rangle$, $\langle u_{ccnt} \leftarrow 0\rangle$, $\langle N_1(u) \leftarrow nil\rangle$, $\langle N_2(u) \leftarrow nil\rangle$.
2: Each node broadcasts a HELLO message.
3: After a lapse of time τ, every node u ascertains its number of neighbours from the number of HELLO messages received and updates its state variable *originalDegree* and *effectiveDegree* as $u_{odegree} \leftarrow u_{edegree} \leftarrow$ number of HELLO messages received.
4: A node u after updating its state variable *originalDegree*, broadcasts a message OWN_INFO $= \langle u_{ID}, u_{odegree}\rangle$.
5: A node v adjacent to u, on receiving OWN_INFO message from u, adds a tuple $\langle u_{ID}, white, u_{odegree}, u_{odegree}\rangle$ to $N_1(v)$.
6: When all the OWN_INFO messages are delivered, each node v broadcasts a message NEB_INFO $= \langle v_{ID}, N_1(v)\rangle$.
7: Every node w, which is a distant-2 neighbour of u, on receiving message NEB_INFO from v, adds all tuples in $N_1(v) - \{\langle w_{ID}, white, w_{odegree}, w_{odegree}\rangle\}$ to $N_2(w)$ with $mutualNeighbour \leftarrow v_{ID}$ and $mnColour \leftarrow$ white.

6.2. Distributed PDS construction

In this phase, each node uses its neighbourhood information (stored in its *1HopNebTable* and *2HopNebTable*) to decide whether it can become a dominator or not. A node on becoming a dominator, virtual-dominator or a dominatee, spread its new status information up to two hops, so that its 1-hop and 2-hop neighbours can update their tables. In each round, one or more nodes become either a dominator or a virtual-dominator. At the beginning of each round, the white nodes check their updated *1HopNebsTable* and *2HopNebsTable*, to decide whether they can become a dominator in the current round or not. When all the nodes change their colour from white to some other colour, the PDS construction is over. This phase constructs the PDS in a distributed manner using the following messages:

- DOMINATOR: A node broadcasts this message when it becomes a dominator.
- DOMINATEE: A node broadcasts this message when it becomes a dominatee.
- VIRTUAL_DOMINATOR: A node broadcasts this message when it becomes a virtual-dominator.
- UPDATE_NEB_INFO: When the *effectiveDegree* of a node is changed, it informs this to its neighbours through this message.
- UPDATE_NODE_COL: This message is sent by a dominatee node, to inform about the change in colour of any of its neighbours to its other neighbours.

The detail procedure for distributed PDS construction is given in Algorithm 2.

6.3. Distributed Steiner Tree construction

At the beginning of this phase, nodes are coloured either black, grey, or yellow. Each of the black and grey node forms a separate component and stores its own *nodeId* in its *cdsList* as it is the only node of its component so far. In each round of this phase, the CDS members (black / grey / blue nodes) of each component send a request message to their adjacent yellow nodes (dominatees) to get their *connectionCount* (number of independent components they are connected to). The dominatees after getting all the request messages, reply to their adjacent CDS members with their own *connectionCount*. The CDS members of each component

Algorithm 2 Distributed PDS construction.

Input: A connected graph $G(V, E)$.
Output: PDS of the graph $G(V, E)$ formed by black and grey nodes.

1: Each white node u, checks itself after each period of time, τ to decide whether it can be a dominator or not, till it is no longer white in colour.
2: A white node u, elects itself as a dominator, if any of the following conditions apply:
  (i) $u_{edegree} > v_{edegree} \forall$ white nodes $v \in N_1(u) \cup N_2(u)$.
  (ii) $u_{edegree} \geq v_{edegree} \forall$ white nodes $v \in N_1(u) \cup N_2(u)$, but $u_{odegree} > w_{odegree} \forall$ white nodes $w \in N_1(u) \cup N_2(u)$ where $u_{edegree} = w_{edegree}$.
  (iii) $u_{edegree} \geq v_{edegree}$ and $u_{odegree} \geq v_{odegree} \forall$ white nodes $v \in N_1(u) \cup N_2(u)$, but $u_{ID} < w_{ID} \forall$ white nodes $w \in N_1(u) \cup N_2(u)$ where $u_{edegree} = w_{edegree}$ and $u_{odegree} = w_{odegree}$.
3: A white node u on becoming a dominator, performs the following operations:
  (i) Updates its colour as $\langle u_{colour} \leftarrow black \rangle$.
  (ii) Updates the colour of each of its 1-hop white neighbours to yellow in $N_1(u)$
  (iii) Update its *componentID* as $\langle u_{cID} \leftarrow u_{ID} \rangle$.
  (iv) Broadcasts message DOMINATOR(u_{ID}).
4: A white node v on receiving DOMINATOR(u_{ID}) message from a node u, performs the following operations:
  (i) Updates its state variable as $\langle v_{colour} \leftarrow yellow \rangle$
  (ii) Updates the colour of node u in $N_1(v)$ and $N_2(v)$ as $\langle u_{colour} \leftarrow black \rangle$.
  (iii) Changes the colour of the node $x \in N_2(v)$ to *yellow* if $x_{colour} = white$ and the *mutualNeighbour* of x is u.
  (iv) Broadcasts message DOMINATEE(v_{ID}, u_{ID}).
5: A white node w on receiving DOMINATEE(v_{ID}, u_{ID}) message from node v, performs the following operations:
  (i) Updates its $effectiveDegree$ as $\left\langle w_{edegree} \leftarrow w_{edegree} - 1 \right\rangle$
  (ii) Updates the colour of node v in $N_1(w)$ and $N_2(w)$ as $\langle v_{colour} \leftarrow yellow \rangle$.
  (iii) Updates the colour of node u in $N_2(w)$ as $\langle u_{colour} \leftarrow black \rangle$.
  (iv) Updates the colour of the *mutualNeighbour* v in $N_2(w)$.[Note that when v becomes a dominatee it is deleted from the network (refer to Step 12 of Algorithm 1 in [31]). So, the 2-hop neighbours of w, only through v, are no more the 2-hop neighbours. Henceforth, the 2-hop neighbours with non-white mutual neighbour only are not considered as 2-hop neighbours of w during dominator election process in the next round (Step 2).]
  (v) Broadcasts UPDATE_NEB_INFO ($w_{ID}, w_{edegree}, v_{ID}$) message .
6: A yellow node w on receiving DOMINATEE(v_{ID}, u_{ID}) message from node v, performs the following operations:
  (i) Updates the colour of node v in $N_1(w)$ as $\langle v_{colour} \leftarrow yellow \rangle$.
  (ii) Updates the colour of node u in $N_2(w)$ as $\langle u_{colour} \leftarrow black \rangle$.
  (iii) Broadcasts UPDATE_NODE_COL ($v_{ID}, yellow$) message.
7: A node p on receiving UPDATE_NEB_INFO ($w_{ID}, w_{edegree}, v_{ID}$) message from w, performs the following operations:
  (i) Updates the colour of node v in $N_1(p)$ and $N_2(p)$ as $\langle v_{colour} \leftarrow yellow \rangle$.
  (ii) Updates the $effectiveDegree$ of w in $N_1(p)$ as $w_{edegree}$.
8: At any instance, when the $effectiveDegree$ of a white node u gets decremented to zero, it become a virtual-dominator and updates its colour as $\langle u_{colour} \leftarrow grey \rangle$. It informs its new role by broadcasting VIRTUAL_DOMINATOR (u_{ID}) message to its neighbours.
9: A yellow dominatee v on receiving the DOMINATOR message from u:
  (i) Updates the colour of node u in $N_1(v)$ as $\langle u_{colour} \leftarrow black \rangle$.
  (ii) Broadcasts UPDATE_NODE_COL ($u_{ID}, black$) message.
10: A yellow node v on receiving the VIRTUAL_DOMINATOR message from u:
  (i) Updates the colour of node u in $N_1(v)$ as $\langle u_{colour} \leftarrow grey \rangle$.
  (ii) Broadcasts UPDATE_NODE_COL ($u_{ID}, grey$) message.
11: A node p on receiving UPDATE_NODE_COL ($nid, ncolour$) message, updates the colour of node $x \in N_1(p) \cup N_2(p)$ with $x_{ID} = nid$ as $\langle x_{colour} \leftarrow ncolour \rangle$.
12: Each white node u after waiting for a period of time τ, broadcasts the message NEB_INFO = $\langle u_{ID}, N_1(u) \rangle$, if the effective degree of any of its 1-hop neighbours has changed in the last round.
13: A node v on receiving NEB_INFO = $\langle v_{ID}, N_1(u) \rangle$ message from u updates its 2-hop neighbours' information present in $N_2(u)$.
14: Phase-I terminates when eventually all nodes change their colour from white to some other colour.

after getting these reply messages, prepare a list of their adjacent dominatees (known as *rivalList*). They circulate their own *rivalList* among other CDS members of the same component to prepare the complete *rivalList* of the whole component. Once the CDS members of a component prepare the complete *rivalList*, they send this *rivalList* along with their *cdsList* together, to their yellow neighbours. The *rivalList* contains the nodeIds of the rival members along with their original degree. Each of the yellow dominatees, after getting the rival information (*rivalList*) of the components they are connected with, arranges the rival nodes in non-increasing order of *connectionCount*. After this, it also decides to participate in this process or not. If a yellow node finds all its rivals are connected to the same component to which it is connected, and each of its 2-hop black neighbours are present in one of the received *cdsLists*, then it decides not to participate in this process anymore. If a yellow node decides to participate and finds itself ranked first in its *rivalList*, then it becomes a connector. A node on becoming a connector, changes its colour to blue, forms a new component by merging the components it connects with itself. It assigns the *componentID* of the new component as the minimum of the *componentIDs* of the merged components and its own *nodeID*. The *cdsList* of the new component contains all the existing CDS member of the merged components and the connector. The connector sends the updated component information to all its neighbours which in turn spread the component information to all the members of the same component. The colour of the connector is also sent to its 2-hop neighbours. After waiting for a period of time, all the CDS nodes start their next round by sending the request messages for getting the *connectionCount* of their yellow neighbours. This phase continues until there is no yellow node that can still participate in this phase.

This phase uses the following messages to constructs the Steiner Tree:

- CONN_INFO_REQ: The black/grey/blue nodes of a component broadcast this message to their yellow neighbours to get their *connectionCount*.
- CONN_INFO_REP: The yellow dominatees send their *connectionCount* to their blue/grey/black neighbours through this message.
- COMP_RIVAL_INFO: The black/grey/blue nodes of a component broadcast this message, to inform the yellow nodes about their *rivalList*. They also send their *cdsList* with this message, which is used by the yellow nodes to decide whether to participate in this phase or not.
- RIVAL_INFO: The black/grey/blue nodes of a component, send this message to their component members, to prepare the complete *rivalList* of the whole component to which they belongs.
- CONNECTOR: A yellow node broadcasts this message, when it becomes a connector to notify its neighbours about its new role.
- UPDATE_COMP_INFO: Black/grey/blue nodes of a component, send this message to their component members, to update their *componentID* and *cdsList*.
- UPDATE_NODE_COL: This message is sent by a dominatee node, to inform about the change in colour of any of its neighbours to its other neighbours.

The detail procedure for distributed construction of Steiner Tree is given in the Algorithm 3.

Algorithm 3 Distributed Steiner Tree construction.

Input: A connected graph $G(V, E)$ with its PDS formed by the black and grey nodes.
Output: Connected Dominating set of the graph $G(V, E)$ formed by black, grey and blue nodes.

1: At the end of PDS construction phase, each black and grey node x, forms isolated separate components and initiates the Steiner Tree construction phase as follows:
   (i) Updates its *cdsList* as $\langle x_{cdsList} \leftarrow \{x_{ID}\}\rangle$.
   (ii) Initializes its *componentID* as its *ID* by $\langle x_{cID} \leftarrow x_{ID}\rangle$
   (iii) Broadcasts the CONN_INFO_REQ($x_{ID}, x_{cID}, x_{cdsList}$) message.
2: Each yellow node u, after getting the CONN_INFO_REQ messages from all of its black/grey/blue neighbours:
   (i) Calculates its *connectionCount* (u_{ccnt}), which is the count of number of independent components it is connected to.
   (ii) Broadcasts a reply message CONN_INFO_REP($u_{ccnt}, u_{ID}, u_{odegree}$) to all its black / grey / blue neighbours.
3: Each black / grey / blue node w, on receiving a CONN_INFO_REP message from one of its yellow neighbours u, updates its *rivalList* by including the node u in it with its details.
4: Each black / grey / blue node w, after receiving CONN_INFO_REP messages from all of its yellow neighbours:
   (i) Circulates its *rivalList* among other component members (if any) to prepare the *rivalList* of the whole component.
   (ii) After preparing the complete *rivalList* of the whole component, it broadcasts the message COMP_RIVAL_INFO($w_{ID}, w_{RivalList}, w_{cdsList}$).
5: Each black / grey / blue node w, to prepare the complete *rivalList* of the whole component, circulates its *rivalList* among other component members in the following way:
   (i) If it is the only member of the component, then its *rivalList* is the *rivalList* of the whole component.
   (ii) Else if it has only one component neighbour, then it sends the message RIVAL_INFO($w_{cID}, w_{RivalList}$) to that component neighbour.
   (iii) Else it waits to receive the RIVAL_INFO message from all its component neighbours except one.
6: Each black / grey / blue node z, on receiving RIVAL_INFO($w_{cID}, w_{rivalList}$) message from the same component neighbour w:
   (i) Updates its *cdsList* as $\langle z_{rivalList} \leftarrow z_{rivalList} \cup w_{rivalList}\rangle$.
   (ii) If it has received the RIVAL_INFO message from all its component neighbours except one, then it sends the message RIVAL_INFO($z_{cID}, z_{rivalList}$) to the component neighbour from which it has not received the RIVAL_INFO message.
   (iii) Else if it has received the RIVAL_INFO message from all its component neighbours, then it sends the message RIVAL_INFO($w_{cID}, w_{RivalList}$) to the component neighbours to which it has not sent the RIVAL_INFO message.
7: Each yellow node after receiving COMP_RIVAL_INFO messages from all its adjacent black / grey / blue nodes, orders its received rival nodes according to the following criteria:
   (i) All the dominatees are arranged in non-increasing order of their *connectionCount*.
   (ii) If two dominatees have the same *connectionCount*, then the one with a higher *originalDegree* is ranked higher.
   (iii) If two dominatees have the same *connectionCount* and *originalDegree*, then the one with a smaller *nodeID* is ranked higher.
8: Each yellow node w, after preparing the *rivalList*, decides whether to further participate in this phase or not. It participates no longer in the process, if it satisfies the following two conditions:
   a) The *conectionCount* of itself and its rivals is 1. This indicates that the yellow node and its rivals are adjacent to the same component.
   b) There is no black node in its *2HopNebsTable* that does not occur in any of the received *cdsList*s from their black/grey/blue nodes.
9: If a yellow node w decides to participate in the process and finds itself ranked first in its *rivalList*, then it becomes a connector and executes the following actions:
   (i) Updates its state variable as $\langle w_{colour} \leftarrow blue\rangle$
   (ii) $w_{cdsList} \leftarrow$ union of *cdsList* of the black/blue nodes it is connected with and its own ID, w_{ID}.
   (iii) $w_{cID} \leftarrow$ minimum of *componentID*s of the black / blue nodes it is connected with and its own ID, w_{ID}.
   (iv) Broadcasts CONNECTOR(w_{ID}, w_{cID}, $w_{cdsList}$) message for its neighbours to notify them about its new role.
10: A node x, on receiving CONNECTOR (w_{ID}, w_{cID}, $w_{cdsList}$) message from w, executes the following:
   (i) Update the colour of w as $\langle w_{colour} \leftarrow blue\rangle$in its $N_1(x)$.
   (ii) Broadcasts UPDATE_NODE_COL ($w_{ID}, blue$) to its neighbours.
   (iii) If $x_{ID} \in w_{cdsList}$
      a) Update its *componentID* as $\langle x_{cID} \leftarrow w_{cID}\rangle$
      b) Update its *cdsList* as $\langle x_{cdsList} \leftarrow w_{cdstList}\rangle$
      c) Broadcast UPDATE_COMP_INFO (w_{cID}, $w_{cdsList}$) to its neighbours if it has not send the same $\langle w_{cID}, w_{cdsList}\rangle$ before through any of the CONNECTOR or UPDATE_COMP_INFO message.
11: A node y, on receiving UPDATE_COMP_INFO (w_{cID}, $w_{cdsList}$) message from w, executes the following:**If** it has not send the same $\langle w_{cID}, w_{cdsList}\rangle$ before through either of the CONNECTOR or UPDATE_COMP_INFO message and $y_{ID} \in w_{cdsList}$ **then**
   (i) Update its *componentID* as $\langle y_{cID} \leftarrow w_{cID}\rangle$.
   (ii) Update its *cdsList* as $\langle y_{cdsList} \leftarrow w_{cdstList}\rangle$.
   (iii) Broadcasts UPDATE_COMP_INFO (w_{cID}, $w_{cdsList}$) message.
12: After waiting for a certain period of time τ, each black / grey / blue node sends the CONN_INFO_REQ(x_{ID}, x_{cID}, $x_{cdsList}$) message to all its neighbours again and the procedure from step 2 onwards is repeated.
13: This phase of connector selection ends when no yellow dominatee participates further. When no yellow node participates further, no new connectors will be created. Due to which no more UPDATE_COMP_INFO messages will be sent. So black / grey / blue nodes will not send any more CONN_INFO_REQ messages.

6.4. Distributed removal of redundant dominators

In this phase, each grey and black node checks whether to downgrade itself or not to reduce the overall CDS size. If a grey node finds that either it is connected to the CDS by only one CDS node or the CDS nodes (in case of multiple connection with CDS nodes) are connected without it, then it downgrades itself to a dominatee, otherwise it upgrades itself to a dominator. After it upgrades / downgrades it sends its new role to its neighbours, which in turn inform their neighbours. However, if a black node satisfies the same condition (as discussed above for a grey node), it has to check whether all its dominatees have some alternative dominators or not. If it finds that all its dominatees have some alternative dominators, then it downgrades itself to a dominatee and informs its neighbours, which in turn inform their neighbours, otherwise it remains as a dominator. A black node, to find out the availability of the alternative dominators of its dominatees, sends a request message to its dominatees and waits for their replies. If it gets the TRUE reply from all of them, then it downgrades itself, otherwise it cancels its previous request by sending a cancel message to all of them. A dominatee which gets a request message to check its alternative dominators, sends a TRUE reply to the first dominator from which it has received the request. After that, it waits for either the change in status of that dominator (to which it has sent the TRUE reply), or the cancel message from it. If it finds that the dominator has downgraded to a dominatee, it sends a FALSE reply to all of the alternative dominator requests after that. However, if it gets a cancel message from the dominator to which it has already sent the TRUE reply, then it sends the TRUE reply to the next dominator out of the dominators waiting in the queue for its reply. This phase removes some of the redundant dominating nodes in a distributed manner by using the following messages:

- UPGRADE_DOM: A grey virtual-dominator broadcasts this message to its neighbours when it decides to change its role from a virtual-dominator to a dominator.
- DOWNGRADE_DOM: This message is sent by either a virtual-dominator or a dominator when it decides to downgrade itself to a dominatee.
- ALT_DOMINATOR_REQ: A black node sends this request message to its dominatees to know whether they have some alternative dominators or not.

- ALT_DOMINATOR_REP: A yellow dominatee sends TRUE reply with this message if it is adjacent to some dominator/connector other than the dominator from which it received the ALT_DOMINATOR_REQ message. Otherwise, it returns FALSE reply with this message.
- ALT_DOMINATOR_REQ_CANCEL: If a black node receives FALSE message from any one of the yellow nodes through the ALT_DOMINATOR_REP message it sends this message to all its yellow neighbours.

The detail distributed procedure for removing the redundant dominating nodes is given in the Algorithm 4.

Algorithm 4 Distributed removal of redundant dominators.

Input: A connected graph $G(V, E)$ with its CDS formed by the black, grey and blue nodes.
Output: A potentially smaller CDS of the graph $G(V, E)$ after removing some of the redundant dominators and virtual-dominators.

1: Each grey node v changes its state according to the following (Steps 2 - 10):
2: **if** there exists only one connector $x \in N_1(v)$ with $x_{colour} = blue$ or there exists at least two connectors $x, y \in N_1(v) \cap N_2(v)$ with $x_{colour} = y_{colour} = blue$ and *mutualNeighbour* corresponding to x being y and vice-versa **then**
3: Updates its colour as $\langle v_{colour} \leftarrow yellow \rangle$.
4: Broadcasts the UPDATE_NODE_COL($v_{ID}, yellow$) message.
5: Broadcasts the DOWNGRADE_DOM(v_{ID}) message.
6: **else**
7: Updates its colour as $\langle v_{colour} \leftarrow black \rangle$
8: Broadcasts the UPDATE_NODE_COL($v_{ID}, black$) message.
9: Broadcasts the UPGRADE_DOM(v_{ID}) message.
10: **end if**
11: Each black node v may changes its state according to the following (Steps 12 - 21):
12: **if** there exists only one connector $x \in N_1(v)$ with $x_{colour} = blue$ or there exists at least two connectors $x, y \in N_1(v) \cap N_2(v)$ with $x_{colour} = y_{colour} = blue$ and *mutualNeighbour* corresponding to x being y and vice-versa **then**
13: Node v broadcasts a request message ALT_DOMINATOR_REQ for all its yellow neighbours and wait for their replies.
14: **if** It receives ALT_DOMINATOR_REP(TRUE) message from all its yellow neighbours **then**
15: Updates its colour as $\langle v_{colour} \leftarrow yellow \rangle$.
16: Broadcasts the UPDATE_NODE_COL($v_{ID}, yellow$) message.
17: Broadcasts the DOWNGRADE_DOM(v_{ID}) message.
18: **else**
19: Broadcasts the ALT_DOMINATOR_REQ_CANCEL message.
20: **end if**
21: **else**
22: The black node v does not change its state.
23: **end if**
24: A yellow node after receiving the first ALT_DOMINATOR_REQ message, sends the ALT_DOMINATOR_REP(TRUE) message to that node and enters into the waiting state if it is connected to some other dominator or connector.Otherwise, it sends the ALT_DOMINATOR_REP(FALSE) message.
25: A yellow node in the waiting state remains in that state until it receives either ALT_DOMINATOR_REQ_CANCEL or DOWNGRADE_DOM message from the node to which it has already sent the ALT_DOMINATOR_REP(TRUE) message.
26: A yellow node in the waiting state, inserts the new nodes in a queue from which it receives the new ALT_DOMINATOR_REQ messages.
27: When a yellow node in the waiting state receives the DOWNGRADE_DOM(v_{ID}) message:
  (i) It sends ALT_DOMINATOR_REP(FALSE) message to the nodes in the queue and comes out of the waiting state.
  (ii) After this if it receives ALT_DOMINATOR_REQ messages from any node, it sends the ALT_DOMINATOR_REP(FALSE) message to that node immediately.
28: When a yellow node in the waiting state receives the ALT_DOMINATOR_REQ_CANCEL message, it sends ALT_DOMINATOR_REP(TRUE) message to the first nodes in the queue (if the queue is non-empty) and remains in the waiting state. In case of empty queue it comes out of the waiting state.
29: A node x on receiving DOWNGRADE_DOM(v_{ID}) from node v:
  (i) Updates the colour of node v in $N_1(x) \cup N_2(x)$ as $\langle v_{colour} \leftarrow yellow \rangle$
  (ii) Broadcasts the UPDATE_NODE_COL($v_{ID}, yellow$) message..
30: A node x on receiving UPGRADE_DOM(v_{ID}) from node v:
  (i) Updates the colour of node v in $N_1(x) \cup N_2(x)$ as $\langle v_{colour} \leftarrow black \rangle$
  (ii) Broadcasts the UPDATE_NODE_COL($v_{ID}, black$) message.

6.5. Phase transition

In any distributed algorithm, phase transition is very important. We handle the phase transition of our distributed algorithm in the following way. Each node after creating its *1HopNebTable* and *2HopNebTable* should start the Distributed PDS Construction phase. A non-white node can begin the Distributed Steiner Tree Construction phase if it finds all its neighbours are non-white. The Distributed Steiner Tree Construction phase messages are queued by any node that still has white neighbours, until all its neighbours become non white. These queued messages are handled by the node when it finds all its neighbours have become non white. In the Distributed Steiner Tree Construction phase, each black dominator and grey virtual-dominator keeps on sending CONNECTION_INFO_REQ messages while they find some of the yellow nodes participate in this phase (See the Line 8 of Algorithm 3 for yellow node participation condition). If none of the yellow nodes participate in this phase, then the black or grey nodes will not receive any UPDATE_COMP_INFO messages. So, if the black or grey nodes do not receive the UPDATE_COMP_INFO messages up to a period of time, then it can be sure that the Distributed Steiner Tree Construction phase is over. Any black dominator or grey virtual-dominator which finds the Distributed Steiner Tree Construction phase is over can start the last phase of the distributed CDS construction algorithm to remove the redundant dominators or virtual-dominators.

7. Algorithm analysis

In this section, first we find the performance ratio of our proposed distributed algorithm. Later we also find the time and message complexity of our proposed scheme. To do this, we use certain lemmas and theorems. The detail proofs of all of these can be found in this section.

Lemma 7.1. *At the end of distributed PDS construction phase, an MIS is formed by the black and grey nodes resulting from the Algorithm 2.*

Proof. The distributed PDS construction terminates when each white node changes its colour. Algorithm 2 ensures that every yellow node is adjacent to at least one black node. Hence, by Definition 1, the set of black and grey nodes form a Dominating Set. We can also observe that when a node changes its colour to black all its neighbours become yellow. Similarly, a node changes its colour to grey when it finds that all its neighbours have changed their colour to yellow. So, no node in the DS will find its neighbour in the set. So, the DS is independent. Also, DS is maximal because every omitted (yellow) node in the graph is dominated. Hence, by Definition 2.1, the set of black and grey nodes form an MIS. □

Theorem 7.2. *Distributed DCMCDS constructs a PDS with the property: the distance between any pair of complementary subsets of the PDS have a distance of exactly two or three hops.*

Proof. In order to prove this property about our constructed PDS, we first need to show that for a $|PDS| > 1$, if $u \in PDS$, then the nearest black or grey neighbour of u in terms of number of hops is separated from u by at most three hops. We prove this by contradiction for any PDS whose cardinality is greater than 1. Let us assume that $u \in PDS$ and the nearest black or grey node to u, in terms of number of hops, is separated from u by more than three hops. Let v be a strictly 2-hop neighbour of u which is not adjacent to u. If such a v does not exist, then it implies that all 2-hop neighbours of u are also its 1-hop neighbours, which in turn indicates that u dominates the whole connected graph. This contradicts $|PDS| > 1$. So, for $|PDS| > 1$, let v be a non-adjacent 2-hop neighbour of u.

Case I : *v is either a dominator or a virtual-dominator*. This implies that v is in PDS. So, we have a node $v \in PDS$ which is two hops away from u. This contradicts our assumption. So this case is not possible.

Case II : *v is neither a dominator nor a virtual-dominator*. By Lemma 7.1, the PDS, which comprises all the black and grey nodes, is a maximal independent set. This implies that v is adjacent to at least one node in the PDS. Let $w \in PDS$ be adjacent to v. This means that u and w are 3-hop neighbours. This also contradicts our assumption. So, this case is not possible as well. Thus, our assumption does not hold true for any $u \in PDS$. This implies that u is separated from its nearest black or grey neighbour by at most three hops. Again, from Lemma 7.1, it follows that any two nodes in the PDS are separated by at least two hops. Therefore, any pair of complementary subsets of the PDS have a distance of exactly two or three hops. □

Lemma 7.3. *The Distributed Steiner Tree construction phase of the proposed scheme (Algorithm 3) constructs a single connected component from the PDS obtained from the Distributed PDS Construction phase.*

Proof. Here, we focus on the situation at the end of the connector selection phase. From the step 8 of Algorithm 3, we know that at the end of the connector selection phase, each yellow node w satisfies:

(i) The conectionCount of itself and its rivals is 1.
(ii) There is no black node in its 2HopNebsTable that does not occur in $w_{cdsList}$.

We show by contradiction that all black, blue and grey nodes are in one component. Suppose otherwise, let *A* be one component and let *B* be a nearest different component (minimum number of hops away). Since, we have considered the network as a connected graph, *A* and *B* must be connected by one or a chain of yellow nodes. Let us consider the *shortest chain* joining *A* and *B*.

Case I: *A and B are joined by a single yellow node*, let *u* be that node. So, the connectionCount of u will be 2, which violates the above condition (i).

Case II : *A and B are joined by a chain of two yellow nodes*, say u (adjacent to A) and v (adjacent to B). Dominatee u must be adjacent to at least one black node. If a black node is adjacent to u belongs to B, then A and B can be joined only by u. In that case, the shortest chain length joining components A and B will be one which contradicts the assumption of this case. In the other hand, if the dominator adjacent to u belongs to a separate component (other than A and B), then B no longer remains the nearest component to A. Therefore, these contradictions imply that u is adjacent to at least one black node that belongs to component A. Similarly, v is adjacent to at least one black node that belongs to component B. Hence, without any loss of generality, we can consider the end nodes in both components A and B to be black. Let u be adjacent to a black node x of component A and v be adjacent to a black node y of component B. So y is a 2-hop neighbour of u. In this case, as y belongs to a different component, u will not find y in its cdsList. This violates the above condition (ii).

Case III: *A and B are joined by a chain of yellow nodes with a chain length greater than two*. Let us consider the second yellow node from the end (nearest to component A). If the second yellow node is adjacent to a black node belonging to A, then we could have made this the first node in the chain contradicting this as our choice of a shortest chain. If the second yellow node is not adjacent to A, then it must have a black node in its 1-hop neighbourhood that is not in A. Either this node is in B (contradicting that the shortest chain is more than 2), or it is in some other component different from both A and B (contradicting *B* being the nearest neighbouring component). So, in all these cases we have a contradiction. Therefore, we conclude that there is only one component at the end of the process. This concludes the proof. □

Theorem 7.4. *From a given a network, DCMCDS constructs a CDS in finite time period.*

Proof. We present the correctness proof of our proposed scheme in two parts. First, we show that DCMCDS operates in finite time and then, we prove that a CDS is definitely obtained. In order to prove that DCMCDS works in finite time, we individually prove that all three phases of the algorithm namely distributed PDS construction, distributed Steiner Tree construction and distributed removal of redundant dominating nodes all takes finite time. In each round, the distributed PDS construction algorithm searches for a potential dominator locally from the remaining white nodes in the local 2-hop neighbourhood. At every round, a white node is selected as dominator and its colour is updated to black and all its adjacent nodes are updated as dominatees by changing their colours to yellow. When any white node discovers all its adjacent nodes to be yellow, it updates itself as virtual-dominator by changing its colour to grey. The PDS construction algorithm terminates when there is no white node left. We now prove by contradiction that this terminating condition must results in termination of the algorithm after a few rounds. Let u be a node which is still white.

Case I : *If all adjacent nodes of u are yellow*, then u must be a virtual-dominator. Hence, u must change its colour to grey.

Case II : *If a black node v is adjacent to u*, then u is a dominatee. Hence, u must change its colour to yellow.

Case III : *If there are one or more white nodes around u*, then one white node among them can be selected as dominator. If u is selected, then u changes its colour to black, otherwise cases I, II and III are followed until u changes its colour after a few rounds. Therefore, u will eventually change its colour from white. Thus, each white node will eventually change its colour either to black, yellow or grey accordingly completing the PDS construction. Lemma 7.3 shows that distributed Steiner Tree construction post-PDS selection results in a single connected component after a finite number of operations. Now, the selective removal of virtual-dominators and dominators (Algorithm 4) takes constant time as each grey node performs these steps independently. Hence, DCMCDS completes execution in finite time. We next show that the proposed algorithm determines a CDS. Lemma 7.1 proves that the set of all black and grey nodes, obtained from PDS construction, is an independent dominating set (MIS). Lemma 7.3 shows that the Steiner Tree construction forms one connected black-blue-grey component. The latter itself is a CDS as it connects all the nodes in the PDS. It can also be shown that after the removal of the selected virtual-dominators and dominators the resulting component is still a CDS as the algorithm takes care of connection and coverage while removing these nodes from the CDS. This concludes the proof. □

Lemma 7.5. *In any UDG, each MIS size is upper-bounded by* $3.8|opt| + 1.2$, *where* $|opt|$ *is the size of the MCDS.*

Proof. Directly from the result found in [32]. □

Lemma 7.6. *Maximum number of Steiner nodes obtained from distributed DCMCDS is* $(1 + \ln 5)|opt|$, *where* $|opt|$ *is the size of any optimal CDS.*

Proof. The proof is direct from the Theorem 2 of [16]. □

Theorem 7.7. *The size of the CDS obtained by DCMCDS is upper bounded by* $(4.8 + \ln 5)|\mathrm{opt}| + 1.2$, *where* $|\mathrm{opt}|$ *is the size of the MCDS.*

Proof. In the first phase, DCMCDS constructs the PDS as an MIS. In the second phase, it finds the Steiner nodes to construct the Steiner Tree. In the last phase, it removes the redundant dominating nodes (both dominators and virtual-dominator) to reduce the CDS size.

Therefore, we have,

$$|\mathrm{CDS}| \le |\mathrm{PDS}| + |\mathrm{Steinernodes}|$$

As PDS is an MIS, from Lemmas 7.5 and 7.6 we have:

$$|\mathrm{CDS}| \le 3.8|\mathrm{opt}| + 1.2 + (1 + \ln 5)|\mathrm{opt}|$$
$$= (4.8 + \ln 5)|\mathrm{opt}| + 1.2$$

Therefore, the performance ratio of DCMCDS is $(4.8 + \ln 5)|\mathrm{opt}| + 1.2$. □

Theorem 7.8. *The time complexity of DCMCDS is* $O(D)$ *time and* $O(D)$ *rounds, where* D *is the network diameter.*

Proof. In the proposed distributed scheme multiple dominators are selected in a single round. After the selection of a dominator from its 2-hop neighbours all its adjacent neighbours become dominatees. In the next round the algorithm selects the dominators from the remaining white nodes. The worst case occurs when in each round only one node is selected as the dominator or virtual-dominator, that means the dominator are selected one after another. The longest stretch of dominators and virtual-dominators should exist along the network diameter. Note that network diameter is the largest of all the shortest distances between any pair of nodes. In the worst case as discussed, the number of rounds is at most $O(D)$. Therefore, the time complexity for PDS construction is $O(D)$ time and $O(D)$ rounds. However, in the proposed scheme there is a chance of selection of multiple dominators in each round. So in average the time complexity is much lower that $O(D)$. Also in the second phase of distributed Steiner Tree construction, there is a chance of selection of multiple Steiner nodes in each round. After the selection of each single connector number of component decrease by one. By the similar argument, the Steiner Tree construction will also need $O(D)$ time and $O(D)$ rounds. In the last phase each dominators and virtual-dominators checks itself whether to upgrade or downgrade. All the dominators and virtual-dominators can do this checking simultaneously. Therefore, only one round is needed to do this. Hence the overall running time of the proposed algorithm is $O(D)$ time and $O(D)$ rounds. □

Theorem 7.9. *DCMCDS has message complexity of* $O(nR)$, *where* n *is the network size and* R *is the maximum between number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes.*

Proof. We present the message complexity of each phase of the distributed DCMCDS to find out the message complexity of the whole algorithm. In the initialization and neighbourhood table creation phase, each node broadcasts the messages HELLO, OWN_INFO and NEB_INFO once each. Therefore, the message complexity of this phase is $\Theta(n)$. In the distributed PDS construction phase, the total number of DOMINATOR and VIRTUAL_DOMINATOR messages broadcast is $\Theta(|\mathrm{PDS}|)$. Similarly, the number of DOMINATEE messages sent is $\Theta(n - |\mathrm{PDS}|)$. So, for each DOMINATOR or VIRTUAL_DOMINATOR message, a total of Δ DOMINATEE and UPDATE_NODE_COL messages are generated in the worst case, where Δ is the maximum degree of all the nodes. As we have a total of $|\mathrm{PDS}|$ dominators/virtual-dominators, the total number of DOMINATEE and UPDATE_NODE_COL messages generated will be $\Delta|\mathrm{PDS}|$. For each DOMINATEE message, a total of Δ UPDATE_NEB_INFO and UPDATE_NODE_COL messages are generated in the worst case. For $n - |\mathrm{PDS}|$ DOMINATEE messages, a total of $\Delta(n - |\mathrm{PDS}|)$ number of UPDATE_NEB_INFO and UPDATE_NODE_COL messages will be generated. Therefore, the total number of UPDATE_NEB_INFO and UPDATE_NODE_COL messages $= \Delta(n - |\mathrm{PDS}|) + \Delta|\mathrm{PDS}| - (n - |\mathrm{PDS}|) = |\mathrm{PDS}| + \Delta n - n = O(n\Delta)$. At the end of each round of this phase, some of the white nodes (those who find a change in effective degree of their 1-hop neighbours in last round) broadcast their updated neighbour information through the NEB_INFO message. So, the message count of this message will be $O(nR_{PDS})$, where R_{PDS} is the number of rounds needed to construct the PDS. Hence, the message complexity of this phase is $O(nR_{PDS})$, assuming R_{PDS} is greater than Δ. To find out the message complexity of distributed Steiner Tree construction phase, let us assume the algorithm runs for R_{ST} rounds to interconnect the PDS nodes. In each round, the total number of CONN_INFO_REQ and CONN_INFO_REP messages sent is n. So the total count of these two messages in all rounds is $O(nR_{ST})$. The COMP_RIVAL_INFO messages are sent by each node of the components once in each round. In the first round, the count of this message is $|\mathrm{PDS}|$. It keeps on increasing up to $|\mathrm{CDS}|$ in the last round. So, the total count of this message in all rounds is $O(R_{ST}|\mathrm{CDS}|)$. As the RIVAL_INFO message is sent by the component members twice in each round, the total count of this message in all rounds is $O(R_{ST}|\mathrm{CDS}|)$. The number of CONNECTOR messages sent in all rounds is $\Theta(|\mathrm{CDS}| - |\mathrm{PDS}|)$. The UPDATE_COMP_INFO message is sent by the component members once in each round. So, the total number of UPDATE_COMP_INFO messages sent in all rounds is $O(R_{ST}|\mathrm{CDS}|)$. For each dominator, Δ UPDATE_NODE_COL messages are sent. Total UPDATE_NODE_COL messages sent in all rounds is $O(n\Delta)$. Hence, the total message complexity of this phase is $O(nR_{ST})$ assuming R_{ST} is greater than Δ. In the last phase of removing redundant dominating nodes, some of the virtual-dominators who want to upgrade themselves to dominators, send the UPGRADE_DOM message, only once. So, the total count of this message sent is $O(|\mathrm{VD}|)$. Similarly, the dominators or virtual-dominators who want to downgrade themselves to dominatees send the DOWNGRADE_DOM message only once. So, the total count of this message is $O(|\mathrm{CDS}|)$. A dominator to downgrade itself, needs to send the ALT_DOMINATOR_REQ message once to its dominatees. The dominatees send the ALT_DOMINATOR_REP message for their dominators. So, the total count of these two messages is $O(|\mathrm{DS}|)$. The dominators which do not receive the TRUE reply through the ALT_DOMINATOR_REP messages from all their dominatees, withdraw their intent to become dominatees by sending the ALT_DOMINATOR_REQ_CANCEL message. So, the count of this cancel message sent is $O(|\mathrm{DS}|)$. For each upgrade / downgrade of virtual-dominators, and downgrade of dominators, Δ number of UPDATE_NODE_COL messages are sent. So, the total UPDATE_NODE_COL messages sent is $O(|\mathrm{DS}|\Delta)$. Hence, the message complexity of this phase is $O(n\Delta)$. Thus, the overall message complexity for DCMCDS is $O(nR)$, where R is the maximum of R_{PDS} and R_{ST}. This completes the proof. □

8. Simulation results

In this section we present the results of the simulations conducted by us to compare our proposed scheme with the existing approaches. The WSN is modelled in a fixed area of dimension 100×100 square units. We have generated the hosts randomly by choosing their abscissa and ordinate using a uniform random num-

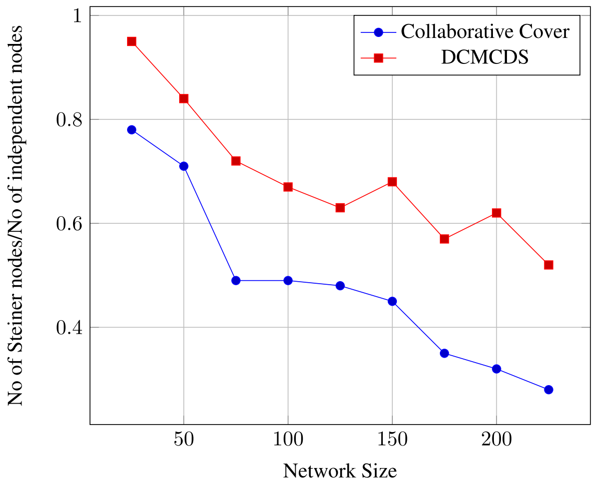


Fig. 7. Performance comparison of number of Steiner nodes and number of independent nodes.

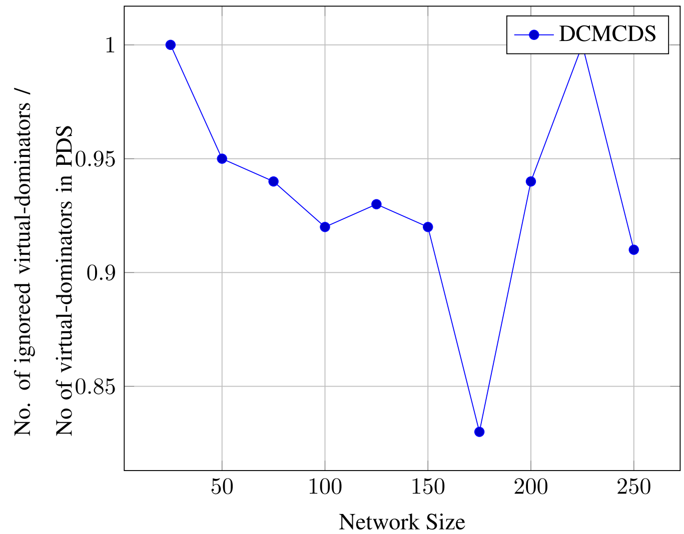


Fig. 8. Ratio of ignored virtual-dominators to total virtual-dominators in pseudo-dominating set for different network sizes.

ber generator. The transmission range of each node was taken as R for each node. Two nodes are connected if their distance is less than equal to R. In the entire experimentation we considered connected networks only. In the simulation we considered $R = 25$. The proposed algorithm is run for 100 times for different network sizes from 50 to 250. The average results are reported in the figures and table. We conducted the entire simulation in NS-2, a network simulator for wireless networks. The simulation experiments considered for analysing the performance are: (i) performance comparison of Stenier nodes with independent set nodes (ii) performance comparison of Stenier nodes against ignored connectors (iii) performance comparison with related techniques. In our first experiment, we found the average effective degree of a connector. It is the ratio of total number of connectors to the number of accepted PDS nodes (connector and virtual-connector). We calculated the same ratio for the best existing algorithm, collaborative cover heuristic [18], for the network sizes varying from 25 to 225. The results are shown in Fig. 7. We found that the average effective degree for collaborative cover is 0.3 and for our proposed scheme it is 0.5. That means in the collaborative cover a Steiner node connects more than three PDS nodes whereas in our algorithm a Steiner node connects nearly two PDS nodes. This is a significant result and it has many positive consequences. Less effective degree of a connector indicates that the connector is less loaded. This enhances the life time of the network.

In our second experiment, we did an analysis on how much PDS nodes change their status in post-Steiner Tree construction phase and what is its impact on reduction of overall CDS size. For varied network sizes from 25 to 250 we determine the ratio of total virtual-dominators being downgraded to dominatee and its impact on the reduction of CDS size. The results in Fig. 8 shows that almost all of the virtual-dominators are downgraded to dominatee. Very few of them retained their earlier state because they are actually bridging two disjoint components of the CDS. The results found in Fig. 9 implies that for networks of larger sizes, by changing the status of some of the PDS nodes from dominator or dominatee according to the specified criteria reduces the CDS size by 10%.

In the next experiment, we compare the performance of our proposed scheme with the existing CDS constructions techniques found in [13,14,16,18,33] in two steps. Firstly, we found the CDS for the network where the nodes are distributed randomly. We considered the network of sizes 20, 40, 60, 80 and 100. The results found are shown in Fig. 10. It is clear from the result that our

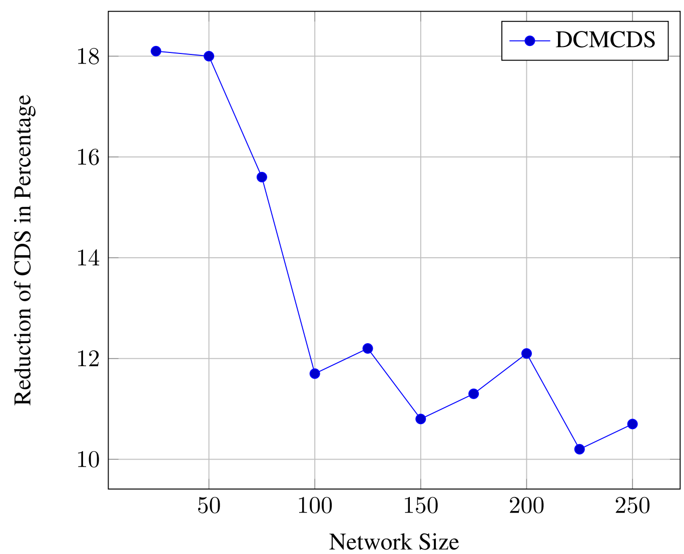


Fig. 9. Reduction in CDS size after discarding redundant dominators and virtual-dominators post-Steiner Tree construction for different network sizes.

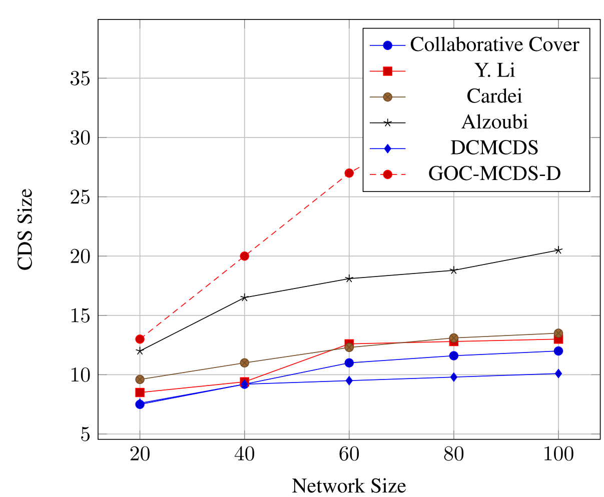


Fig. 10. Performance comparison of CDS construction algorithms.

Table 1
Comparison with collaborative cover for uniform distribution of nodes.

Distribution	CDS size by DCMCDS / CDS size by collaborative cover for network size				
	50	100	150	200	250
Uniform	0.74	0.75	0.67	0.78	0.91

scheme outperforms all above mentioned CDS construction techniques in identifying smaller CDSs. Our result is at least 16% better than the collaborative cover heuristic which is best among the above mentioned algorithms. We are getting better results because of our "Steiner Tree Construction" phase and "Removal of redundant dominator" phase. During Steiner Tree construction we select the connectors which connects maximum number of components. In the redundant dominator removal phase it omits the redundant dominating nodes cleverly without any loss in coverage or connectivity. Secondly, we compare our proposed scheme with the collaborative cover based heuristic [18], for ideal uniform distribution of nodes. Illustrations provided in [18] shows that the coverage based heuristic achieves significantly better results in optimizing CDS size for uniform hexagonal distributions by identifying optimal sub-structures than previously reported degree-based schemes. We vary the network size from 50 to 250 and determine the ratio of CDS sizes obtained by the DCMCDS scheme and collaborative cover heuristic for uniform hexagonal distribution. The results summarized in Table 1 illustrate that DCMCDS produces smaller CDS sizes for the above discussed distribution.

A good CDS construction algorithm not only should constructs the CDSs of smaller sizes but also should construct it in less time. In our next experiment, we compare the number of rounds needed to construct a CDS by our proposed scheme with 8-approximate CDS algorithm [14]. We use the number of rounds as the metric to compare the performance of our scheme with the existing approaches where, a round is the total time needed by the nodes to receive messages from their neighbours in the previous round, execute local computations and consequently send messages to their neighbours. The reason for comparing with only 8-approximate CDS algorithm [14] is, it represents the class of CDS construction techniques which first forms an MIS and then connect the MIS nodes greedily using different techniques. The S-MIS approach [16] and collaborative cover heuristic [18] belongs to this class of algorithms. However, the number of rounds required by all of the algorithms is hardly any different. For the comparative study, we varied the size of the networks from 25 to 250. For each of the network size, we find the number of rounds for constructing the CDS both by DCMCDS and 8-approximate CDS algorithm. The comparison is shown in Fig. 11. Theoretically the upper bound of both these algorithm is $O(D)$ rounds, D, being the network diameter. However, our proposed scheme needs fewer rounds than 8-approximate CDS. For large network sizes the number of rounds required by the proposed algorithm is nearly $(\frac{2}{3})$rd of that required by 8-approximate CDS. This reduces 33% execution time. We also observe that the slope of the curve representing the proposed scheme is significantly smaller than that for 8-approximate CDS. This means with increase in network size, the corresponding increment in the number of rounds is much smaller for the proposed scheme than other CDS construction techniques.

Next, we analyse the message exchanges needed for CDS construction of DCMCDS by comparing with previous degree-based [14] and collaborative cover [18] approaches. We vary the network size N from 100 to 500 and compare the mean number of messages required. From the Fig. 12, one can observe that, the mean number of broadcasted messages by our proposed algorithm is closer to degree-based approach [14] and better than the collaborative cover heuristic [18]. The result is justified because the message complexity of 8-approximate degree based CDS scheme is $O(n\Delta)$ and that of collaborative cover is $O(n\Delta^2)$, where n is the number of nodes in the network and Δ is the maximum degree of a node. The message complexity of DCMCDS is $O(nR)$, where R is the number of rounds. Although the message complexity of both degree-based 8-approximate CDS algorithm [14] and DCMCDS are linear, but the former requires slightly fewer messages than the latter. This is because 8-approximate CDS technique [14] uses only 1-hop connectivity information whereas DCMCDS uses 2-hop neighbourhood knowledge to obtain a better CDS. So from this experiment we can conclude that DCMCDS constructs the CDSs of smaller sizes using a slightly higher expense of number of messages exchanged as compared to previous degree-based CDS construction techniques.

The advantages of DCMCDS is that it identifies much smaller CDSs than 8-approximate CDS or collaborative cover for all network sizes. In fact after the network density crosses a certain threshold, the CDS size will hardly change much irrespective of how many nodes more added to the fixed deployment area. Fig. 13 explains this scenario. For DCMCDS this threshold is reached earlier than collaborative cover or 8-approximate CDS. Therefore a significant increment in average dominator degree coupled with a marginal increment in CDS size for DCMCDS would lead to a slower rise in energy dissipation for small as well as large scale networks.

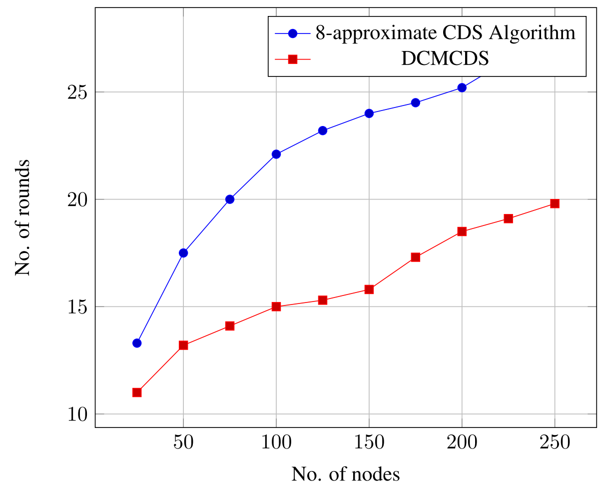


Fig. 11. Performance comparison of number of rounds in CDS construction.

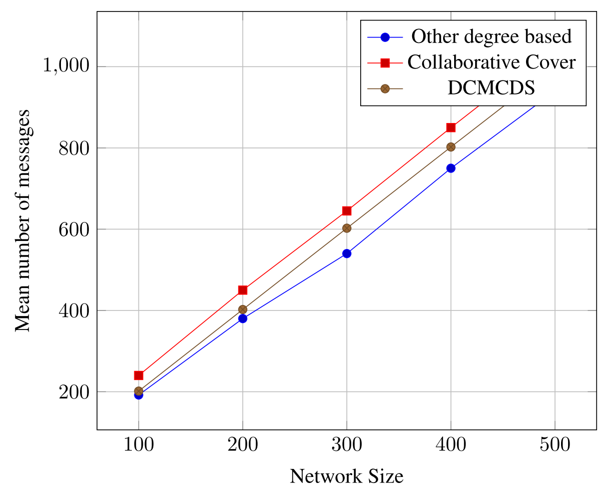


Fig. 12. Comparison of message exchanges in CDS construction algorithms.

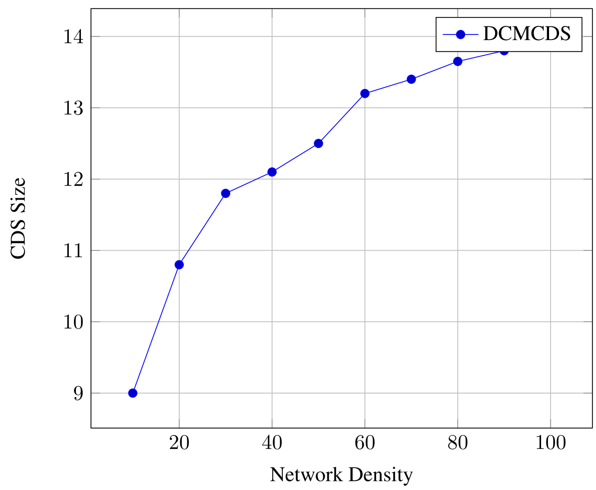


Fig. 13. CDS Size corresponding to DCMCDS for different network densities.

9. Conclusion

In this paper, we studied various techniques of CDS construction in the wireless network. We proposed a novel distributed degree-based greedy approximation algorithm. The time and message complexities of our algorithm are $O(D)$ and $O(nR)$ respectively, where n is the network size, D is the diameter of the network and R is the maximum between number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes. The approximation ratio of our proposed scheme is found to $(4.8 + \ln 5)|opt| + 1.2$, where $|opt|$ is the size of any optimal CDS. The simulation results shows that the proposed algorithm constructs smaller CDSs in comparison to all other existing CDS construction algorithms. The localized nature of our algorithm makes it useful for situations where a centralised approach is not suitable. The proposed algorithm can construct the CDS construction in the network of nodes with different transmission ranges.

References

[1] A. Salhieh, J. Weinmann, M. Kochhal, L. Schwiebert, Power efficient topologies for wireless sensor networks, in: Parallel Processing, 2001. International Conference on, IEEE, 2001, pp. 156–163.
[2] B. Das, R. Sivakumar, V. Bharghavan, Routing in ad hoc networks using a spine, in: Computer Communications and Networks, 1997. Proceedings., Sixth International Conference on, IEEE, 1997, pp. 34–39.
[3] B.N. Clark, C.J. Colbourn, D.S. Johnson, Unit disk graphs, Discrete Math. 86 (1–3) (1990) 165–177.
[4] R. Misra, C. Mandal, Rotation of CDS via connected domatic partition in ad hoc sensor networks, IEEE Trans. Mobile Comput. 8 (4) (2009a) 488–499.
[5] R. Misra, C. Mandal, Efficient clusterhead rotation via domatic partition in self--organizing sensor networks, Wireless Commun. Mobile Comput. 9 (8) (2009) 1040–1058.
[6] M. Rabbat, R. Nowak, Distributed optimization in sensor networks, in: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, ACM, 2004, pp. 20–27.
[7] A. Ephremides, J.E. Wieselthier, D.J. Baker, A design concept for reliable mobile radio networks with frequency hopping signaling, 75, IEEE, 1987, pp. 56–73.
[8] S. Guha, S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica 20 (4) (1998) 374–387.
[9] C. Adjih, P. Jacquet, L. Viennot, Computing connected dominated sets with multipoint relays, Ad Hoc Sens. Wireless Netw. 1 (1–2) (2005) 27–39.
[10] J. Wu, H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, in: Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, ACM, 1999, pp. 7–14.
[11] K.M. Alzoubi, P.-J. Wan, O. Frieder, Message-optimal connected dominating sets in mobile ad hoc networks, in: Proceedings of the 3rd ACM International Symposium on Mobile ad hoc Networking & Computing, ACM, 2002, pp. 157–164.
[12] I. Stojmenovic, M. Seddigh, J. Zunic, Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks, IEEE Trans. Parallel Distrib. Syst. 13 (1) (2002) 14–25.
[13] P.-J. Wan, K.M. Alzoubi, O. Friede, Distributed construction of connected dominating set in wireless ad hoc networks, in: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 3, IEEE, 2002, pp. 1597–1604.
[14] M. Cardei, M.X. Cheng, X. Cheng, D.-Z. Du, Connected domination in multihop ad hoc wireless networks, in: JCIS, 2002, pp. 251–255.
[15] Y.L.S. Zhu, M.T.D.-Z. Du, Localized construction of connected dominating set in wireless networks, in: Proceedings of US National Science Foundation International Workshop Theoritical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks, 2004.
[16] Y. Li, M.T. Thai, F. Wang, C.-W. Yi, P.-J. Wan, D.-Z. Du, On greedy construction of connected dominating sets in wireless networks, Wireless Commun. Mobile Comput. 5 (8) (2005) 927–932.
[17] A. Das, M. Aasawat, C. Mandal, C. Reade, An improved greedy construction of minimum connected dominating sets in wireless networks, in: Wireless Communications and Networking Conference (WCNC), 2011 IEEE, IEEE, 2011, pp. 790–795.
[18] R. Misra, C. Mandal, Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks, IEEE Trans. Parallel Distrib. Syst. 21 (3) (2010) 292–302.
[19] R.K. Jallu, P.R. Prasad, G.K. Das, Distributed construction of connected dominating set in unit disk graphs, J. Parallel Distrib. Comput. 104 (2017) 159–166.
[20] T. Nieberg, J. Hurink, A ptas for the minimum dominating set problem in unit disk graphs, International Workshop on Approximation and Online Algorithms, Springer, Berlin Heidelberg, 2005, pp. 296–306.
[21] Z. Zhang, J. Zhou, Y. Mo, D.-Z. Du, Performance-guaranteed approximation algorithm for fault-tolerant connected dominating set in wireless networks, in: Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on, IEEE, 2016, pp. 1–8.
[22] W. Wang, B. Liu, D. Kim, D. Li, A new constant factor approximation to construct highly fault-tolerant connected dominating set in unit disk graph, IEEE/ACM Trans. Networking (99) (2016) 1.
[23] L. Ding, W. Wu, J. Willson, H. Du, W. Lee, D.-Z. Du, Efficient algorithms for topology control problem with routing cost constraints in wireless networks, IEEE Trans. Parallel Distrib. Syst. 22 (10) (2011) 1601–1609.
[24] C. Liu, H. Huang, H. Du, X. Jia, Performance-guaranteed strongly connected dominating sets in heterogeneous wireless sensor networks, in: Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on, IEEE, 2016, pp. 1–9.
[25] T. Shi, S. Cheng, Z. Cai, Y. Li, J. Li, Exploring connected dominating sets in energy harvest networks, IEEE/ACM Trans. Networking (2017).
[26] L. Liu, Y. Song, H. Zhang, H. Ma, A.V. Vasilakos, Physarum optimization: A biology-inspired algorithm for the Steiner Tree problem in networks, IEEE Trans. Comput. 64 (3) (2015) 818–831.
[27] F. Cadger, K. Curran, J. Santos, S. Moffett, A survey of geographical routing in wireless ad-hoc networks, IEEE Commun. Surv. Tutorials 15 (2) (2013) 621–653.
[28] D. Das, R. Misra, A. Raj, Approximating geographic routing using coverage tree heuristics for wireless network, Wireless Netw. 21 (4) (2015) 1109–1118.
[29] S. Shukla, R. Misra, A. Agarwal, Virtual coordinate system using dominating set for GPS-free adhoc networks, Ann. Telecommun. 72 (3–4) (2017) 199–208.
[30] D.-Z. Du, L. Wang, B. Xu, et al., The Euclidean Bottleneck Steiner Tree and Steiner Tree With Minimum Number of Steiner Points, 1, 2001, pp. 509–518.
[31] J.P. Mohanty, C. Mandal, C. Reade, A. Das, Construction of minimum connected dominating set in wireless sensor networks using pseudo dominating set, Ad Hoc Netw. 42 (2016) 61–73.
[32] W. Wu, H. Du, X. Jia, Y. Li, S.C.-H. Huang, Minimum connected dominating sets and maximal independent sets in unit disk graphs, Theor Comput. Sci. 352 (1–3) (2006) 1–7.
[33] H. Du, W. Wu, Q. Ye, D. Li, W. Lee, X. Xu, CDS-based virtual backbone construction with guaranteed routing cost in wireless sensor networks, IEEE Trans. Parallel Distrib. Syst. 24 (4) (2013) 652–661.

Jasaswi Prasad Mohanty is a Ph.D. candidate in the Department of Computer Science & Engineering, IIT Kharagpur, India. He received his Master Degree in Computer Science from Utkal University, Bhubaneswar, India in 2006. He is working as a senior Assistant Professor with the Department of Computer Science and Engineering, Silicon Institute of Technology, Bhubaneswar, India. His current research interests includes wireless networks and algorithms.

Chittaranjan Mandal received his Ph.D. from the Indian Institute of Technology (IIT), Kharagpur in 1997. He is currently working as a Professor with the department of Computer Science & Engineering, IIT Kharagpur. Prior to joining IIT, he served as a reader with Jadavpur University, Calcutta. His research interests include networked systems, formal modelling and design and web technologies. Professor Mandal has been an Industrial Fellow of Kingston University, London, U.K., since 2000. He was a recipient of a Royal Society Fellowship in 2006.

Chris Reade received his Ph.D. in 1984 and lectured in Computer Science until 1999. He became head of Department of Informatics and Operations Management at Kingston University from 2000 to 2013 and is now a consultant in mathematics and computing. His interests are in formal modelling, functional programming, finite methods, geometric modelling and networked systems.