INTEGER PROGRAMMING FORMULATION FOR CONTENTION AWARE CONNECTED DOMINATING SET IN WIRELESS MULTI-HOP NETWORK

CHOWDHURY NAWRIN FERDOUS
Bachelor of Science, Bangladesh University of Professionals, 2014

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

INTEGER PROGRAMMING FORMULATION FOR CONTENTION AWARE CONNECTED DOMINATING SET IN WIRELESS MULTI-HOP NETWORK

CHOWDHURY NAWRIN FERDOUS

Date of Defence: August 12, 2020

Dr. Daya Gaur Thesis Supervisor	Professor	Ph.D.
Dr. Rossitsa Yalamova Thesis Examination Committee Member	Professor	Ph.D.
Dr. Robert Benkoczi Thesis Examination Committee Member	Associate Professor	Ph.D.
John Sheriff Chair, Thesis Examination Committee	Assistant Professor	Ph.D.

Dedication

To my parents, parents-in-law and my loving husband

Abstract

Efficient data propagation across the mobile nodes is an essential concern in wireless networks. Broadcasting with Minimum Connected Dominating Set (*MCDS*) is used to reduce redundant transmission. Contention occurs when a group of nodes want to transmit over a shared channel at the same time. During contention, nodes defer transmissions for a random time. Using Contention-aware Connected Dominating Set (*CACDS*) to minimize contention is a new concept. We study computationally (using CPLEX) Integer Programming for *MCDS* and *CACDS* and use Benders Decomposition to solve the problem. To find a connected dominating set, we use one state-of-art approach based on the shortest path algorithm, and ours one is based on the number of connected components.We propose IP formulation of selection forwarding-nodes based on Dominant Pruning and Contention-aware Dominant Pruning. The result shows that our approach performs better than the state-of-art approach in large networks. *CACDS* results better in minimizing contention.

Acknowledgments

First of all, I would like to declare that all the appraisals belong to the Almighty. I would like to express my deep gratitude to my supervisor Dr. Daya Gaur for his continuous guidance, suggestions and whole hearted supervision throughout the progress of this work. I thank him for his patience in reviewing my drafts, for correcting my proofs, and encouraging me to continue my research work. I also want to thank my committee members Dr. Robert Benkoczi and Dr Rossitsa Yalamova for their constant support and motivation.

It is a great pleasure to thank my parents, parents-in-law and sister for encouraging me throughout my life. I would like to thank specially Sowkat Alam Shakil, Farhana Aklam Mitu, Asif Mahmud, Parinaz Bairami, Leila Karimi, Farzina Islam and Sakib Mahmud Khan for being part of my journey these two years in Canada.

Finally, I would like to thank my husband, Chowdhury Hasan Ibne Obayed for all his support and his trust on me. Without his constant motivation, I would not be in this position today. Thank you for always being there for me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A wireless multi-hop network uses one or more intermediate nodes to convey information from source to destination. One type of wireless multi-hop network is mobile ad-hoc network which does not rely on any pre-existing infrastructure. A mobile ad-hoc network is temporarily constructed without any centralized control and it is used in emergencies like disaster relief, conferences, military operations etc [38].

A multi-hop scenario occurs when a mobile node is not able to communicate directly with other nodes due to radio power limitation, channel utilization, and power-saving concerns [38]. The mobile nodes are resource-constrained devices with limited energy and power. If the limited resources of the mobile nodes are not used efficiently, the outcome can be devastating, and could lead to a network partition. A big challenge in a mobile ad-hoc network is its constantly changing topology because of the high mobility of the nodes. Hence, the selection of which nodes that forward data is made dynamically based on the network connectivity and the routing algorithm in use.

The nodes in a multi-hop wireless network frequently broadcast control messages for route discovery and other network services. The simplest way to broadcast a message through the entire network is *Blind Flooding* [38]. It requires every node to forward the broadcast message once. This leads to severe performance bottleneck due to redundant traffics, contention, and collision, altogether known as **Broadcast Storm Problem** [38]. A

general solution to minimize the redundant broadcasting is the use of **Minimum Connected Dominating Set (*MCDS*)** [30] as the basis of routing. Only the nodes in *MCDS* are allowed to forward a message. These nodes are known as the **"forwarding nodes"** of the network.

Contention in a network is a situation where more than one nodes aim to propagate a message at the same time using the common medium shared by those nodes. Medium Access Control (MAC) layer of Open System Interconnection (OSI) Model [9] is primarily responsible for regulating access to the shared medium. Upon contention, the nodes defer their transmissions for a random amount of time. Sometimes it causes longer wait time and additional traffic, which slows down the data transmission in the network [9].

Contention aware Connected Dominating Set (*CACDS*) [16] is a special type of the *MCDS*. Selecting the forwarding nodes to be in *MCDS* will reduce the redundant transmission. However, it does not minimize the contention among the forwarding nodes. The *CACDS* selects the forwarding nodes to ensure there is minimum contention among the forwarding nodes. As a result, there will be no/fewer nodes that need to defer their transmissions, and speeds up the broadcasting process.

In a centralized network, the global topology information is already known. Here, an admin node runs the algorithm and selects the forwarding nodes (a subset of *MCDS*) for the network. Only the nodes selected by the admin node are eligible to forward data in the network. However, for a mobile ad-hoc network, the construction of a distributed *MCDS* is needed because of the lack of centralized administration. Also, it is not possible to gather the complete topological information of an ad-hoc network. Instead of a central admin node, here each node is eligible to participate in decision making and routing by forwarding data to other nodes. Various methods have been proposed to effectively broadcast a message in ad-hoc network based on the neighborhood information. **Dominant Pruning (*DP*)** [26] is one of the promising approaches. It uses neighborhood (2-hop) information of each node to ensure complete network coverage while reducing redundancy. **Distributed CACDS** or

Contention aware Dominant Pruning (*CADP*) [16] is a special version of *DP*, which uses the same principle of *DP* and minimizes contention among the forwarding nodes.

1.1 Our Motivation

The *MCDS* problem is an NP-hard problem [18]. Several heuristics [26] and approximation algorithms [21, 28] have been developed to solve the *MCDS* problem. The Integer Programming (IP) formulation and a version of the **Branch and Cut** algorithm to solve this problem is discussed in [35]. Two exact algorithms based on **Benders Decomposition** and **Branch and Cut** method for the *MCDS* problem have been proposed in [19].

Dominant Pruning (*DP*) [26] is an effective approach to handle redundant transmission in a distributed system, but there is no prior work on the use of Integer Programming formulation to select forwarding nodes. In this thesis, we propose an IP formulation for selecting forwarding nodes in a distributed ad-hoc network following the basic principle of Dominant Pruning.

Contention aware Connected Dominating Set (*CACDS*) is a new concept in the ad-hoc wireless networks used to minimize the contention. Some heuristic algorithms [16] have recently been proposed for finding *CACDS*. However, the mathematical formulation of *CACDS* has not attracted any attention so far. In this thesis, we propose to fill this notable gap by introducing and studying an Integer Programming formulation of Contention aware Connected Dominating Sets (*CACDS*) for both centralized and distributed system. To the best of our knowledge, this is the first study done on IP formulation for *CACDS* in centralized and distributed system.

1.2 Our Contribution

The contributions in this thesis are:

- We study computationally (using *CPLEX*) an Integer Programming formulation for Minimum Connected Dominating Set (*MCDS*) and use the Iterative version of Benders Decomposition [19] to solve the problem. The framework used in this work is from [19]. In an intermediate step of Benders Decomposition, we use two different approaches to find the connected dominating set. One is based on the Shortest Path (SP) algorithm (*SP* based) [19], and we provide a different approach based on the number of connected components and maximum degree (Δ based).

- We also formulate an Integer Program for Contention aware Connected Dominating Set (*CACDS*) and use the same approach to solve the problem as used for *MCDS*.

- As mentioned earlier, we need global topology information to calculate the *MCDS*/*CACDS*. As, it is not possible to know the complete topological information for an ad-hoc wireless network, we propose an Integer Program to select forwarding nodes of the distributed network based on the Dominant Pruning (*DP*) [26] approach using only two-hop neighborhood information of each node. We also focus on the selection of contention aware forwarding nodes in the distributed network.

- We evaluate the IP models of centralized and distributed network. The evaluation criteria used are the number of forwarding nodes, a measure of contention, the time to solve the problem and the number of cuts generated. The computational result shows that the Benders Decomposition (*MCDS*/*CACDS*) performs better in dense graphs compared to sparse graphs. *MCDS* selects the same number of forwarding nodes for both the approaches. Our approach (Δ based) performs better than the *SP* based approach [19] in terms of time to find *CDS*.

We reached a similar conclusion for *CACDS*. The objective values generated by both the approaches are equal and Δ based approach generates the solution faster than the *SP* based approach [19]. *CACDS* minimizes contention.

For the distributed system, Contention aware Dominant Pruning selects less contended forwarding nodes than Dominant Pruning [26]. However, the difference between the execution time of both the algorithm is not that high.

1.3 Organization of the Thesis

There are seven chapters, including this chapter. In chapter 2, we present the definitions needed to explain our work more precisely. We review the state-of-the-art research works in chapter 3. We describe the Integer Programming formulations for centralized *MCDS* and *CACDS* in chapter 4 and discuss Benders Decomposition. We discuss the Integer Programming formulation to select the forwarding nodes for a distributed system in chapter 5. Chapter 6 is on the simulation and performance evaluation. Chapter 7 concludes with a discussion on the limitations of our work and possible future extensions.

Chapter 2

Preliminaries

A network can be represented by a graph where the vertices represent the communicating nodes (hosts), and a direct edge from one vertex to another vertex means that the first-mentioned vertex can send data directly to the later one. The data propagation conditions can be modelled by considering "transmission range" within which communication is possible, and outside of which it is impossible. If all nodes have equal transmission ranges, then the graph becomes undirected. We use a symmetric graph $G = (V, E)$ to represent a

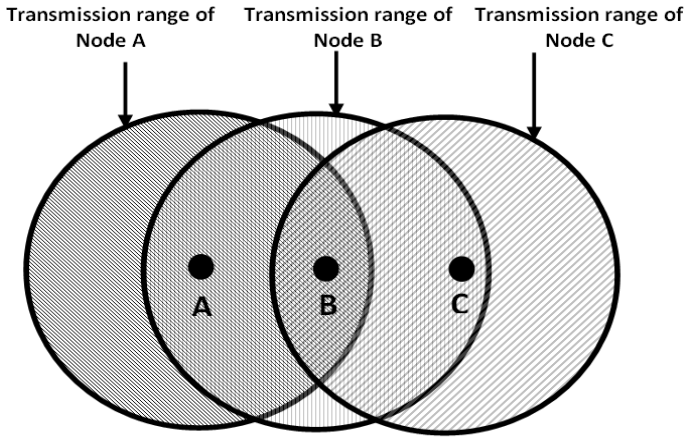


Figure 2.1: A sample wireless network

network, where V represents a set of wireless mobile nodes (hosts), and $E = \{(i, j) : i \ \& \ j$ share a communication link$\}$. Two nodes i and j are called 1-hop neighbors or adjacent if there is an edge between them i.e. $(i, j) \in E$. The edge between two nodes (i, j) also

indicates that the nodes i and j are within their transmission range. Figure 2.1 represents a

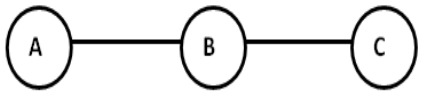


Figure 2.2: A graph representation of network shown in Figure 2.1

wireless network of three nodes- node A, B and C. The circle centered at the node represents the transmission range of the nodes. The nodes that are within the transmission range of node A, and node C is node B, wheres node A and node C are in the transmission range of node B. Nodes within the transmission range of each other are known to be neighboring nodes. So, node B is the neighbor of both node A and node C; however, node A and node C are not considered as neighbors as they are not within the transmission range of each other. As node A exists in the transmission range of node B, they can directly communicate with each other; hence in the network graph, there is an edge between them. The similar condition holds for node B and node C as well. Figure 2.2 is graph representation of network shown in Figure 2.1. When a node sends a broadcast packet, all of its 1-hop neighbors receive the packet.

2.1 Flooding and Broadcast Storm Problem in the Network

A wireless multi-hop network uses multiple hops/nodes for data dissemination. There is no centralized admin node, so each node can participate in forwarding messages to others to reach out to the entire network. Wireless networks need to broadcast messages for various services such as route discovery, periodic data dissemination, erasing an invalid route, locating a node, duplicate IP address detection or even sending alarm signals in the entire network.

Blind Flooding [38] is a natural process to conduct broadcasting in a wireless multi-hop network. In blind flooding, upon receiving a broadcast message for the first time, the node is obliged to rebroadcast the message. This is how the message is transmitted to the entire network. Though blind flooding ensures full coverage at high mobility but unfortunately, it results in redundant transmissions, contention and collision in the network, which is collaboratively known as **Broadcast Storm Problem** [38]. In a network, drawbacks of blind flooding include:

- **Redundant Rebroadcast:** A node doesn't need to broadcast a message if all of its neighbors have already received the message. This unnecessary broadcasting is called redundant rebroadcast. The scenario is illustrated in Figure 2.3. Suppose node *A* and node *C* have already broadcast the message. Node *H*'s broadcasting is redundant as all of its neighbors- node *A* and node *G* have already received the message beforehand.

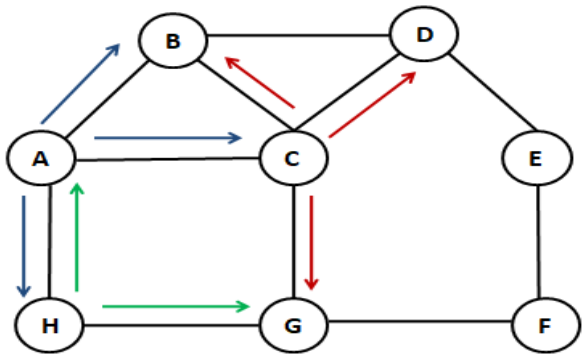


Figure 2.3: Scenario of Redundant Rebroadcast: The broadcast of node *H* is redundant

- **Medium Contention:** Contention means competition for resources. When a node broadcasts a message, many of its neighbors need to rebroadcast the message to spread it in the whole network. When more than one nodes within the same transmission area try to broadcast a message at the same time, their transmission may severely contend with each other, leads to a contention problem. Upon receiving

message from node *A*, when node *B* and node *C* will try to rebroadcast the message at the same time, they will face contention as they are within each others transmission range. The scenario is shown in Figure 2.4.

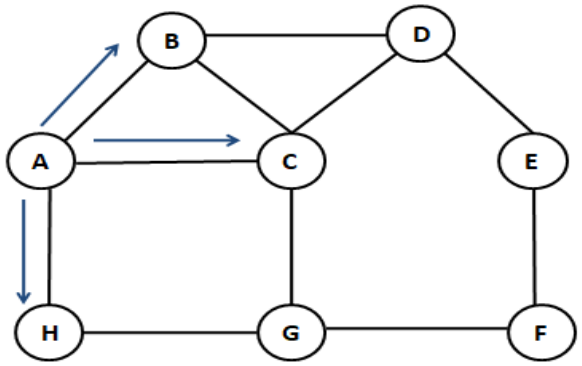


Figure 2.4: Scenario of Contention: node *B* & *C* are contending for medium

- **Packet Collision:** While contention occurs at the sender side, packet collision can occur at the destination side because of blind flooding. If nodes *G* and *E* broadcast at approximately the same time, there is a possibility of a packet collision at node *F*, shown in Figure 2.5.

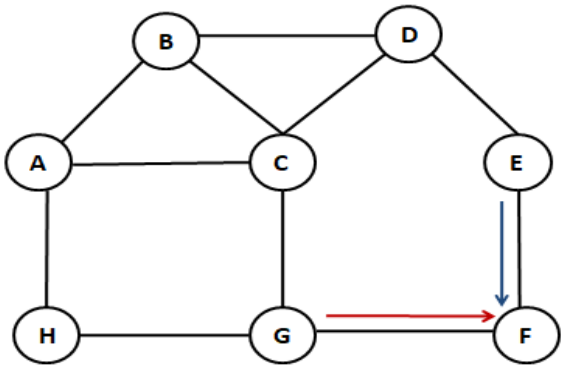


Figure 2.5: Scenario of Collision: Packet collision occurs at node *F* when node *G* & *E* transmit at the same time.

2.2 Controlling Contention & Medium Access Control (MAC) Layer

The Medium Access Control (MAC) layer is one of two sub layers that constitutes the Data Link Layer of the Open Systems Interconnection (OSI) model [9]. In most of the networks, multiple nodes share a common communication medium for transmitting their data packets. MAC layer is primarily responsible for regulating access of the nodes to the shared medium. It specifies when a node can access the media and resolve potential conflicts among the conflicting nodes. There are two types of protocols - one is contention-free MAC protocol, and another is contention-based MAC protocol [9].

1. **Contention-free Protocol:** There are two types of contention-free protocol - Fixed assignment strategy and Dynamic assignment strategy. In Fixed assignment strategy, contentions are avoided by ensuring that each node can exclusively use its allocated resources. Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA), Code Division Multiple Access (CDMA) are examples of Fixed assignment strategy [9]. These strategies are not efficient, as it is difficult to reallocate the idle resources in this protocol. Also, the schedules of resource allocation need modification with the change of network topology or traffic characteristics. Generating schedules for the entire network is a time-consuming task as well. Dynamic assignment strategy allows nodes to access the medium based on demand. Polling-based, token passing and reservation-based protocols are examples of Dynamic assignment strategy [9]. The disadvantage of this protocol is that it requires more space, and the calculations and analysis are increased.

2. **Contention-Based Protocol:** In contrast to contention-free techniques, contention-based protocols allow nodes to contend for getting access to the medium simultaneously but also provide mechanisms to reduce the number of contentions and recover from such contentions. ALOHA, CSMA (Carrier Sense Multiple Access) are examples of contention-based Protocol [9].

A popular contention-based MAC scheme is CSMA which includes its variations like Collision Detection (CSMA/CD) and Collision Avoidance (CSMA/CA) [9]. In CSMA/CD, the node is allowed to transmit the data when the medium is idle. If the node finds that the medium is busy, it waits for a certain amount of time before attempting to transmit it again, which is known as back-off operation.

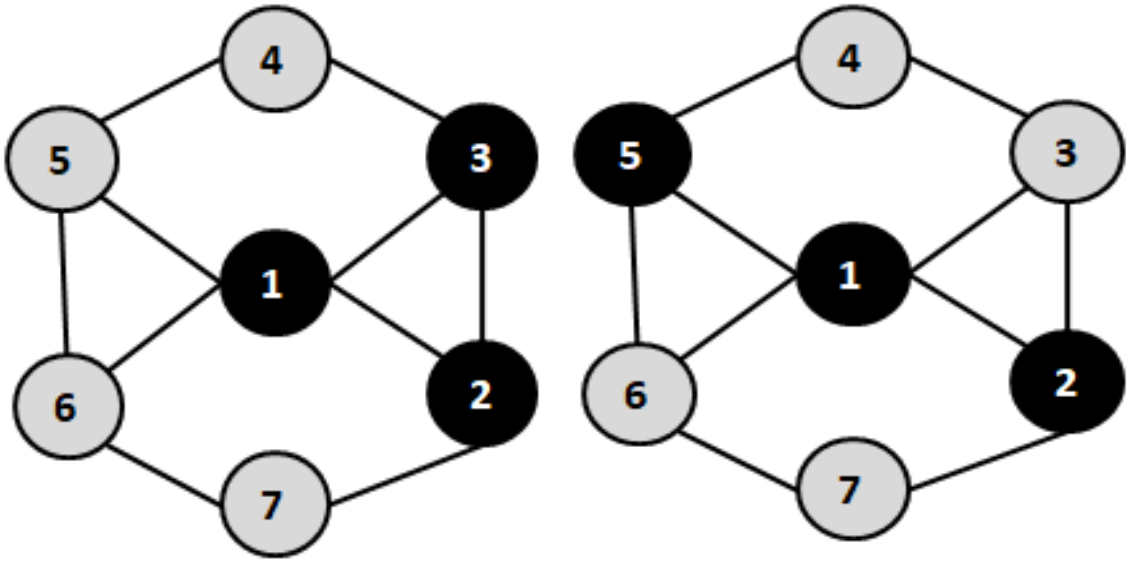
CSMA/CA [9] is used mostly in the wireless system, which attempts to avoid contention in the first place. In CSMA/CA, after finding the medium idle, a node waits for at least a time period called the DCF interframe space (DIFS) before start transmitting. If the node finds the medium busy, it defers its transmission by executing a back-off algorithm. The algorithm randomly selects a number of time slots to wait and store the value in the back-off counter. For every time the slot passes on the network without any activity, the counter is decremented. When the counter reaches zero, the node attempts to transmit. If any activity is detected before the counter reaches zero, the node will wait until the medium is idle for a DIFS period before the counter value continues to decrease. After a successful transmission, a receiver node responds with an acknowledgment after waiting for a time period called the short interframe space (SIFS). No other device accesses the channel before the receiver node can transmit its acknowledgment.

Though this strategy helps to prevent contention and data loss, it causes longer waiting times and creates additional traffic; slowing down the data transmission process.

2.3 Important Definitions

We are going to discuss some of the essential definitions that will need to explain our work more precisely.

i **Dominating Set (DS):** The dominating set (DS) of a graph $G = (V, E)$ is defined as a



(a): CDS Construction with Contention (b): CDS Construction without Contention

Figure 2.6: Sample network with 7 nodes, Black nodes are the forwarding nodes

subset of V ($DS \subseteq V$), where each node in the graph is either an element of DS or 1-hop neighbor (adjacent) to at least one of the elements of DS. The set of possible DS for the graph shown in Figure 2.6 is,

$$\{DS\} = \{\{3,6\},\{2,5\},\{1,2,3\},\{1,2,5\},\{1,5,6\},\{1,4,7\},\{1,2,3,4\}.....\}$$

ii **Connected Dominating Set (CDS):** If any node in DS can reach any other node in DS by a path that stays entirely within DS, then the subset is called connected dominating set (CDS). The possible CDS for the graph in Figure 2.6 are,

$$\{1,2,3\},\{1,2,5\},\{1,5,6\},\{1,2,3,4\},\{2,3,4,7\}.....$$

iii **Minimum Connected Dominating Set (MCDS):** CDS with the minimum cardinality is known as minimum connected dominating set ($MCDS$). The possible $MCDS$ for the

graph in Figure 2.6 are,

$$\{1,2,3\},\{1,2,5\},\{1,5,6\}.....$$

iv **Contention Aware Connected Dominating Set (CACDS):** Contention occurs when two or more adjacent nodes try to transmit a packet simultaneously. At the result of contention, the nodes have to wait for a random amount of time to use the medium and sometimes it also causes packet loss in the network [9].

The concept of contention aware connected dominating set (*CACDS*) is new in the field of wireless network to minimize broadcast storm problem. The idea of *CACDS* was first introduced in [16]. *CACDS* is the problem of finding a *CDS* with a minimum contention number. The definition of the contention number is in Section 6.2.2 (pp 54).

In Figure 2.6, if we form the *CDS*, there is a possibility that node 1, 2 and 3 can form *CDS*. Upon receiving packet from node 1, if nodes 2 and 3 rebroadcast the message at the same time, they face contention, as they share the same communication channel (within the transmission range of each other). This contention can be avoided with a different *CDS*.

If node 1 broadcasts the message, nodes 2, 3, 5 and 6 will receive the message. If we select node 2 and node 5 to cover node 7 and node 4 respectively, instead of selecting node 2 and node 3, the scenario will be contention-free. So, the contention aware connected dominating set (*CACDS*) for Figure 2.6 are-

$$\{1,2,5\},\{1,3,6\}$$

Figure 2.6(a) and 2.6(b) are the examples of minimum connected dominating set with and without contention respectively.

v **Neighborhood of a Node:** The nodes can gather the neighborhood knowledge by send-

ing the periodic "hello" messages in the network [6]. This information can also be found on demand when the nodes attach their neighborhood information to the packet header. In **Proactive approaches** [25] of broadcasting packet in wireless network, the transmitting node takes the decision that which of the neighboring node will forward/broadcast the message next. The list of the next forwarding/broadcasting nodes known as **"forwarding list"** is attached to the packet header before rebroadcasting. Dominant Pruning [25] is an well-known example of proactive approach which uses neighborhood information for broadcasting the message. Details of this process is discussed in Section 5.1.

To define the terms related to neighborhood of a node, let us assume, upon receiving a packet from node i, node j is the next node that has been selected to forward the packet. Now, node j will create a forward list and append the list in the packet header before rebroadcasting.

- $N(j)$**:** $N(j)$ is the set of all 1-hop neighbors (adjacent) of node j. These nodes are within communication range of node j. Note that, node j itself also is a member of this set, $\{j\} \in N(j)$. For the graph shown in Figure 2.6, the set of 1-hop neighbors of node 2 is:

$$N(2) = \{1,2,3,7\}$$

- $N(N(j))$**:** $N(N(j))$ is the set of all nodes within 2-hop of node j, $N(j) \subseteq N(N(j)) \subseteq V$. For the graph shown in Figure 2.6,

$$N(N(2)) = \{1,2,3,4,5,6,7\}$$

- $N(N(j)) - N(j)$**:** The nodes that are exactly 2-hop away from j are in this set. For the graph shown in Figure 2.6,

$$N(N(2)) - N(2) = \{4,5,6\}$$

- **F_j:** The list of 1-hop neighbors of node j that are chosen as next forwarding nodes. Node j will attach this list in the packet header before broadcasting the message. Upon receiving the message from node j, if a node p finds itself in this list; i.e. $p \in F_j$; then node p will participate in broadcasting the message, otherwise it will not broadcast the message.
- **B_j:** The set of 1-hop neighbors of node j that can be selected as a member of the forward list F_j. When node j receives a packet from node i and $j \in F_i$, j selects its own forward list (F_j). Node i and node j may have some common neighbors, so while selecting forwarding nodes, node j does not need to consider those common neighbors as they were already considered by node i. $B_j = N(j) - N(i)$ and node j selects forwarding nodes from B_j. That is, $F_j \subseteq B_j$. Details have been discussed in Section 5.1.
- **U_j:** This is the set of those nodes that needs to receive the message when the nodes in F_j will broadcast. Details have been mentioned in Section 5.1.

vi **Degree of a node:** $N(j)$ is the set of all the nodes which are adjacent to j and $j \in N(j)$. The degree of $j \in V$ is defined as $deg(j) = |N(j)| - 1$.

vii **Maximum Node Degree:** For a graph $G = (V,E)$, the maximum degree of G denoted by x, is the degree of the node with the greatest number of edges incident to it.

viii **Density of a graph:** The density (d) of an undirected graph is given by:

$$d = \frac{2|E|}{|V|(|V|-1)}$$

Chapter 3

Related Work

In this chapter, we discuss the different approaches for constructing *MCDS* in both centralized and distributed networks that has been proposed by many researchers to mitigate the **Broadcast Storm Problem** [38]. We also focus on the Integer Programming formulation for *MCDS* and other approaches to solve the problem.

3.1 Studies on Broadcasting in Centralized Network

Various methods have been proposed for finding an optimal broadcasting tree [24]. An optimal broadcast tree guarantees that all nodes in the network will hear the message if only the nodes in the tree transmit it, and the number of the nodes of the tree is minimum possible.

Finding an optimal broadcast tree in NP-complete [24] as this is similar to finding a minimum connected dominating set in the network. Ephermides [14] first proposed using a connected dominating set (*CDS*) as a basis of broadcasting. Many algorithms have been proposed to use *CDS* construction to reduce the redundant broadcast in the network throughout time [4, 5, 8, 21, 33].

The usage of *CDS*s occurs in various protocols that perform a wide range of communication functions in wireless ad-hoc networks. Various protocols including media access

coordination [1, 36], location-based [11] and multicast/broadcast routing [39, 40]; conservation of energy [7, 34]; and topology control [12, 13] all use the concept of *CDS*. Another application of *CDS* is to assist resource discovery in mobile ad-hoc network [22]. This resource discovery is also known as Backbone based routing [3], dominating set based routing or spine based routing [10].

Guha and Khuller [21] first proposed a greedy heuristic to construct *MCDS*. The algorithm starts with coloring in which all the nodes are colored white. The node with maximum degree is then selected and colored black. All the 1-hop neighbors of that node are colored gray. A gray node having maximum number of white neighbors is then selected. The selected gray node is then colored black and all its white neighbors are colored gray. The selection process continues until all the nodes are either colored black or gray. The set with all the black nodes is *MCDS*. Many more algorithms for *CDS* construction are known [4, 5, 8, 33].

Many researchers have worked on mathematical formulation of the *MCDS* problem throughout the time. Fan and Watson [15] presented an Integer Programming. Any feasible solution to the IP formulation forms a dominating set, *DS* of the given graph, *G*. The formulation has an exponential number of constraints to ensure that the nodes in the dominating set are connected. As a result, it is computationally expensive to solve for large graphs. Later, the authors [15] presented IP approaches with polynomial number of constraints. Based on the fact that a spanning tree is connected, some of the models used to solve the spanning tree problem have been applied to solve the *MCDS* problem. The authors [15] studied four mixed integer programming approaches to ensure the connectivity in the dominating set *DS*. The approaches included **Miller-Tucker-Zemlin Constraints**, **Martin Constraints**, **Single Commodity Flow Constraints** and **Multi Commodity Flow Constraints** [15]. Exact algorithms were obtained by providing these formulations as an input to the IP solver CPLEX. The authors observed that computing a dominating set without im-

posing connectivity constraints is much faster without the connectivity constraints. Among the four different approaches for the connectivity constraints, **Miller-Tucker-Zemlin Constraints** performed the best in terms of CPU time. Other constraints took more than 24h for large graphs consisting of 73 nodes and 108 edges [15].

Simonetti et al. [35] presented an IP formulation of *MCDS* problem and used **Branch and Cut (BC)** [29] method to solve the problem. Based on a relation between the *MCDS* and **Maximum Leaf Spanning Tree (***MLST***)** problem, the authors presented a formulation for *MCDS*. The key idea to ensure the connectivity among the nodes in the dominating set generated by their IP formulation is to select the edges that ensure the generation of a spanning tree from the sub graph, induced by the dominating set. The constraints of the model in [35] guarantees that, the number of selected edges is one unit less than the number of selected nodes. The selected nodes make a dominating set and the selected edges make a tree (Generalized Subtour Breaking Constraints (*GSEC*)) [35]. They also presented a **Cut Inequality** (described in Section 4.2), which states that at least one of the edges in the **"Cut"** must be chosen for the solution. The authors used two types of decision variables: $y_i \in \{0,1\},\ i \in V$: to select which nodes are to be included ($y_i = 1$) or not ($y_i = 0$) in the dominating set; $x_e \in \{0,1\}$, $e \in E$: to choose edges so that the dominating set is connected. They used Branch and Cut algorithm. Let, $(\overline{x},\overline{y})$ be the solution obtained using the linear programming. $\overline{G} = (\overline{V},\overline{E})$ be the subgraph of G implied by $(\overline{x},\overline{y})$, (where $\overline{V} := \{i \in V : \overline{y_i} > 0\}$ and $\overline{E} := \{e \in E : \overline{x_e} > 0\}$). In their method, they use a heuristic method to separate *GSEC*s, and they mention that the exact separation of *GSEC*s can be performed by **Max-flow (Min Cut)** [23]. The edges in $\overline{E}$ are sorted in decreasing order of $\overline{x_e}$ values. Using the Kruskal's algorithm, the authors find a forest of maximum cardinality in $\overline{G}$, giving preference to include the edges with higher $\overline{x_e}$ values. Each edge selected in this method merges two connected components into a large one. For every new connected component, the node in the component are checked for *GSEC*s violation. The process continues until a forest is found with maximum cardinality. If no violated *GSEC*s are found

with the separation heuristic, they branched on y variables. An upper bound on *MCDS* was determined using dynamic greedy heuristic stated in [27]. This heuristic is based on generating a spanning tree of G with as many leaves as possible. The heuristic starts with initialing $DS = \{v\}$ and $L = N(v) \setminus v$ for some $v \in V$. The main idea is to add nodes in list DS from L until a *CDS* has been found. Assume, node i has been selected to be moved to DS in the next iteration, so $DS = DS \cup \{i\}$ and $L = L \setminus \{i\} \cup (N(i) \setminus DS)$. For inclusion in DS, preference is given to those nodes which have maximum numbers of neighboring nodes that are not included in L yet. The process continues until $V = DS \cup L$ where DS consists of the nodes of *CDS*, and L represents the leaf nodes in the tree.

To solve the *MCDS* problem, Gendron et al. [19] presented two exact algorithms. The algorithms are based on **Benders Decomposition** and **Branch and Cut**. They also developed a hybrid algorithm combining both approaches. They demonstrated two variants for each of the methods: *'Stand Alone'* version and the *'Iterative Probing'* version.

In Benders Decomposition, the problem has two parts: *Master Problem*, and *Sub Problem*. In the master problem, some constraints of the original problem are relaxed. In *MCDS*, the connectivity constraint is relaxed in the master problem. A solution to the master problem is a dominating set, *DS*. The sub problem checks whether the resulting *DS* is connected or not, and if it is not connected, then it generates additional cuts to add to the master problem (known as feasibility cuts). They obtain a lower bound of the feasibility cut based on the idea that, if the nodes in the *DS* are not connected, then there must be at least one node outside the nodes in *DS* needed for connectivity. They also strengthened the lower bound by using the number of nodes in a shortest path needed to connect some components. The iterative probing variation is based on the idea that, if there is no connected dominating set of a given cardinality, there is no connected dominating set with lower cardinality. We will discuss Benders Decomposition in Section 4.1.1.

The Branch and Cut [29] method is also used to solve the *MCDS* problem. The authors

[19] use the approach in [27]. The computational result showed that Benders Decomposition performed better in dense graphs whereas Branch and Cut method worked better in the sparse graphs. The authors [19] also presented a hybrid algorithm combining; both the Benders Decomposition and Branch and Cut method for a better computational result. In each iteration of the hybrid algorithm, the master problem is solved by the Branch and Cut process. The master problem at the root node is initialized with the cuts generated in the previous iterations [19]. The computational result shows that the hybrid approach performs better than Benders Decomposition and Branch and Cut approach.

Contention aware CDS has not received as much attention as *MCDS* in the literature. In [16], the authors presented one heuristic to find a contention aware *CDS*. The algorithm starts all nodes colored white. The node with the maximum degree is colored black, and all of its neighbors are colored grey. A gray node with minimum black neighbor and the maximum number of white neighbor is selected next and colored black, and all its white neighbors are colored grey. The selection process runs until no white node exists. The set with all the black nodes is the contention aware connected dominating set (*CACDS*). This is similar to the approximation algorithm of Guha and Khuller [21].

3.2 Studies on Broadcasting in a Distributed Network

Due to the lack of a centralized management in the wireless network, it is more effective to construct *CDS* in a distributed way. The large size of a network also hinders the computation of a *CDS*. Das and Bharghavan [10] provided a distributed implementation of the two centralized algorithms by Guha and Khuller [21]. Both implementations suffer from high message complexities. Various heuristics based on the neighborhood information of nodes have been proposed to minimize redundant broadcasting which we examine next. These heuristics can be divided into two categories: Reactive and Proactive.

In **Reactive approaches**, after receiving a packet, the receiver node itself decides whether to forward the packet or not. One of the most well-known algorithms based on the reactive approach is **Self Pruning**, proposed by Lim and Kim [25]. In Self Pruning, while forwarding a packet, node u attaches its neighborhood list, $N(u)$ to the packet's header. When another node v in the transmission range of node u receives the packet, it compares its neighborhood list, $N(v)$ with the sender's neighborhood list $N(u)$. If the rebroadcasting of node v covers at least one new node, then node v will transmit the packet. In other word, if $N(v) - N(u) \neq \emptyset$, node v transmits the packet. This can lead to redundant transmission as there can be many receivers of node u, who cover the same node w, and the nodes take the decision of broadcasting without coordinating among themselves. Another well-known reactive approach is **Improved Self Pruning** [31], where each node makes the forwarding decision based on 3-hop neighbor information. As a result, this approach performs much better than the traditional Self Pruning approach.

In **Proactive approaches**, the transmitting node decides which of its neighboring nodes should forward the message next and attaches the forward list with the packet's header. Upon receiving the packet, if the node finds itself in the forward list, it starts constructing its forward list from its neighbors and rebroadcasts the packet attaching its forward list; otherwise, the node does not participate in forwarding. The process of rebroadcasting ends when no node forwards. There are various types of proactive approaches.

One of the well-known proactive approaches is **Dominant Pruning (*DP*)** proposed by Lim and Kim [25]. The selection of forwarding nodes is based on 2-hop neighborhood information. When a node, v receives a packet from a node u and $v \in F_{\mathrm{u}}$ (where F_{u} is the forward list constructed by u), it selects the minimum number of nodes from its 1-hop neighbors minus the neighbors of u, $N(v) - N(u)$ to cover all the nodes of the set $U_{\mathrm{v}} = N(N(v)) - N(v) - N(u)$. $N(v)$ is discarded as they will receive when v will forward. As node u has already forwarded, so all the nodes in $N(u)$ have already received the message.

This algorithm performs better than blind flooding and self pruning. Two variants of *DP* - Partial Dominant Pruning and Total Dominant Pruning are described in [26].

- **Partial Dominant Pruning (*PDP*):** It uses 2-hop neighborhood information, and it works more effectively. In addition to deduct $N(v)$ and $N(u)$ from $N(N(v))$, it also deducts the neighbor of common neighbors of node u and v. This reduction of sets reduces the number of nodes in the forward list.

- **Total Dominant Pruning (*TDP*):** This requires 2-hop neighbors of the immediate sender node to be piggybacked with the broadcast packet. Thus, a node v receiving a packet from node u also deducts $N(N(u))$ from $N(N(v))$. The detail of this approach is described in Section 5.1.

Enhanced Partial Dominant Pruning (*EPDP*) [32] is an extended version of *PDP*. *EPDP* introduces a delay before forwarding the packet. It takes advantage of the fact that the same node may hear the same packet several times from its neighbors. This solution offers better performance than *DP* and *PDP*. *EPDP* is a combination of the reactive and proactive approach. As the nodes in the forward list from the immediate sender defers its transmission, and after the defer time, the node itself decides whether to rebroadcast the message. A new heuristic named **Extended Neighbor Information based Dominant Pruning (ExDP)** [2] optimizes *DP* by exploiting neighbor information of nodes from 2-hop to 3-hop while broadcasting the packet in the network at the cost of little additional overhead.

A contention aware heuristic for the distributed network is presented in [16]. This heuristic is based on Dominant Pruning approach and reduces contention at the cost of additional forwarding nodes compared to *DP*. The integer programming formulation of distributed *CDS* has not been studied in the literature till now, neither for regular one nor for the contention aware one.

In summary, we can see that there are several approaches to reduce redundant broadcasts using *MCDS*. Some of the algorithms use global topology information to minimize redundancy. However, to achieve better performance in a mobile environment, distributed algorithms are needed. Many researchers have also studied the IP formulation of *MCDS* and different approaches to solve the IP. Until now, there is no work done on *IP* formulation of distributed *CDS*. Contention aware *CDS* is a new topic, and very few studies exist. The integer programming formulation of *CACDS* both for the centralized and distributed network has not been studied to the best of our knowledge.

Chapter 4

Centralized Model

In this chapter, we discuss the Integer Programming (IP) formulation for the Minimum Connected Dominating Set (*MCDS*) problem, as well as the Contention aware Connected Dominating Set (*CACDS*) problem.

In order to present an IP formulation for *MCDS*/*CACDS*, let us use the following decision variables:

$$x_i = \begin{cases} 1, & \text{if node } i \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

and

$$e_{ij} = \begin{cases} 1, & \text{if edge } (i,j) \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

For each node i in the network, the corresponding decision variable x_i is set 1 if the node i is selected to be a member of *MCDS*/*CACDS*; otherwise it is 0. If edge (i,j) is chosen for the solution, then e_{ij} is set to 1 otherwise it is 0.

4.1 The IP Model of Minimum Connected Dominating Set (MCDS)

The IP model of *MCDS* can be stated as follows:

$$minimize \sum_{i \in V} x_i \tag{4.1}$$

Subject to:

$$\sum_{j \in N(i)} x_j \geq 1 \qquad \forall i \in V \tag{4.2}$$

$$\sum_{(a,b) \in (S, V \setminus S)} e_{ab} \geq x_i + x_j - 1 \qquad \forall S \subset V : N(S) \neq V, N(V \setminus S) \neq V, \forall i \in V, \forall j \in V \tag{4.3}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{4.4}$$

$$e_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{4.5}$$

The objective function defined by Eq. 4.1 is to find the minimum number of nodes that will act as a connected dominating set of the given graph. Only the nodes in *MCDS* are used as the broadcasting/forwarding nodes in the network. The constraint (4.2) is the dominating set constraint. $N(i)$ is the set of adjacent nodes of node i. Note that, node i itself also is a member of this set, $\{i\} \in N(i)$. We use the **Cut formulation** to ensure the connectivity among the chosen dominating nodes in the network. Constraint (4.3) is the cut-constraint. More precisely, suppose, $S \subset V$, $S \neq \emptyset$, $N(S) := \bigcup_{i \in S} N(i)$. The edges in the cut implied by S is denoted by $(S, V \setminus S) := \{(a,b) \in E : a \in S, b \notin S\}$. Whenever, $N(S) \neq V$ and $N(V \setminus S) \neq V$, at least one edge in $(N(S), V \setminus N(S))$ must be chosen. This is true as the nodes in a *CDS* cannot be exclusively limited to S or to $V \setminus S$.

This is an exponential-sized IP. Hence, it is not a great idea to write down all the constraints and solve it directly. One of the ways to solve this problem is by using Benders Decomposition [20]. We discuss the implementation of Benders Decomposition algorithm in the context of the *MCDS* problem in the next section.

4.1.1 Benders Decomposition with Iterative Probing Strategy

The Benders Decomposition [20] has been applied effectively to a wide variety of difficult problems. This algorithm divides the problem into a master problem and a sub problem. Given the initial master problem and sub problem, the algorithm iterates between them (starting with the master problem) until it finds an optimal solution. Figure 4.1 is a schematic representation of the Benders Decomposition method.

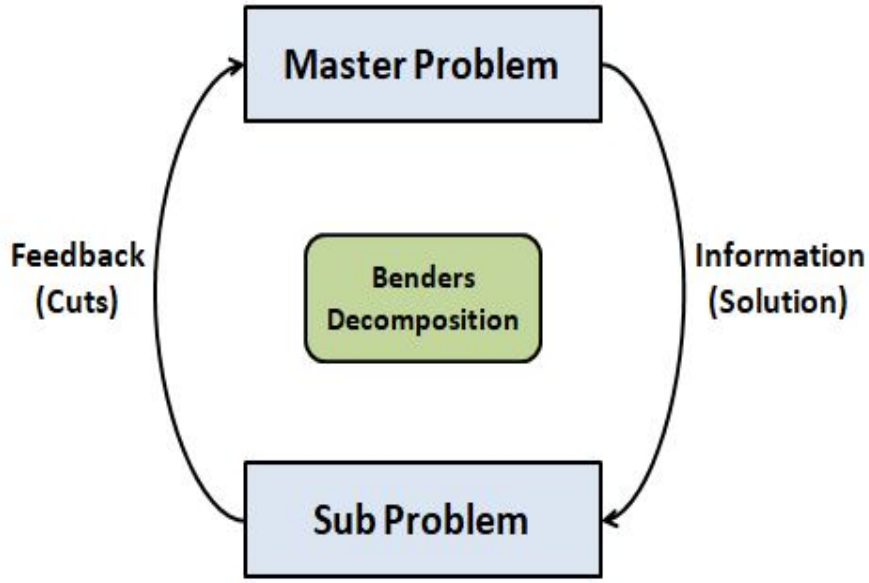


Figure 4.1: A Schematic representation of the Benders Decomposition

By relaxing some of the constraints of the original model, the master problem of the Benders Decomposition is constructed. This induces a lower bound on the optimal objective function value. Based on the result of the master problem, the sub problem tries to find a solution that satisfies all the constraints in the original model.

In the case of *MCDS*, we relax the cut-constraint (4.3). So, the master problem in our model consists of constraints (4.1), (4.2), (4.4), (4.5) and some additional constraints called Benders Cut. By solving the master problem, we get a dominating set *DS* and a sub graph

$G = (DS, E(DS))$ induced by that dominating set. The next step is to verify whether the sub graph $G = (DS, E(DS))$ is connected or not. We can determine the connectivity of $G = (DS, E(DS))$ by applying a graph traversal algorithm like **Depth First Search (DFS)** [37] which can be performed in $O(|DS| + |E(DS)|)$ time. The sub problem checks the connectivity of $G = (DS, E(DS))$. If $G = (DS, E(DS))$ is connected, we add **Benders Optimality Cut** [19] in the next iteration. Otherwise, the solution is not feasible, and additional cuts are generated and added to the master problems which needs to be solved in the next iteration. These cuts are called **Benders Feasibility Cut** [19]. The algorithm continues until we find an optimal solution. Let us now discuss **Benders Optimality Cuts** and **Benders Feasibility Cuts** next.

- **Benders Optimality Cut:** The traditional way of obtaining a feasible solution of value z in Benders Decomposition is to impose a constraint that the objective value must be less than z. The initial value of z can be obtained by any the heuristic algorithm [27] of constructing *MCDS*.

$$\sum_{i \in V} x_i \leq z - 1 \qquad \forall i \in V \tag{4.6}$$

  This optimality cut constraint defined by Eq. 4.6 is added to the master problem iteratively.

  We use an iterative probing strategy with Benders Decomposition [19]. The iterative probing strategy maintains a basic property of a connected dominating set. For a given graph, if there exists a connected dominating set, *CDS* of size $k < |V|$, where $k = |CDS|$, there also exists a connected dominating set of size $k+1$. We can easily observe this. Assume there is a *CDS* of size k. If we add any other node from $V \backslash CDS$, it will be a connected dominating set of size $k+1$. As a consequence of this property, we can say that in the absence of a connected dominating set of size $k > 0$, there is no connected dominating set of size $k-1$.

Based on the above-stated property, the iterative probing strategy can be formulated as follows: Assume, we have a connected dominating set, *CDS* of size k. If $k = 1$, the algorithm stops and the resultant *CDS* is the minimum connected dominating set (*MCDS*). This case can easily be verified. Suppose i is the node selected as a connected dominating set. If $|N(i)| = |V|$ for any $i \in V$, then we can say that, the minimum connected dominating set is consisted of only one node, that is i, and the problem is solved.

If $k > 1$, the algorithm will search for a new connected dominating set of size $k-1$. If no such connected dominating set of $k-1$ exists, the algorithm stops and current *CDS* is the minimum *CDS* of size k. However, if the algorithm returns a new connected dominating set *CDS* of size $k-1$, update k with $k-1$ and iterate.

The whole process is shown in the following flowchart 4.2. Based on the method of iterative probing, our new optimality cut equation is defined as Eq. 4.7.

$$\sum_{i \in V} x_i \leq k \qquad \forall i \in V \tag{4.7}$$

So, the master problem of our model consists of Eq. 4.1-4.2, 4.4-4.5 and the optimality cut defined by Equation 4.7.

- **Benders Feasibility Cut:** Benders feasibility cuts are added if we obtain a disconnected sub graph after solving the Benders master problem. Suppose *DS* is the dominating set returned after solving the master problem, and the sub graph induced by the nodes in *DS* is not connected. So, we need at least one node from the set-$\overline{DS} = \{V \backslash DS\}$ to make the sub graph connected.

$$\sum_{i \in \overline{DS}} x_i \geq 1 \qquad \forall i \in \overline{DS} \tag{4.8}$$

There are finite number of these cuts and the graph will eventually be connected. This

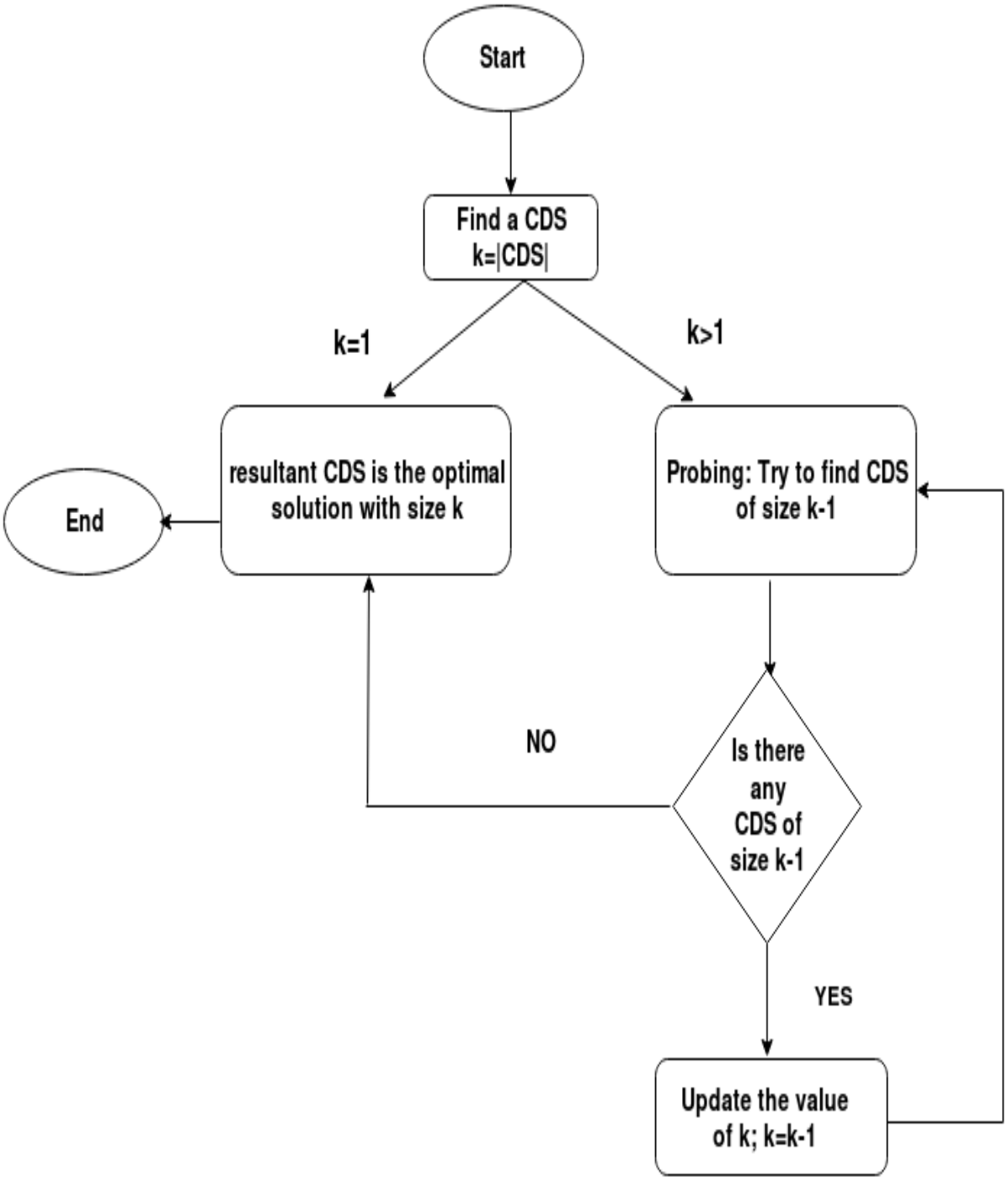


Figure 4.2: Iterative Probing Strategy

gives an optimal *MCDS*. So, our the mathematical model of *MCDS* consists of Eq. 4.1-4.2,4.4-4.5, 4.7-4.8.

However, this cut inequality (Eq. 4.8) is not so strong as we may need several nodes to make the graph $G = (DS, E(DS))$ connected. By using the depth-first search (DFS) algorithm, we can find the list of connected components from $G = (DS, E(DS))$. A component, sometimes called a connected component, of an undirected graph is a maximal sub graph in which any two nodes are connected to each other by paths within the component. So, the nodes that belong to a connected component are connected to each other by a path within the component, but two connected components are not connected by a path with each other in the sub graph $G = (DS, E(DS))$ induced by DS. Now, our goal is to find a good lower bound on minimum number of nodes (ms) from $\overline{DS} = \{V \backslash DS\}$, needed to connect all the components.

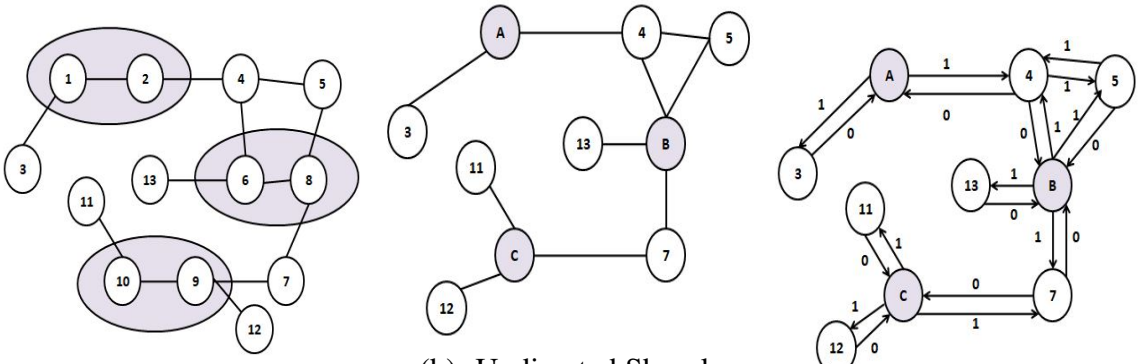
Lower Bound on Minimum Number of Nodes needed to Connect the Components:

Let, ms be a lower bound on the minimum number of nodes needed in order to make the components connected. We use two different approaches to compute the value of ms. The first approach is based on using the shortest path algorithm (SP based) [19], and our approach is based on the number of connected components and the maximum node degree of the graph in each step (Δ based).

1. **Based on the Shortest Path Algorithm (SP based):** To find a lower bound on the number of nodes needed to connect, we replace the original undirected graph with a new directed **shrunk graph**. Each connected component is replaced by a new node called *shrunk node*. Shrunk nodes is also called as **terminal nodes**. All the other nodes $i \in \overline{DS}$ is called as **non terminal nodes**. We convert an undirected graph to a directed one by replacing each edge with two edges, one in each direction.

The edges have costs. The cost of the directed edges that point to the **terminal nodes** (shrunk nodes) is set to 0; otherwise, it is set to 1.

A step by step process of constructing a shrunk graph is shown in Figure 4.3. Suppose we have a network of 12 nodes. After solving the master problem, we get, $\{DS\} = \{1,2,6,8,9,10\}$. There are three connected components $\{1,2\}$, $\{6,8\}$, $\{9,10\}$ highlighted in Figure 4.3(a). Each connected component is replaced by a **terminal node** (shrunk node), shown in Figure 4.3(b). Finally, the directed shrunk graph with associated edge cost is shown in Figure 4.3(c).



(a): Sample Graph with 12 nodes and connected components.

(b): Undirected Shrunk Graph (Connected components are replaced with new nodes)

(c): Directed Shrunk graph with Edge Cost

Figure 4.3: Step by step construction of the shrunk graph

The next step is to find a minimum cost path between each pair of **terminal nodes**. The shortest distance between each pair of nodes is the minimum number of **non terminal nodes** that are needed to connect that pair. To calculate the shortest path, we use the well known shortest path algorithm (**Dijkstra**) in the new shrunk graph with the associated edge costs. Finally, the maximum of these minimum distances between the terminal nodes leads us to the lower bound (LB) on the number of nodes needed to connect the components.

The maximum of all the minimum distances between each pair of terminal

nodes is a good lower bound for the feasibility cut equation. So, the feasibility cut can be lifted with this new value of *ms*.

2. **Based on Number of connected component and the maximum degree: (Δ based)** Our approach to calculate the value of *ms* is by using the number of connected components and the maximum degree of non terminal nodes in each step. Let us define the following variables.

   m : the number of terminal nodes which is the same as the number of connected

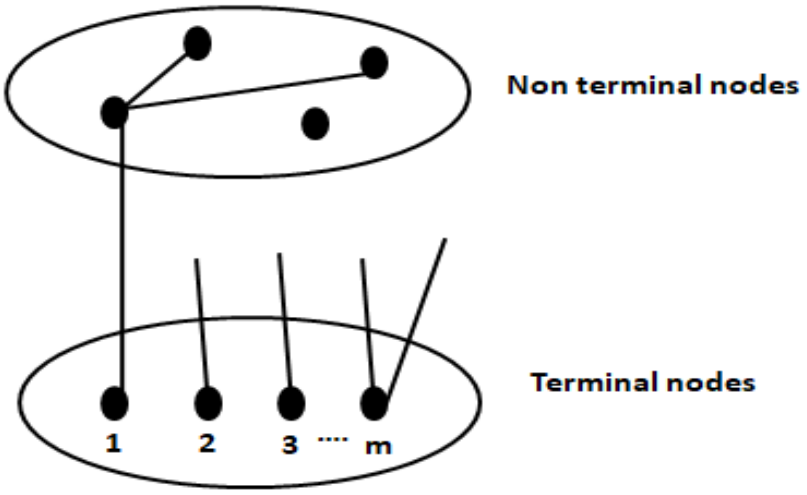


Figure 4.4: Connected component and node degree

components.

Δ : the maximum degree of the non terminal nodes in the undirected shrunk graph.

T : set of all terminal nodes.

NT : set of all non-terminal nodes.

d_T: the sum of degrees of all terminal nodes in the undirected shrunk graph.

$$d_T = \sum_{i \in T} deg(i) \qquad \forall i \in T$$

d_{NT}: The sum of degrees of all non terminal nodes in the undirected shrunk

graph.

$$d_{NT} = \sum_{j \in NT} deg(j) \qquad \forall j \in NT$$

We need at least $m+1$ nodes to make the terminal nodes connected; hence we need at least m edges.

Each edge incident on T has other end point in NT shown in Figure 4.4. Therefore,

$$d_{NT} \geq m \tag{4.9}$$

By definition,

$$d_{NT} \leq |NT| \times \Delta \tag{4.10}$$

We can proof the validity of inequality 4.10 by following argument:

As, Δ is the maximum degree of the non terminal nodes - that is, the largest degree of any node in the non terminal set. We have,

$$deg(j) \leq \Delta \qquad \forall j \in NT$$

for all $j \in NT$, and therefore we may take the sum of both sides of the inequality above, for each $j \in NT$, to get the inequality

$$\sum_{j \in NT} deg(j) = d_{NT} \leq \sum_{j \in NT} \Delta$$

The second sum here is a sum of constants: for every $j \in NT$, we are taking the same value Δ, not depending on j. This is because Δ is an upper bound on each term of the first sum. We are not taking a sum over maximum degree nodes; the sum still ranges over all nodes j, it is just ignoring the actual value of $deg(j)$

and instead using the upper bound Δ.

Because the second sum is a sum of constants, its value is just the number of terms, multiplied by the value of a term. So we get

$$d_{NT} \leq \sum_{j \in NT} \Delta = |NT| \times \Delta$$

Hence 4.10 is proved.

Each of the non terminal node has at most Δ edges, as Δ is the maximum node degree among all the node degrees of non terminal nodes. So, the number of non terminal nodes, we need to connect m terminal nodes is at least $\frac{m}{\Delta}$.

To ensure the connectivity among the non terminal nodes, we need at least $\frac{m}{\Delta} - 1$ nodes. So, the new lower bound on the number of non-terminal node needed to connect the terminal nodes with each other can be

$$|NT| \geq \left\lceil \frac{m}{\Delta} \right\rceil + \left\lceil \frac{m}{\Delta} \right\rceil - 1 = \left\lceil \frac{2m}{\Delta} \right\rceil - 1 \tag{4.11}$$

This is called as Δ based approach throughout this thesis.

Again, if h is the length of shortest path to make the non terminals connected, we can write,

$$d_{NT} \geq m + (h-1)$$

From Eq. 4.10,

$$|NT| \geq \left\lceil \frac{m + (h-1)}{\Delta} \right\rceil \tag{4.12}$$

As we are looking for a lower bound, the maximum of Eq. 4.11 and 4.12 acts as a good lower bound on the number of non terminal nodes needed to connect

the terminal nodes to each other denoted as *ms* for our problem.

$$ms \geq max\{\left\lceil \frac{m+(h-1)}{\Delta} \right\rceil, \left\lceil \frac{2m}{\Delta} \right\rceil - 1\} \tag{4.13}$$

Eq. 4.13 can easily be verified. Given, $a \geq x$ and $a \geq y$. a, x and y are positive integers. Two conditions are possible.

$$a \geq x \geq y, \qquad \text{when } x \geq y.$$

$$a \geq y \geq x, \qquad \text{when } y \geq x.$$

Comparing both the cases we can say that,

$$a \geq max(x, y)$$

Hence proved Eq. 4.13.

Above are the ways we have calculated the lower bound to have minimum number of non-terminal nodes needed to connect the terminal nodes to each other.

So, given a **disconnected dominating set**, *DS* by the master problem and *ms* (computed by *SP* based approach or Δ based approach) be the lower bound on the number of nodes needed from the set $\overline{DS} = \{V \setminus DS\}$ to obtain a connected dominating set, the new strengthened feasibility cut [19] is stated in Eq. 4.14.

$$\sum_{i \in \overline{DS}} x_i \geq ms \qquad \forall i \in \overline{DS} \tag{4.14}$$

So, our final mathematical model of *MCDS* consists of Eq. 4.1-4.2, 4.4-4.5. The optimality cut is stated in Eq: 4.7 and the feasibility cut is as Eq. 4.14.

4.1.2 The Outline of the whole Algorithm

The iterative probing version of Benders Decomposition is stated as follows:

1. Start with a connected dominating set including all nodes in the network, i.e.: $k = |CDS| = |V|$.
2. Solve the master problem with the optimality cut.
3. Check whether there is a feasible solution or not. If NO, the algorithm ends, and resultant *DS* is our *MCDS*.
4. If a feasible solution is found, then check whether the new sub graph $G = (DS, E(DS))$ induced by *DS* is connected or not by using the Depth First Search.
5. If the new sub graph $G(DS, E(DS))$ is connected, update the right-hand side of Eq. 4.7 with $k = k - 1$ and go to step 2.
6. If the new $G = (DS, E(DS))$ is not connected, generate the feasibility cut referred in Eq: 4.14 and go to step 2. For generating the feasibility cut, we need the minimum number of nodes (*ms*) needed to connect the nodes selected by the master problem. The value of *ms* can be generated by using any of the two ways discussed in Section 4.1.1.

4.2 IP formulation of Contention Aware Connected Dominating Set (CACDS)

A Contention aware Connected Dominating Set $(CACDS)$ is a special version of the traditional *MCDS* problem. If we use the regular *MCDS* for this problem, it can select the nodes in such a way that generates a cycle of odd length among the selected nodes. If there is a cycle among the selected nodes, that will definitely create contention among the nodes

at the time they will forward any message. So, our goal is to select the nodes (members of *CDS*) to avoid contention. A *CDS* which induces a tree does not have any contention as there are no cycles. If there is a cycle of odd length among the nodes in *CDS* and if the nodes receive a message in a specific order, then contention can arise. Our integer program does not minimize the number of (induced) odd cycles directly. It tries to reduce the extra edges (induced) in addition to the edges needed for connectivity.

A dominating set which induces a tree will definitely lead to no contention while transmitting the message through the network. This motivates the following integer programming.

$$\textit{minimize} \quad \alpha * \sum_{(i,j)\in E} e_{ij} - \sum_{i\in V} x_i + 1 \tag{4.15}$$

Subject to:

$$\sum_{j\in N(i)} x_j \geq 1 \qquad \forall i \in V \tag{4.16}$$

$$x_i \geq e_{ij} \qquad \forall (i,j) \in E \tag{4.17}$$

$$x_j \geq e_{ij} \qquad \forall (i,j) \in E \tag{4.18}$$

$$e_{ij} \geq x_i + x_j - 1 \quad \forall (i,j) \in E \tag{4.19}$$

$$\sum_{(a,b)\in(S,V\setminus S)} e_{ab} \geq x_i + x_j - 1 \qquad \forall S \subset V : N(S) \neq V, N(V \setminus S) \neq V, \forall i \in V, \forall j \in V \tag{4.20}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{4.21}$$

$$e_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{4.22}$$

The objective function of this model is to find out a tree of minimum size from the given graph to minimize contention. Note that, the graph induced by the *MCDS* is not guaranteed to be a tree, we hope that it is. As we know, any tree with n number of nodes has $n-1$

number of edges. In our objective function defined by Eq. 4.15, α is chosen as sufficiently large value to obtain desired subgraph.

The constraint (4.16) defines the dominating set and constraint (4.20) is the connectivity constraints same as we used for the *MCDS* model discussed in Section 4.2.

In order to minimize the contention, a new constraint has been added to the model. If any two nodes are selected, then the edges between them should be selected.

$$e_{ij} \geq x_i.x_j \qquad \forall (i,j) \in E \tag{4.23}$$

Minimizing the number of edges minimize contention. Constraints (4.17)-(4.19) represent the linear transformation of Eq: 4.23.

We solve the problem by using a combination of iterative probing strategy and Benders Decomposition as discussed in Section 4.1.1.

The computational time of constructing *CACDS* using this model is quite high. So, instead of using this model to construct *CACDS* , we use the result of *MCDS* as a starting point for this algorithm. If there is a contention among the nodes of *MCDS*, We calculate

$$p = \alpha * E_{MCDS} - N_{MCDS} + 1 \tag{4.24}$$

where, $E_{MCDS} =$ no of edges selected by the *MCDS* algorithm, $N_{MCDS} =$ no of nodes selected by the *MCDS*. We apply a new constraint to the *CACDS* model,

$$\alpha * \sum_{(i,j) \in E} e_{ij} - \sum_{i \in V} x_i + 1 \leq p - 1 \tag{4.25}$$

Therefore, instead of changing the objective function, we have added constraint (4.25) to the model (discussed in Section). The following steps are added to the previous algorithm

(discussed in 4.1.2).

1. Solve the model with the additional constraint (4.25).
2. If there is no feasible solution, stop the algorithm. Previous *CACDS* is the resultant set.
3. If there is a feasible solution with zero contention, stop the algorithm.
4. If there is a feasible solution with non-zero contention, update the value of p with the newly selected no of edges and nodes and go to 1.

Chapter 5

Distributed System

Wireless networks and MANET lack of centralized administration, therefore a construction of distributed *CDS* algorithm is needed. Furthermore, it is time consuming to compute centralized *CDS* if the topology size is large.

5.1 Basic Idea of Dominant Pruning (DP)

Dominant Pruning (DP) [26] uses neighborhood information to construct a dominating set, which acts as a forwarding node set for the network. The neighborhood information up to 2-hop apart is considered. The nodes gather the 2-hop neighborhood information by exchanging the neighborhood list. Neighborhood information is the basis of most of the routing algorithms. The nodes in the wireless network periodically send "whoami" messages to announce their presence in the network. The sender node also piggybacks its neighborhood list in the header of the packet before forwarding it. Thus, it is easier for node v to gather its 2-hop neighborhood information, $N(N(v))$ from its 1-hop neighbors, $N(v)$.

The sender nodes selects some of its adjacent nodes to relay the packet. The IDs of the selected nodes are registered in the header of the packet. This list of selected nodes is called the forwarding list. Upon receiving a packet, only the nodes whose IDs are there in

the forwarding list attached to the header of the receiving packet, rebroadcast the packets. Nodes while forwarding a packet will update the forwarding list by selecting some of the nodes from its neighborhood list. This process continues until the broadcast is complete.

Now we discuss the process of selection of forwarding nodes.

In Dominant Pruning [26], in addition to forwarding packet, each node u determines the set of forwarding nodes, F_u from its 1-hop neighbor list, $N(u)$ for covering all the 2-hop neighbors, $N(N(u))$ of u. In order to minimize redundant transmission, node v selects the next forwarding nodes from $B_u \subseteq N(u)$ that will cover all the nodes in $U_u \subseteq N(N(u))$. (B_u and U_u are defined later.) It is possible to model the selection of F_u as a set cover problem, as depicted in Figure 5.1.

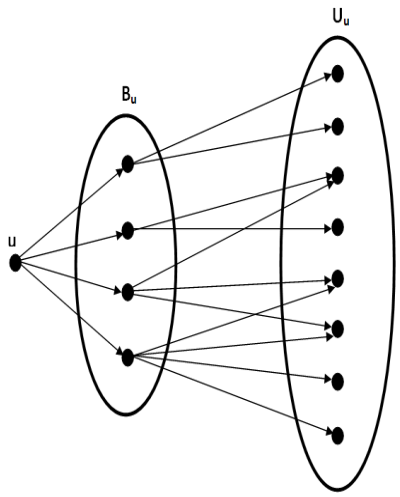


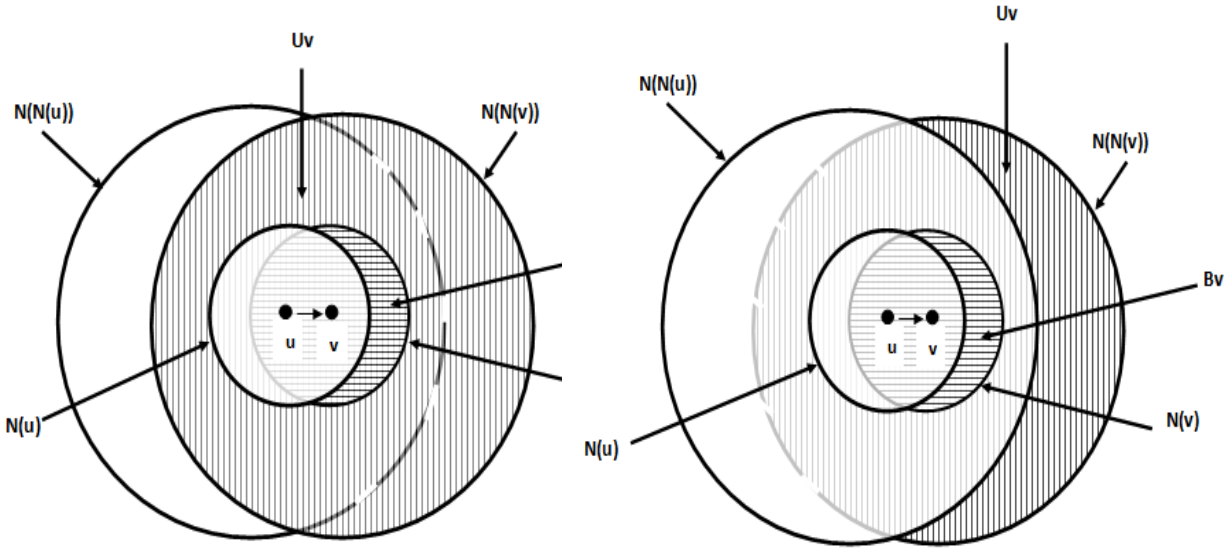
Figure 5.1: Connectivity among node u, B_{u}, U_{u}

Suppose a node u is the broadcast initiator. When node u sends a packet, it also sends the forward list, F_u with the packet header. Upon receiving the packet from node u, if node v finds itself in the forwarding list, F_u (i.e. $v \in F_u$) attached to the packet header, then node v will forward. Node v will construct its own forwarding list, F_v that will cover all its 2-hop neighbors, $N(N(v))$ and will insert it in the packet header. In other word, all the nodes in $N(N(v))$ will receive the packet once the nodes in F_v will forward it. Before constructing

the forwarding list, node v will construct a list U_v, the uncovered 2-hop neighbors of node v. Among the nodes in $N(N(v))$, node u is the source node, so node $N(u)$ already received the packet when node u had broadcast the packet. All the nodes consist of $N(v)$ will receive the packet when node v broadcasts. So, the only uncovered nodes in $N(N(v))$ are:

$$U_v = N(N(v)) - N(v) - N(u)$$

Node u also considered nodes in $N(u)$ before constructing its own forwarding list, so node v does not need to consider the nodes in $N(u)$ for its own forward list. Therefore, node v constructs its own forward list F_v from the list B_v, where,



(a): $U_v = N(N(v)) - N(v) - N(u)$, $B_v = N(v) - N(u)$

(b): $U_v = N(N(v)) - N(N(u))$, $B_v = N(v) - N(u)$

Figure 5.2: Illustration of Dominant Pruning

$$B_v = N(v) - N(u) \tag{5.1}$$

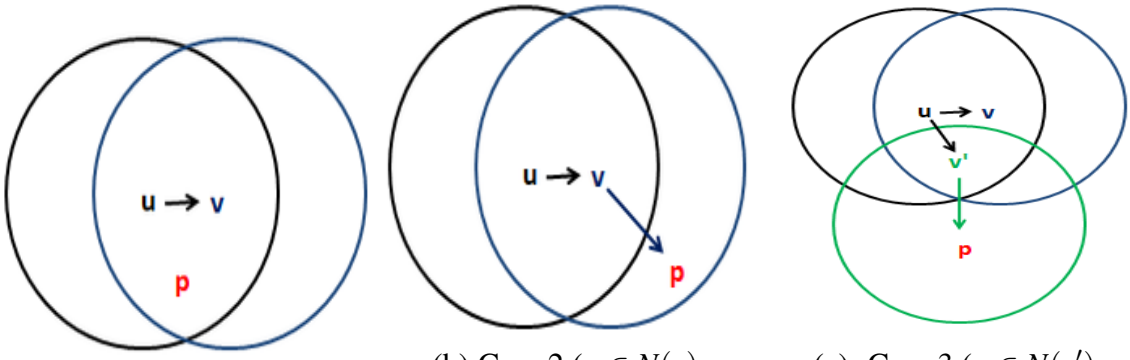
to cover all its uncovered 2-hop neighbors, U_v.

If node v can receive a packet with a list of nodes in $N(N(u))$ (node v gets the packet from node u), then the uncovered 2-hop neighbors of node v can be updated as,

$$U_v = N(N(v)) - N(N(u)) \tag{5.2}$$

The pictorial view of the scenario is shown in Figure 5.2.

The correctness of Eq. 5.2 can be shown by following lemma.



(a): Case 1 ($p \in N(u)$)

(b):Case 2 ($p \in N(v)$, $v \in N(u)$ & $v \in F_u$)

(c): Case 3 ($p \in N(v')$, $v' \in N(u)$ & $v' \in F_u$)

Figure 5.3: Cases for *Lemma 1*

Lemma 1: If we have a node p such that, $p \in N(N(v))$ as well as $p \in N(N(u))$, it is possible to eliminate node p from U_v.

Proof: As previously stated, u is the broadcast initiator, $v \in F_u$, U_v is the set of nodes that needed to be covered by the forwarding list of v. Let p is a node such that $p \in N(N(v))$ and if p also belongs to $N(N(u))$, the following three scenarios can happen:

- Case 1: p is a member of $N(u)$ ($p \in N(u)$)[including p is v itself]. When node u broadcast the message, node p already received the message. The scenario is depicted in Figure 5.3 (Case 1).
- Case 2: p is not a member of $N(u)$ ($p \notin N(u)$). Node u selects node v (where $v \in N(u)$ & $v \in F_u$) to cover node p (where $p \in N(v)$). The scenario is illustrated in Figure 5.3 (Case 2).

- Case 3: Node p can not be covered by node v ($p \notin N(v)$), node u selects another node of its neighbor $v' \in N(u)$ & $v' \in F_u$ to cover node p where $p \in N(v')$. The scenario is illustrated in Figure 5.3 (Case 3).

So, node v can exclude node p while constructing its forwarding list as node u already considered that node p receives the message as node p is in the 2-hop neighborhood list of node u.

Lemma 2: Let the set $U_v = N(N(v)) - N(N(u))$ and $B_v = N(v) - N(u)$, then $U_v \subseteq N(B_v)$

Proof: Let, X and Y are two sets. By using the fact that $N(X) - N(Y) \subseteq N(X - Y)$, we can prove this lemma. For any $p \in N(N(v)) - N(N(u))$, we have $p \in N(N(v) - N(u))$. Therefore, $N(B_v) = N(N(v) - N(u))$ can cover $U_v = N(N(v)) - N(N(u))$.

However, this process will consume more bandwidth as 2-hop neighborhood information of each sender node has to be piggybacked in the broadcasting packet.

5.1.1 IP Formulation of Selecting Forwarding Nodes in Each Step

In this section, we discuss the Integer Programming formulation for selecting the forwarding nodes in each step based on the Dominant Pruning (DP) [26] algorithm. Assume, node v has received a packet from node u, and node v is constructing its own forwarding list. Let us use the following decision variable:

$$x_i = \begin{cases} 1, & \text{if node } i \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

For each node, $i \in B_v$, the corresponding decision variable x_i is set 1 if node i is selected to be a member of forwarding list, otherwise 0. The IP model can be stated as follows.

$$minimize \quad \sum_{i \in B_v} x_i \tag{5.3}$$

Subject to:

$$\sum_{i \in B_v \cap N(w)} x_i \geq 1 \qquad \forall w \in U_v \tag{5.4}$$

$$x_i \in \{0,1\} \qquad \forall i \in B_V \tag{5.5}$$

The objective function defined by Eq. 5.3 is to find the minimum number of forwarding nodes among all the nodes in B_v. We have to select the nodes from B_v that will cover all the nodes in U_v. $N(w)$ is the neighborhood nodes of each node $w \in U_v$. Eq. 5.4 is the covering constraint.

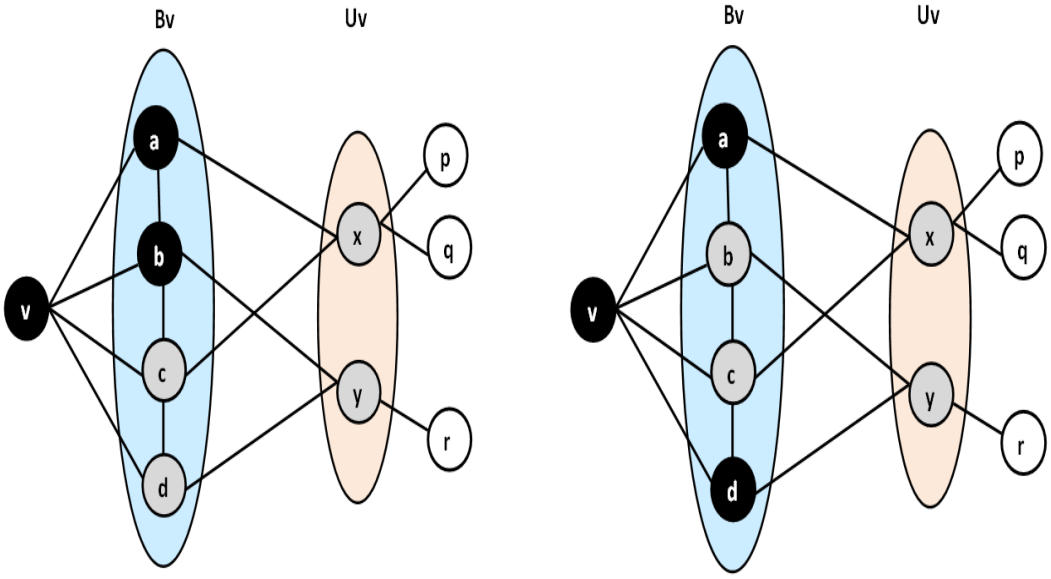
5.2 Basic idea of Contention aware Dominant Pruning (*CADP*)

As we stated earlier, if two or more neighboring nodes try to forward the packet at the same time, they will contend with each other for the shared medium. So, as in a centralized system, the selection of forwarding nodes in *DP* can also lead to contention. Let us discuss the situation, see Figure 5.4. The forwarding nodes are colored black. The nodes that already received the message are colored grey. The nodes yet to receive the message is colored white.

Suppose node v is constructing its own forwarding list, F_v. Node a, b, c and d are members of B_v, which are the possible candidates of the forwarding list. Set U_v consists of node x and node y. The following are the possible minimal set of forwarding nodes:

$$F_v = \{a,b\},\{b,c\},\{c,d\},\{a,d\} \quad .$$

If node-set $\{a,b\}$, $\{b,c\}$ or $\{c,d\}$ is selected as forwarding list by node v, it will directly leads to contention. If $\{a,d\}$ is selected as the forwarding set, it will cover all the nodes in U_v and there will be no contention among the forwarding nodes. The scenario is depicted



(a): Selection of forwarding nodes to minimize redundancy

(b): Selection of forwarding nodes to minimize contention

Figure 5.4: Regular Dominant Pruning and Contention Aware Dominant Pruning

in Figure 5.4(b).

If there exists a minimal independent set in B_v, such that each node in the independent set will act as a dominating set for the nodes in U_v, the problem of contention is solved. However, in the real-life scenario, it is not possible to find such independent sets as the nodes are quite close to each other. As a result of contention, a packet will not reach to the entire network. So, we propose another new approach that will minimize the contention and it will ensure that the packet is delivered to all the nodes in the network. We account for the edges between the nodes in B_v and try to pick up the nodes that is a dominating set for the nodes in U_v and the number of edges in the dominating set is minimized. The contention will be minimized and the packet will reach to all the nodes in the network.

5.2.1 IP Formulation of Selecting Contention Aware Forwarding Nodes in Each Step

In this section, we discuss the Integer Programming formulation for selecting Contention aware forwarding nodes in each step. Assume, node v has received a packet from

node u, and node v is constructing its forwarding list. We have the following decision variables:

$$x_i = \begin{cases} 1, & \text{if node } i \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

and

$$e_{ij} = \begin{cases} 1, & \text{if edge } (i,j) \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

For each node $i \in B_v$, the corresponding decision variable x_i will be set 1 if the node i is selected as forwarding node, otherwise 0. Another decision variable e_{ij} is defined for every edges (i,j) in the network. If edge (i,j) is chosen for the solution, then e_{ij} is 1 otherwise it is 0.

The model can be written as follows. This model is an extension of the IP model discussed in Section 5.1.1.

$$\textit{minimize} \quad \sum_{i \in B_v} x_i + \alpha * \sum_{(i,j) \in E} e_{ij} \tag{5.6}$$

Subject to:

$$\sum_{i \in B_v \cap N(w)} x_i \geq 1 \qquad \forall w \in U_v \tag{5.7}$$

$$x_i \geq e_{ij} \qquad \forall (i,j) \in E \quad \& \quad \forall i \in B_v \tag{5.8}$$

$$x_j \geq e_{ij} \qquad \forall (i,j) \in E \quad \& \quad \forall j \in B_v \tag{5.9}$$

$$e_{ij} \geq x_i + x_j - 1 \quad \forall (i,j) \in E \quad \& \quad \forall i \in B_v, \forall j \in B_v \tag{5.10}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{5.11}$$

$$e_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{5.12}$$

The objective function defined by Eq: 5.6 is to minimize the number of nodes as well as the number of edges between them in order to minimize the contention. α is a sufficiently large value to obtain desired hierarchic objective value. Constraint (5.7) is the dominating set constraint. In order to minimize the contention, a new constraint is added to the model. If any two nodes are selected, then the edges between them should be selected. Hence,

$$e_{ij} \geq x_i.x_j \qquad \forall (i,j) \in E \tag{5.13}$$

Without selecting the edges, we can not find out whether there will be contention or not. Constraints (5.8-5.10) represent the linear transformation of Eq. 5.13.

5.3 The Final Algorithm

The process of selecting the forwarding nodes is shown in Algorithm 1. Variable *Nodes* represents all the nodes in the network. We use a boolean array named $already\_forward$ to keep track of the forwarding nodes. *queue* is an array used to keep the nodes to be processed in the future. Function $add()$ is called to add an node in the array and function $remove()$ deletes the node from the array. Additionally function $top()$ returns the first element of the array. x_i is the decision variable retrieved from the model. The $initiator\_node$ represents the initiator of the broadcasting.

At the start, we add the $initiator\_node$ to the *queue*. Node v represents the first element of the *queue*. If node v did not participate in the forwarding before, then v will calculate B_v and U_v defined by Eq. 5.1 and Eq. 5.2 respectively. After having specified the necessary inputs, the algorithm calls the model defined in Section 5.1.1 (for regular DP) or Section 5.2.1 (for Contention aware DP). The result of the model gives the forwarding list, F_v of node v. The nodes in F_v are inserted in the *queue* for further processing. Node v is then

Algorithm 1 Selection of forwarding nodes

```
1:  u ← NULL
2:  Nodes = all nodes p ∈ V;
3:  queue = ∅;
4:  for all node p ∈ Nodes do
5:     already_forward[p] = 0;
6:  end for
7:  queue.add(initiator_node);
8:  while queue ≠ ∅ do
9:     F_v = ∅;
10:    v = queue.top();
11:    if already_forward[v] == 0 then
12:       Calculate U_v and B_v;                          {using Eq. 5.2 and 5.1}
13:       Solve the model;          {Section 5.1.1 for DP, Section 5.2.1 for CADP}
14:       for all i ∈ Nodes do
15:          if x_i == 1 then
16:             F_v.add(i);
17:             queue.add(i);
18:          end if
19:       end for
20:       already_forward[v] = 1;
21:       queue.remove(v);
22:       u = v;
23:    end if
24: end while
```

removed from the *queue*, and now it will be a sender node for subsequent iterations. The whole process iterates until the *queue* is empty.

Chapter 6

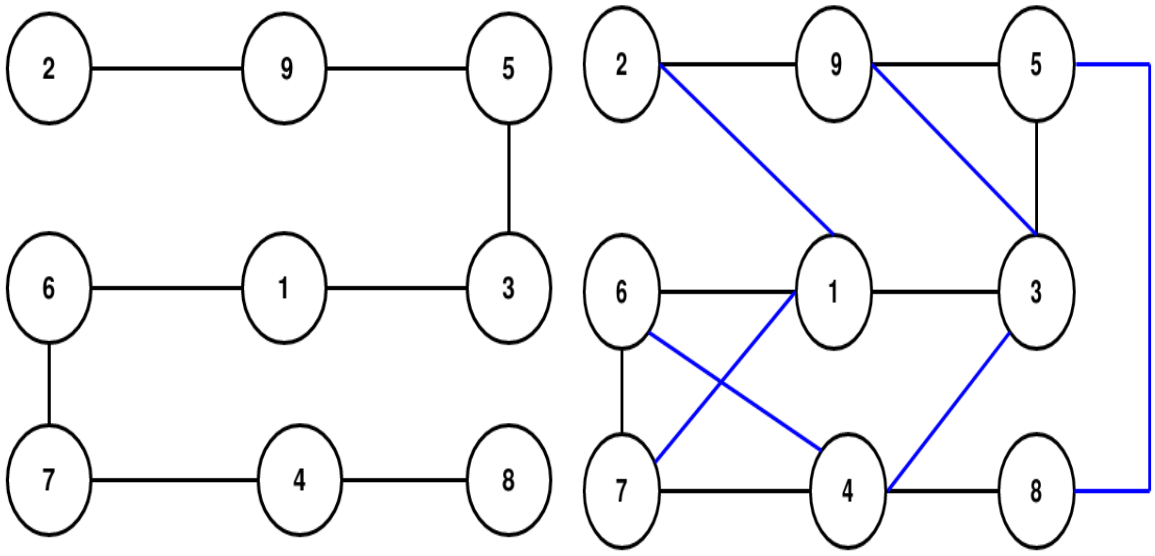
Simulation and Performance Evaluation

In this chapter, we discuss the computational results of the centralized and distributed system. As we discussed in Section 4.1.1, for computing the centralized *MCDS*/*CACDS*, we use two separate approaches to measure the lower bound on the number of nodes needed to obtain a *CDS*. A prior approach is based on the shortest path algorithm (*SP* based) [19], and ours is based on the number of connected components and maximum node degree (Δ based). We discuss the impact of these two different approaches on Benders Decomposition in calculation of *MCDS* and *CACDS*. We see how *CACDS* reduces contention. We also analyze the outcome of Dominant Pruning and Contention aware Dominant Pruning formulations.

6.1 Instances

The instances used for our experiments are generated randomly using a two-step procedure. At first, we create a **Hamiltonian path**, connecting all the nodes. The nodes are ordered randomly on the path. To obtain a uniform random permutation of the nodes, we create an array of n nodes and initialize the array elements as $1...n$. Then we use **Fisher-Yates shuffle algorithm** [17] to shuffle the elements of the array. The algorithm starts from the last element of the array and swaps that element with a randomly selected element from the entire array. We reduce the size of the array to $n-1$ and repeat the whole procedure

until we reach the first element. The run time complexity of the Fisher-Yates Shuffle Algorithm is $O(n)$. This gives the first $|V|-1$ edges. It also ensures that the generated graph is always a connected graph. Let $n = 9$, after application of the Fisher-Yates Shuffle Algorithm, assume the order of the nodes is $\{2,9,5,3,1,6,7,4,8\}$. The Hamiltonian path is the graph shown in Figure 6.1(a).



(a): Construction of Instances (Step 1) (b): Construction of Instances (Step 2)

Figure 6.1: Construction of Instances

In the second step, we generate additional edges by connecting pairs of randomly selected nodes. This step continues until a previously defined density for the graph is reached. Figure 6.1(b) shows the resulting graph, used as an input instance for our work.

6.2 Simulation Environment and Performance Metrics

6.2.1 Simulation Environment

We use the optimization programming language (*OPL*) V-4.0 and CPLEX Optimization Studio V-12.10 to implement and solve the models. CPLEX optimizer can solve very larger integer linear programs defined using the *OPL*. The simulation is carried on a 2.30 GHz

Intel(R) Core-i5 machine with 8 GHz of RAM (Random Access Memory). We evaluate our algorithms on instances varying in density from 5% to 70% and the number of nodes $n \in \{20, 30, 50, 60, 70, 100, 120\}$. Each instance name is a combination of the number of nodes and the density of graph, identified in the Column 1 of Table 6.1, 6.2 and 6.3 as n_d.

6.2.2 Performance Metrics

For centralized algorithms, we use four different parameters to evaluate our methods - the number of nodes selected (size of the $\{MCDS\}$ / $\{CACDS\}$), execution time (in seconds), the number of constraints and the number of contentions among the selected nodes.

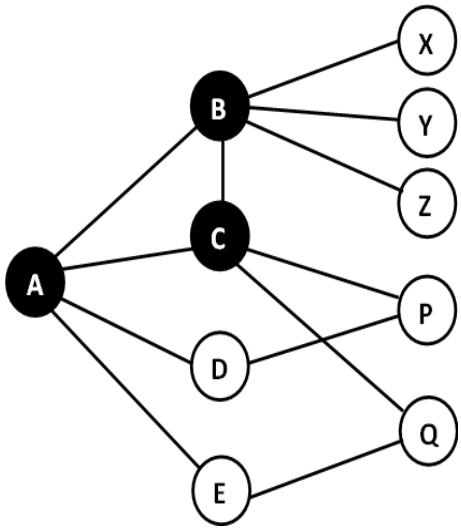
For the distributed system, we use the number of nodes selected, the number of contentions and the execution time (in seconds) as evaluation criteria.

Let us discuss the performance metrics in details.

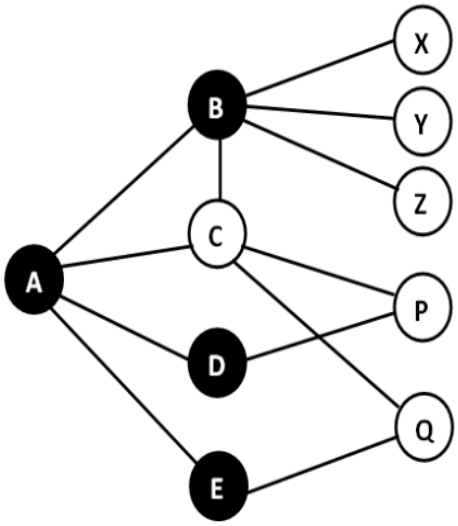
i **Number of Selected Nodes:** The number of selected nodes is the size of the *CDS*. It is also the value of the objective function. *MCDS* is used as a virtual backbone of a network to reduce the redundancy problem [38]. *CACDS*, with the goal to minimize contention selects another set of nodes or sometimes it selects some extra nodes than *MCDS*. *CACDS* finds a minimum number of broadcasting nodes with minimum contention. The scenario is shown using Figure 6.2.

In Figure 6.2, the black nodes are the selected nodes. In Figure 6.2(a), the size of *MCDS* is three. That means three nodes broadcast the message in the network. Suppose, node *A* is the broadcast initiator. After *A* broadcasts, when nodes *B* and *C* try to rebroadcast, contention arises. Figure 6.2(b) shows a *CACDS* in black nodes. The number of selected node here is four. Instead of node *C*, it selects node *D* and *E* to minimize contention.

The same scenario occurs while constructing *DP* and *CADP*. The number of selected



(a): Selection of forwarding nodes to minimize redundancy (Output of *MCDS*)



(b): Selection of forwarding nodes to minimize contention (Output of *CACDS*)

Figure 6.2: Effect of algorithms in term of forwarding nodes

nodes can increase in *CADP* that minimizes contention.

ii **Execution Time:** Execution time is the CPU time to find an optimal solution. For each instance, we run the algorithms for atmost 3600 CPU seconds. Whenever the time limit exceeds and an optimal solution is not obtained for the instances, we put a character '-' in the time entry.

iii **Number of Constraints:** The number of cut-constraints generated to solve the problem is another criteria we use to evaluate our centralized algorithms.

iv **Contention Number:** The contention number [16] among the selected nodes is determined using the Algorithm 2 (on the next page) while constructing *MCDS*/*CACDS*. $\{\text{MCDS}\}$ is the set of selected nodes.

We use the set $\{CACDS\}$ to calculate the contention for *CACDS* formulation.

The contention number for *DP* and *CADP* [16] for a broadcast is determined as follows. The number of nodes in the forwarding list of any node v that are neighbors of each other is defined as "*Per-hop Contention*". If a node in F_v is within the transmission

Algorithm 2 Calculate contention for *MCDS*

1: $Contention = 0$;
2: **for all** $i \in \{MCDS\}$ **do**
3: $X = \emptyset$;
4: $X = (N(i) - \{i\}) \cap \{MCDS\}$;
5: $\{MCDS\}$=$\{MCDS\}$-{i};
6: $Contention = Contention + \sum_{w \in X} \left(\left| (N(w) - \{w\}) \cap \{MCDS\} \cap N(i) \right| \right)/2$;
7: **end for**

range to another node, then those nodes face contention when they try to forward the packet. Thus *Per-hop Contention* (P) is calculated as:

$$P(v) = \sum_{w \in F_{\mathrm{v}}} \left(\left| (N(w) - \{w\}) \cap F_{\mathrm{v}} \right| \right)/2$$

Suppose, to complete a broadcast the nodes in $S = \{v_1, v_2, v_3, ..., v_z\}$ forward the packet, i.e., S is the set of selected broadcasting nodes by $DP/CADP$ and z is the number of these nodes. Let F represents the set of all the forwarding lists that are created at each hop.

$$F = \{F_{v_1}, F_{v_2}, F_{v_3}, ..., F_{v_z}\}$$

When we sum up *Per-hop Contention* of all the forwarding nodes in the network, we get the total number of contention that would occur for a broadcast. The total contention is defined:

$$Contention = \sum_{v_i \in S} P(v_i) = \sum_{v_i \in S} \sum_{w \in F_{v_i}} \left(\left| (N(w) - \{w\}) \cap F_{v_i} \right| \right)/2$$

6.3 Analysis and Presentation of Experimental Result of Constructing MCDS

Table 6.1 shows the detailed computational results for *MCDS*. We use two different lower bounds on the number of nodes needed to obtain a *CDS* (described in Section 4.1.1).

Table 6.1 shows the detailed result for both the methods.

Table 6.1: Detailed computational results: Benders Decomposition for *MCDS*

Instances	***SP* based**				**Δ based**			
n_d	**No of Selected Nodes**	**t(s)**	**No of Constraints**	**Contention**	**No of Selected Nodes**	**t(s)**	**No of Constraints**	**Contention**
20_*d*10	16	30.977	199	0	16	30.12	220	0
20_*d*20	6	1.02	29	0	6	0.99	39	0
20_*d*30	4	2.84	51	0	4	1.12	51	0
20_*d*40	3	0.105	21	0	3	0.059	21	0
20_*d*50	3	0.67	24	1	3	0.527	24	1
20_*d*60	3	0.605	23	1	3	0.488	23	1
20_*d*70	2	0.132	21	0	2	0.119	21	0
30_*d*10	15	51.84	71	3	15	50.87	78	3
30_*d*20	7	6.915	54	1	7	5.25	56	1
30_*d*30	5	1.57	34	1	5	1.46	34	1
30_*d*40	4	1.434	35	1	4	0.665	35	1
30_*d*50	3	0.226	31	1	3	0.15	31	1
30_*d*60	2	0.553	31	0	2	0.137	31	0
30_*d*70	2	0.382	31	0	2	0.159	31	0
50_*d*05	-	-	-	-	-	-	-	-
50_*d*10	12	23.217	74	3	12	21.142	79	2
50_*d*20	6	2.665	54	1	6	0.934	59	1
50_*d*30	5	2.195	52	2	5	0.502	52	1

Continued on next page

Table 6.1 – continued from previous page

Instances	***SP* based**				**Δ based**			
n_d	**No of Selected Nodes**	**t(s)**	**No of Constraints**	**Contention**	**No of Selected Nodes**	**t(s)**	**No of Constraints**	**Contention**
50_*d*40	5	1.05	51	1	5	0.19	51	1
50_*d*50	3	0.891	51	0	3	0.191	51	0
50_*d*60	2	1.55	52	0	2	0.396	52	0
50_*d*70	2	1.17	51	0	2	0.232	51	0
60_*d*05	-	-	-	-	-	-	-	-
60_*d*10	13	366.37	89	3	13	341.87	96	4
60_*d*20	7	12.799	63	2	7	1.119	65	2
60_*d*30	5	2.664	62	2	5	0.668	62	2
60_*d*40	4	2.385	62	1	4	0.613	62	1
60_*d*50	3	3.35	63	0	3	0.917	63	0
60_*d*60	3	1.314	61	1	3	0.322	61	1
60_*d*70	2	1.361	61	0	2	0.288	61	0
70_*d*05	-	-	-	-	-	-	-	-
70_*d*10	-	-	-	-	-	-	-	-
70_*d*20	7	7.7	71	3	7	1.344	75	3
70_*d*30	5	1.737	71	2	5	0.315	71	3
70_*d*40	4	3.588	72	2	4	0.53	72	2
70_*d*50	3	4.874	73	1	3	0.763	73	1
70_*d*60	3	1.904	71	0	3	0.309	71	1

Continued on next page

Table 6.1 – continued from previous page

Instances	***SP* based**				**Δ based**			
n_d	**No of Selected Nodes**	**t(s)**	**No of Constraints**	**Contention**	**No of Selected Nodes**	**t(s)**	**No of Constraints**	**Contention**
70_*d*70	2	1.908	71	0	2	0.331	71	0
100_*d*05	-	-	-	-	-	-	-	-
100_*d*10	13	1034.4	121	5	13	513.68	134	5
100_*d*20	8	9.107	102	2	8	1.659	102	3
100_*d*30	6	14.796	103	2	6	2.535	103	2
100_*d*40	5	18.798	104	2	5	3.365	104	2
100_*d*50	4	4.509	101	1	4	0.519	101	0
100_*d*60	3	4.704	101	0	3	0.537	101	0
100_*d*70	2	4.795	101	0	2	0.526	101	0
120_*d*05	-	-	-	-	-	-	-	-
120_*d*10	13	621.43	219	7	13	312.14	231	7
120_*d*20	9	79.13	121	4	9	31.063	121	4
120_*d*30	7	9.136	127	4	7	3.059	127	4
120_*d*40	6	8.128	121	3	6	1.672	121	3
120_*d*50	4	8.633	123	1	4	2.548	123	1
120_*d*60	3	7.757	121	2	3	1.894	121	2
120_*d*70	3	8.174	119	1	3	0.958	119	1

The results of the experiments on *MCDS* can be summarized as follows:

- **Effects on number of Selected Nodes:** The number of selected nodes indicates the size of *MCDS* ($|MCDS|$) as well as the model's objective function (described in Section 4.1). Only these nodes relay the message over the network. Table 6.1 shows that the number of nodes chosen by both methods is the same. To put it another way, we have the same objective value for both of the approaches. For a fixed no of nodes and with increasing density, the size of *MCDS* decreases. As the density of the graph increases, the graph becomes more connected, and we need fewer nodes to forward the message to the entire network. Table 6.1 shows for $n = 20$, the number of selected nodes decreases from 16 to 2 as the density of the graph increases to 10% to 70%.

- **Effects on the Execution Time:** We run the algorithms for atmost 3600 seconds on each instance. Character '-' in table means that the time limit exceeded and no solution was obtained. In terms of time, Benders Decomposition is faster for a dense graph rather than a sparse graph. Out of 54 instances, our algorithm solves 48 instances using both the approaches. Only sparse instances numbered - 50_d05, 60_d05, 70_d05, 70_d10, 100_d05 and 120_d05 could not be solved by the algorithm within the allowed time.

  The execution time for finding *MCDS* using Δ based approach (for all instances) is less than that of using *SP* based approach. The difference between the execution time of both the methods is noteworthy for larger graphs with more nodes. When the graph size is small, the execution time is almost the same. The gap between the two methods increases with the number of nodes in the graph. For example, for instance 60_d20, *SP* based approach takes about 13 seconds, Δ based approach provides a solution within 1 second. Figure 6.3 shows the effect on the execution time for $n = 30$ and $n = 120$ for both the approaches. The X-axis represents the graph's density in percentage, and Y-axis represents the execution time in seconds. We can see, Δ based approach is faster in both the scenarios. Although the time difference is not significant for $n = 30$ in both the approaches, but as the number of nodes increases

to 120, the difference is quite noticeable.

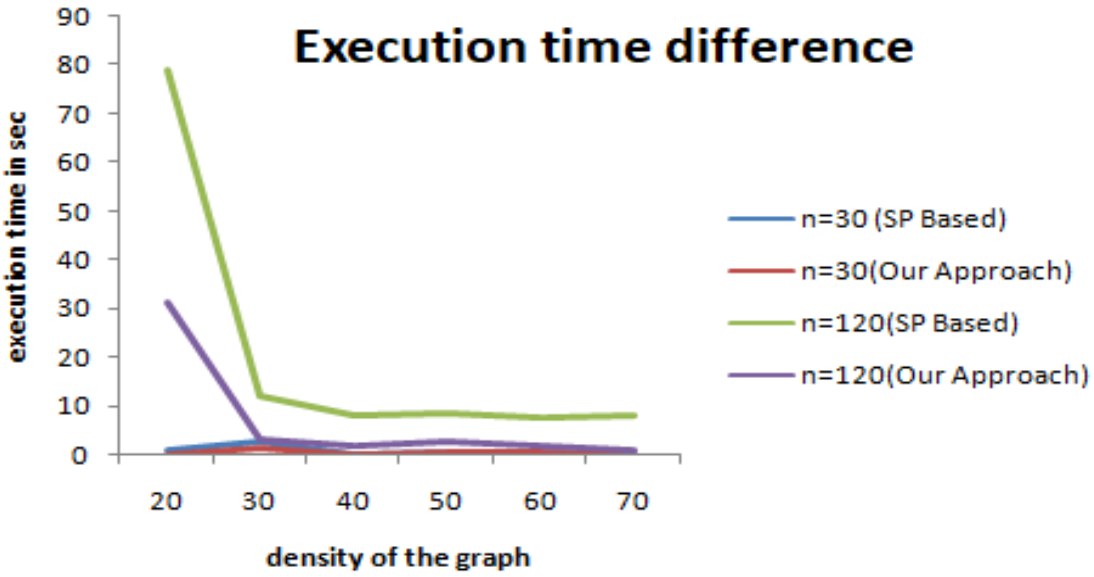


Figure 6.3: Execution time when $n = 30$ and $n = 120$ varying the density from 20% to 70%

- **Effects on number of Constraints:** The number of cut constraints generated by *SP* based approach for the sparse graph (graph density $< 30\%$) is slightly less than that of the number generated by Δ based approach. From Table 6.1, we see that the number of cut constraints for *SP* based approach is less than Δ based approach when the graph density is 10% -20%. However, both the approaches generated the same number of cut constraints when the graph density is 30% or more.

- **Effects on Contention:** We find contention in 32 instances, among 48 solved instances. In certain cases, for both approaches, the contention among the selected nodes is different. These instances include 50_d10, 50_d30, 60_d10, 70_d30, 70_d60, 100_d20, 100_d50. As two approaches select different sets of nodes while solving the problem, the contention varies with that. Overall there is little difference in the contention number.

From the above analysis, we conclude that, we get the same objective value using both the approaches. Δ based approach is faster than *SP* based approach for all the cases. As if we

know the number of connected components and the maximum degree in the graph, we do not need to traverse the whole graph and generate the shortest path to obtain a good lower bound on the minimum number of nodes need to construct a connected dominating set (as in *SP* based approach). *SP* based approach is time-consuming for larger networks.

6.4 Experimental Result for Constructing CACDS: Analysis

Table 6.2 shows the detailed computational results for constructing *CACDS*. We use similarly generated instances and evaluation parameters to evaluate *CACDS* (that we use for *MCDS* and discussed in Section 6.2.2).

Table 6.2: Detailed computational results: Benders Decomposition for Contention-aware CDS

Instances	***SP* based**				**Δ based**			
n_d	**No of Selected Nodes**	**t(s)**	**No of Constraints**	**Contention**	**No of Selected Nodes**	**t(s)**	**No of Constraints**	**Contention**
20_*d*10	16	32.12	413	0	16	30.98	422	0
20_*d*20	6	1.034	456	0	6	1.03	461	0
20_*d*30	4	2.98	517	0	4	1.312	525	0
20_*d*40	3	0.213	636	0	3	0.091	636	0
20_*d*50	3	0.791	809	0	3	0.57	809	0
20_*d*60	3	0.845	926	0	3	0.607	926	0
20_*d*70	2	0.15	1070	0	2	0.134	1070	0
30_*d*10	16	55.76	659	0	16	52.13	670	0

Continued on next page

Table 6.2 – continued from previous page

Instances	***SP* based**				**Δ based**			
n_d	**No of Nodes**	**t(s)**	**No of Constraints**	**Contention**	**No of Nodes**	**t(s)**	**No of Constraints**	**Contention**
30_*d*20	7	11.794	781	0	7	9.231	811	0
30_*d*30	5	10.42	1076	0	5	8.434	1087	0
30_*d*40	5	9.226	1475	0	5	7.333	1475	0
30_*d*50	3	7.308	1814	0	3	7.041	1814	0
30_*d*60	2	0.564	2120	0	2	0.143	2120	0
30_*d*70	2	0.382	2424	0	2	0.159	2424	0
50_*d*05	-	-	-	-	-	-	-	-
50_*d*10	14	25.136	984	0	14	23.15	996	0
50_*d*20	6	3.638	1966	0	6	1.465	1967	0
50_*d*30	5	12.614	2990	0	5	5.859	2990	0
50_*d*40	5	10.929	4071	0	4	6.253	4071	0
50_*d*50	3	0.93	4902	0	3	0.19	4902	0
50_*d*60	2	1.939	6077	0	2	0.678	6077	0
50_*d*70	2	1.88	6168	0	2	0.315	6168	0
60_*d*05	-	-	-	-	-	-	-	-
60_*d*10	15	370.12	3478	0	15	361.34	3512	0
60_*d*20	8	17.201	4019	0	8	15.148	4102	0
60_*d*30	6	3.038	4366	0	6	0.576	4366	0
60_*d*40	4	3.302	5758	0	4	0.679	5758	0
60_*d*50	3	3.82	7210	0	3	0.973	7210	0

Continued on next page

Table 6.2 – continued from previous page

Instances	***SP* based**				**Δ based**			
n_d	**No of Nodes**	**t(s)**	**No of Constraints**	**Contention**	**No of Nodes**	**t(s)**	**No of Constraints**	**Contention**
60_*d*60	3	1.864	8508	0	3	1.114	8508	0
60_*d*70	2	1.496	9878	0	2	0.473	9878	0
70_*d*05	-	-	-	-	-	-	-	-
70_*d*10	-	-	-	-	-	-	-	-
70_*d*20	7	241.78	4157	0	7	172.80	4230	0
70_*d*30	5	3.885	5857	0	5	1.335	5857	0
70_*d*40	4	35.657	8071	0	4	11.959	8071	0
70_*d*50	3	5.844	9801	0	3	3.15	9801	0
70_*d*60	3	8.983	11651	0	3	2.8	11651	0
70_*d*70	2	2.408	13624	0	2	0.838	13624	0
100_*d*05	-	-	-	-	-	-	-	-
100_*d*10	13	1056.13	5287	0	13	985.79	5287	0
100_*d*20	10	159.84	7863	0	10	73.783	7863	0
100_*d*30	6	120.12	10618	0	6	61.12	10618	0
100_*d*40	5	102.19	16035	0	5	59.823	16035	0
100_*d*50	4	7.641	19998	0	4	2.91	19998	0
100_*d*60	3	5.681	23727	0	3	0.637	23727	0
100_*d*70	2	4.766	28166	0	2	0.601	28166	0
120_*d*05	-	-	-	-	-	-	-	-
120_*d*10	16	1034.12	49356	0	16	987.74	50167	0

Continued on next page

Table 6.2 – continued from previous page

Instances	***SP* based**				**Δ based**			
n_d	**No of Nodes**	**t(s)**	**No of Constraints**	**Contention**	**No of Nodes**	**t(s)**	**No of Constraints**	**Contention**
120_*d*20	9	519.23	41529	0	9	412.23	41529	0
120_*d*30	8	429.13	38267	0	8	221.28	38267	0
120_*d*40	6	222.59	22787	0	6	159.89	22787	0
120_*d*50	4	393.40	29359	0	4	248.60	29359	0
120_*d*60	3	85.424	34687	0	3	37.71	34687	0
120_*d*70	3	12.518	38762	0	3	7.387	38762	0

The result of the experiments can be summarized as follows:

- **Effects on the number of Selected Nodes:** As we see in Table 6.2, the number of selected nodes by both the approaches is the same. That is the value of the objective function (described in Section 4.2) is the same for both the approaches. However, if we compare the number of selected nodes in *CACDS* and *MCDS*, there is some variation in number of selected nodes. This increase has been explained in Section 6.2.2.

  30_*d*10, 30_*d*40, 50_*d*10, 60_*d*10, 60_*d*20, 60_*d*30, 100_*d*20, 120_*d*10, and 120_*d*30 are the instances where the number of nodes selected by *CACDS* exceeds that of *MCDS*. Although some extra nodes are selected to minimize contention, the difference between the size of both the sets is not that high. For instances, *MCDS* selects five nodes and there are two contentions among the selected nodes for 60_d30, whereas *CACDS* generates contention-free forwarding nodes with addition of one

extra node.

- **Effects on Execution Time:** Out of the 54 instances, *CACDS* optimally solves 48 instances using both the approaches. Only 50_*d*05, 60_*d*05, 70_*d*05, 70_*d*10, 100_*d*05 and 120_*d*05 are not solved by the algorithm within the specified time allowed (3600 seconds).

  Like *MCDS*, the construction of *CACDS* shows the same behaviour with respect to the execution time. The method using Δ based approach is more efficient than *SP* based approach. The method performs better on dense graphs.

- **Effects on the number of Constraints:** The number of generated cut constraints is nearly the same in both the approaches. Δ based approach creates a few more constraints than *SP* based approach on sparse graph when the density around 05%-20%. However as the density increases, the number of cut constraints is the same for both the approaches.

- **Effects on Contention:** We got contention in 32 instances out of 48 for *MCDS*. However, *CACDS* provides us *CDS* set free of contention for all the solvable instances.

From the above analysis, we conclude that Δ based approach performs better than SP based approach in terms of execution time. They both select the same number of nodes as forwarding nodes. *CACDS* by choosing another set of nodes or in addition to some extra nodes other than *MCDS*, *CACDS* generates contention-free forwarding nodes. The nodes do not face contention with each other, as the differ time will be reduced, and the message will transmit faster in the network.

6.5 Analysis and Presentation of Experimental Result for Constructing DP & CADP

Table 6.3 shows the detailed computation result for the distributed system. We use the same instances to evaluate *DP* (discussed in Section 5.1.1) and *CADP* (discussed in Section 5.2.1) that we use to evaluate the centralized system.

Table 6.3: Detailed computational results: Dominant Pruning & Contention Aware Dominant Pruning

Instances	**DP**			**CADP**		
n_d	**No of Selected Nodes**	**t(s)**	**Contention**	**No of Selected Nodes**	**t(s)**	**Contention**
20_*d*10	19	9.047	2	19	11.64	0
20_*d*20	14	8.201	0	14	8.233	0
20_*d*30	14	7.843	1	14	11.115	0
20_*d*40	4	1.789	1	4	1.946	1
20_*d*50	3	1.429	0	3	0.935	0
20_*d*60	3	.962	1	3	0.95	0
20_*d*70	2	0.99	0	2	1.001	0
30_*d*10	26	49.432	9	28	52.953	4
30_*d*20	19	47.236	4	19	49.93	4
30_*d*30	13	31.338	4	16	53.236	2
30_*d*40	5	6.165	2	6	4.929	1
30_*d*50	3	4.242	1	4	8.981	0
30_*d*60	3	4.34	1	3	4.39	0
30_*d*70	3	4.405	1	3	4.437	0

Continued on next page

Table 6.3 – continued from previous page

Instances	**DP**			**CADP**		
n_d	**No of Nodes**	**t(s)**	**Contention**	**No of Nodes**	**t(s)**	**Contention**
50_*d*05	46	886.63	2	46	969.876	0
50_*d*10	37	671.236	8	37	676.08	8
50_*d*20	7	26.60	6	7	28.231	6
50_*d*30	6	28.63	6	6	28.94	3
50_*d*40	5	30.464	1	5	28.802	1
50_*d*50	4	30.17	1	4	31.87	0
50_*d*60	3	32.26	1	3	31.08	0
50_*d*70	3	34.84	1	3	32.46	0
60_*d*05	-	-	-	-	-	-
60_*d*10	47	1656.80	5	47	1782.498	3
60_*d*20	8	54.12	3	9	54.68	2
60_*d*30	6	55.013	4	7	58.316	1
60_*d*40	4	60.91	3	6	60.95	0
60_*d*50	4	62.585	2	4	64.118	0
60_*d*60	3	63.722	1	4	66.791	0
60_*d*70	3	63.748	1	3	68.889	0
70_*d*05	-	-	-	-	-	-
70_*d*10	33	1401	19	36	1418	8
70_*d*20	24	1001.7	10	27	1013.6	6
70_*d*30	15	592.82	5	18	577.10	2
70_*d*40	7	105.14	3	9	112.62	2
70_*d*50	5	103.14	2	6	111.65	0

Continued on next page

Table 6.3 – continued from previous page

Instances	**DP**			**CADP**		
n_d	**No of Nodes**	**t(s)**	**Contention**	**No of Nodes**	**t(s)**	**Contention**
70_*d*60	4	118.7	1	4	108.84	0
70_*d*70	3	122.03	1	3	122.60	0
100_*d*05	-	-	-	-	-	-
100_*d*10	43	599.69	20	50	603.7	12
100_*d*20	11	499.69	7	12	526.82	5
100_*d*30	7	467.41	4	8	552.64	1
100_*d*40	5	540.54	2	6	573.85	0
100_*d*50	4	576.64	1	4	622.14	0
100_*d*60	3	561.98	0	3	599.10	0
100_*d*70	3	573.19	0	3	593.03	0
120_*d*05	-	-	-	-	-	-
120_*d*10	-	-		-	-	-
120_*d*20	8	1517.63	6	10	1528.9	4
120_*d*30	7	1437.8	4	8	1443.8	0
120_*d*40	5	1123.92	1	5	1131.6	0
120_*d*50	4	997.24	1	4	997.98	0
120_*d*60	3	1081.95	0	3	1098.6	0
120_*d*70	3	1106.211	0	3	1152.9	0

The result in Table 6.3 can be summarized as follows:

- **Effect on number of Selected Nodes:** For a fixed set of nodes, as the density increases, the number of forwarding nodes decreases. In term of minimizing the con-

tention, *CADP* selects some extra nodes than *DP* for some instances. For $n = 100$, *DP* selects 43, 11, 7 and 5 nodes, whereas *CADP* selects 50, 12, 8, and 6 nodes for density 10%, 20%, 30% and 40% respectively. For density 50%, 60% and 70%, both the algorithms select same number of forwarding nodes.

- **Effect on the Execution Time:** Just like in previous simulations, we run the distributed algorithms for 3600 seconds. If the algorithms cannot produce solution within this particular time, we add a character '-" to the corresponding entry. Among the 54 instances, both *DP* and *CADP* can not solve five instances. 60_*d*05, 70_*d*05, 100_*d*05, 120_*d*05 and 120_*d*10 are the instances for which our distributed algorithms can not provide solution. The gap in time between the execution time of the two algorithms is not as large.

- **Effect on Contention:** *CADP* is better than *DP* at reducing the contention. There are 43 instances among 54 where *DP* can not generate contention-free forwarding nodes, whereas for 29 instances *CADP* generates contention-free broadcast nodes. Although *CADP* may not generate contention-free forwarding nodes in some cases, it does select the forwarding nodes with minimal contention. 30_*d*10, 30_*d*30, 50_*d*30, 60_*d*10, 60_*d*20, 70_*d*10, 70_*d*20, 100_*d*10, 120_*d*20 etc are such instances where *CADP* can not choose contention free forwarding nodes. For each of these cases, *CADP* chooses nodes with contention that is less than that of *DP*.

In summary, Benders Decomposition performs better in dense graph than sparse graph. The objective value obtained by *SP* based approach and Δ based approach is the same while constructing *MCDS*. The same scenario occurs in *CACDS*. *CACDS* reduces contention. *SP* based approach is time-consuming for larger graphs. Δ based approach is faster than *SP* based approach for all the cases for *MCDS* and *CACDS*. *CADP* performs better in minimizing contention that *DP* with a little increase in the execution time.

Chapter 7

Conclusion & Future Work

In this chapter, we discuss the main contribution of this thesis and outline directions for further research.

7.1 Summary

In this thesis, we study Integer Programming formulations for minimum connected dominating set (*MCDS*) and contention aware connected dominating set (*CACDS*). We use Iterative version of Benders Decomposition to solve the two problems. To find a connected dominating set, we study one state-of-art approach based on the shortest path algorithm (*SP* based) [19] and ours one is based on number of connected components and maximum degree (Δ based). We also develop an IP formulation for selecting the forwarding nodes in a distributed network based on Dominant Pruning [26] algorithm (Discussed in Section 5.1, Page 40-44), where each node has only two-hop neighborhood information. We also describe how to select contention aware forwarding nodes in the distributed network.

We use *CPLEX* optimization studio V-12.10 to implement and solve the models. The computational results in Chapter 6 show that Benders Decomposition is faster for dense graphs compared to sparse graphs to construct *MCDS*/*CACDS*. The objective value obtained by both the approaches (*SP* and Δ based) is the same as it should be. Δ based ap-

proach performs better than *SP* based approach to find *CDS*, specially when the network is large with large number of nodes. *SP* based approach is time-consuming for larger graphs. *CACDS* reduces contention. For a distributed system, the execution time for constructing forwarding nodes in the both the algorithms- *DP* and *CADP* is not significant. *CADP* reduces contention at the cost of a small number of additional forwarding nodes.

7.2 Future Work

Our method of Benders Decomposition with iterative probing provided good results on our instances, being extremely effective on dense instances. It would be interesting to explore the behavior of a similar approach on other specific graph classes.

The computational time of constructing *CACDS* using the model defined in Section 4.2 is quite high. So, we used an *MCDS* as the starting point for the algorithm for constructing *CACDS*. It would be an interesting to add constraints to the *CACDS* model so that a good starting point is not needed.

Bibliography

[1] Beongku An and Symeon Papavassiliou. A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks. *International Journal of Network Management*, 11(6):387–395, 2001.

[2] Tasmiah Tamzid Anannya and Ashikur Rahman. Extended neighborhood knowledge based dominant pruning (exdp). In *2018 5th International Conference on Networking, Systems and Security (NSysS)*, pages 1–9. IEEE, 2018.

[3] Sivakumar R Bevan Das and V Bharghavan. Routing in ad-hoc networks using a virtual backbone. In *Proceedings of the 6th International Conference on Computer Communications and Networks (IC3N'97)*, pages 1–20, 1997.

[4] Sergiy Butenko, Xiuzhen Cheng, Carlos A Oliveira, and Panos M Pardalos. A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks. *Recent developments in cooperative control and optimization*, 3:61–73, 2004.

[5] Sergiy Butenko, C Oliveira, and Panos M Pardalos. A new algorithm for the minimum connected dominating set problem on ad hoc wireless networks. *Proceedings of CCCT'03*, pages 39–44, 2003.

[6] Ian D Chakeres and Elizabeth M Belding-Royer. The utility of hello messages for determining link connectivity. In *The 5th International Symposium on Wireless Personal Multimedia Communications*, volume 2, pages 504–508. IEEE, 2002.

[7] Benjie Chen. An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Proceedings of ACM/IEEE MOBICOM'01*, 2001.

[8] Xiuzhen Cheng, Min Ding, and Dechang Chen. An approximation algorithm for connected dominating set in ad hoc networks. In *Proc. of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peer-to-Peer Networks (TAWN)*, volume 2, 2004.

[9] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.

[10] Bevan Das and Vaduvur Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *Communications, 1997. ICC'97 Montreal, Towards the Knowledge Millennium. 1997 IEEE International Conference on*, volume 1, pages 376–380. IEEE, 1997.

[11] Susanta Datta, Ivan Stojmenovic, and Jie Wu. Internal node and shortcut based routing with guaranteed delivery in wireless networks. *Cluster computing*, 5(2):169–178, 2002.

[12] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. Multi-resolution state retrieval in sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pages 19–29. IEEE, 2003.

[13] Min Ding, Xiuzhen Cheng, and Guoliang Xue. Aggregation tree construction in sensor networks. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 4, pages 2168–2172. IEEE, 2003.

[14] Anthony Ephremides, Jeffrey E Wieselthier, and Dennis J Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, 75(1):56–73, 1987.

[15] Neng Fan and Jean-Paul Watson. Solving the connected dominating set problem and power dominating set problem by integer programming. In *International conference on combinatorial optimization and applications*, pages 371–383. Springer, 2012.

[16] Chowdhury Nawrin Ferdous and Ashikur Rahman. A contention aware connected dominating set construction algorithm for wireless ad-hoc networks. In *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8. IEEE, 2018.

[17] Ronald A Fisher and Frank Yates. *Statistical tables: For biological, agricultural and medical research*. Oliver and Boyd, 1938.

[18] Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[19] Bernard Gendron, Abilio Lucena, Alexandre Salles da Cunha, and Luidi Simonetti. Benders decomposition, branch-and-cut, and hybrid algorithms for the minimum connected dominating set problem. *INFORMS Journal on Computing*, 26(4):645–657, 2014.

[20] Arthur M Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972.

[21] Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[22] Ulaş C Kozat and Leandros Tassiulas. Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues. *Ad Hoc Networks*, 2(1):23–44, 2004.

[23] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999.

[24] H Lim and C Kim. Flooding in wireless ad hoc networks. *Computer Communications*, 24:353–363, 02 2001.

[25] Hyojun Lim and Chongkwon Kim. Flooding in wireless ad hoc networks. *Computer Communications*, 24(3):353–363, 2001.

[26] Wei Lou and Jie Wu. On reducing broadcast redundancy in ad hoc wireless networks. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2003.

[27] Abilio Lucena, Nelson Maculan, and Luidi Simonetti. Reformulations and solution algorithms for the maximum leaf spanning tree problem. *Computational Management Science*, 7(3):289–311, 2010.

[28] Madhav V Marathe, Heinz Breu, Harry B Hunt III, Shankar S Ravi, and Daniel J Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995.

[29] John E Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of applied optimization*, 1:65–77, 2002.

[30] Jasaswi Prasad Mohanty, Chittaranjan Mandal, Chris Reade, and Ariyam Das. Construction of minimum connected dominating set in wireless sensor networks using pseudo dominating set. *Ad Hoc Networks*, 42(C):61–73, 2016.

[31] Raqeebir Rab, Shaheed Ahmed Dewan Sagar, Nazmus Sakib, Ahasanul Haque, Majedul Islam, and Ashikur Rahman. Improved self-pruning for broadcasting in ad hoc wireless networks. *Wireless Sensor Network*, 9(02):73, 2017.

[32] Ashikur Rahman, Md Endadul Hoque, Farzana Rahman, Sabuj Kumar Kundu, and Pawel Gburzynski. Enhanced partial dominant pruning (EPDP) based broadcasting in ad hoc wireless networks. *JNW*, 4(9):895–904, 2009.

[33] Lu Ruan, Hongwei Du, Xiaohua Jia, Weili Wu, Yingshu Li, and Ker-I Ko. A greedy approximation for minimum connected dominating sets. *Theoretical Computer Science*, 329(1-3):325–330, 2004.

[34] Jamil A Shaikh, Julio Solano, Ivan Stojmenovic, and Jie Wu. New metrics for dominating set based energy efficient activity scheduling in ad hoc networks. In *Local Computer Networks, 2003. LCN'03. Proceedings. 28th Annual IEEE International Conference on*, pages 726–735. IEEE, 2003.

[35] Luidi Simonetti, Alexandre Salles Da Cunha, and Abilio Lucena. The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm. In *International Conference on Network Optimization*, pages 162–169. Springer, 2011.

[36] Raghupathy Sivakumar, Prasun Sinha, and Vaduvur Bharghavan. Cedar: a core-extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in communications*, 17(8):1454–1465, 1999.

[37] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[38] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, 8(2):153–167, 2002.

[39] Jie Wu and Bing Wu. A transmission range reduction scheme for power-aware broadcasting in ad hoc networks using connected dominating sets. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 5, pages 2906–2909. IEEE, 2003.

[40] Jie Wu, Bing Wu, and Ivan Stojmenovic. Power-aware broadcasting and activity scheduling in ad hoc wireless networks using connected dominating sets. *Wireless Communications and Mobile Computing*, 3(4):425–438, 2003.