12-4-2006

An Enhanced Algorithm to Find Dominating Set Nodes in Ad Hoc Wireless Networks

Naresh Nanuvala

AN ENHANCED ALGORITHM TO FIND DOMINATING SET NODES IN AD HOC WIRELESS NETWORKS

by

Naresh Nanuvala

Under the Direction of Yi Pan

ABSTRACT

A wireless ad hoc network is a collection of wireless mobile nodes forming a temporary network without the aid of any established infrastructure or centralized administration. A connection is achieved between two nodes through a single hop transmission if they are directly connected or multi-hop transmission if they are not.

The wireless networks face challenges to form an optimal routing protocol. Some approaches are based on a dominating set, which has all the nodes either in the set or within its neighborhood. The proposed algorithm is an enhancement of the distributed algorithm proposed by Wu and Li. The simulation results from the new algorithm are compared to results from Wu and Li's algorithm. The simulation results show that the average dominating set of nodes decreased considerable after applying the new algorithm. The decrease in number of dominate set nodes is not very much noticeable in low density space.

INDEX WORDS: Wireless Mobile Ad-hoc networks, Connected Dominating Set, Routing Protocol, Position-based Routing.

AN ENHANCED ALGORITHM TO FIND DOMINATING SET NODES IN AD HOC WIRELESS NETWORKS

by

Naresh Nanuvala

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Science

Georgia State University

2006

AN ENHANCED ALGORITHM TO FIND DOMINATING SET NODES IN AD HOC WIRELESS NETWORKS

by

Naresh Nanuvala

Major Professor:	Yi Pan
Committee:	Anu Bourgeois
	Raj Sunderraman

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

Dec 2006

Acknowledgements

I would like to thank my advisor, Dr. Yi Pan for his support and invaluable guidance throughout my thesis work. I am also very grateful to the other members of my thesis committee, Dr. Anu Bourgeois and Dr. Raj Sunderraman for their advice and spending their valuable time in reviewing the material.

I would also like to thank Dr. Hui Liu for her guidance and advice and I also want to thank my friends and family for their encouragement without their support I could not have completed this.

Table of Contents

Acknowledgements ..... **iv**

List of Figures ..... **vii**

List of Tables ..... **ix**

List of Abbreviations ..... **x**

1 INTRODUCTION **1**

1.1 MOBILE AD HOC WIRELESS NETWORKING 2

1.2 CHALLENGES AND STATUS OF ROUTING PROTOCOLS 4

1.3 RESEARCH QUESTION 5

1.4 ORGANIZATION OF THIS THESIS 6

2 LITERATURE REVIEW **7**

2.1 TRADITIONAL ROUTING PROTOCOLS 7

2.2 POSITION-BASED ROUTING PROTOCOLS 8

2.3 DYNAMIC SOURCE ROUTING PROTOCOLS 9

2.4 SUBGRAPH-ROUTING PROTOCOLS 9

2.5 OTHER ROUTING PROTOCOLS 10

2.6 SUMMARY 10

3 AN ENHANCED ALGORITHM TO FIND DOMINATING SET IN AD-HOC WIRELESS NETWORKS. **11**

3.1 DOMINATING SET PROBLEM AND DEFINITION 11

3.2 WU AND LI'S ALGORITHM 12

3.3 NEW ALGORITHM 17

4. IMPLEMENTATION OF THE NEW ALGORITHM **19**

4.1 PROCEDURES TO IMPLEMENT NEW ALGORITHM 19

4.2 EXPLANATION OF NEW ALGORITHM 20

5. RESULTS 29

6. CONCLUSION 44

7. BIBLIOGRAPHY 45

Appendix – A 48

Appendix – B 62

List of Figures

Figure 1.1 A simple mobile wireless network ……....2

Figure 1.2 Simple unconnected wireless network..…….3

Figure 3.1 A Domain with 10 randomly located nodes…….....14

Figure 3.2 A Connected Graph..……....14

Figure 3.3 A Graph Marked by Basic Rule...…….....15

Figure 3.4 A Graph Marked by extensional rule 1……15

Figure 3.5 A Graph Marked by extensional rule 2……16

Figure 4.1 Generate randomly located nodes. Each number represents a node……22

Figure 4.2 Connect two nodes if their distance is less than or equal to transmission range…….23

Figure 4.3 Verify if the graph is connected……...24

Figure 4.4 Mark dominating set nodes using basic rule...…….25

Figure 4.5 Unmark some nodes to blue nodes by extensional rule 1……26

Figure 4.6 Unmark some nodes to green nodes by extensional rule 2...……...27

Figure 4.7 Unmark some red nodes using the new extensional rule…….28

Figure 5.1 Average number of dominate set nodes relative to varying density for 20 nodes……...31

Figure 5.2 Average number of dominate set nodes relative to varying density for 40 nodes……...32

Figure 5.3 Average number of dominate set nodes relative to varying density for 60 nodes……...33

Figure 5.4 Average number of dominate set nodes relative to varying density for 80 nodes……...34

Figure 5.5 Average number of dominate set nodes relative to varying density for 170 nodes……...35

Figure 5.6 Average number of dominate set nodes relative to varying density for 180 nodes……36

Figure 5.7 Average number of dominate set nodes relative to varying density for 190 nodes……37

Figure 5.8 Average number of dominate set nodes relative to varying density for 200 nodes……38

Figure 5.9 Average number of dominating set nodes relative to number of nodes for constant density d = 6……...39

Figure 5.10 Average number of dominating set nodes relative to number of nodes for constant density d = 12……40

Figure 5.11 Average number of dominating set nodes relative to number of nodes for constant density d = 18……41

Figure 5.12 Average number of dominating set nodes relative to number of nodes for constant density d = 24……42

List of Tables

Table 5.1 Average number of dominate set nodes relative to varying density with node number N = 20.....31

Table 5.2 Average number of dominate set nodes relative to varying density with node number N = 40…….32

Table 5.3 Average number of dominate set nodes relative to varying density with node number N = 60…….33

Table 5.4 Average number of dominate set nodes relative to varying density with node number N = 80…….34

Table 5.5 Average number of dominate set nodes relative to varying density with node number N = 170……...35

Table 5.6 Average number of dominate set nodes relative to varying density with node number N = 180……...36

Table 5.7 Number Average number of dominate set nodes relative to varying density with node number N = 190……...37

Table 5.8 Average number of dominate set nodes relative to varying density with node number N = 200……...38

Table 5.9 Average number of dominating set nodes for constant density d = 6……..39

Table 5.10 Average number of dominating set nodes for constant density d = 12……..40

Table 5.11 Average number of dominating set nodes for constant density d = 18……..41

Table 5.12 Average number of dominating set nodes for constant density d = 24……..42

List of Abbreviations

BFS – Breadth First Search

DS - Dominating Set

CDS – Connected Dominating Set

LAN – Local Area Network

MCDS – Minimum Connected Dominating Set

DSR – Dynamic Source Routing

MANET – Mobile Ad Hoc Network

WRP – Wireless Routing Protocol

WLAN – Wireless Local Area Network

1 INTRODUCTION

Wireless networks can provide mobile users with widespread communication capability and easy information access regardless of location. There are currently two variations of mobile wireless networks. The first kind is known as infrastructured networks, i.e., those networks with fixed and wired gateways, the bridges for these networks are known as stations. Typical applications of this kind of network are WLAN's and cellular networks. The second type of mobile wireless networks is the infrastructure less mobile network, known as self-organized network, and which is also referred to as Mobile Ad hoc Network (a term used in MANET [5]), Mobile Packet Radio Networking, Mobile Mesh Networking and Mobile, Multi-hop, Wireless Networking [5,29].

Self-organized networks consist of mobile radio nodes (hosts, routers or switches) forming a temporary network, without any aid of existing network infrastructure or centralized system administration. Network nodes, when out of the transmission range of each other, may communicate with intermediate nodes to forward their packets in a multi-hop mode. These networks are suitable in situations when an instant infrastructure is needed; typical applications include mobile computing in remote areas (e.g., sharing files in the field), tactical communications, law enforcement operations, and disaster recovery.

A connection is achieved either through a single-hop radio transmission if two nodes are located within wireless transmission range of each other, or through relays by intermediate nodes that are willing to forward packets for them. Mobile wireless networks have increased dramatically during the past few years. They can be quickly deployed in many applications.

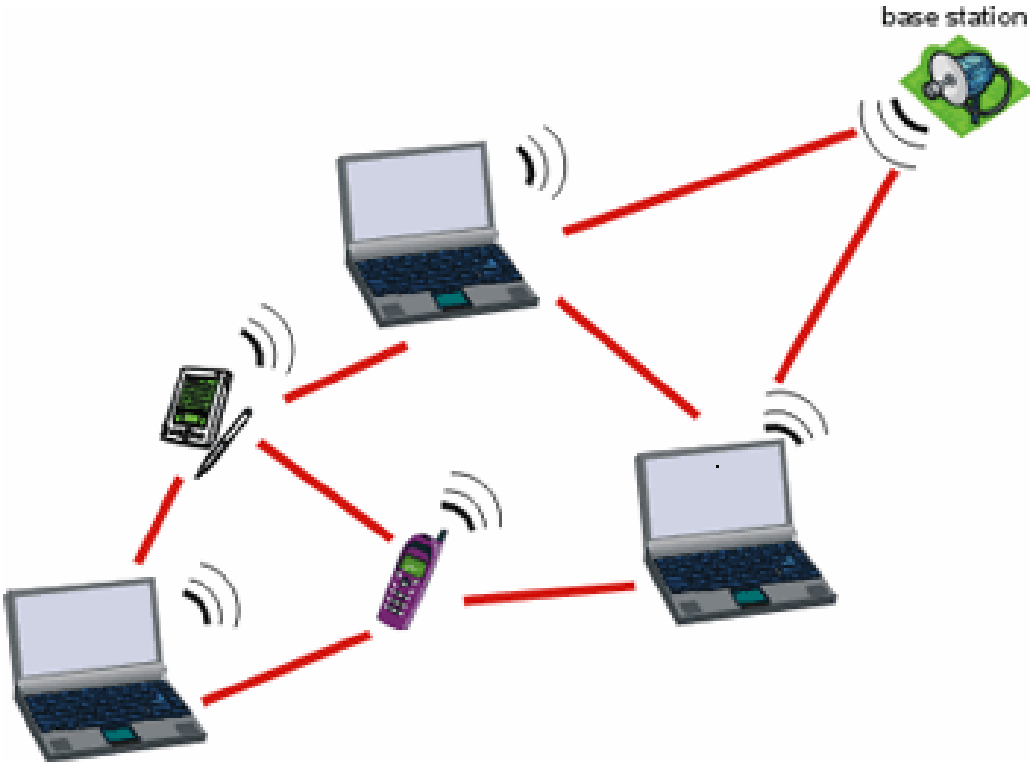

Figure 1.1: A simple mobile wireless network

Figure 1.1 shows a scenario where a wireless network is established between a single base station serving 5 connected wireless enabled devices, such as laptops, PDA's, cell phones.

1.1 MOBILE AD HOC WIRELESS NETWORKING

Ad hoc wireless network is an autonomous system consisting of mobile hosts (or routers) connected by wireless links [5]. It is a system of compatible wireless routers that set up a possibly short-lived network just for communication needs.

Following are some of the characteristics of ad hoc wireless networks

1) Nodes are mobile
2) Each node has limited power
3) Low bandwidth

An ad hoc wireless network is established without the aid of any infrastructure or centralized network [29].

A wireless ad hoc network can be represented as a simple graph G (V, E), where V represents a set of mobile nodes and E represents a set of edges. An edge (u, v) in E indicates that nodes u and v are neighbors, and that u is within v's range of transmission, while v is within u's range [34].

The following two assumptions are made:

1) The transmission range of all nodes is identical.
2) The graph is undirected. The edge between two nodes has no direction. If two nodes in the network are not directly connected, they need other nodes to forward packets between them.

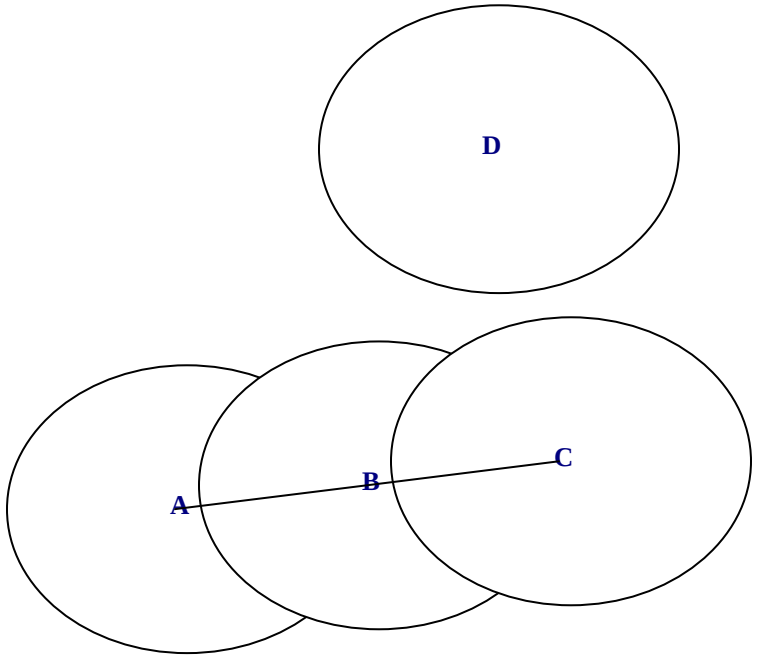

Figure 1.2: Simple Unconnected wireless networks

There are four nodes in the graph shown in Figure 1.2. Nodes A, B are within a certain transmission range, say 100 units. Nodes B and C are also within the same transmission range. But Node A and Node C are too far away that their distance is greater than the transmission range. Node D is too far away from all other nodes in the network hence there is no connection to node D from any other node in the network. If the wireless network needs to relay packets from Node A to Node C, Node B should be used as an intermediate node to forward packets between them. Other alternative paths could be used if there are other nodes between Node A and Node C. Most mobile nodes usually function as end nodes and connecting gateways simultaneously.

1.2 CHALLENGES AND STATUS OF ROUTING PROTOCOLS

Many unique characteristics of self-organized network have posed new challenges on routing protocol design: dynamic network topology, energy constraints, lack of network scalability and a centralized entity, and the different characteristics between wireless links and wired links such as limited bandwidth, unidirectional links, and poor security. Address migration, locality migration and other critical properties related to computing mobility are also key challenges.

Volatility of the network i.e.; host mobility causes network topological changes, multi-hop communication and limited resources (lower bandwidth, low battery power, limited CPU) are some properties that pose credible challenges to mobile networks. All these induce more failures in mobile networks. Several routing protocols have been proposed to address these problems.

Routing problem is to find a route for sending a packet from a source to a given destination. There are two main classes of routing protocols [16, 27].

1) Topology based
2) Position based

Topology based routing protocols are based on information about the links. Position based routing protocols use additional information about physical positions. Shortest path algorithm does not work well in MANETS because some nodes may become temporarily inactive or nodes might move.

Wireless networks require localized algorithms; traditional routing protocols that use link state or distance vector in wired networks are not suitable in ad hoc wireless networks [14]. Greedy routing, Face algorithm, combination of Greedy Face Greedy (GFG) algorithm are other position based algorithms. Dynamic source protocol (DSR) has also been proposed [17]. DSR allows the network to be completely self-organizing and self-configuring. The protocol includes two parts: Route Discovery and Route Maintenance.

Some researches proposed a new approach where a sub-graph of the ad hoc wireless network is selected and then the sub-graph is searched for routing. This reduces the running time. Dominating set based routing is one such kind of sub graph routing. Wu and Li proposed an efficient algorithm to calculate connected dominating set [34].

1.3 RESEARCH QUESTION

The research presented in this thesis implements Wu and Li's algorithm to calculate the connected dominating set nodes in ad hoc wireless networks and extends further by adding an extra extensional rule to decrease the size of dominating set nodes. The simulation results from the new algorithm are compared to results from Wu and Li's algorithm.

The research question for this thesis is:

"How does the new extensional rule decrease the size of the dominating set of nodes according to the topology of the wireless networks?"

1.4 ORGANIZATION OF THIS THESIS

This thesis is divided into six chapters: Introduction (Chapter1), Literature Review (Chapter 2), Dominating Set in Ad hoc Wireless networks (Chapter 3), New Extensional Rule Implementation (Chapter 4), Results and Explanation (Chapter 5) and Conclusion (Chapter 6)

Chapter 1 is an introduction for this thesis. It describes definition of mobile wireless network, ad hoc wireless network and its characteristics. The challenges ad hoc wireless network provide for finding a route. In addition to this a summary of status of routing protocols, the research question and overview of this thesis are outlined.

Chapter 2 gives the literature survey about the status of routing protocols. Mainly there are two kinds of protocols, topology based routing protocol and position based routing protocol. The characteristics and disadvantage of some protocols are described in this chapter.

Chapter 3 describes the definition of Dominating Set and Wu and Li's algorithm is introduced. Also the new extensional rule is described.

Chapter 4 describes implementation process of Wu's algorithm and the new extensional rule.

Chapter 5 presents simulation results and explanation. Tables and figures are provided based on the two parameters used to run the algorithms (number of nodes, transmission range).

In Chapter 6 relevant conclusions are drawn. The contributions of this work are briefly discussed followed by future areas of research that might be investigated in order to build upon the work presented in this thesis.

2 LITERATURE REVIEW

Various routing protocols have been proposed in recent years to address the routing problem in ad hoc wireless networks. Mainly they are classified into two routing classes.

One type is topology based routing protocol, based on the information about the links. The other is position based routing protocol that uses additional information about physical location.

2.1 TRADITIONAL ROUTING PROTOCOLS

Distributed algorithms for Minimum Connected Dominating Set (MCDS) in mobile ad hoc networks were first developed by Das et al. [3,10]. These algorithms provide distributed implementations of the two centralized algorithms given by Guha and Khuller [13]. The shortest path algorithm does not work very well in MANETS because some nodes maybe temporarily inactive or some might move. Wireless networks require localized algorithms in which nodes make routing decisions based on the neighboring nodes information.

Other traditional routing protocols that use link state or distance vector in wired networks are not suitable for ad hoc wireless networks [14,12,17,18]. Lower bandwidth in wireless networks makes information collection expensive. The power limitation leads users to get disconnected from mobile host frequently. Routing information needs to be localized to adapt quickly to topology changes caused by node movements. Link state routing algorithms are closer to the centralized shortest path algorithm. Each node maintains a view of the network topology with a cost for each link. Each node periodically floats the link cost between it and all other nodes. If a node gets the information, it updates its view of topology and applies shortest path algorithm to select next hop. Although link state routing generally requires each node to know the entire topology, there are some link state algorithms where

each node only maintains partial information of the network.

The distance vector routing algorithms use the distributed version of Bellman-Ford algorithm (DBF), each node maintains for each destination a set of distances. A node selects next hop node if that node has the minimum distance for a destination. Compared to link state algorithm, it requires less storage space and less network bandwidth overhead. But this algorithm might be effective only when network topological changes are rare [12,17,18].

The details of these two routing protocols and their problems are discussed in the papers [16,18].

2.2 POSITION-BASED ROUTING PROTOCOLS

In position-based routing protocols, forwarding decision of a node depends on the destination node's position and it's one hop neighbors position. One method, called Greedy routing algorithm, is a position based protocol. Each node forwards packet to its neighbor that is closet to destination based on the location information. The greedy algorithm may fail to find a path if the node does not have a neighbor that is closer to destination than the node itself. When that problem arises, the message needs to be forwarded to the node with the least backward distance; this introduces another problem of looping packets. Greedy algorithm's route is very close to the shortest path algorithm, but it has high failure rate because of loop or low degree graphs.

To solve the local maximum problem, another algorithm called FACE algorithm is provided. It guarantees the package delivery in connected graph, but it has longer route. Face algorithm is to forward the packet on faces of the planar sub-graph, which are progressively closer to the destination. It also increases the hop count. GFG algorithm is a combination of these two algorithms. First Greedy algorithm is run, when it fails, face algorithm is run, and

then greedy algorithm is run. The GFG algorithm combines the two algorithms advantages: it guarantees the package delivery and a relative short route [12].

2.3 DYNAMIC SOURCE ROUTING PROTOCOLS

Other researches propose a dynamic source routing protocol [17]. DSR is a simple and efficient routing protocol, specially in multi-hop wireless ad hoc networks. The network is allowed by DSR to be completely self-organizing and self-configuring. The protocol has two parts: Route Discovery and Route Maintenance. They work together to allow nodes to discover and maintain source routes to arbitrary destinations in the ad hoc wireless network. The algorithm does not need to construct any routing tables. All protocols operate on-demand, allowing the routing packet overhead of DSR to scale automatically to only that needed to be changed in the routes currently in use. This protocol adapts quickly to routing changes when node movement is frequent.

2.4 SUBGRAPH-ROUTING PROTOCOLS

Some researches try to find a sub-graph of ad hoc wireless network and search the routing in the sub-graph and reduce the running time. These approaches are known as dominating–set based routing protocols. Another type of routing protocols known as Cluster based algorithm, divides a graph into several overlapping clusters [19]. Each cluster is a clique, which is a complete sub-graph. The routing protocol is completed in two phases: cluster formation and cluster maintenance. The routing process centralizes the whole network into a small connected sub-network so that if the network topological changes do not affect this centralized part of the network, there is no need to recalculate routing tables in the sub-network [21, 22]. Dominating-set-based routing is also one kind of sub-graph routing [1, 2, 21, 33].

If every vertex not in the subset is adjacent to at least one vertex in this subset, the subset is called dominating set. The dominating-set-based routing is based on the theory of dominating theory. This approach reduces the routing and search process to a reduced sub-graph. The efficiency of the approach depends on the process of finding a connected dominating set and the size of the dominating set nodes.

2.5 OTHER ROUTING PROTOCOLS

Some protocols aim to consider the power problem in ad hoc wireless networks [12,18], because nodes are power-constrained in ad hoc wireless networks. One tries to select different nodes as route to balance the power assumption in the nodes, other designs an energy efficient routing protocols that dynamically makes local routing decisions so that a near optimal power efficient end to end route is formed for forwarding data packets. Because in ad hoc wireless networks, geographical routing protocols take advantage of location information, it heavily depends on the existence of scalable location management services. Therefore some researches studied the location management scheme in mobile ad hoc networks [18]. Grid's location service (GLS) is a new distributed location service, which tracks mobile node location.

2.6 SUMMARY

All of these studies are based on different assumption and try to achieve different objectives. Quite a few algorithms are based on the dominating set based principle. This research focuses on the dominating-set-based routing protocol, particularly Wu and Li's algorithm [34].

3 AN ENHANCED ALGORITHM TO FIND DOMINATING SET IN AD-HOC WIRELESS NETWORKS.

3.1 DOMINATING SET PROBLEM AND DEFINITION

A Dominating Set (DS) is a subset of nodes such that each node is either in DS or has a neighbor in DS. A Connected Dominating Set (CDS) is a connected DS, that is, there is a path between any two nodes in CDS that does not use nodes that are not in CDS. It is favorable to have few nodes in the CDS or DS. This is known as the Minimum Connected Dominating Set (MCDS) problem. Given an arbitrary undirected graph finding a MCDS or CDS is a NP-hard problem. Various algorithms have been proposed to address this problem. One such approach is dominating-set-based routing theory.

Assume a wireless ad hoc network is deployed in a two-dimensional space where each node has equal maximum transmission range. Thus the topology of an ad hoc network can be described as a unit-disk graph (UDG). A graph is a unit graph if and only if its vertices can be put in one to one correspondence with equi-sized circles in a plane in such a way that two vertices are joined by an edge if and only if the corresponding circles intersect [5].

A wireless ad hoc network can be represented as a simple graph G (V, E), where V represents a set of mobile nodes and E represents a set of edges. An edge (u, v) in E indicates that nodes u and v are neighbors, and that u is within v's range of transmission, while v is within u's range.

A dominating set (DS) is a subset of vertices of a graph G where every vertex that is not in the subset is adjacent to at least one vertex in the dominating set (DS) subset. A connected dominating set (CDS) is a dominating set that induces a connected sub-graph.

This approach reduces the routing and searching process to a reduced sub-graph, therefore it simplifies the routing and search process. Several algorithms have been proposed based on the dominating set theory [2, 12, 22]. One such algorithm is Das's algorithm [3, 10]. In Das' algorithm, a CDS is found by growing a set U starting from a vertex with the maximum node degree. It then iteratively adds to U a node that is adjacent to the maximum number of nodes not yet in U until U forms a dominating set. Finally, it assigns each edge with a weight equal to the number of neighbors not in U, and then finds a minimum spanning tree T in the resulting weighted graph. All the non-leaf nodes form a CDS. There are several advantages to this approach but the main drawback of this algorithm is that the process of constructing a spanning tree is almost sequential, that is, it needs a non-constant number of rounds to determine a CDS. Further more, the algorithm suffers from high implementation complexity and message complexity.

3.2 WU AND LI'S ALGORITHM

Wu and Li [34] proposed a simple and efficient distributed algorithm that can quickly find a DS in a mobile ad hoc network. In an ad hoc wireless network represented by a graph G= (V, E) all vertices are unmarked initially. m (v) is a marker for vertex v.

Vertex v is marked by setting m (v) = T (marked) and unmarked by setting m (v) = F (unmarked). The open neighbor of vertex v is represented by N (v) = {u| {v, u} <E}.

The basic marking rule is:

1) Every node v exchanges its open neighbor N (v) with its neighbors.
2) If node v finds any two of its neighbors x, y, that are not directly connected, the node is marked to be a dominating set node (gateway node) m(v) = T.

Using the basic marking rule, there are too many dominating set nodes. Wu and Li proposed two extensional rules to eliminate the number of dominating nodes.

Extensional rules are defined as:

1) Any two nodes u, v $\in$ dominating sets, If N[v] $\subseteq$ N[u] and id (v) < id (u), change the m (v) to F.
2) Any three nodes u, v, w $\in$ dominating sets, u and w are two marked neighbors of v.

If N (v) $\subseteq$ N (u) $\cup$ N (w) and id (v) < id (u) and id (v) < id (w), change the m (v) to F.

Where N[v] $\cup$ {v} = N[v], it is the closed neighbor set of v. Another condition is that assign a distinct id, id (v) to each vertex in the dominating set.

The main idea of the extensional rules is that if a dominating set node A can be covered by another dominating set node(s) (B, C …) and A's id is the smaller, it can be unmarked to be a non dominating set node. "Cover" means the N[v] $\subseteq$ N[u] or N (v) $\subseteq$ N (u) $\cup$ N (w) etc. By applying the extensional rules, some nodes can be unmarked and the size of the dominating set is reduced. The number of dominating set nodes is largely reduced and is proved by the simulation tests in this thesis.

The above algorithm proposed by Wu and Li is distributed and constant number of rounds is needed for the marking process. The dominating set includes all intermediate nodes of any shortest path. The efficiency of the approach depends on the size of the dominating set nodes. Wu and Li also proved that the dominating set is connected and closed to minimum. Figure 3.1 is a domain with 10 nodes randomly located. Figure 3.2 shows the results of Wu's algorithm using basic rule and extensional rules 1 and 2.

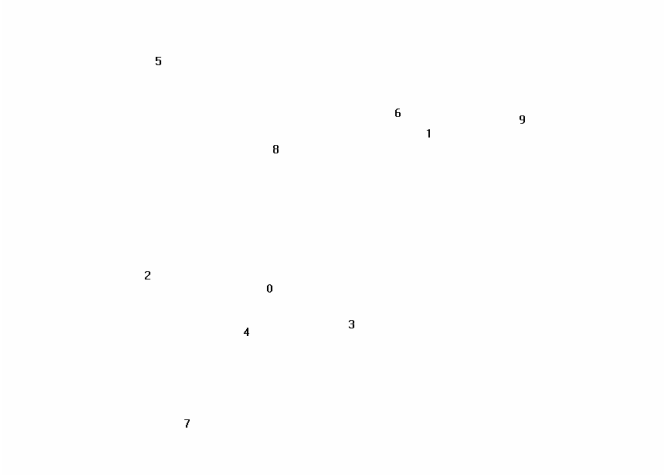

Figure 3.1: A Domain with 10 randomly located nodes

Figure 3.1 shows 10 nodes randomly distributed in a 2D space.

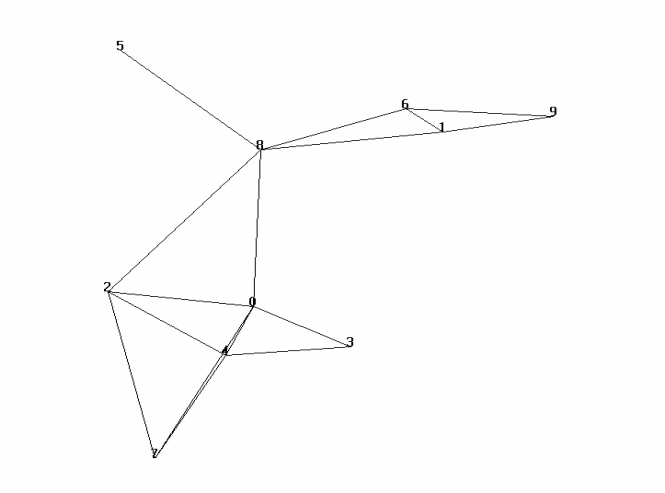

Figure 3.2: A Connected Graph

Figure 3.2 shows connections between nodes. A connection is established if the distance between node A and node B is less than the transmission range. For example, the distance between node 1 and node 6 is less than the range and a connection is established.

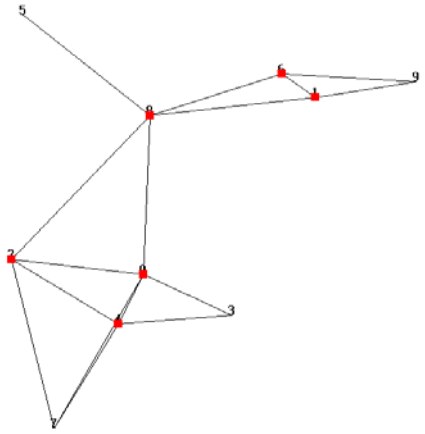

Figure 3.3: A Graph Marked by Basic Rule

Figure 3.3 shows the graph after applying basic rule. A node is marked red if that node has any two nodes that are not directly connected. For example node 4 has 0, 2, 3, 7 as neighbors. Since there is no direct connection between node 3 and node 7, node 4 can be marked red. The same rule is applied for all other nodes in the graph.

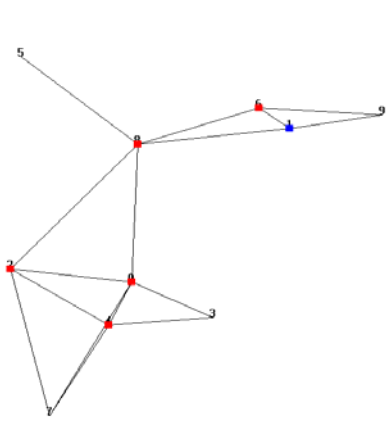

Figure 3.4: A Graph Marked by extensional rule 1

Figure 3.4 shows the graph after applying extensional rule 1. A node is unmarked blue if extensional rule 1 applies for that node. For example N[1] = {1,6,8,9}, N[6] = {1,6,8,9}, N[1] $\subseteq$ N[6] and id(1) < id(6), therefore 1 can be unmarked by extensional rule 1. In above example there are no other nodes where this rule satisfies.

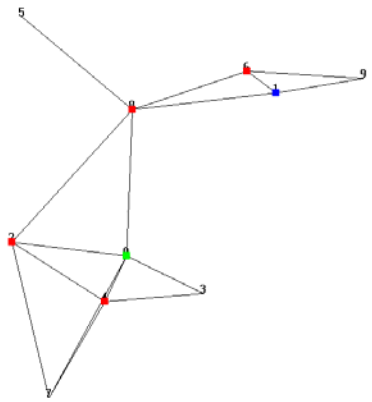

Figure 3.5 A Graph Marked by extensional rule 2

There are 10 nodes randomly located in 2-D space. Nodes are connected if there are within the transmission range. Nodes are marked by red if the node has any two-neighbor nodes that are not connected directly. For example, node 0 has neighbor nodes 2, 3, 4, 7, 8 and has some nodes for example nodes 3 and 7 are not connected; therefore node 0 is marked by basic rule. In the figure nodes 3, 5, 7, 9 are not marked because all of their corresponding neighbors are connected. For example, for node 9, 1 and 6 are neighbors. There are lines between 9 and 1, 9 and 6, 1 and 6. Applying extensional rule 1, node 1 is marked.

For example N [1] = {1, 6, 8, 9}

N [6] = {1, 6, 8, 9} and

N [1] $\subseteq$ N [6] and id (1) < id (6). Therefore node 1 can be unmarked (marked blue)

Applying extensional rule 2, node 0 can unmarked (marked green). Node 0 can be unmarked by extensional rule because it can be covered by other nodes (2, 4, 8). This is shown in Figure 3.5. Nodes that are unmarked by extensional rule 1 can also be unmarked by extensional rule 2. There is some overlap between extensional rule 1 and rule 2.

3.3 NEW ALGORITHM

In this section we introduced a new rule to extend Wu and Li's algorithm. This section describes the new rule and a sample result is also shown

Rule 3:

Node u is covered by two connected neighbors v and w if and only if

N (u) $\subseteq$ N (v) $\cup$ N (w) and one of the following conditions is satisfied:

a) key(v) < key(u) < key(w) and u has no neighbor z such that key(v) < key(z) and N(v) $\subseteq$ N(u) $\cup$ N(z).

b) key(v) < key(u) and key(w) < key(u) and

u has no neighbor z such that key(v) < key(z) and N(v) $\subseteq$ N(u) $\cup$ N(z), and

u has no neighbor z such that key(w) < key(z) and N(w) $\subseteq$ N(u) $\cup$ N(z).

In high density networks new extensional rule should unmark some dominating set nodes and decrease the size of DS. Based on the above rule, Wu and Li's algorithm can be enhanced by the following steps:

1) Finding the dominating set nodes using basic rule (Basic rule)
2) Using extensional rules to eliminate the number of dominating set nodes. (Wu and Li's extensional rule 1, extensional rule 2)

3) Add the new rule and implement the new algorithm to find dominating set nodes.

4. IMPLEMENTATION OF THE NEW ALGORITHM

In this chapter we describe how the new Algorithm was implemented and how the simulation was carried out. It is important that we test the performance of the algorithm by simulating it over a wide range of simulation parameters.

We start off this chapter by describing how the code is implemented and how the code operates. In the next chapter we present some results and analyze the results so obtained. Finally, we present the conclusions drawn from our simulation study.

4.1 PROCEDURES TO IMPLEMENT NEW ALGORITHM

The new algorithm was implemented in Windows 2000 or XP operating systems by using Visual C++ 6.0. Microsoft Foundation Classes (MFC) was used to implement the interface.

Two assumptions are postulated:

1) Each mobile host has some transmission radius.
2) Graph is undirected.

There are 6 steps in the new algorithm.

1) Generate randomly distributed nodes.
2) Connect two nodes if their distance is less than or equal to the transmission range.
3) Use the depth-first search algorithm (DFS) to check if the graph is connected or not. If the graph is connected, do step 4 else go back to step 1 again.
4) Use basic rule to mark dominating set nodes.
5) Use Wu and Li's extensional rule 1 and rule 2 to unmark some dominating set nodes.
6) Use new extensional rule to unmark some dominating set nodes.

Compare the results from the new algorithm with basic rule and Wu and Li's extensional rule 1 and extensional rule 2.

4.2 EXPLANATION OF NEW ALGORITHM

We improved the existing algorithm proposed by Wu and Li. Wu and Li [34] proposed a simple and efficient distributed algorithm that can quickly find a DS in a mobile ad-hoc network.

The basic rule of the algorithm proposed by Wu and Li is:

- A node is marked as dominating node it has two unconnected neighbors.

Since this returns a large set of dominate set two extensional rules were proposed:

- In the first extension rule if a dominating set node can be covered by another dominating node with higher id, then it can be removed from dominating set of nodes.
- In the second extension rule if a dominating set node can be covered by two dominating nodes with higher id's, then it can be removed from dominating set of nodes.

We implement the basic rule and the two extensional rules first and extend it further by introducing these two rules:

- If a dominating set node can be covered by two dominating nodes, one with higher ID and one with lower ID, then it can be removed from dominating set of nodes.
- If a dominating set node can be covered by two dominating nodes with lower id's, then it can be removed from dominating set of nodes.

We model a mobile ad-hoc network as a set of mobile nodes deployed in a predetermined rectangular area of dimension 600 × 600 square units. Each node has a unique ID. In our model we assume that mobile nodes do not move out of the deployed area.

The following shows the steps in our simulation:

1) Generate n pairs of random numbers, representing nodes in wireless network.
2) Connect two nodes if the distance between the two nodes is less than the transmission range (factor of density).
3) Verify if the graph is connected. If the graph is not connected go to step 1. Use the BFS algorithm to check the connectivity of the graph.
4) Using Wu and Li's basic rule mark dominating set nodes.
5) Apply Wu's extensional rules 1 and 2 to unmark some dominating set nodes.
6) Apply the new rule to unmark some dominating set nodes.
7) Compare the results using three variables, number of nodes, density, number of running circles.

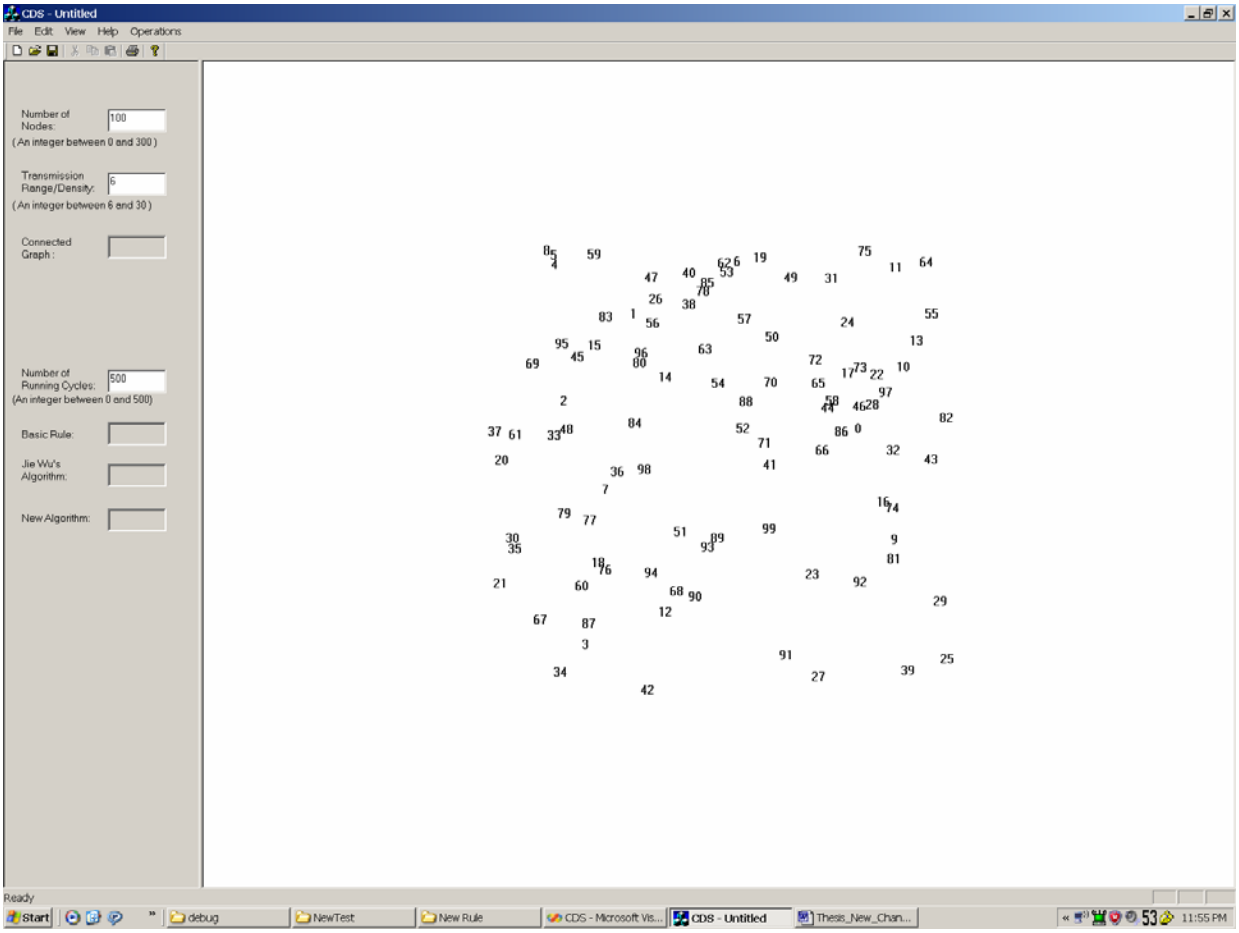

Figure 4.1: Generate randomly located nodes.

Figure 4.1 shows 100 nodes randomly located in 2D space. Each number represents a node. The density of the network d = 6. The following figures show sample results after applying basic marking rule, extensional rule 1 and extensional rule 2.

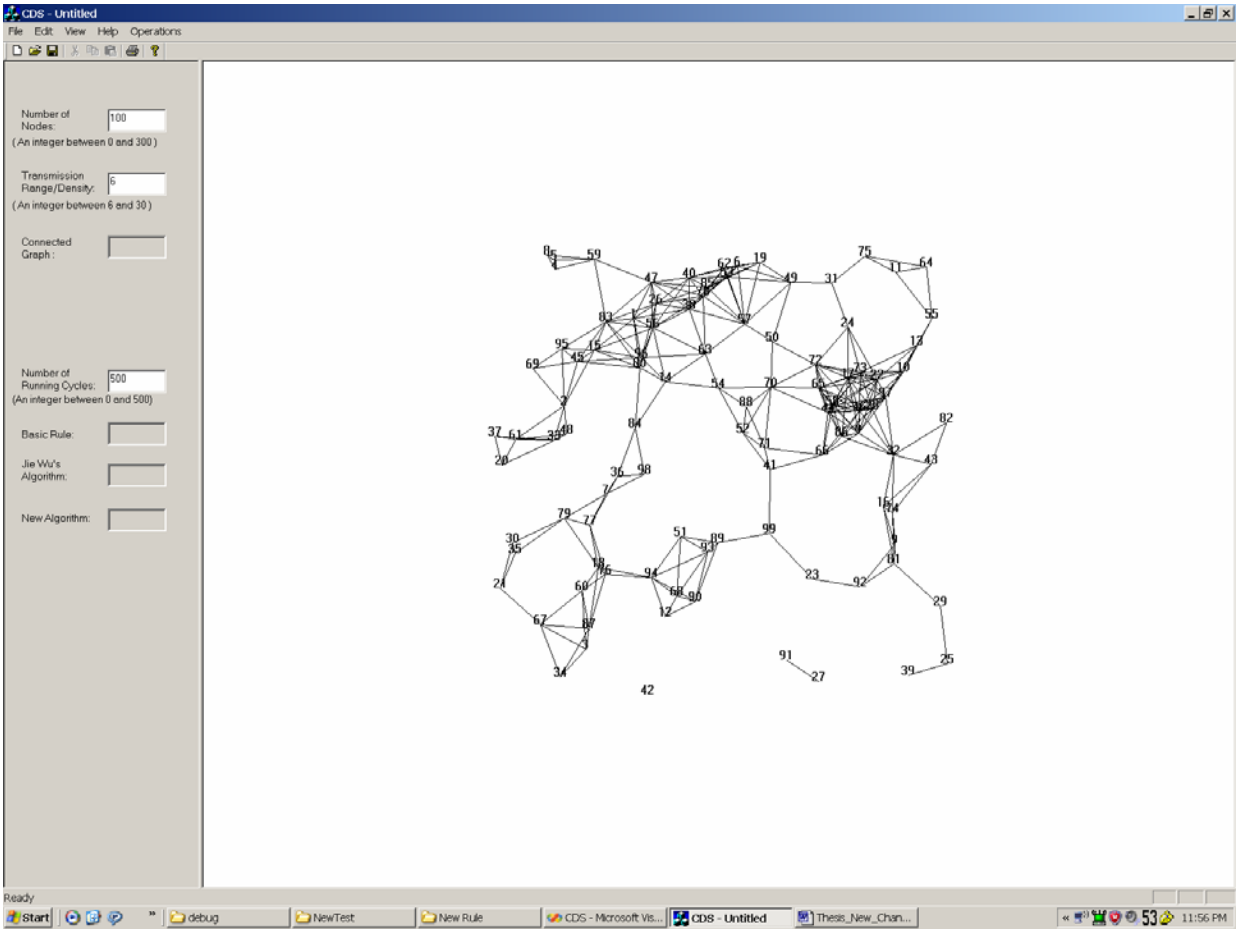

Figure 4.2: Connect nodes if their distance is less than transmission range

The above figure shows connections between nodes if the distance between them is less than the transmission range. For example, node 42 is not connected to the rest of the network. So the above figure is an unconnected graph. Unconnected graph can be discarded and a new graph has to be generated.

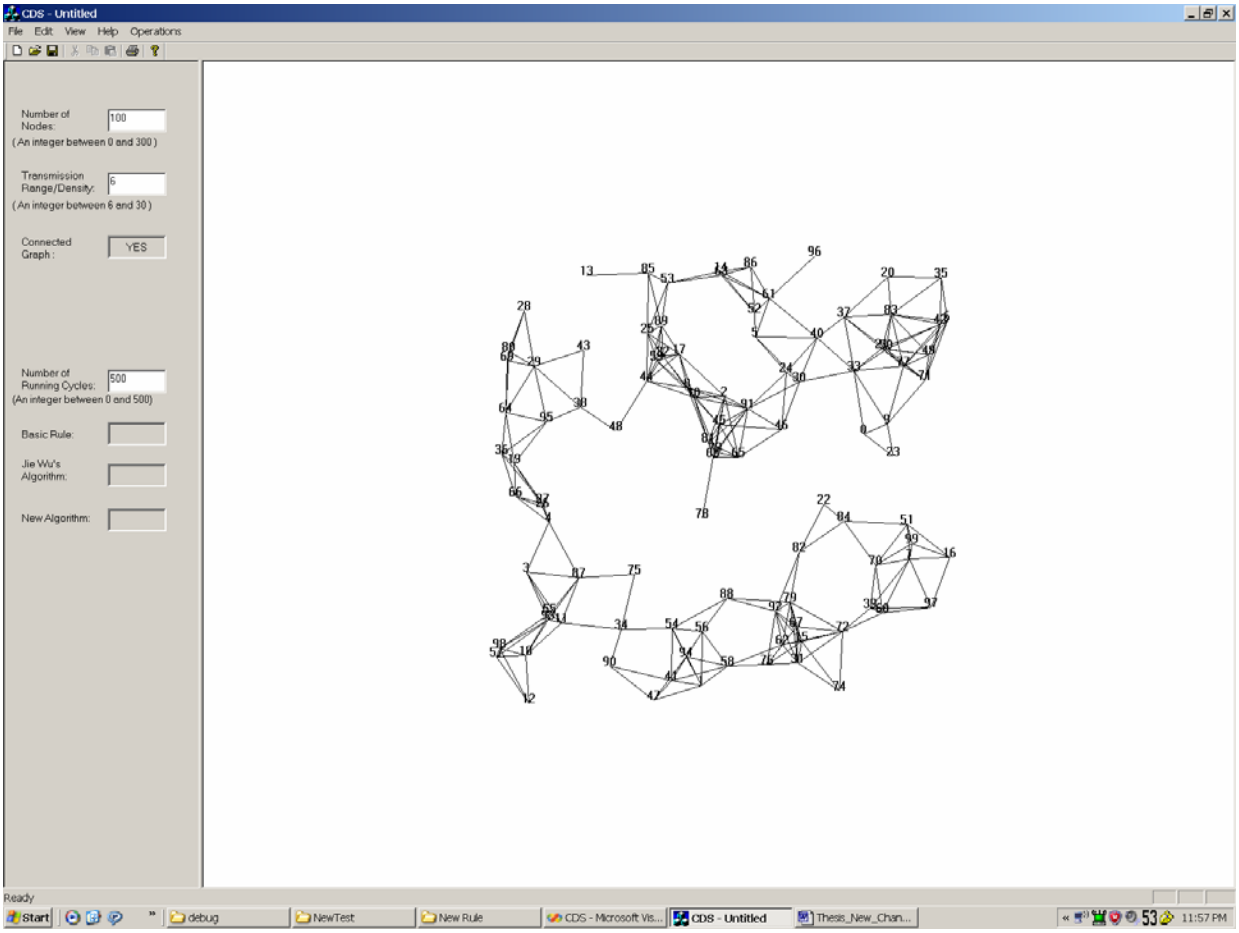

Figure 4.3: Verify if the graph is connected.

The above figure shows connections between nodes if the distance between them is less than the transmission range. The above figure is a connected graph. Basic marking rule is applied only after a connected graph is generated.

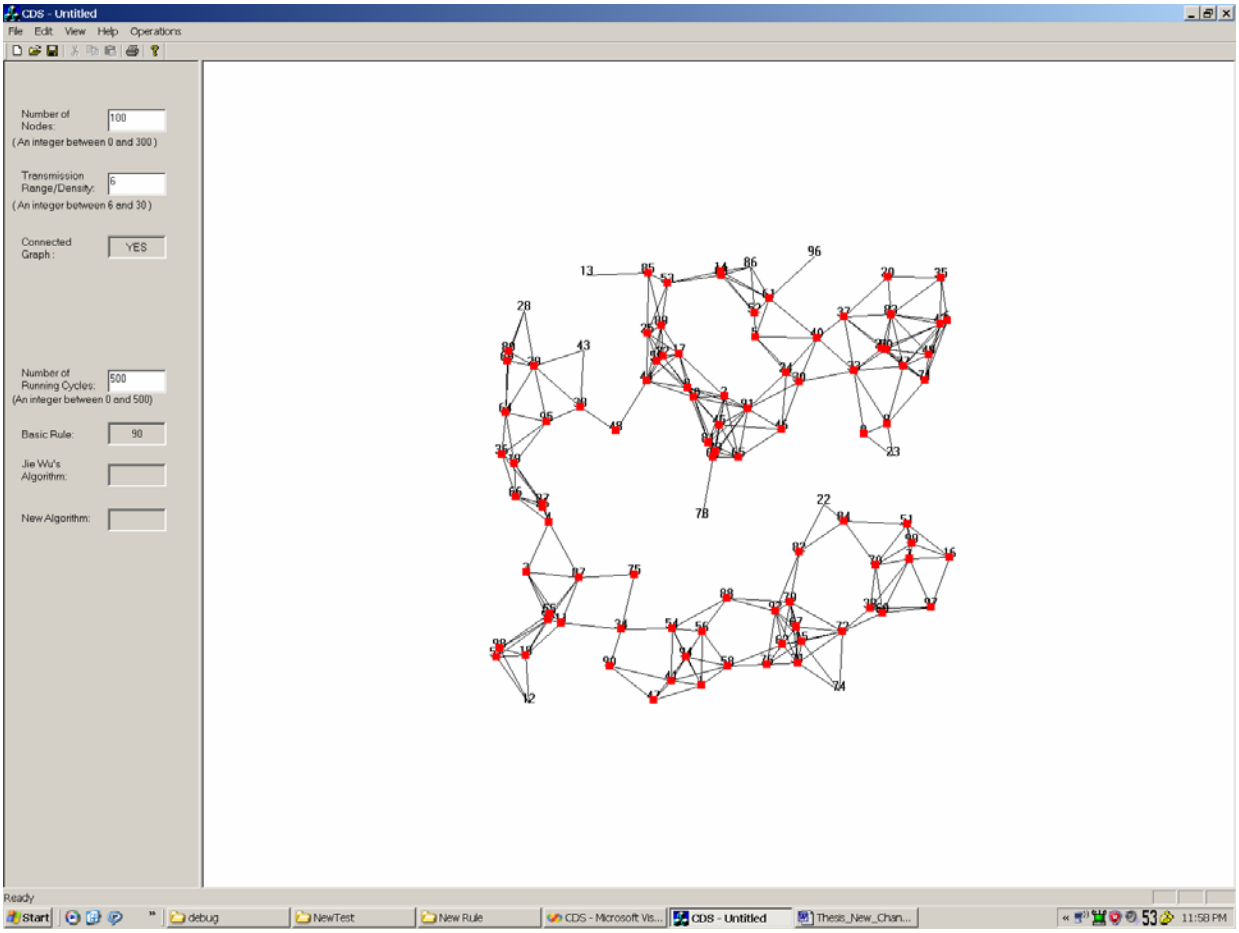

Figure 4.4: Mark dominating set nodes using basic rule

Initially all the nodes are unmarked. Use the basic marking rule to mark dominating nodes in red. The above figure shows the graph after the basic rule is applied. The number of dominate nodes after applying basic rule is high; it is almost equal to the total number of nodes in the network.

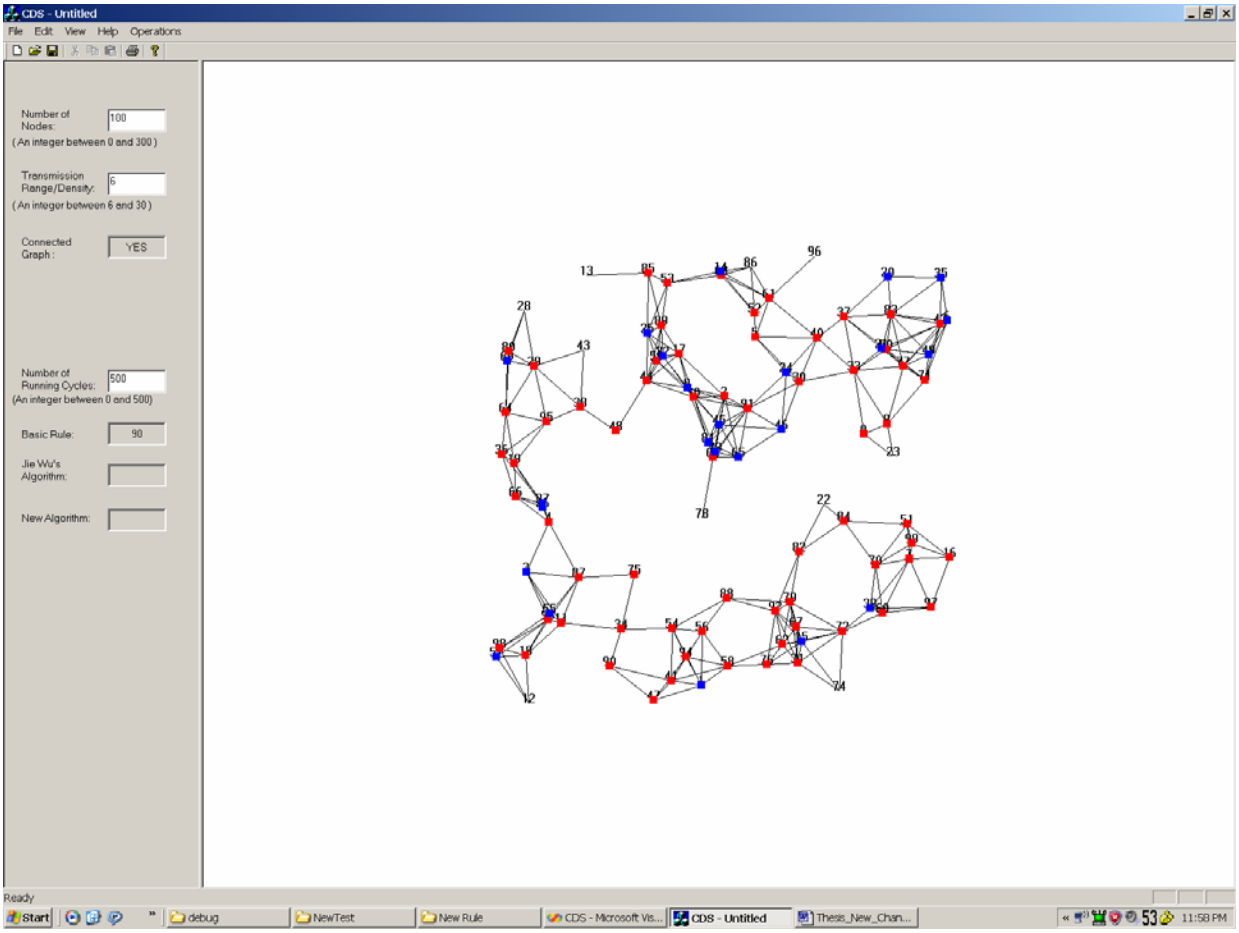

Figure 4.5: Unmark some nodes to blue nodes by extensional rule 1

The above figure shows the graph after Wu and Li's extensional rule 1 is applied. Unmark the nodes by turning them to blue. There is a considerable decrease in the number of dominate set of nodes.

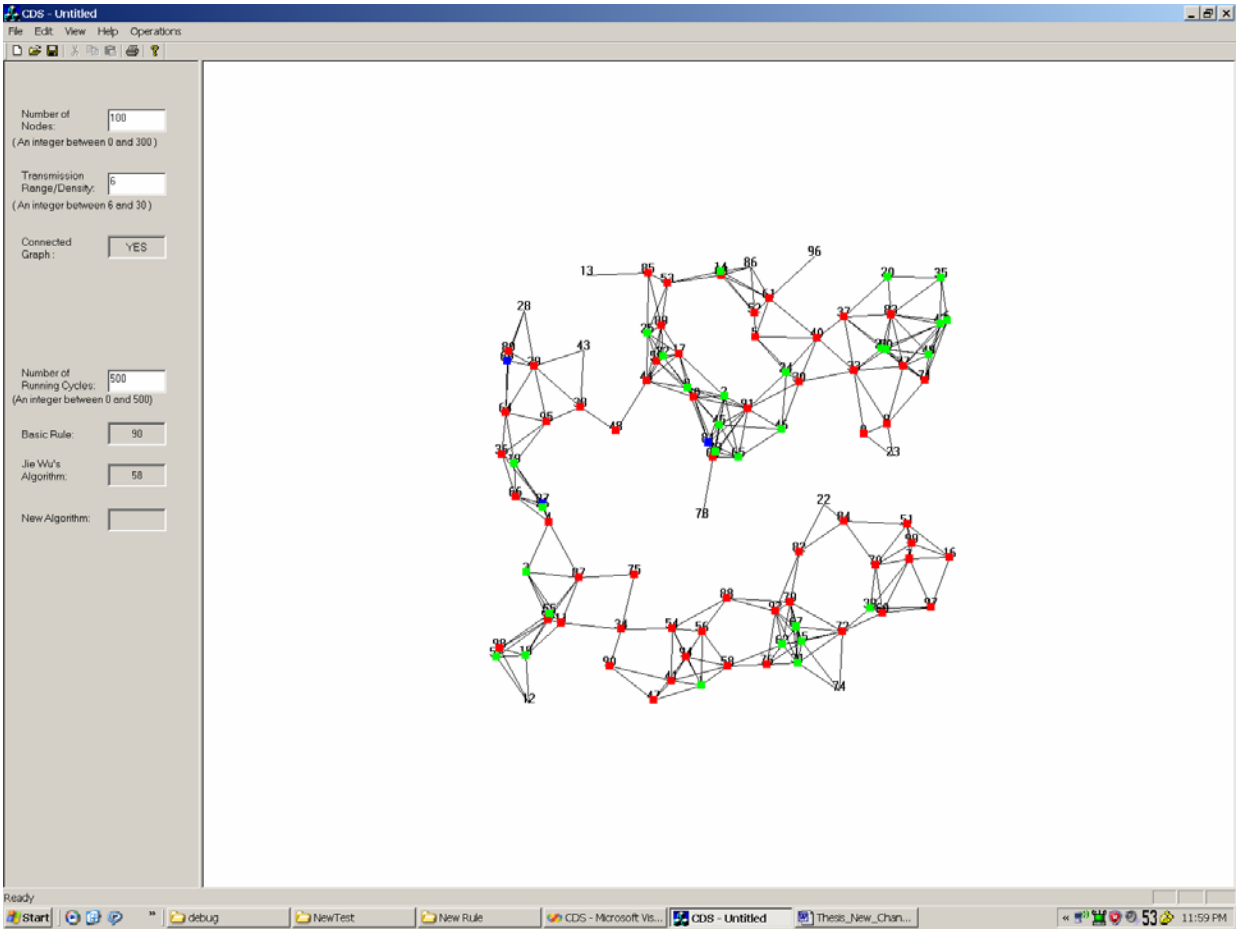

Figure 4.6: Unmark some nodes to green nodes by extensional rule 2

The above figure shows the graph after Wu and Li's extensional rule 2 is applied. Unmark the dominating nodes by turning them to green. There is a considerable decrease in the number of dominating set of nodes after Wu and Li's two extensional rules are applied. There is certain overlap between nodes unmarked by rule 1 and rule 2.

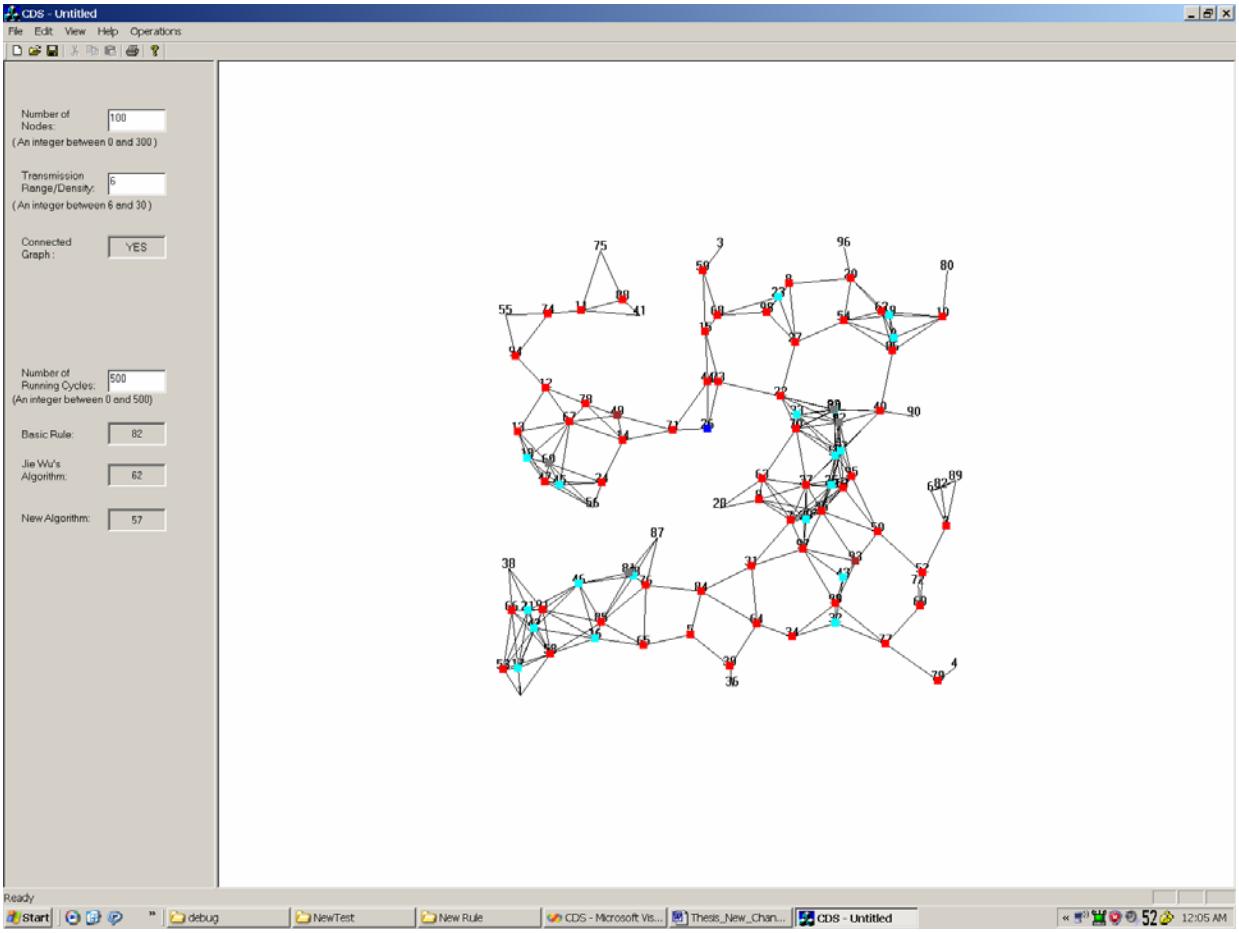

Figure 4.7: Unmark some red nodes using the new extensional rule

The above figure shows the graph after the new extensional rule is applied. Unmark the nodes where the new algorithm applies. The nodes in light blue, maroon, and grey are nodes unmarked after the new rule is applied. There is further decrease in the number of dominate set of nodes after the new rule is applied. The results are shown on the left panel. In this particular example, Basic rule = 82, Wu's rules = 62, New Algorithm = 57. The next chapter shows the simulation results.

5. RESULTS

In this section we conducted the simulation study which computes the average size of the CDS derived from our algorithm and compared with results from existing Wu and Li's algorithm. We have simulated three algorithms: Wu and Li's basic rule, Wu and Li's extensional rules and our new algorithm. The results were not compared with results from other enhancements of the algorithm like Ni's [26] here because Ni's simulation uses transmission radius as a parameter whereas we used density as a parameter for our simulation.

In our simulation environments, random graphs are generated in 600 × 600 square units of a 2-D simulation area, by randomly inducing a certain number of mobile nodes. We assume that each mobile node has the same transmission range *r*, thus the generated graph is undirected. If the distance between any two nodes is less than radius *r*, then there is a connection link between the two nodes. If generated graph is disconnected, simply discard the graph. Otherwise continue the simulation.

Note that, for a constant *r*, the network density, in terms of the average vertex degree d, will increase rapidly as the network size (*n*) increases. Simulation is carried out by varying average degree *d* of the network (i.e. the average number of neighbors of a node in the network), such that the impact of network size can be observed independent of density. The transmission range can be set as a function of *d*, number of nodes *n*, and the network area using relation $r^2 = (d * 600 * 600) / \pi * (n\text{-}1)$. In order to observe the impact of density, each simulation is repeated on various average vertex degrees (d = 6, 12, 18, 24, 30). Since the topology of ad hoc networks change very dynamically, our simulation takes snapshots on dynamic ad hoc networks. For each

average vertex degree *d*, the number of nodes *n* is varied from 20 to 200. For each *n*, the number of running times is 500 times.

Table 5.1: Number of dominating set nodes relative to varying density for 20 nodes

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
20	6	15	11	10
20	12	17	9	8
20	18	18	7	6
20	24	18	6	5
20	30	19	5	4

Table 5.1 shows the average number of dominate set of nodes for a network size of 20 with varying densities. The size of dominate set increases as density increases for Basic rule whereas the size of the dominate set of nodes decreases for Wu and Li's extensional rules and for the new rule as expected (Refer to Figure 5.1).

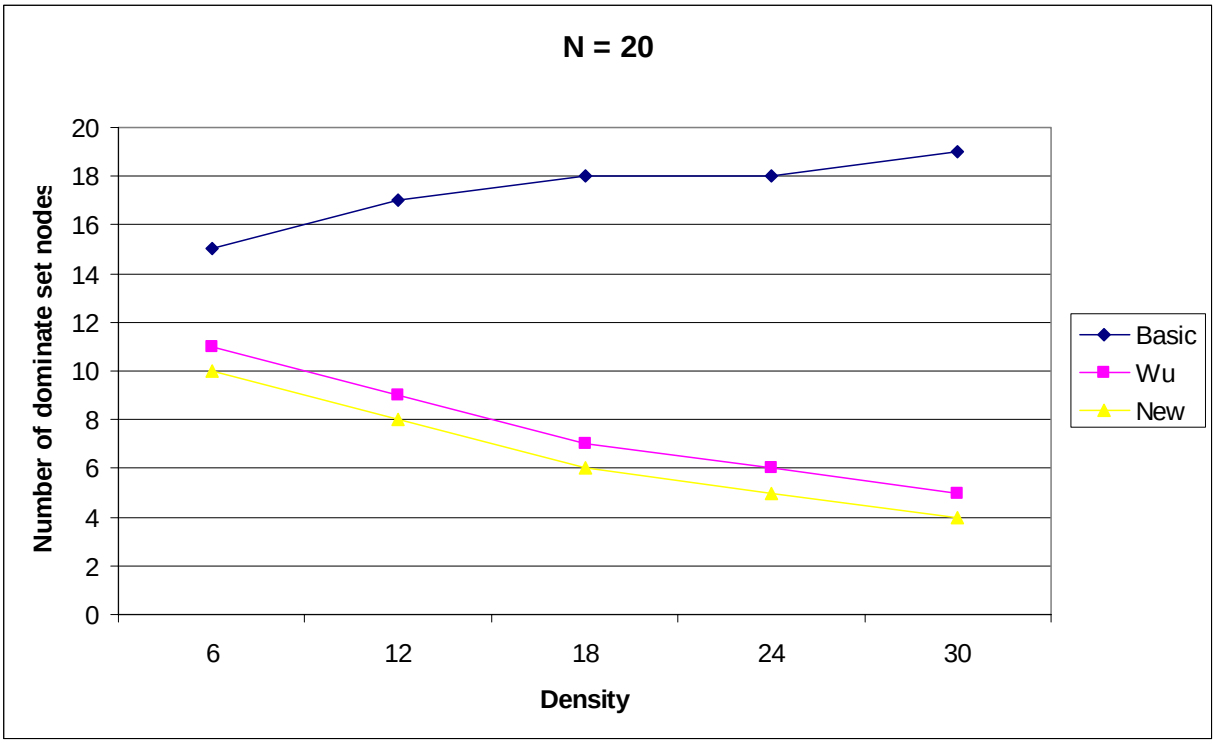

Figure 5.1: Average number of dominate set nodes relative to varying density for 20 nodes

Table 5.2: Number of dominating set nodes relative to varying density for 40 nodes

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
40	6	32	23	21
40	12	36	21	17
40	18	37	17	14
40	24	38	15	12
40	30	39	13	11

Table 5.2 shows the average number of dominate set of nodes for a network size of 40 with varying densities. The size of dominate set increases as density increases for Basic rule whereas the size of the dominate set of nodes decreases for Wu and Li's extensional rules and for the new rule as expected (Refer to Figure 5.2).

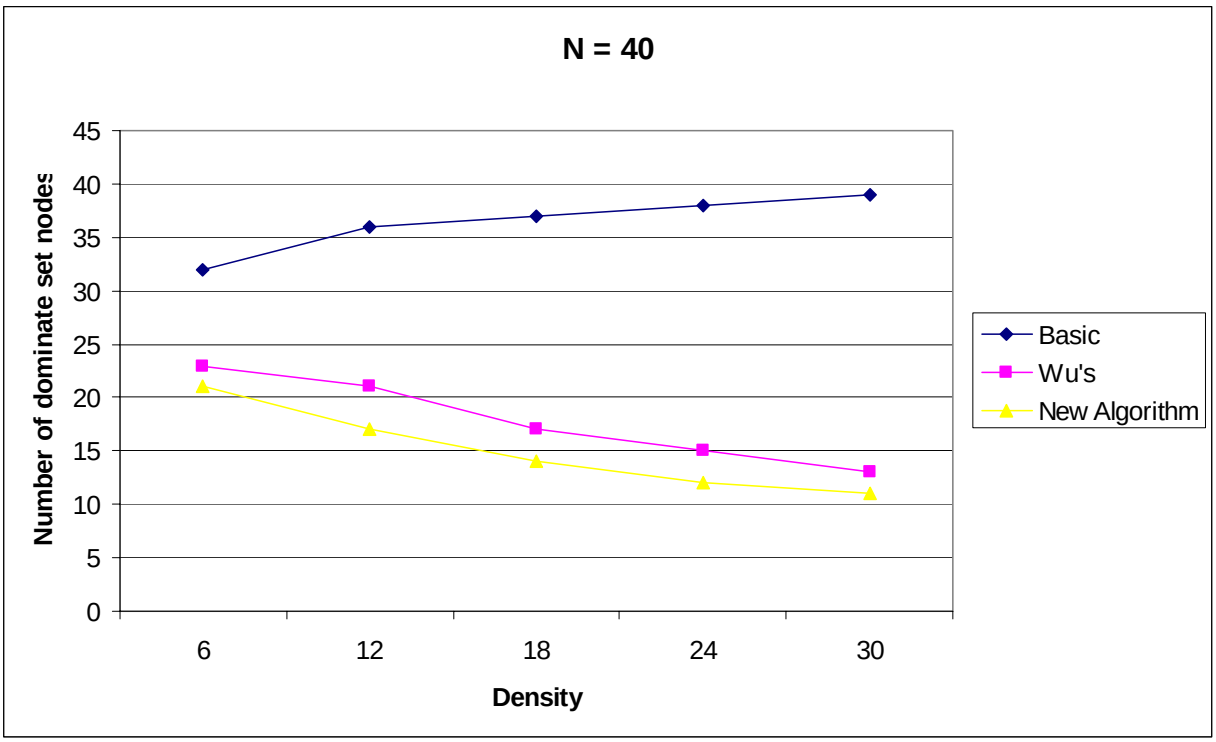

Figure 5.2: Average number of dominate set nodes relative to varying density for 40 nodes

Table 5.3: Number of dominating set nodes relative to varying density for 60nodes

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
60	6	50	36	32
60	12	56	32	26
60	18	58	28	22
60	24	58	25	19
60	30	59	22	17

Table 5.3 shows the average number of dominate set of nodes for a network size of 60 with varying densities. The size of dominate set increases as density increases for Basic rule whereas the size of the dominate set of nodes decreases for Wu and Li's extensional rules and for the new rule as expected (Refer to Figure 5.3).

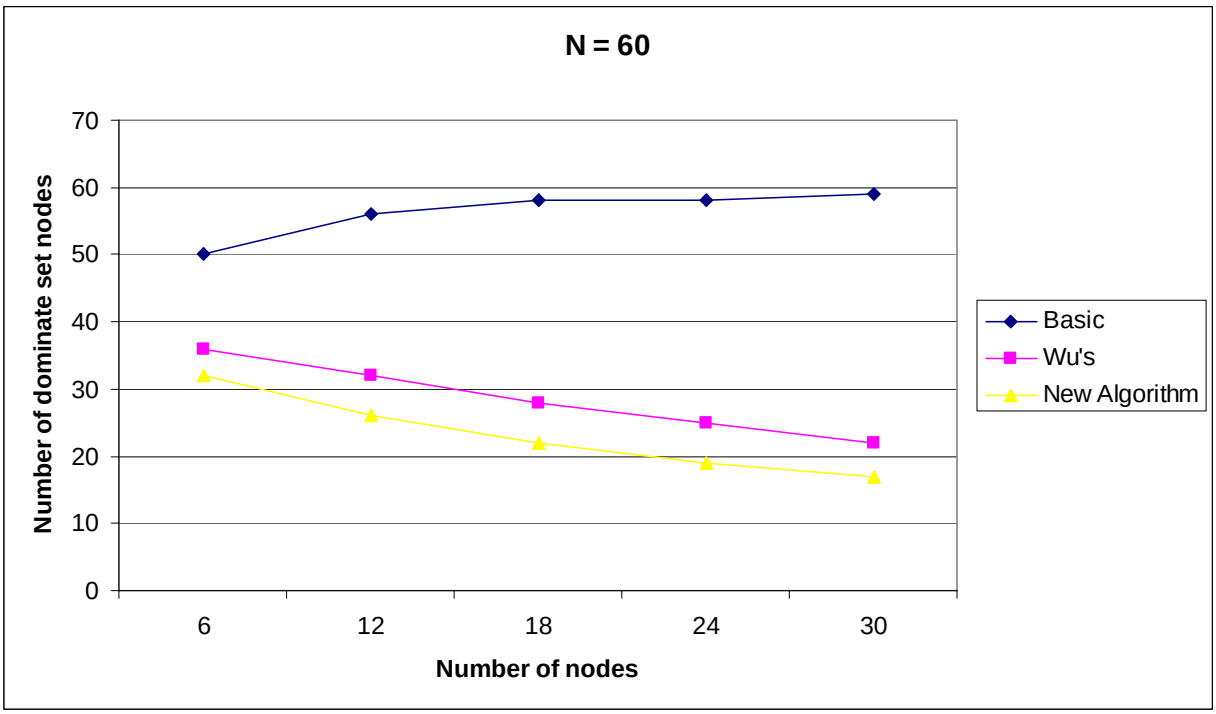

Figure 5.3: Average number of dominate set nodes relative to varying density for 6 nodes

Table 5.4 Number of dominating set nodes relative to varying density for 80 nodes

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
80	6	67	49	43
80	12	75	44	36
80	18	77	39	30
80	24	78	34	26
80	30	79	31	23

Table 5.4 shows the average number of dominate set of nodes for a network size of 80 with varying densities. The size of dominate set increases as density increases for Basic rule whereas the size of the dominate set of nodes decreases for Wu and Li's extensional rules and for the new rule as expected (Refer to Figure 5.4).

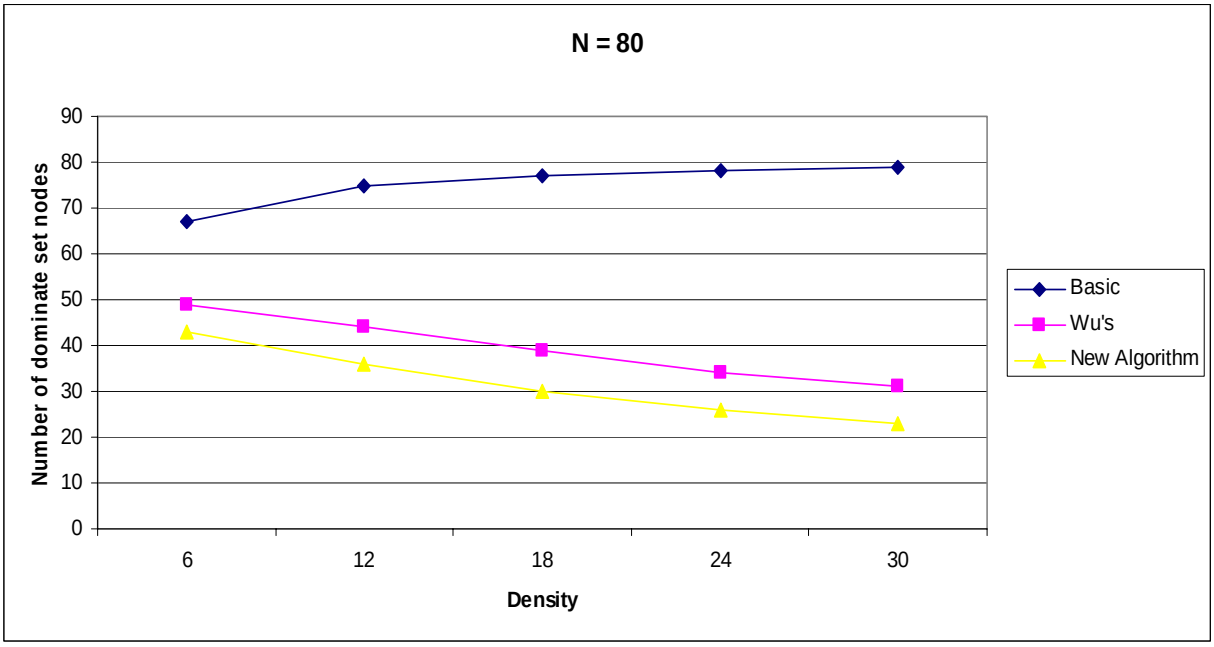

Figure 5.4: Average number of dominate set nodes relative to varying density for 80 nodes

Table 5.5: Number of dominating set nodes relative to varying density for 170 nodes

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
170	6	147	107	96
170	12	164	100	79
170	18	167	88	66
170	24	168	78	58
170	30	169	70	51

Table 5.5 shows the average number of dominate set of nodes for a network size of 170 with varying densities. The size of dominate set increases as density increases for Basic rule whereas the size of the dominate set of nodes decreases for Wu and Li's extensional rules and for the new rule as expected (Refer to Figure 5.5).

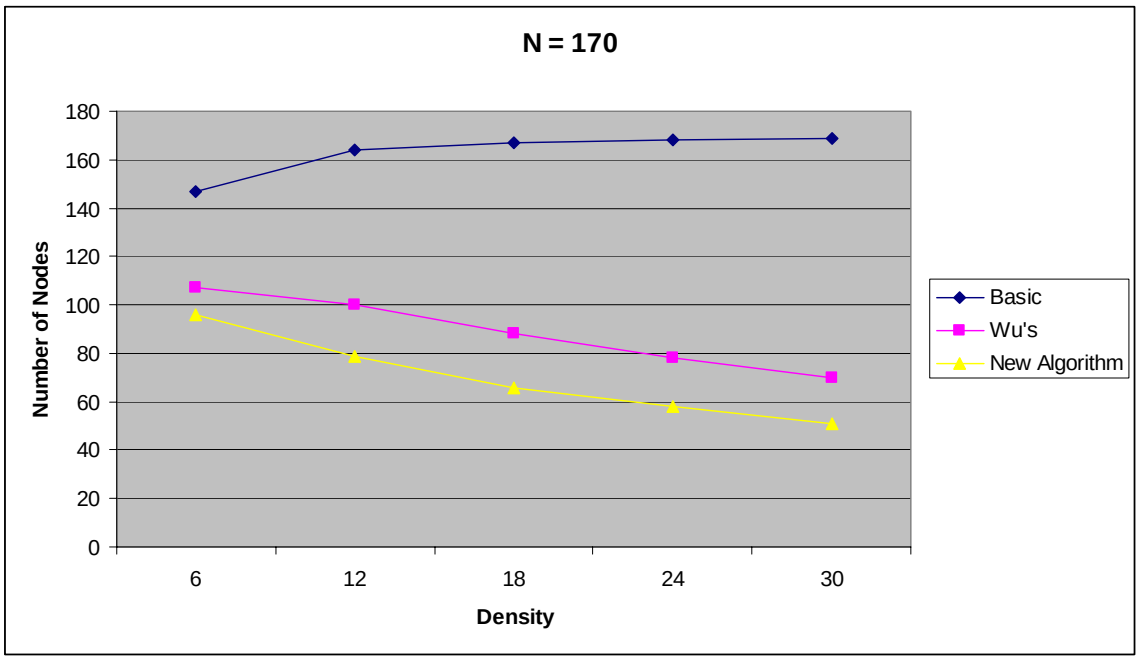

Figure 5.5: Average number of dominate set nodes relative to varying density for 170 nodes

Table 5.6: Number of dominating set nodes relative to varying density for 180 nodes

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
180	6	155	114	101
180	12	173	106	84
180	18	177	93	70
180	24	178	83	61
180	30	179	75	54

Table 5.6 shows the average number of dominate set of nodes for a network size of 180 with varying densities. The size of dominate set increases as density increases for Basic rule whereas the size of the dominate set of nodes decreases for Wu and Li's extensional rules and for the new rule as expected (Refer to Figure 5.6).

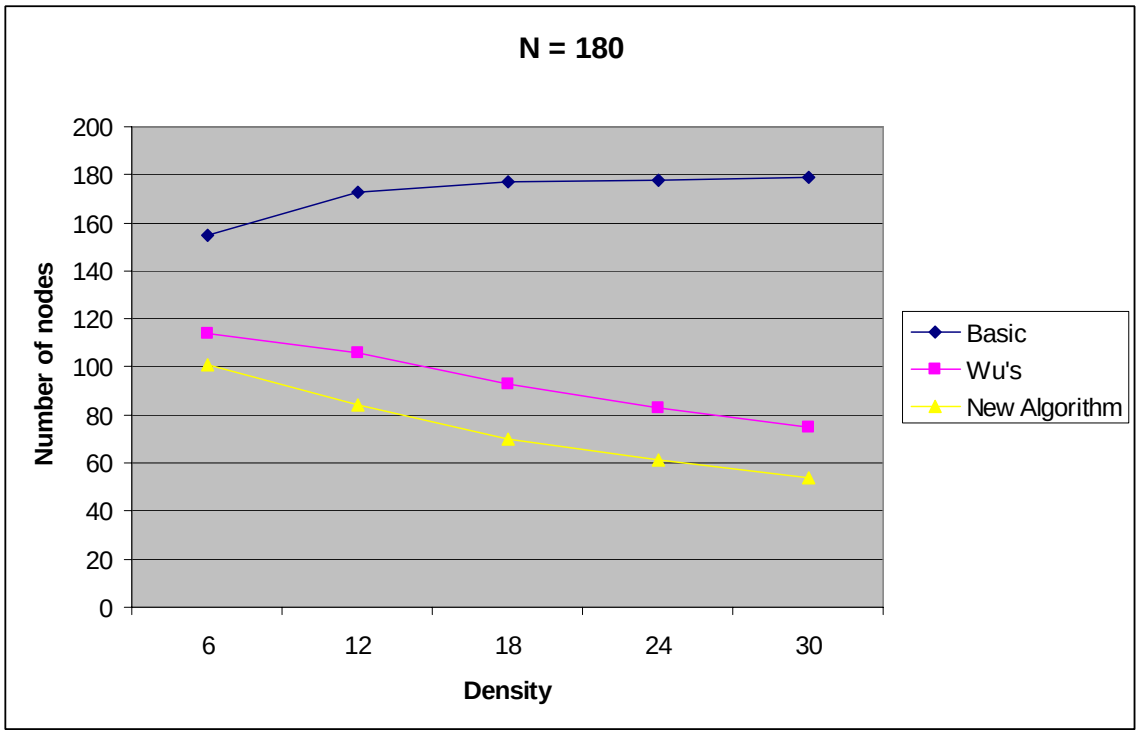

Figure 5.6: Average number of dominate set nodes relative to varying density for number of nodes N = 180

Table 5.7: Number of dominating set nodes relative to varying density for N = 190

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
190	6	165	121	107
190	12	183	112	88
190	18	187	99	74
190	24	188	88	65
190	30	188	79	58

Table 5.7 shows the average number of dominate set of nodes for a network size of 190 with varying densities. The size of dominate set increases as density increases for Basic rule whereas the size of the dominate set of nodes decreases for Wu and Li's extensional rules and for the new rule as expected (Refer to Figure 5.7).

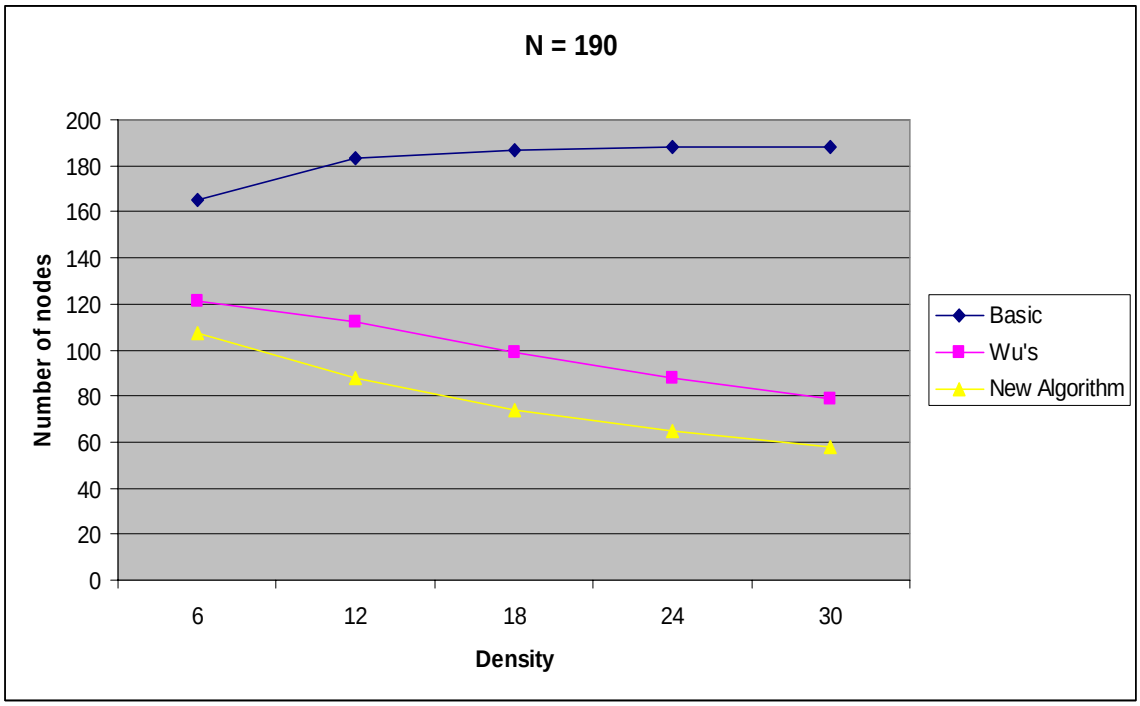

Figure 5.7: Average number of dominate set nodes relative to varying density with number of nodes N = 190

Table 5.8: Number of dominating set nodes relative to varying density for N = 200

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
200	6	173	127	112
200	12	193	118	94
200	18	197	104	78
200	24	198	93	68
200	30	199	84	61

Table 5.8 shows the average number of dominate set of nodes for a network size of 200 with varying densities. The size of dominate set increases as density increases for Basic rule whereas the size of the dominate set of nodes decreases for Wu and Li's extensional rules and for the new rule as expected (Refer to Figure 5.8).

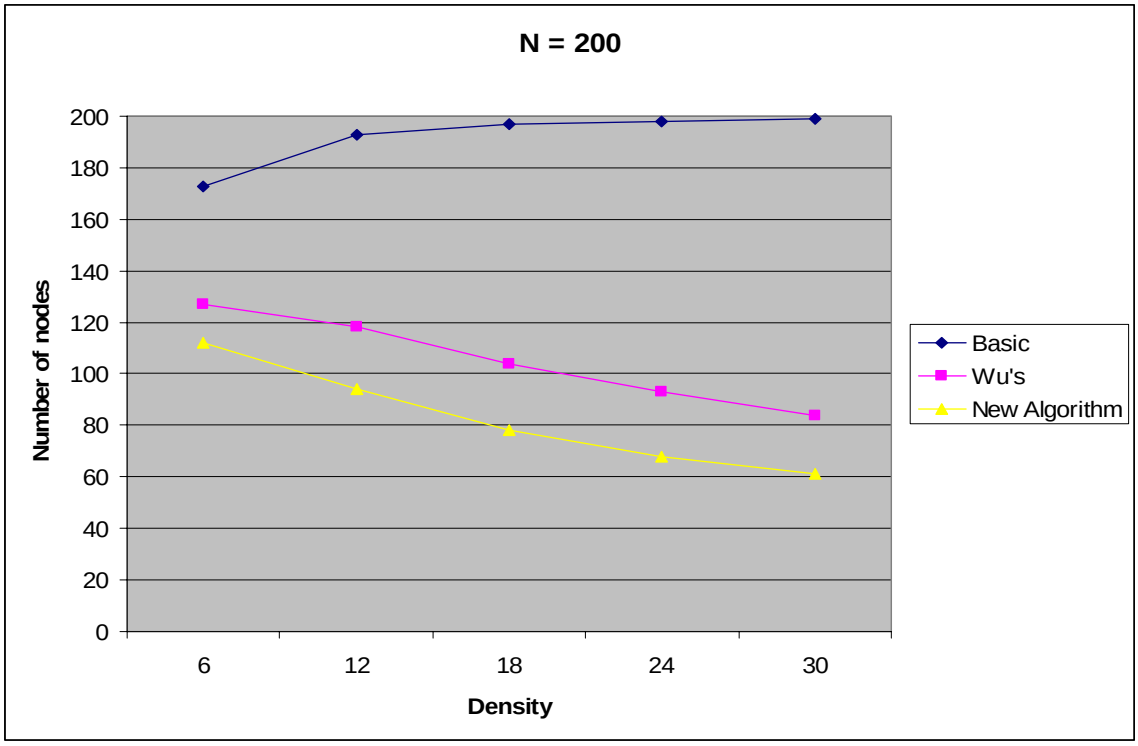

Figure 5.8: Average number of dominate set nodes relative to varying density for number of nodes N = 200

Table 5.9: Average number of dominating set nodes for constant density d = 6

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
20	6	15	11	10
40	6	32	23	21
60	6	50	36	32
80	6	67	49	43
100	6	85	62	55
120	6	102	75	66
140	6	120	87	80
160	6	137	99	88
180	6	155	114	101
200	6	173	127	112

Table 5.9 shows the average number of dominate set of nodes for a constant density of 6 with varying network sizes. The size of dominate set increases as the network size increases for all three rules as expected (Refer to Figure 5.9).

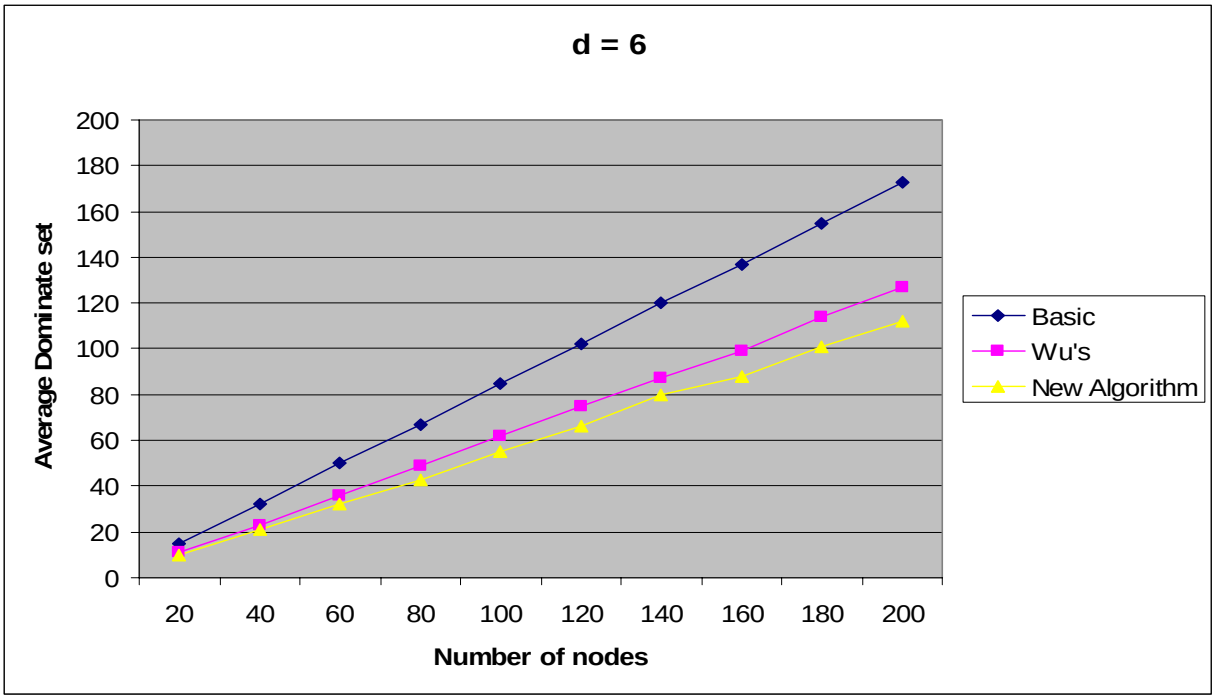

Figure 5.9: Average number of dominating set nodes relative to number of nodes for constant density d = 6

Table 5.10: Average number of dominating set nodes for constant density d = 12

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
20	12	17	9	8
40	12	36	21	17
60	12	56	32	26
80	12	75	44	36
100	12	95	57	45
120	12	114	69	55
140	12	134	81	64
160	12	155	93	75
180	12	173	106	84
200	12	193	118	94

Table 5.10 shows the average number of dominate set of nodes for a constant density of 12 with varying network sizes. The size of dominate set increases as the network size increases for all three rules as expected (Refer to Figure 5.10).

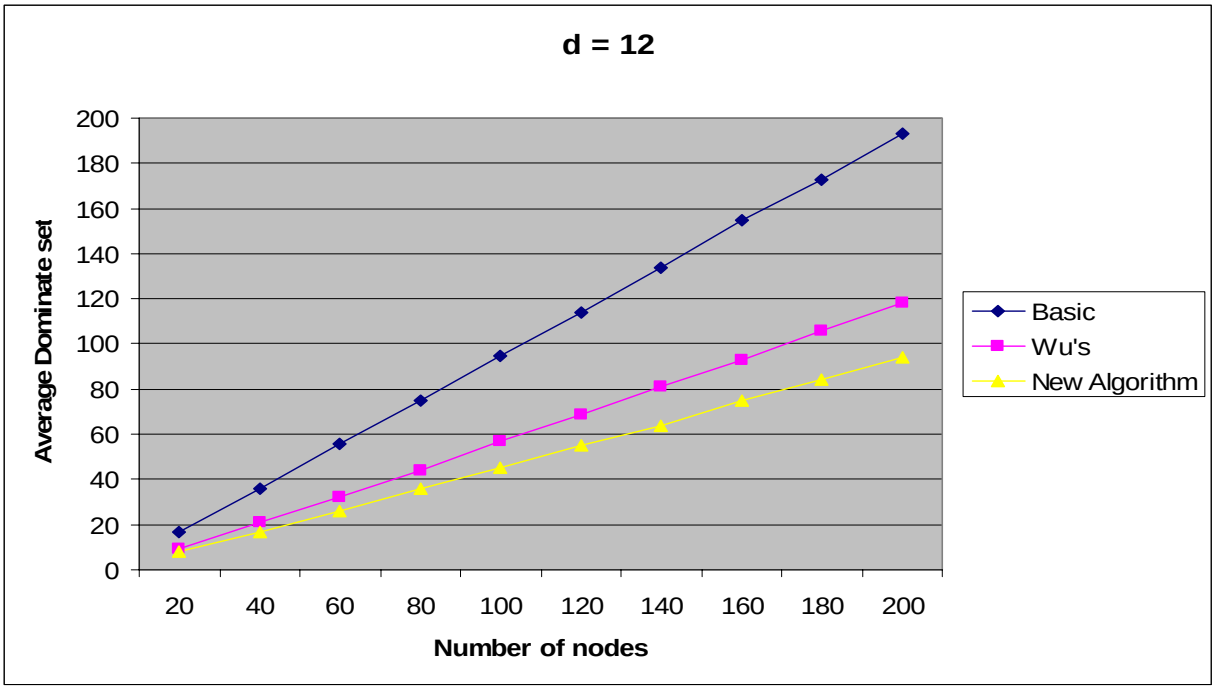

Figure 5.10: Average number of dominating set nodes relative to number of nodes for constant density d = 12

Table 5.11: Average number of dominating set nodes for constant density d = 18

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
20	18	18	7	6
40	18	37	17	14
60	18	58	28	22
80	18	77	39	30
100	18	97	50	38
120	18	117	61	46
140	18	137	71	54
160	18	157	83	62
180	18	177	93	70
200	18	197	104	78

Table 5.11 shows the average number of dominate set of nodes for a constant density of 6 with varying network sizes. The size of dominate set increases as the network size increases for all three rules as expected (Refer to Figure 5.11).

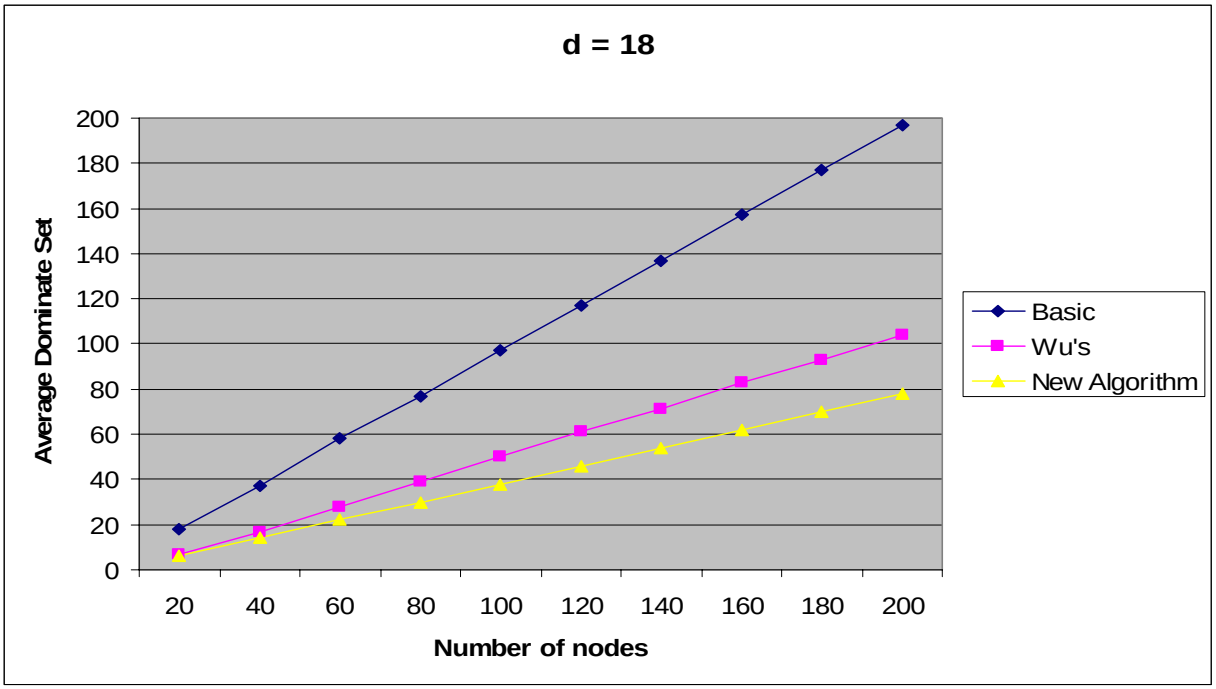

Figure 5.11: Average number of dominating set nodes relative to number of nodes for constant density d = 18

Table 5.12: Average number of dominating set nodes for constant density d = 24

Number of Nodes	Density	Basic Rule	Wu's Algorithm	New Rule
20	24	18	6	5
40	24	38	15	12
60	24	58	25	19
80	24	78	34	26
100	24	98	44	33
120	24	118	54	40
140	24	138	61	46
160	24	158	71	52
180	24	178	83	61
200	24	198	93	68

Table 5.12 shows the average number of dominate set of nodes for a constant density of 6 with varying network sizes. The size of dominate set increases as the network size increases for all three rules as expected (Refer to Figure 5.12).

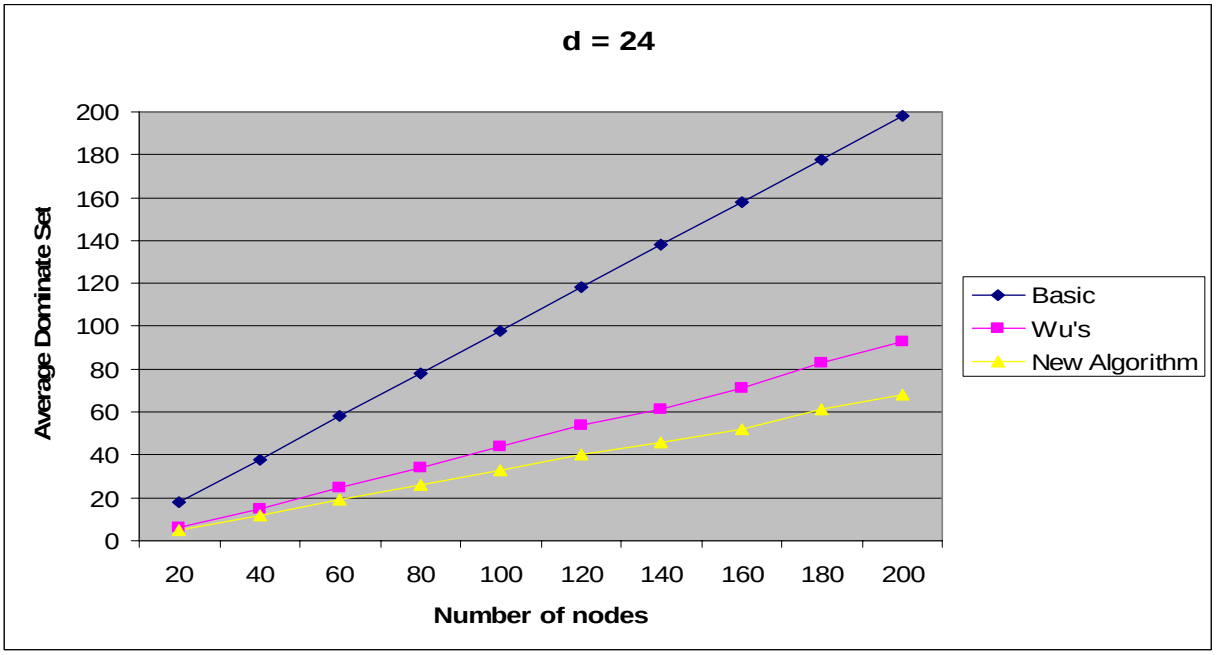

Figure 5.12: Average number of dominating set nodes relative to number of nodes for constant density d = 24

The above tables show the number of dominating set nodes versus the number of nodes in the space for the increasing order of density. The averages are recorded in the tables by using basic rule, Wu's extensional algorithms and the new algorithm. All these figures show that the number of dominating set nodes increased by the number of nodes. The efficiency of basic rule is poor because the ratio of the line is almost 1, especially when the density is very high. That means the number of dominating set nodes is almost equal to the number of nodes i.e.; almost every node is dominating set node decided by basic rule. The range (maximum-minimum) and SD of the number of dominating set is decreased by the nodes and density increasing because the number of dominating set tends to equal the number of nodes. But after applying the two extensional rules of Wu's algorithm, the size of dominating set decreased considerably, especially when the density is increased. The ratio of Wu's algorithm of nodes over the number of nodes changes from 0.64 to 0.25. The range (maximum-minimum) and SD of the number of dominating set is increased. By using the new algorithm, the size of the dominating set decreased further, it is less than the nodes decided by Wu's algorithm. The ratio of dominate nodes over the number of nodes changes from 0.55 to 0.2. In low density nodes space, every node has fewer neighbors, sometimes a node has less than three neighbors that it can't even use the new extensional rule. There is only some change in low density space. In high density space the size of the dominating set nodes is already decreased a lot.

6. CONCLUSION

An Ad hoc wireless network is a special kind of wireless network without the aid of any established infrastructure or centralized administration. The routing of packets between any two nodes not directly connected can be achieved through intermediate nodes. Finding an optimal route faces many challenges in ad hoc networks. Dominating-set-based routing is one kind of routing protocol proposed to reduce running time. A dominating set has all the nodes with in the set or within its neighborhood. Wu and Li proposed an efficient algorithm to calculate the connected dominating set. The research presented in this thesis extended Wu and Li's algorithm to calculate dominating set in ad hoc wireless network.

Our simulation results verify that Wu and Li's algorithm results from using basic rule only is poor and generates a large dominating set. Wu and Li's extensional rules 1 and 2 decrease the dominating set nodes considerably. After applying the new algorithm the decrease in numbers is more evident in high-density medium. The simulation results also show that results from the new algorithm constantly out performs Wu and Li's extensional rules.

The future research direction is to compare the new rule with other enhancements of Wu and Li's algorithm. Ni's [26] enhancement uses transmission range as a parameter and we used density as a parameter. One future area of work would be to compare these two enhancements using one single parameter either density or transmission range. Another direction for extending this research is to observe the simulation results using other parameters; we have used density, transmission range and number of nodes as the parameters in this research.

7. BIBLIOGRAPHY

[1] K. M. Alzoubi, P. J. Wan and O. Frieder, *New distributed algorithm for connected dominating set in wireless ad hoc networks*, *Proc. 35th Hawaii Int. Conf. On System Sciences*, 2002, pp. 1-7.

[2] K. M. Alzoubi, P. J. Wan and O. Frieder, *Distributed heuristics for connected dominating sets in wireless ad hoc networks*, J. Comm. and Networks, 4 (2002), pp. 22-29.

[3] V. Bharghavan and B. Das, *Routing in ad hoc networks using minimum connected domination sets, in Proc. Int. Conf. Commun*, Montreal, Canada, 1997.

[4] I. Castineyra, N. Chiappa, and M. Sttenstrup, *The nimrod routing architecture*, 1996.

[5] X. Cheng, X. Huang, D. Li, and D.Z. Du, *Polynomial-Time Approximation Scheme for Minimum Connected Dominating Set in Ad Hoc Wireless Networks* Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 02-003, 2003.

[6] M. S. Corson and A. Ephremides, *A distributed routing algorithm for mobile wireless networks*, ACM/Baltzer Wireless Networks, 1 (1995), pp. 61-81.

[7] S. Corson and J. Macker, *Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*, RFC 2501, January 1999.

[8] F. Dai and J. Wu, *Distributed dominant pruning in ad hoc wireless networks*, In Proc. Of IEEE International Conference on Communications (ICC), 2003.

[9] F. Dai and J. Wu, *An extended localized algorithm for connected dominating set formation in ad hoc wireless networks*, IEEE Trans. Parallel and Distributed Systems, 15 (2004), pp. 908-920.

[10] B. Das, R. Sivakumar and V. Bharghavan, *Routing in ad hoc networks using a spine*, *in Proc. Int. Conf. Comput. and Commun. Networks*, Las Vegas, NV, 1997.

[11] S. Datta, I. Stojmenovic, and J. Wu, "Internal node and shortcut based routing with guaranteed delivery in wireless networks," Cluster Computing, vol. 5, 2002.

[12] Elizabeth M. Royer, Chai-Keong Toh, *A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks* ,IEEE Personal Communications, April, 1999

[13] S. Guha and S. Khuller, *Approximation algorithms for connected dominating sets*, Algorithmica, 20 (1998), pp. 374-387.

[14] P. Jacquet, P. Muhlethaler and A. Qayyum, *Optimized link state routing protocol, Internet-Draft, draft-ietf-manet-olsr-00.txt*, 1998.

[15] R. Jain, A. Puri and R. Sengupta, *Geographical routing using partial information for wireless ad hoc networks*, IEEE Personal Communications (2001), pp. 48-57.

[16] D.B. Johnson, *Routing in ad hoc networks of mobile hosts*, presented at Workshop on Mobile Computing Systems and Applications, 1994.

[17] D.B. Johnson, D. A. Maltz, Y.-C. Hu and J. G. Jetcheva, "The dynamic source routing protocol for mobile ad hoc networks", *Internet Draft* http://www.ietf.org/internet-drafts/draftietf-manet-dsr-07.txt,March 2001.

[18] H. Hartenstein, M. Kasemann, H. Fubler, and M. Mauve, " A simulation study of a location service for position-based routing in mobile ad hoc networks," Department of Science, University of Mannheim TR-02-007, 2002.

[19] P. Krishna, M. Chaterjee, N. H. Vaidya, and D. K. Pradhan, "A cluster-based approach for routing in ad-hoc networks," presented at the Second USENIX Symposium on Mobile and Location-Independent Computing, 1995.

[20] H. Liu, J. Li, Y. Pan and Y. Zhang, *An adaptive genetic fuzzy multi-path routing protocol for wireless ad-hoc networks, the 1st ACIS International Workshop on Self-Assembling Wireless Networks (SAWN 2005)*, Maryland, U.S.A, 2005.

[21] H. Liu and Y. Pan, *A scalable location management scheme for position-based routing in mobile ad hoc networks*, in Y. Xiao, J. Li and Y. Pan, eds., *Security and Routing in Wireless Networks*, Nova Science Publishers, 2005.

[22] H. Liu, Y. Pan and J. Cao, *An improved distributed algorithm for connected dominating sets in wireless ad hoc networks*, *Proc. Int. Symp. on Parallel and Distributed Processing and Applications (ISPA) Lecture Notes in Computer Science* 2004, pp. 340-351.

[23] J.M. McQuillan and D.C. Walden, *The ARPA network design decisions*, Computer Networks, vol. 1, pp.243-289, 1997.

[24] S. Murthy and J. J. Garcia-Luna-Aceves, *An efficient routing protocol for wireless networks*, ACM/Baltzer Mobile Networks and Applications special issue on Routing in Mobile Communications Networks, 1 (1996), pp. 183-197.

[25] A. Nasipuri and S. R. Das, *On-demand multipath routing for mobile ad hoc networks*, *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN)*, Boston, MA, 1999, pp. 64-70.

[26] Ni, C., Liu, H., Bourgeois, A.G. and Pan, Y. *'An enhanced approach to determine connected dominating sets for routing in mobile ad hoc networks'*, Int. J. Mobile Communications 2005, Vol. 3, No.3 pp. 287 – 302.

[27] C.E. Perkins and P. Bhagwat, *Highly dynamic destination-sequences distance vector routing (DSDV) for mobile computers*, Computer Communications Review, vol.24, pp.234-244, 1994.

[28] E. M. Royer and C. K. Toh, *A review of current routing protocols for ad hoc mobile wireless networks*, IEEE Personal Communications, 6 (1999), pp. 46-55.

[29] B. Shrader, "A proposed definition of 'Ad hoc network'", Royal Institute of Technology (KTH), Stockholm, Sweden 2002.

[30] I. Stojmenovic, *Data gathering and activity scheduling in ad hoc and sensor* networks, Proc. International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peer-to-Peer Networks, Chicago, Illinois, 2004.

[31] I. Stojmenovic, *Home region based location updates and destination search schemes in ad hoc wireless networks*, SITE, University of Ottawa, 1999.

[32] I. Stojmenovic and X. Lin., *Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks*, IEEE Transactions on Parallel and Distributed Systems, 12 (2001), pp. 1023-1032.

[33] I. Stojmenovic, M. Seddigh and J. Zunic, *Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks,* Proc. IEEE Hawaii Int. Conf. On System Sciences, 2001.

[34] J. Wu and H. Li, "A dominating-set-based routing scheme in ad hoc wireless networks," Telecommunication Systems Journal, vol. 3, pp.63-84, 2001.

Appendix – A

VC++ code for simulating the new algorithm [26]:

```
// CDSView.cpp : implementation of the CCDSView class
// Other files and include files not listed here
//

#include "stdafx.h"
#include "CDS.h"
#include <afxwin.h>

#include "CDSDoc.h"
#include "CDSView.h"
#include "NodeSizeDlg.h"
#include "TransmitionRangeDlg.h"
#include "FormCommandView.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
//////
// CCDSView

IMPLEMENT_DYNCREATE(CCDSView, CView)

BEGIN_MESSAGE_MAP(CCDSView, CView)
      //{{AFX_MSG_MAP(CCDSView)
      ON_COMMAND(ID_OPERATIONS_GENERATENODES,
OnOperationsGenerateNodes)
      ON_WM_PAINT()
      ON_COMMAND(ID_OPERATIONS_CONNECTINGNODES,
OnOperationsConnectingNodes)
      ON_COMMAND(ID_OPERATIONS_DS1, OnOperationsDs1)
      ON_COMMAND(ID_OPERATIONS_DS2, OnOperationsDs2)
      ON_COMMAND(ID_OPERATIONS_DS3, OnOperationsDs3)
      ON_COMMAND(ID_OPERATIONS_DS4, OnOperationsDs4)
      ON_COMMAND(ID_OPERATIONS_READ, OnOperationsRead)
      ON_COMMAND(ID_CheckConnected, OnCheckConnected)
      ON_COMMAND(ID_OPERATIONS_RUN, OnOperationsRun)
      ON_COMMAND(ID_OPERATIONS_DS5, OnOperationsDs5)
      //}}AFX_MSG_MAP
      // Standard printing commands
      ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
      ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
      ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()
```

```
////////////////////////////////////////////////////////////////////////
//////
// CCDSView construction/destruction

CCDSView::CCDSView()
{
	// TODO: add construction code here

}

CCDSView::~CCDSView()
{
}

BOOL CCDSView::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: Modify the Window class or styles here by modifying
	//  the CREATESTRUCT cs

	return CView::PreCreateWindow(cs);
}

////////////////////////////////////////////////////////////////////////
//////
// CCDSView drawing

void CCDSView::OnDraw(CDC* pDC)
{
	CCDSDoc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);
	// TODO: add draw code for native data here
}

////////////////////////////////////////////////////////////////////////
//////
// CCDSView printing

BOOL CCDSView::OnPreparePrinting(CPrintInfo* pInfo)
{
	// default preparation
	return DoPreparePrinting(pInfo);
}

void CCDSView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: add extra initialization before printing
}

void CCDSView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: add cleanup after printing
}

////////////////////////////////////////////////////////////////////////
//////
// CCDSView diagnostics
```

```
#ifdef _DEBUG
void CCDSView::AssertValid() const
{
	CView::AssertValid();
}

void CCDSView::Dump(CDumpContext& dc) const
{
	CView::Dump(dc);
}

CCDSDoc* CCDSView::GetDocument() // non-debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CCDSDoc)));
	return (CCDSDoc*)m_pDocument;
}
#endif //_DEBUG

////////////////////////////////////////////////////////////////////////
//////
// CCDSView message handlers

void CCDSView::OnOperationsGenerateNodes()
{
	// TODO: Add your command handler code here
	OperationsGenerateNodes();
}

bool CCDSView::OperationsGenerateNodes()
{
	CFormCommandView*	pCommondView	=	(CFormCommandView*)
	GetFormCommandView();
	m_Data.GenerateRandomNumber(pCommondView->m_Nodes_Size);
	operations=0;
	this->InvalidateRgn(NULL);
	return true;
}

void CCDSView::OnOperationsConnectingNodes()
{
/*	CTransmitionRangeDlg Dlg;
	if(Dlg.DoModal() != IDOK)
		return;  Dlg.m_Transmition_Range*/
	operations=1;
	CFormCommandView*	pCommondView	=	(CFormCommandView*)
	GetFormCommandView();
	m_Data.Set_Transmition_Range(pCommondView->m_Transmission_Range);

	this->InvalidateRgn(NULL);
	return;
}

void CCDSView::OnCheckConnected()
{
	OperationsCheckConnected();
}
```

```
bool CCDSView::OperationsCheckConnected()
{
	CString strYes="YES";
	CString strNo="NO";
	CFormCommandView*	pCommondView	=	(CFormCommandView*)
	GetFormCommandView();
	int count=0;
	for(int p=0; p<m_Data.m_Nodes.GetSize(); p++)
		m_Data.m_Nodes[p].m_colour=0;
	DFS(0,count);
	if(count==m_Data.m_Nodes.GetSize())
	{
		pCommondView->m_Connected=strYes;
		pCommondView->UpdateData(FALSE);
		return true;
	}
	else
	{
		pCommondView->m_Connected=strNo;
		pCommondView->UpdateData(FALSE);
		return false;
	}
}


void CCDSView::OnOperationsDs1()
{
    operations=2;
	  this->InvalidateRgn(NULL);
	  return;
}

void CCDSView::OnOperationsDs2()
{
    operations=3;
	  this->InvalidateRgn(NULL);
	  return;
}

void CCDSView::OnOperationsDs3()
{
    operations=4;
	  this->InvalidateRgn(NULL);
	  return;
}

void CCDSView::OnOperationsDs4()
{
	operations=5;
//    bool connected=OperationsCheckConnected();
	this->InvalidateRgn(NULL);
	return;
}

void CCDSView::OnOperationsDs5()
{
	operations=7;
```

```
//    bool connected=OperationsCheckConnected();
      this->InvalidateRgn(NULL);
      return;
}

void CCDSView::OnOperationsRun()
{
      CFormCommandView*      pCommondView      =      (CFormCommandView*)
      GetFormCommandView();

      CString str1="ab";
      CString str2="ab";
      CString str3="ab";
      CString str4="ab";
      int numberOfCycles=(pCommondView->m_Running_Cycles);
      int avgBasic=0;
      int avgJie=0;
      int avgNew=0;
      int avgMyRule=0;

      int i=0;
      do
      {
            OnOperationsGenerateNodes();
            OnOperationsConnectingNodes();
            OnPaint();
            if(OperationsCheckConnected())
            {
                  OnOperationsDs5();
                  OnPaint();

                  avgBasic=avgBasic+numberDS;
                  avgJie=avgJie+numberDS-numberDSRuleTotal;
                  avgMyRule     =     avgMyRule+numberDS-numberMyRuleADS-
                  numberMyRuleBDS-numberMyRuleCDS;

                  i++;
            }
      }while(i<numberOfCycles);

      avgBasic=(int)(avgBasic/double(numberOfCycles)+0.5);
      avgJie=(int)(avgJie/double(numberOfCycles)+0.5);
      avgNew=(int)(avgNew/double(numberOfCycles)+0.5);
      avgMyRule=(int) (avgMyRule/double(numberOfCycles)+0.5);
      str1.Format("%s%d", "", avgBasic);
      str2.Format("%s%d", "", avgJie);
      str3.Format("%s%d", "", avgNew);
      str4.Format("%s%d", "", avgMyRule);
      str1="avgBasic is: "+str1+" || avgJie is: "+str2+" || avgNew is:
      "+str3+" || avgMyRule is: "+str4;
    AfxMessageBox(str1);

//    this->InvalidateRgn(NULL);

}

void CCDSView::OnPaint()
```

```
{
	CFormCommandView*	pCommondView	=	(CFormCommandView*)
	GetFormCommandView();
	CRect rect;
	GetClientRect(rect);

	numberDS=0;
	numberDSRuleTotal=0;
	int x=(rect.right-600)/2;
	int y=(rect.bottom-600)/2;

	CPaintDC dc(this); // device context for painting

	dc.SetTextAlign(TA_BASELINE | TA_CENTER);
	dc.SetTextColor(::GetSysColor(COLOR_WINDOWTEXT));
	dc.SetBkMode(TRANSPARENT);

	CString str = "Some Data";
	// TODO: Add your message handler code here

	for(int i=0; i<m_Data.m_Nodes.GetSize(); i++)
	{
		str.Format("%s%d", "", i);
		dc.TextOut((int)(x+m_Data.m_Nodes.GetAt(i).m_x),
		(int)(y+m_Data.m_Nodes.GetAt(i).m_y) , str);
	}

	if(operations>=1)
	{
		for(int p=0; p<m_Data.m_Nodes.GetSize(); p++)
		{

	m_Data.m_Nodes[p].m_ConnectedVertexIndices.RemoveAll();
			m_Data.m_Nodes[p].m_ClosedVertexIndices.RemoveAll();
			m_Data.m_Nodes[p].m_colour=0;
			m_Data.m_Nodes[p].m_Marked=0;
			m_Data.m_Nodes[p].m_MarkedRule1=0;
			m_Data.m_Nodes[p].m_MarkedRule2=0;
			m_Data.m_Nodes[p].m_MarkedRule3=0;
		}
		for(int j=0; j<m_Data.m_Nodes.GetSize()-1; j++)
		{
			int x0 = x+(int)m_Data.m_Nodes.GetAt(j).m_x;
			int y0 = y+(int)m_Data.m_Nodes.GetAt(j).m_y;
			m_Data.m_Nodes[j].m_ClosedVertexIndices.Add(j);
			for(int k=j+1;k<m_Data.m_Nodes.GetSize();k++)
			{
				dc.MoveTo(x0, y0);
				int x1 = x+(int)m_Data.m_Nodes.GetAt(k).m_x;
				int y1 = y+(int)m_Data.m_Nodes.GetAt(k).m_y;
				if(((x0-x1)*(x0-x1)+(y0-y1)*(y0-
				y1))<((m_Data.Transmition_Range * 600 * 600 )/
				(3.14 * (m_Data.m_Nodes.GetSize()-1))))

				{
				dc.LineTo(x1,y1);
```

```
      m_Data.m_Nodes[j].m_ConnectedVertexIndices.Add(k);


      m_Data.m_Nodes[k].m_ConnectedVertexIndices.Add(j);
                        m_Data.m_Nodes[j].m_ClosedVertexIndices.Add(k);
                        m_Data.m_Nodes[k].m_ClosedVertexIndices.Add(j);
                        }

                  }
            }
            int size1=m_Data.m_Nodes.GetSize()-1;
            m_Data.m_Nodes[size1].m_ClosedVertexIndices.Add(size1);
      } //end operations=1

      //Code for Basic Rule
      if(operations>=2)
      {
      for(int j=0; j<m_Data.m_Nodes.GetSize(); j++)
      {
      int
indices_size=m_Data.m_Nodes[j].m_ConnectedVertexIndices.GetSize();

      if(indices_size>1)
      {
       for(int m=0;m<indices_size-1;m++)
       {
       int u=m_Data.m_Nodes[j].m_ConnectedVertexIndices[m];
       for(int n=m+1;n<indices_size;n++)
       {
       int v=m_Data.m_Nodes[j].m_ConnectedVertexIndices[n];
       bool marked=true;
for                                                              (int
l=0;l<m_Data.m_Nodes[u].m_ConnectedVertexIndices.GetSize();l++)
      {
            if(m_Data.m_Nodes[u].m_ConnectedVertexIndices[l]==v)
            {
                  marked=false;
            l  =  m_Data.m_Nodes[u].m_ConnectedVertexIndices.GetSize()  -
1;
            }
      }
            if(marked==true)
            {
            m_Data.m_Nodes[j].SetMark(marked);
            n = indices_size - 1;
            }
      }
      if(   m_Data.m_Nodes[j].m_Marked)
            m = indices_size - 2;
      }
      }
      }
      for(int z=0; z<m_Data.m_Nodes.GetSize(); z++)
      {
            if(m_Data.m_Nodes[z].m_Marked==true)
```

```
            {
            DrawDotCircle(dc,                   x+m_Data.m_Nodes[z].m_x,
            y+m_Data.m_Nodes[z].m_y,RGB(255, 0, 0));
            numberDS++;
            }
      }
            str.Format("%s%d", "", numberDS);
            pCommondView->m_Basic=str;
            pCommondView->UpdateData(FALSE);

      }//end if operations=2



//Code for Extensional Rule 1
      if(operations>=3)
      {

      for(int j=0; j<m_Data.m_Nodes.GetSize(); j++)
      {
            if(m_Data.m_Nodes[j].m_Marked==true)  //if@2
            {
            int                                                 size1=
m_Data.m_Nodes[j].m_ConnectedVertexIndices.GetSize();

            for(int m=0;m<size1;m++)
            {
            int u=m_Data.m_Nodes[j].m_ConnectedVertexIndices[m];

            int
size2=m_Data.m_Nodes[u].m_ClosedVertexIndices.GetSize();

            if(m_Data.m_Nodes[u].m_Marked==true && j<u && size1<size2)
            {
            bool
            marked=subset(m_Data.m_Nodes[j].m_ConnectedVertexIndices,
            m_Data.m_Nodes[u].m_ClosedVertexIndices);

                  if(marked==true)
                  {
                  m = size1-1;
                  }
            }//end if@2
            }
            }//end if@1
            } //end for every node

            numberDSRule1=0;
            for(int z=0; z<m_Data.m_Nodes.GetSize(); z++)
            {
                  if(m_Data.m_Nodes[z].m_MarkedRule1==true)
                  {
                        DrawDotCircle(dc,       x+m_Data.m_Nodes[z].m_x,
                        y+m_Data.m_Nodes[z].m_y,RGB(0, 0, 255));
                        numberDSRule1++;
                  }
```

```
      }
} //end operations=3

//Code for Extensional Rule 2
if(operations>=4)
{
intArray C;

for(int j=0; j<m_Data.m_Nodes.GetSize()-1; j++)
{
int size1= m_Data.m_Nodes[j].m_ConnectedVertexIndices.GetSize();

bool marked=false;

      if(m_Data.m_Nodes[j].m_Marked==true && size1>=2)
      {
      for(int m=0;m<size1-1;m++)
      {

      int u=m_Data.m_Nodes[j].m_ConnectedVertexIndices[m];

      if(u>j&&m_Data.m_Nodes[u].m_Marked)
      {
      for(int n=m+1;n<size1;n++)
            {
            C.RemoveAll();
            int v=m_Data.m_Nodes[j].m_ConnectedVertexIndices[n];
            if(m_Data.m_Nodes[v].m_Marked)
            {

                  unionArray(m_Data.m_Nodes[u].m_ConnectedVertexI
            ndices,    m_Data.m_Nodes[v].m_ConnectedVertexIndices,
            C);


marked=subset(m_Data.m_Nodes[j].m_ConnectedVertexIndices,C);

                  if(marked)
                  n=size1-1;
            }
            }

            if(marked)
            {
                        m_Data.m_Nodes[j].SetMarkRule2(marked);
                  m=size1-2;
            }
      }
      }
      }//end if@1
      } //end for every node

int numberDSRule2=0;
for(int z=0; z<m_Data.m_Nodes.GetSize(); z++)
{
      if(m_Data.m_Nodes[z].m_MarkedRule2)
      {
```

```
            DrawDotCircle(dc,                    x+m_Data.m_Nodes[z].m_x,
            y+m_Data.m_Nodes[z].m_y,RGB(0, 255, 0));
            numberDSRule2++;
            }
      }

      for(int w=0; w<m_Data.m_Nodes.GetSize(); w++)
      {
      if(m_Data.m_Nodes[w].m_Marked                                    &&
      (m_Data.m_Nodes[w].m_MarkedRule1||m_Data.m_Nodes[w].m_MarkedRule2
      ))
            {
                  numberDSRuleTotal++;
            }
      }
      str.Format("%s%d", "", numberDS-numberDSRuleTotal);
      pCommondView->m_JieWu=str;
      pCommondView->UpdateData(FALSE);

      }


      if(operations==7)
      {
      intArray C1,C2,C3a,C3b;
      for(int u=0; u<m_Data.m_Nodes.GetSize()-1; u++)
      {

      int
indices_size=m_Data.m_Nodes[u].m_ConnectedVertexIndices.GetSize();

      if(m_Data.m_Nodes[u].m_Marked && indices_size>=2)
      {
       for(int m=0;m<indices_size-1;m++)
         {
         int v=m_Data.m_Nodes[u].m_ConnectedVertexIndices[m];

         for(int n=m+1;n<indices_size;n++)
         {
            int w=m_Data.m_Nodes[u].m_ConnectedVertexIndices[n];

            C1.RemoveAll();
            unionArray(m_Data.m_Nodes[v].m_ConnectedVertexIndices,
            m_Data.m_Nodes[w].m_ConnectedVertexIndices, C1);

            if (subset(m_Data.m_Nodes[u].m_ConnectedVertexIndices,C1))
            {
            bool markeda = false;
            if (u<v && u<w)
            {
                        markeda = true;

                        m_Data.m_Nodes[u].SetMyRuleA(markeda);
            }

            if (v<u && w<u)
      {
```

```
            bool markedc = true;

            for(int o=m+1;o<indices_size;o++)
            {
            int z=m_Data.m_Nodes[u].m_ConnectedVertexIndices[o];

                  if (z!=w && z!=v)
            {
                  C3a.RemoveAll();
            unionArray(m_Data.m_Nodes[u].m_ConnectedVertexIndices,
            m_Data.m_Nodes[z].m_ConnectedVertexIndices, C3a);
            if   (subset(m_Data.m_Nodes[v].m_ConnectedVertexIndices,C3a)
            && subset(m_Data.m_Nodes[w].m_ConnectedVertexIndices,C3a))
                  {
                  markedc = false;
                        }
            }
      }
            if (markedc == true)
            {
            m_Data.m_Nodes[u].SetMyRuleC(markedc);
            }
      }


            if (v<u && u<w)
      {
            bool markedb = true;

      for(int o=m+1;o<indices_size;o++)
      {
            int z=m_Data.m_Nodes[u].m_ConnectedVertexIndices[o];

                  if (z!=w)
            {

                  C2.RemoveAll();
            unionArray(m_Data.m_Nodes[u].m_ConnectedVertexIndices,
      m_Data      .m_Nodes[z].m_ConnectedVertexIndices, C2);


                  if
(subset(m_Data.m_Nodes[v].m_ConnectedVertexIndices,C2))
                  {
                  markedb = false;

                        }
            }
      }

            if (markedb == true)
            {
      m_Data.m_Nodes[u].SetMyRuleB(markedb);
            }
      }
}
}
```

```
}
}
}


      numberMyRuleADS=0;
      for(int z=0; z<m_Data.m_Nodes.GetSize(); z++)
      {
            if(m_Data.m_Nodes[z].m_MarkedMyRuleA)
            {
            DrawDotCircle(dc,                   x+m_Data.m_Nodes[z].m_x,
            y+m_Data.m_Nodes[z].m_y,RGB(0, 255, 255));
            numberMyRuleADS++;
            }
      }
      numberMyRuleBDS=0;
      for(int z=0; z<m_Data.m_Nodes.GetSize(); z++)
      {
            if(m_Data.m_Nodes[z].m_MarkedMyRuleB&&(!m_Data.m_Nodes[z].m
            _MarkedMyRuleA)&&(!m_Data.m_Nodes[z].m_MarkedMyRuleC))
            {
            DrawDotCircle(dc,                   x+m_Data.m_Nodes[z].m_x,
            y+m_Data.m_Nodes[z].m_y,RGB(165, 42, 42));
            numberMyRuleBDS++;
            }
      }
      numberMyRuleCDS=0;
      for(int z=0; z<m_Data.m_Nodes.GetSize(); z++)
      {
            if(m_Data.m_Nodes[z].m_MarkedMyRuleC&&(!m_Data.m_Nodes[z].m
            _MarkedMyRuleA)&&(!m_Data.m_Nodes[z].m_MarkedMyRuleB))
            {
            DrawDotCircle(dc,       x+m_Data.m_Nodes[z].m_x,       y+m_Da
            ta.m_Nodes[z].m_y,RGB(122, 122, 122));
            numberMyRuleCDS++;
            }
      }

      str.Format("%s%d",  "",  numberDS-numberMyRuleADS-numberMyRuleBDS-
      numberMyRuleCDS);
      pCommondView->m_New=str;
      CString str1="ab";
      CString str2="ab";
      CString str3="ab";

      str1.Format("%s%d", "", numberMyRuleADS);
      str2.Format("%s%d", "", numberMyRuleBDS);
      str3.Format("%s%d", "", numberMyRuleCDS);

      str1="Total  is:  "+str+"  ||  RuleA  is:  "+str1+"  ||  RuleB  is:
      "+str2+" || RuleC is: "+str3;

      pCommondView->UpdateData(FALSE);

}//end if operations=7

} //end OnPaint()
```

```
bool CCDSView::DrawDotCircle(CPaintDC &dc, double x, double y, COLORREF
clr )
{
	int radius = 5;
	dc.FillSolidRect((int)x-radius,	(int)y-radius,	2*radius,
2*radius,clr);
	return true;
//	FillRect( LPCRECT lpRect, CBrush* pBrush );

}


bool CCDSView::subset(intArray &A, intArray &B)
{
	int position=0;
	bool marked=false;
	for(int p=0;p<A.GetSize();p++)
	{
		if(position==B.GetSize())
			break;
		int s=A[p];
		for(int q=position;q<B.GetSize();q++)
		{
			int t=B[q];
			if(s==t)
			{
				position=q+1;
				if((position==B.GetSize())&&(p==A.GetSize()-1))
					marked=true;
				q=B.GetSize()-1;
			}
			else if(s<t)
			{
				marked=false;
				q=B.GetSize()-1;
				p=A.GetSize()-1;
			}

		}
	}
	return marked;
}

void CCDSView::unionArray(intArray &A, intArray &B, intArray &C)
{
	int sizeA=A.GetSize();
	int sizeB=B.GetSize();
	int position=0;
	for(int i=0;i<sizeA;i++)
	{
		for(int j=position;j<sizeB;j++)
		{
			if(A[i]<B[j])
			{
				C.Add(A[i]);
				position=j;
```

```
                        j=sizeB-1;
                  }
                  else if(A[i]==B[j])
                  {
                        C.Add(A[i]);
                        position=j+1;
                        j=sizeB-1;
                  }
                  else
                        C.Add(B[j]);

            }
      }
      if(A[sizeA-1]>B[sizeB-1])
            C.Add(A[sizeA-1]);
      else if(A[sizeA-1]<B[sizeB-1])
            C.Add(B[sizeB-1]);

      for(int p=0;p<C.GetSize();p++)
      {
            int temp=C[p];
      }
}

void CCDSView::OnOperationsRead()
{
      m_Data.ReadNodesFromFile();
    operations=0;
      this->InvalidateRgn(NULL);
}

void CCDSView::DFS(int start, int &count)
{
      m_Data.m_Nodes[start].m_colour=1;
      count++;
      if(count==m_Data.m_Nodes.GetSize())
            return;
      int
size=m_Data.m_Nodes[start].m_ConnectedVertexIndices.GetSize();
      for(int i=0;i<size;i++)
      {
            int j=m_Data.m_Nodes[start].m_ConnectedVertexIndices[i];
            if(m_Data.m_Nodes[j].m_colour==0)
                  DFS(j,count);
      }
}

CView * CCDSView::GetFormCommandView()
{
      CCDSApp *pApp = (CCDSApp *)AfxGetApp();
      CMainFrame *pMainFrame = (CMainFrame *)pApp->m_pMainWnd;
      CView *pView = (CView *)pMainFrame->m_wndSplitter.GetPane(0,0);
      return pView;
 }
```

Appendix – B

Sample Input file: This file has 80 nodes.

```
135 556
285 239
508 340
321 456
383 2
270 218
426 227
525 299
548 274
25 232
186 215
107 39
102 410
473 381
477 84
209 207
576 239
248 208
130 154
426 352
526 501
122 419
567 546
421 196
460 47
8 471
422 525
113 264
350 317
535 447
138 301
244 192
512 281
203 593
336 527
315 105
472 473
109 340
562 53
506 274
591 483
51 394
210 366
363 244
480 133
400 105
414 499
263 445
178 129
160 311
322 422
```

253 266
516 383
191 255
344 301
469 413
248 210
211 578
367 434
118 541
159 407
375 387
158 65
453 558
350 599
225 398
129 468
302 206
296 263
421 155
88 312
393 514
480 430
41 120
157 95
237 534
199 56
258 244
269 56
193 140