ROUTING IN SENSING - COVERED AD HOC NETWORKS

Tarek El Salti

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

August 2007

Your file Votre référence
ISBN: 978-0-494-34435-4
Our file Notre référence
ISBN: 978-0-494-34435-4

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Tarek El Salti**

Entitled: **Routing in Sensing - Covered Ad Hoc networks**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair

_____ Examiner

_____ Examiner

_____ Examiner

_____ Supervisor

Approved _____

Chair of Department or Graduate Program Director

_____ 20 _____ _____

Dr. Nabil Esmail, Dean

Faculty of Engineering and Computer Science

Abstract

Routing in Sensing - Covered Ad Hoc networks

Tarek El Salti

We introduce a new approach in *ad hoc* networks for obtaining a sensing-covered network in the 2-D environment. This approach is based on a grid technique and the unit disk graph (UDG). This technique makes sure that a region is fully covered by a small number of sensor nodes. To experiment with variations of the generated graph, we introduce a new subgraph of the UDG. This graph is a generalization of the *Yao* graph in 2-D environment where the cones used are adaptively centered on a set of nearest neighbors for each node, thus creating a directed or undirected spanning subgraph. We also permit the apex of the cones to be positioned anywhere along the line segment between the node and its nearest neighbor, leading to a class of *Yao*-type subgraphs. We give an extension of the $DAAY$ (both directed and undirected) from the 2-D environment to the 3-D environment. For routing on such a 2-D sensing-covered sensor network topology, we propose new routing protocols. Some of these new routing algorithms include hybrid routing algorithms which are based on the $BVGF$ routing protocol of Xing *et al.* (2006) and our own protocols. The reason for these combinations is that we show that the $BVGF$ routing protocol does not guarantee delivery all the time on general sensing covered networks. Our hybrid algorithms do guarantee delivery on such networks. We demonstrate through simulations that our proposed routing protocols perform much faster compared to some existing routing protocols. We also compare the routing protocols based on the path lengths (hops or Euclidean distances). Our routing protocols show good performance in terms of these metrics.

Acknowledgments

At first, I would like to express my appreciation to Professor Thomas Fevens for his invaluable advice and guidance during my master research. As my supervisor, he has constantly forced me to remain focused on achieving my goal. His observation and comments helped me to establish the overall direction of the research and to move forward with investigation in the depth.

Second, I thank A.Abdallah and L.Harutyunyan for their cooperation concerning the literature of DAAY subgraphs since it was the base of our work presented in Chapter 3.

Finally, I thank my parents, my sister, my two brothers, and all my friends. They have been an inspiration throughout my life. They have always supported my dreams and aspirations. I'd like to thank them for all they are, and all they have done for me.

Contents

List of Figures

Chapter 1

Introduction

1.1 Wireless Ad hoc Networks

An *ad hoc* network [MWH01] is a system of wireless autonomous hosts that can communicate with each other without having any fixed infrastructure. Thus, it is one type of computer network. Each host in this network can communicate with all other hosts within its transmission range [BFNO03, BCSW98], which we will assume to be a fixed range R_C for all hosts. The hosts in an ad hoc network can collaborate so that data can be transported. These hosts are considered as endsystems (receive, send and process data) and as routers (data forwarding). The decision of forwarding packets depends on the network connectivity. This is in contrast to older traditional network technologies where there are special nodes like routers, firewalls, hubs, and switches that are responsible for forwarding the packets. Since ad hoc networks do not need specific configurations and can be deployed quickly, ad hoc networks are suitable in remote or hostile environments. For example, this technology can be used in a salvational mission when a natural disaster occurs. Wireless Ad Hoc networks have three types: Wireless Mobile ad hoc networks (referred to as MANET), Wireless Sensor networks (referred to as WSN), and Wireless Mesh networks. We will focus on the first two types since our work primarily relates to them.

A wireless mobile ad hoc network contains either static or mobile hosts. The communication between these hosts is over wireless links without the use of a static

infrastructure. There will be hosts that are not within the transmission range of each other. Thus, multihop routing will be used where intermediate hosts are used to transmit the message.

A wireless sensor networks is an integration of processing, sensing and wireless communication. Even though wireless sensor networks have many similarities with the usual ad hoc networks, their sensing applications need additional requirements to be included. For example, many applications (e.g., distributed detection [Var96]) present a sensing coverage requirement that is not included in the usual ad hoc networks. What we mean by a sensing-covered network is that every single point in the geographic area is covered by the sensing range of at least one sensor node. The sensor nodes are used to monitor conditions, e.g., environmental conditions (temperature, pollutants, etc.).

1.2 Routing in MANETs and WSNs

In mobile ad hoc and sensor networks, two nodes, u and v, can communicate with each other if they are within their transmission range. In this case there is no need for a routing protocol, otherwise a routing protocol is used. This routing protocol uses the intermediate nodes (between u and v) to forward the packets hop by hop until reaching, if possible, the node v. This is referred to as a multihop routing protocol. MANETs and WSNs work with a lower bandwidth than wired networks. If the information collected by such traditional routing algorithms (link-state routing protocol [JMQ+01] and distance vector routing protocol [PB94]) to generate a routing table, this would be impractical and expensive. This is because these protocols need the global topology information to be available at each node. Therefore if the nodes are mobile, then all the nodes using these protocols have to update their routing tables whenever the topology changes. This increases network congestion and may shorten the network lifetime since the nodes' energies are being used for every route discovery. Therefore improving a routing protocol is a challenging task in MANETs and WSNs, and is essential for fundamental network operation. Also, the underlying networks

use channels that provide lower frequency than the wired networks. Thus the routing protocols here must be efficient in using the network bandwidth. In other words, these protocols must consume the minimum amount of overhead while computing the routes. Another issue to be mentioned here is the battery power limitation. The routing protocols must be power efficient in order to prolong the lifetime of the network (MANET and WSN).

Routing research in MANETs and WSNs is a huge field. We focus on the unicast approach to delivering a packet between a single source and a single destination. Several unicast routing protocols have been presented based on 2-D graphs during the past few years [GSB03]. Most of these protocols assume that the network is static and models the network as a unit disk graph (UDG) in the 2-D environment.

One of the popular classes of routing protocols is the geographic routing protocol (position-based routing protocols). In these protocols, a node forwards a packet based on the positions of itself, its neighbors, and the destination [GSB03]. The positions of the nodes can be acquired by GPS system or any other technique. Various versions of the *Greedy* algorithm [Fin87, TK84, KSU99] are good examples for position-based routing protocols.

1.3 Contribution of Thesis

We introduce a new approach for obtaining a static sensing-covered network in the 2-D environment. This approach is based on a grid technique and the UDG. This technique makes sure that a region is fully covered by a small number of sensor nodes, thus creating a fully connected network and using minimal energy. This energy saving would prolong the lifetime of the network. To experiment with variations of the generated graph, we introduce a new subgraph of UDG. This subgraph is referred to as $DAAY$ (Displaced Apex Adaptive Yao). It is a generalization of the *Yao* graph in 2-D environment where the cones used are adaptively centered on a set of nearest neighbors for each node, thus creating a directed or undirected spanning subgraph. We also permit the apex of the cones to be positioned anywhere along the line segment

between the node and its nearest neighbor, leading to a class of *Yao*-type subgraphs. Unlike the *Yao* graph [Yao82], the $DAAY$ graph is orientation-invariant and it can be tuned (various cone angles, continuous in value, may be used). We present only one version of $DAAY$ in the sensor network. In this version, the cone apex is at the current node since moving the cone apex needs further investigation. Even though $DAAY$ reduced connectivity in sensor network under various cone angles, the network is still strongly connected and has a small number of edges. We give an extension of the $DAAY$ from 2-D environment (both directed and undirected) to 3-D environment. We do not use the 3-D $DAAY$ in our sensor network since it needs further investigation (e.g., what is the notion of coverage in 3-D?) and the grid technique should be modified in order to work in a 3-D environment.

We show that the Bounded Voronoi Greedy Forwarding ($BVGF$) routing protocol [XLPH06] does not always achieve 100% delivery on a general sensing covered network. This is because, even though the current node in the $BVGF$ routing protocol has neighbors with positive progresses, the Voronoi regions of these neighbors do not always intersect the line joining the source and the destination. This leads to a failure in packet delivery.

Using the above sensor network topology ($DAAY$ combined with the grid technique), we propose new routing protocols and compare them to *Greedy Forwarding* (GF) routing protocol. All these routing protocols achieve 100% delivery rate. Among these new routing algorithms, two new hybrid versions of $BVGF$ (the original $BVGF$ routing algorithm was mentioned by Xing *et al.* [XLPH06]) and two new hybrid versions of our Sensing Circles Close to the Line (SCL) routing protocols are proposed. The first version is a combination between $BVGF$ and the Smallest Angle To The Line routing protocol (SAL) which is referred to as $BVGF$:SAL routing protocol (the same applies to SCL:SAL). The second version is a combination between $BVGF$ and the GreedyClose2 ($GC2$) which is referred to as $BVGF$:$GC2$ routing protocol (the same applies to SCL:$GC2$). These hybrid versions are proposed since $BVGF$ and SCL routing algorithms do not guarantee delivery all the time. After introducing these versions, we compare them to the other proposed routing algorithms and the

GF routing algorithm. We show that, using the undirected and directed versions of *DAAY*, our new routing protocols and *GF* routing protocol perform much faster in terms of execution time if the angle of the cone is larger.

We also compare our routing protocols to the *GF* routing protocol in terms of the path length (number of hops or Euclidean distances). In terms of number of hops, we find that even though the *GF* routing protocol outperforms our routing protocols, the *SCL:GC2*, *SCL:SAL*, and *GC2* routing protocols still outperform the *BVGF:SAL* and *BVGF:GC2* routing protocols. In terms of Euclidean distances, the two proposed versions of the *BVGF* routing protocol outperform the *SCL:GC2*, *SCL:SAL*, and *GC2* routing protocols, but when θ in the *DAAY* graph increases, all of them become very similar in their performance. Using the same metric, our routing protocols outperform the *GF* routing protocol. We show that *SCL:GC2*, *SCL:SAL*, *GC2*, and *GF* routing protocols are almost the same in their time performance but all of them are much faster than the *BVGF:SAL* and *BVGF:GC2* routing protocols. We also show the difference in the performance between the mentioned routing algorithms on the directed Displaced Apex Adaptive Yao graph and the same routing protocols on the undirected Displaced Apex Adaptive Yao graph.

1.4 Thesis Organization

The rest of this Thesis is organized as follows. In Chapter 2, we explain some geometric concepts and introduce some existing subgraphs based on *UDG* graph in the 2-*D* environment and briefly mention some of them in 3-*D* environment. We also introduce some existing position-based routing protocols. In Chapter 3, we introduce both directed and undirected versions of DAAY in 2-*D* environment and their extensions in 3-*D* environment, and demonstrate some of their properties experimentally. In Chapter 4, we show a new approach for achieving a sensing-covered network with a small number of nodes. In Chapter 5, we introduce new routing protocols which are based on the position-based approach and compare them to *GF* routing protocol and each other. In Chapter 6, we conclude this Thesis and list our interests for further

research.

Chapter 2

Background

As mentioned in Sec. 1.3, we present new results related to graphs and routing protocols. In order to demonstrate their properties clearly later on, we illustrate in this Chapter some related works concerning geometric graphs and routing protocols.

In Sec. 2.1, we give a quick review of some subgraphs and elaborate on their related geometric concepts. The description includes subgraphs in 2-D and some in 3-D. We also demonstrate some relevant work concerning coverage and connectivity in sensor networks. In Sec. 2.2, we present some existing position-based routing algorithms in the 2-D environment and briefly mention some in the 3-D environment.

2.1 Geometric Graphs

A geometric graph is a graph embedded in a d-dimensional Euclidean space such that its vertices are points with coordinates and its edges are straight-line segments. We are particularly interested in 2-D and 3-D geometric graphs.

The set of n wireless hosts can be represented as a point set S in the Euclidean two-dimensional plane $\mathbb{R}^2$, each point possessing a geometric location. We will assume that S is static. On S, a (Euclidean) graph can be modeled as a weighted (undirected or directed) graph $G(S, E)$ where E is a subset of the pairs of nodes of S and the weight of an edge uv between nodes u and v is the Euclidean distance between the nodes which we denote as $|uv|$. In 2-D, the Euclidean distance is $|uv|=\sqrt{(u_x - v_x)^2 + (u_y - v_y)^2}$.

In 3-D, the Euclidean distance is $|uv|=\sqrt{(u_x - v_x)^2 + (u_y - v_y)^2 + (u_z - v_z)^2}$.

The weight of a graph is the sum of its edge weights. Between any pair of nodes, there are two types of path lengths. One refers to the Euclidean length which is the sum of all the hops' Euclidean distances in a path. The other one refers to the network length which is the hop count of a path.

In our wireless host model, two nodes are connected by an undirected edge if the Euclidean distance between them is at most R_c, the transmission range of the nodes. The resulting graph is called a unit disk graph. For node u, we denote the set of its neighbors by $N(u)$. The number of the neighbors of u is the degree of u.

A unit disk graph (UDG) is a common geometric graph to represent ad hoc networks. The UDG is considered poor for some routing protocols for various reasons. The 2-D UDG graph is typically a non-planar graph which means that it has crossing edges. Also since the UDG is a dense graph, it can have a high average node degree, the routing protocols performed on it will take considerable time.

2.1.1 Quality Measurements of Graphs

Following the notation of Xing *et al.* [XLPH06], let $L_G(u,v)$ refer to the shortest network length between any two nodes, u and v, in the graph $G(V,E)$. Thus, if we have a subgraph $Z(V,E')$, $E' \subseteq E$, that is a network t-spanner of graph $G(V,E)$ if $\forall u,\ v \in V,\ L_Z(u,v) \leq t \cdot L_G(u,v)$. The term t refers to the network stretch factor [Epp96]. Assuming also a graph $EU_G(u,v)$ which refers to the shortest Euclidean length between any two nodes, u and v, in the graph $G(V,E)$. Thus, if we have a subgraph $Z(V,E')$, $E' \subseteq E$, that is a Euclidean a-spanner of graph $G(V,E)$ if $\forall\ u,v \in V,\ EU_Z(u,v) \leq a \cdot EU_G(u,v)$. The term a refers to the Euclidean stretch factor [Epp96]. Spanner graphs have been heavily studied in computational geometry [Epp00] and mainly provide us with two things: short paths and energy optimal paths.

We will define our stretch factors as follows. Let $S_G(u,v)$ refers to the shortest network length between any two nodes, u and v, in the graph $G(V,E)$. Assume also that we have a subgraph $Z(V,E')$, $E' \subseteq E$.

Definition 1 *The hop number stretch factor is defined by:*

$$D_h = max_{u,v \in V} \frac{S_Z(u,v)}{S_G(u,v)}$$

Now assume a graph $EU_G(u,v)$ which refers to the shortest Euclidean length between any two nodes, u and v, in the graph $G(V,E)$. Assuming also that we have a subgraph $Z(V,E')$, $E' \subseteq E$.

Definition 2 *The Euclidean stretch factor is defined by:*

$$D_e = max_{u,v \in V} \frac{EU_Z(u,v)}{EU_G(u,v)}$$

The stretch factor in general is represented by the dilation with regard to an ultimate wireless network. A network is referred to as an ultimate network if it has a path with network length $\left\lceil \frac{|uv|}{R_C} \right\rceil$ and a path with Euclidean distance $|uv|$ for any pair of nodes u and v. We adopt two definitions from [XLPH06]. See definitions 3 and 4 below:

Definition 3 *The network dilation is defined by:*

$$D_n = max_{u,v \in V} \frac{L_G(u,v)}{\left\lceil \frac{|uv|}{R_C} \right\rceil}$$

Definition 4 *The Euclidean dilation is defined by:*

$$D_e = max_{u,v \in V} \frac{EU_G(u,v)}{|uv|}$$

You can see from the above definitions that the dilation is the upper bound of the stretch factor with regard to any potential wireless network that has the same set of nodes.

Our simulations depend on the above two dilation definitions. Even though the definitions here are for gauging the quality of graphs, we will also use them later in Sec. 2.2.3 while discussing routing algorithms. This is because our simulations use them to gauge the performance of the routing algorithms.

For a geometric graph G, a Euclidean Minimum Spanning Tree $EMST(G)$ is a minimum weight spanning tree of G. There are some subgraphs that contain the

EMST(G) as a subgraph. Another desired property of subgraphs is to be strongly connected which means that each node u in G can communicate with every other node in G. Also subgraphs aim to be t-spanners, and orientation-invariant (which refers to the fact that if G is rotated by an arbitrary angle to give G' then the resulting subgraph Z' of G' is a rotation of the subgraph Z of G) and to have bounded out-degree.

2.1.2 Geometric Subgraphs

In general, the main objective of the subgraphs is to resolve some *UDG*'s issues such as high node degree and non-planarity. Some of these subgraphs are described in the following subsections in 2-D environment and some of them are briefly mentioned in 3-D environment.

Voronoi Diagram and Delaunay Triangulation

Assume V is the set of nodes in the 2-D environment. The Voronoi diagram[For92] is acquired by partitioning the plane into k Voronoi regions for all nodes in V. The Voronoi diagram is not necessarily based on *UDG*. Each node in V has one Voronoi region. A Voronoi region for a node $f \in V$ is termed $Vor(f)$. A point z lies in $Vor(f)$ if and only if the node f has the shortest Euclidean distance to z. The boundary between two adjacent Voronoi regions is called a Voronoi edge. A Voronoi edge is perpendicular to the segment connecting the two adjacent nodes. When Voronoi edges intersect, we will have a Voronoi vertex p. See Fig. 1.

The dual graph of a Voronoi diagram is the Delaunay Triangulation [For92] which is denoted by *DT*(V). The Delaunay Triangulation has an edge which is not related to *UDG* between nodes u and v if and only if the Voronoi regions for these two nodes share the same Voronoi edge. The *DT*(V) is a planar graph. The stretch factor of a Delaunay Triangulation is $\frac{4\pi\sqrt{3}}{9}$ as determined by Keil and Gutwin [KG92].

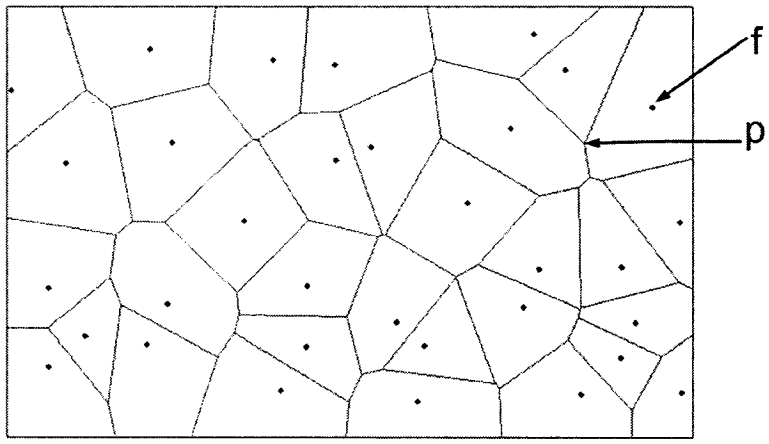

Figure 1: Voronoi diagram.

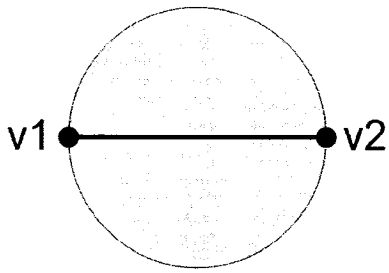

Figure 2: An edge u,v in GG

Gabriel Graph

A 2-D Gabriel graph [GS69] [MS80] (GG) is a planar subgraph of a $UDG(V)$, contains minimum energy paths (where a minimum energy path allows messages to be sent with the minimum energy usage) and is connected. An edge binding two vertices, v_1 and $v_2 \in$ is in GG if the minimal diameter circle that circumscribes v_1 and v_2, whose diameter is the line segment that has both v_1 and v_2 as its endpoints, has no other vertex of V. See Fig. 2 and Fig. 3 respectively. In other words, there exists an edge (v_1,v_2) in $GG(V)$ if there is no $w \in V$ where $|v_1w|^2+|v_2w|^2 < |v_1v_2|^2$. The 3-$D$ Gabriel graph would have the same definition as the 2-D version [KFO05]. Let S(p, R_c) be the sphere with center point p with a radius R_c and let an edge (u, v) have the midpoint q. This edge would be in GG if another node w does not exist which has an edge (u, w) and an edge (v, w) in the sphere $S(q, \frac{|uv|}{2})$.

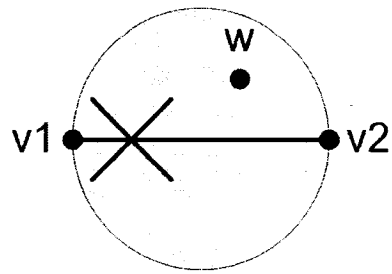

Figure 3: An edge u and v is not in GG

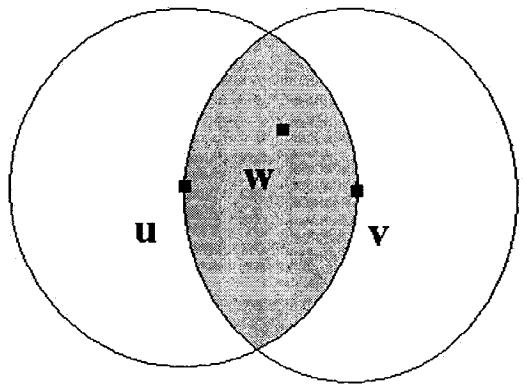

Figure 4: The edge (u,v) is not in $RNG(G)$.

Relative Neighborhood Graph

The 2-D Relative Neighborhood Graph (RNG) of $UDG(V)$ [Tou80] is a planar subgraph of a set of points V. An edge binding two vertices, u and $v \in V$, is in RNG if the intersection of the two disks centered on two vertices with radii $|uv|$ does not contain any other vertex $w \in V$. See Fig. 4. Bose *et al.* [BDEK01] showed that the stretch factor of RNG is at most n-1. The 3-D RNG [KFO05] would have the same definition as the 2-D version. The edge (u, v) would be in RNG if another node w does not exist which has an edge (u, w) and an edge (v, w) inside the lens region formed by the intersection of two spheres $S_u(u, |uv|)$ and $S_v(v, |uv|)$.

In other words, you can see from Fig. 4 that there should not be a vertex w in the gray area, defined by the intersection between the disks for the two vertices, u and v.

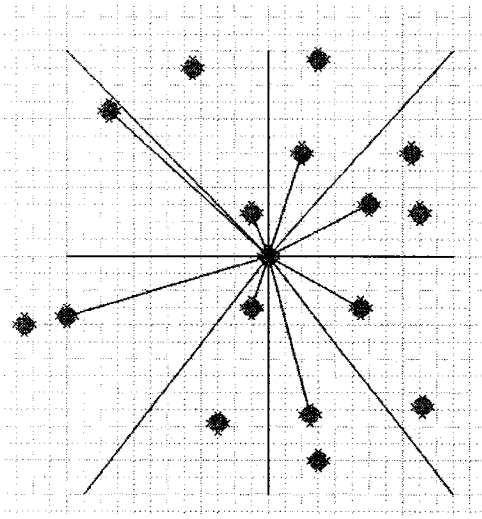

Figure 5: Yao graph for one node located in the center with k=8,

YAO Graph

The *YAO* graph [Yao82] is constructed by partitioning the space around each node into cones with equal angles. Each node connects itself to the nearest neighbor in each of these cones. See Fig. 5.

In other words, for a geometric graph G, a *YAO* Graph (also called a *Theta* Graph [BGM04]) $YG_k(G)$ with an integer parameter $k \geq 6$ is defined as follows. First, we will define a directed *Yao* graph, *D-*$YG_k(G)$, for G. At each node u in G, k equally-separated rays originating at u define k cones. In each cone, only the directed edge (u, v) to the nearest neighbor v, if any, is part of *D-*$YG_k(G)$. Ties are broken arbitrarily. Let $YG_k(G)$ be the undirected graph obtained if the direction of each edge in *D-*$YG_k(G)$ is ignored, yielding a subgraph which may have crossing edges if G=*UDG*. The graph $YG_k(G)$ is a $1/(1–2\sin(\omega/k))$-spanner of G [LWW02], has an out-degree of at most k, and contains the *EMST*(G) as a subgraph [Yao82]. One drawback of the $YG_k(G)$ graph is that it is not orientation-invariant. That is, if the G is rotated by an arbitrary angle to give G' then the resulting $YG_k(G')$ subgraph is not necessarily a rotation of $YG_k(G)$.

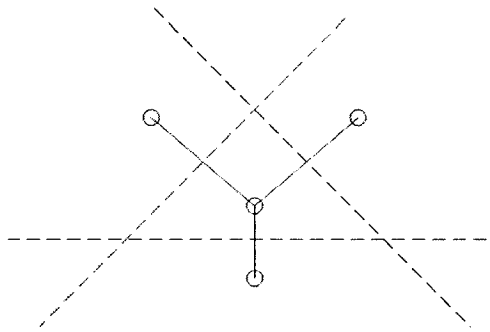

Figure 6: *HSP* graph

Half Space Proximal Graph (*HSP*)

For a geometric graph G, *HSP*(G) is defined as follows [CDK+05]. As with the *YAO* Graph, first a directed *D-HSP*(G) is defined. At each node u in G, the following iterative procedure is performed until all the neighbors of u are either discarded or are connected with an edge. See Fig. 6. A directed edge $[u, v]$ is formed with the nearest neighbor v. An open half plane is defined by a line perpendicular to $[u, v]$, intersecting $[u, v]$ at its midway point, and containing v. All the nodes in this half plane are then discarded. The procedure then continues with the next nearest non-discarded neighbor and so on until all the nodes have been discarded. The selected directed edges determine the *D-HSP*(G). The undirected *HSP*(G) is obtained by ignoring the direction of the edges, yielding a subgraph that may still have crossing edges. Among the properties shown in [CDK+05] for the *HSP* subgraph that it is strongly connected, has an out-degree of at most six, contains the *EMST*(G) as its subgraph, and is orientation-invariant. Bose *et al.* [BCC+07] show that *HSP* has a stretch factor of at least $3 - \epsilon$. One drawback of the *HSP*(G) graph is that, since the forbidden region is always defined by a straight line, there is no control over the degree of a node.

2.1.3 Issues Concerned In Sensor Networks

For sensor networks, in terms of graphical representations, using the subgraphs mentioned above is not enough since other issues arise. Because of the impact of the

distribution of sensor nodes in a region on performance, additional issues concerning sensor networks are required and they are the following: coverage, connectivity and power usage issues. Coverage refers to the fact that every single point in a region is at least covered by the sensing range of one node. A sensor node covers a point in a region by its sensing range (sensing radius) that keeps track of the surrounding area. Connectivity refers to the fact that any node can reach any other node. After dealing with these two issues, the power saving issue comes into consideration. It refers to the procedure for turning off redundant nodes while keeping the area covered. This is considered as an efficient way to conserve the energy and prolong the lifetime of the sensor network while maintaining sufficient coverage of the field.

There is some previous work proposed to create networks to solve the issues mentioned above. As far as we know, most these works are based on the decimation approach. In other words, the previous methods deal with the fact that the network is already dense such that every single point is covered by at least one sensor node. Our method solves the above issues as well and is based on an incrementation approach. In our approach, we place one sensor node at a time at the region, until we make sure that every single point in the region is covered by at least one sensor node, reducing the number of required nodes.

The work by Meguerdichian *et al.* [MKPS01] proposes Surveillance which is a sensor coverage metric that is used as a quality of service measurement provided by a particular sensor network, and centralized optimum algorithms are presented to assess the paths that are best and least monitored. Another work by Meguerdichian *et al.* [MKQP01] deals of how a moving object can be monitored along a path with a random velocity in a sensor network.

Xing *et al.* [XWZ+05] mentioned an efficient way that determines whether the network is being covered or not by the sensor nodes. They use the following algorithm:

1. For each sensor s_j of s_i such that $|s_i,s_j| \leq 2R_s$ (where $|s_i,s_j|$ is the distance between a node s_i and its neighbor s_j and R_s is the sensing radius) determine the arch angle of s_i, denoted by $[\alpha_j f,\alpha_j g]$, that is perimeter covered by s_j.

2. For all the neighbors s_j of s_i, such that $|s_i,s_j| < 2R_s$, place all the points $\alpha_j f$ (left) and $\alpha_j g$ (right) on the line segment [0,2π] and sort all these points in an ascending order into a list L. Also properly mark each point as a left or right boundary of a coverage range.

3. Traverse the line segment [0,2π] by visiting each element in the sorted list L from left to right and determine the perimeter-coverage of s_i.

Huang *et al.* [HTW07] deal with the fact that the relationship between R_c and R_s is arbitrary and they could still have a sensor network that is fully covered and fully connected. They mentioned that if the network is not covered, it may be disconnected. The coverage issue is based on the work presented by Huang and Tseng [HT03]. The work by Zhang and Hou [ZH04] focuses on the coverage, connectivity, and power saving issues. Their work makes sure that the network is fully covered, fully connected and has the least number of sensor nodes. The authors assumed that the network is a dense network such that any sensor node can be found at any desirable point. They solve the connectivity issue by presenting a fact that if a finite number of sensor nodes exist in a finite area, then the condition $R_c \geq 2xR_s$ is necessary to ensure that coverage implies connectivity. The problem is reduced to the coverage problem. For this, the authors proposed a completely localized density control algorithm, called OGDC (Optimal Geographical Density Control Algorithm). The algorithm assumes that all the nodes have 'undecided' states. The algorithm works as follows. See Fig. 7. The time is divided into rounds. In each round, a random starting node is chosen. Let us say this node is A. Then, one of its neighbors is chosen with an approximate distance $\sqrt{3}R_s$. Let us say this node is B. In order to cover the crossing point O, the node whose position is the closest to the optimal position C is then chosen. In this case node P is chosen. The process continues until making sure that all the nodes switch their states to 'on' or 'off' states. When a node switches to the 'off' state, this means that redundant nodes are turned off while ensuring that the network is still fully covered. The protocol outperforms other protocols like: the PEAS algorithm [YZLZ03], the hexagon-based GAF-like algorithm [YZLZ02], and the sponsor area

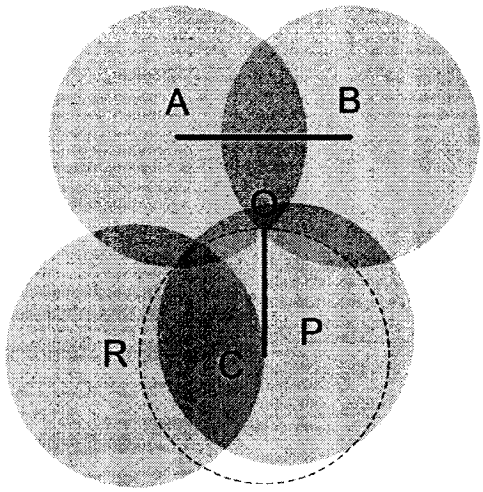

Figure 7: Node p is picked since it is the closest to the optimal point C

algorithm [TG02] with respect to the number of working nodes needed.

The work by Xing *et al.* [XWZ+05] presents a coverage configuration protocol (*CCP*). The *CCP* protocol is constructed of a coverage eligibility algorithm that determines if a node is eligible to become active or not. This algorithm works as follows. An intersection point occurs when two or more sensing circles intersect each other. A node is eligible to be an active node if all the intersection points inside the node's sensing circle are not covered by at least K_s sensor nodes. Another condition for eligibility is that if there are no intersection points inside the sensing circle of a node and there are K_s or more sensor nodes (other than this sensor node) that are located at the same position of this node, then this sensor node would be ineligible to become active. The reason is that CCP checks if every single point on the region is covered by at least K_s sensor nodes. The *CCP* protocol keeps a table of the sensing neighbors using beacon messages (HELLO messages) that are received from these neighbors. A HELLO message from a node contains its position. Sensor nodes can communicate with each other if they are within each other's the communication range. Xing *et al.* assume two cases for the communication range: 1)$R_c \geq 2R_s$ and 2)$R_c < 2R_s$. In this Thesis, we will use the first assumption, since the second one is based on the assumption that instead of using sensing circles of the nodes, irregular sensing shapes are used. In our Thesis, these irregular shapes are not presented due

to their complexities. Therefore, the *CCP* protocol deals with both connectivity and coverage problems and ensures that only necessary nodes are being used in the network to ensure energy conservation.

As mentioned above, all these works are based on the assumption that the network used is a dense network. A dense network means that the nodes are already placed at the field. In our view, using a dense network in advance made the correctness of the previous coverage techniques easily proven. These techniques appear as if they are checking if the network is covered or not rather than making a covered network. Another issue is that a dense network would mean that a large variable number of nodes have to be deployed. This would make any coverage technique take sometimes a very long time. Also, simply having dense networks do not necessarily mean that every single point in the region is actually fully covered. Our work proposes a grid technique as mentioned in Sec. 1.3 which is based on the incrementation approach. This technique is used to fill out the field and uses the least number of sensor nodes while having a fully connected sensor network based on the assumption $R_c \geq 2R_s$ [XWZ+05]. The power saving issue was addressed by our approach by only placing the necessary nodes at the field based on the grid calculations. The placement procedure is a random procedure meaning that each time we will have a random position for the new sensor node. Before placing a node, the sensor controller generates a random position for the new sensor node, then it checks if this prospective position is already covered by some previous sensor nodes. If not, then the sensor node will be placed at this position, otherwise the sensor controller will generate a new random position and checks again, and so on. The drawback of our approach and the previous approaches is that they are considered global approaches.

2.2 Routing Algorithms

After presenting geometric graphs to model our networks in Sec. 2.1, we can now present routing protocols based on geometric graphs. Many studies focused on routing algorithms in mobile ad hoc and sensor networks.

Routing algorithms can be divided into two main classes [MWH01]: position-based routing algorithm and topology-based routing algorithm. The former makes a decision based on the local position information available. The latter makes a decision based on a routing table using the global routing information.

Topology-based routing algorithms can be classified into 3 classes: proactive, reactive, and hybrid. Proactive techniques use one of the two routing methods which are distance-vector routing [PB94] or link-state routing [JMQ+01]. They maintain the information about all the available paths (including the unused paths). The shortcoming here is that each node has to maintain a large routing table for the whole network. Another shortcoming is that periodic dissemination of routing information is required by proactive techniques so that all nodes can calculate routes to other nodes. As opposed to proactive techniques, reactive techniques [PC97] are known as on-demand route acquisition systems. A node sends a route request (RREQ) whenever it wants to send a message to another node for which this route does not already exist. Reactive techniques generate less traffic in the network. Thus they are scalable techniques. Therefore they are appropriate for dynamic ad hoc networks. Hybrid techniques [HP98] combine the proactive and reactive strategies. Even though hybrid and reactive schemes are much more efficient and scalable, they may have to perform a routing discovery prior sending one packet, and to maintain a routing table for as many as all the nodes in the network. Therefore, these algorithms have limited tolerance to any topological change that may occur later on.

Giordano *et al.* [GSB03] distinguished several classes of position-based routing protocols. We mention two of them. The first class includes Basic Distance, Progress, and Direction Based Methods. In this type of class, a node A will forward the packet either based on the Euclidean distance to the destination, projected distance to the destination or the direction to the destination. Such protocols under this class are: GEDIR [Fin87], DIR [KSU99], and MFR [TK84]. The second class includes Partial Flooding and Multi-Path Based Path Strategies. In this type of class, a node Q sends a message to its neighbors whose direction is closer to the direction of the destination node $Q1$. To control flooding, flooding-based methods require from nodes to memorize

past traffic, so that the same message would not be forwarded more than once. Such protocols under this class are: DREAM [BCSW98], LAR [KV00], V-GEDIR [Sto99], and CH-MFR [Sto99]. Due to our research interest, we will only focus on the first class.

Position-based routing algorithms are locally distributed algorithms [SL01][BMSU01] [KK00][TK84]. In these routing algorithms, each node makes a decision to forward a packet to a specific neighbor based on the locations of this node, its neighbors, and the destination. This indicates that these routing algorithms do not need to gather the global topology information of the network as a whole. Thus, the bandwidth and limited storage resources can be more efficiently utilized. These characteristics also make the position-routing algorithms quick to adapt to network topology changes. We will consider only position-based routing algorithms in this thesis.

Several characteristics of position-based routing protocols are presented by Giordano *et al.* [GSB03]. The purpose of these characteristics is to evaluate the performance of these routing protocols. Some of these characteristics are mentioned in the following:

1.Guaranteed delivery: If a packet can be assured by a routing algorithm to be delivered to the appropriate destination in a connected graph, we say that this algorithm has a guaranteed delivery. This feature will be present in our routing algorithms discussed later.

2. Memorization: If a routing protocol does not need to memorize the past traffic, we say that this protocol is without memorization. Even though memorization is being used to resolve some issues such as avoiding routing loops, it has several serious drawbacks. First, each node has limited resources (memory, battery power) to maintain this extra information. Second, each node requires extra time to memorize the past traffic.

3. Loop Freedom: If a loop can be avoided by a routing protocol without memorizing the past traffic, we say that this protocol is a loop freedom routing protocol. The loop means that the packet is being trapped between nodes without being delivered.

2.2.1 Localized Progress-Based Routing Algorithms in Ad Hoc Networks

In the previous section, we briefly mentioned some of the classes of position-based routing protocols. Localized progress-based routing algorithms are under the class that includes Basic Distance, Progress, and Direction Based Methods. In these algorithms, all the nodes know their geometric positions and the geometric positions of their neighbors within the transmission range R_c. The source node in this algorithm also knows the position of the destination node.

If we have a source node S and a destination node D, S will pick one of its neighbors that have the most positive progress towards the destination. If S picks C, then C will repeat the same procedure till reaching, if possible, the node D. The main goal is to always forward the packet towards the destination. We refer to such protocols as progress-based routing protocols. We will discuss the versions of Greedy routing protocol as examples of progress-based routing protocols.

Even though these protocols can be used in both 2-D and 3-D environments, our work mainly deals with the 2-D version presented in Sec. 5.6.

One popular example of position-based routing algorithms is the *Greedy Forwarding* routing algorithm (denoted by *GF*). *GF* takes decisions based on n-hop neighbor information, where $n \geq 1$. This feature avoids the need of gathering and maintaining the global topology information for the whole network. In other words, the decision made by *GF* regarding forwarding the packets depends on the information of n-hops (neighbors) away, where n is constant.

GF has three types. One is based on the Euclidean distance to the destination [Fin87] which is referred to as geographic distance (GEDIR). See Fig. 8. In this figure, a current node (yellow) forwards the packet to the neighbor (orange) which is the closest one to the destination (red).

Another one is based on the projected distance to the destination (the projection is on the straight line joining the current node and the destination node) [TK84], see Fig. 9, which is referred to Most Forward within Radius (MFR). As you can

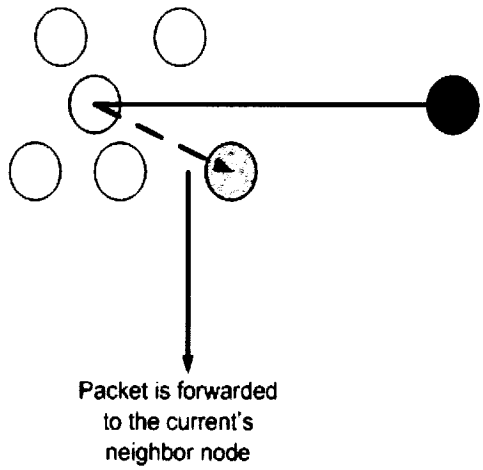

Figure 8: *GF* is based on the Euclidean distance to the destination (GEDIR)

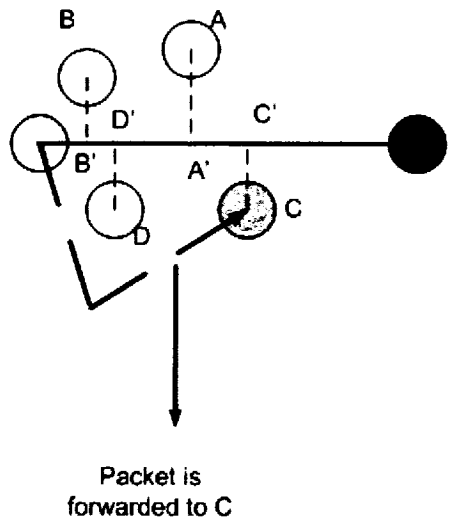

Figure 9: *GF* is based on the projected distance to the destination (MFR)

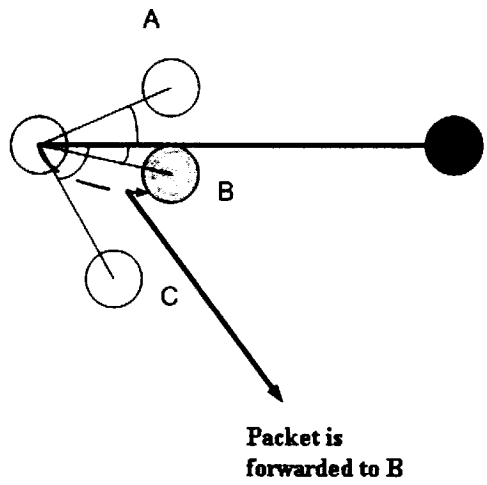

Figure 10: *GF* is based on the smallest angle (DIR)

see, each node has a projected point on the straight line joining the current node (yellow) and destination node (red). The node C has a projected point C' which is the closest among other projected points to the destination node, thus the packet will be forwarded to C.

The last one is based on the direction to the destination (this is measured by calculating the angle between the line joining the current node and destination node and the line joining the current node and the neighbor node [KSU99], see Fig. 10, which is referred to Compass routing (DIR). As you can see that node B (orange) has the smallest angle to the line that joins the current node and the destination node (red). Hence node B is the chosen one.

In general, progress-based routing protocols may suffer from local minima [BMSU01], that is, a current node has no other neighbors that make better progress than itself, or two adjacent neighbors that are equally close to the destination. Hence a loop can occur. Even if GEDIR protocol can stop forwarding a packet once detecting a loop, there will be a failure in delivering the packet.

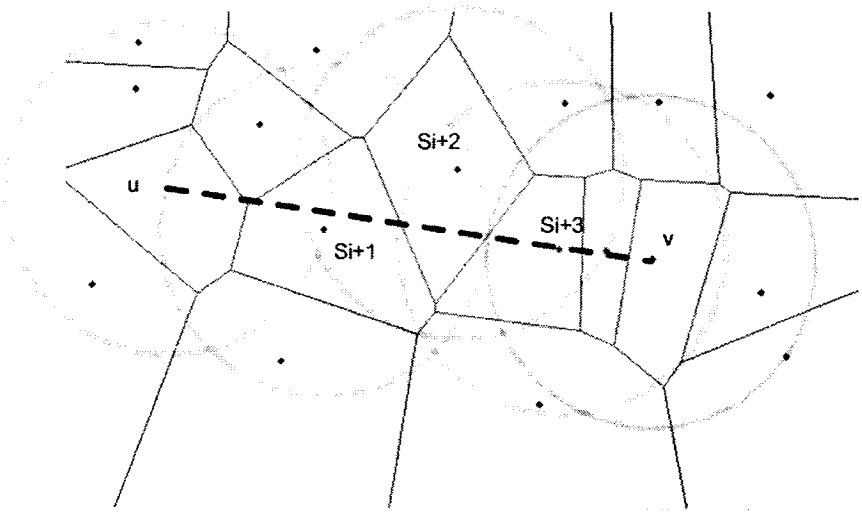

Figure 11: *BVGF* algorithm.

2.2.2 Greedy Forwarding Routing Protocol for Sensing-Covering Networks

The sensing-covered network is assumed to cover every single point in a region. Also some routing protocols require to have the Voronoi diagram in advance in sensor networks.

There are several routing protocols that have been studied in sensor networks. Some of these include: GEDIR, DIR, MFR (and their 2-hop version [SL01]) and Bounded Voronoi Greedy Forwarding (*BVGF*) [XLPH06] routing protocols.

Since we already gave a clear idea about the versions of *GF* routing protocol in Sec. 2.2.1, we would focus on the *BVGF* routing protocol. The *BVGF* routing algorithm works as follows: A current node checks if the Voronoi regions of its neighbors intersect the line segment joining the source and the destination. See Fig. 11. After having this set of nodes, the algorithm picks the one which is the closest one to the destination.

The algorithm continues like this until reaching the destination. This algorithm is based on a strongly connected graph. According to the Theorem 5 in [XLPH06], this algorithm guarantees that the packet is delivered to the destination. In this theorem, the authors assume that there is always a neighbor that has a positive progress and

whose Voronoi region intersects the line. The proof is associated with the theorem in [XLPH06].

2.2.3 Metrics For Routing Algorithms Performance

As mentioned in Sec. 2.1.1, the stretch factor is represented by the dilation with regards to an ultimate wireless network for measuring the quality of graphs. The presentation includes Euclidean and network dilations. Xing *et al.* [XLPH06] represented the stretch factor in terms of Euclidean distance and hop counts using routing algorithms. This representation is used to evaluate the performance of routing algorithms. Therefore, this new representation is illustrated in this section.

A routing algorithm's performance depends on two things. The former is the network length which is the hop count between any pair of nodes. The latter is the Euclidean length which is the accumulated Euclidean distance of each hop of the routing path. The shortest Euclidean and network paths may be different for the same pair of nodes. The subgraph's path quality can affect routing algorithm performance.

The network dilation is said to be $D_n(U)$. This means that the network dilation is based on U (routing algorithm) if $L_G(u, v)$ in definition 3 in Sec. 2.1.1 represents the network length of the routing path (from node u to node v) chosen by U. The network dilation evaluates the performance of the routing algorithm with regards to the ultimate wireless network. This network has a path with $\left\lceil \frac{|uv|}{R_c} \right\rceil$ hops between any two nodes u and v, where R_c is the connectivity radius. The same applies for the Euclidean dilation of U but using definition 4 in Sec. 2.1.1.

Chapter 3

Displaced Apex Adaptive Yao Graphs in 2-D and 3-D

To add a degree of variability to our sensor network simulations, we will use a generalization of the Yao graph. This generalization is based on the Yao graph, presented in Sec. 2.1.2.

In this Chapter, we demonstrate this Yao graph generalization in both 2-D and 3-D. The cones used here are adaptively centered on a set of nearest neighbors for each node, thus creating a directed or undirected spanning subgraph of a given unit disk graph (UDG). We also permit the apex of the cones to be positioned anywhere along the line segment between the node and its nearest neighbor, leading to a class of (Yao)-type subgraphs. We show that these locally constructed spanning subgraphs are orientation-invariant. Since a continuous set of cone angles are possible, these subgraphs also permit control over the degree of the graph. This new subgraph is referred to as the Displaced Apex Adaptive Yao graph ($DAAY$). We demonstrate through simulations that these subgraphs of the UDG combine the desirable properties of the Yao and the Half Space Proximal subgraphs of the UDG.

3.1 Displaced Apex Adaptive Yao in 2-D

In this section, we give a formal definition of a class of *Yao*-type graphs and prove some basic properties of the graphs. Let S be a set of N points in the Euclidean two dimensional plane, each point possessing a geometric location. For the following, define the cone angle θ to be the half-angle of the cone's apex.

We will use the parameter s to parametrize the closed line segment between u and v: $(1-s)u + sv$, $0 \leq s \leq 1$. Any particular choice of s represents the position of the apex of the cone. We will use a second parameter α, $0 \leq \alpha \leq 1$, to determine θ as a fraction of the maximum cone angle, $\theta_m(s, |uz|)$, which we define shortly, which is a function of s and the distance from the current node u to the nearest neighbor z for which the cone is determined. If the cone's half-angle θ is increased more than $\theta_m(s, |uz|)$, then the *DAAY* becomes disconnected.

Algorithm 1 Displaced Apex Adaptive Yao(G, α, s) graph algorithm

Input: *A graph G with the node set S, an angle parameter α, and a parameter s.*
Output: *A list of directed edges L for each node $u \in S$ which represent the Displaced Apex Adaptive Yao subgraph of G, DAAY(G, α, s).*
for all $u \in S$ **do**
  Create a list of neighbors of u: $LN(u) = N(u)$.
  repeat
    (a) Remove the nearest neighbor z node from $LN(u)$ and add the directed edge uz to L.
    (b) Determine $\theta_m(s, |uz|)$. Let $\theta = \alpha \cdot \theta_m(s, |uz|)$.
    (c) Let $r = (1-s)u + sv$ be a point on the line segment uz.
    (d) Consider the cone C with its apex at r with a cone angle θ and z in its interior, such that the line uz bisects the cone C into two equal halves (i.e., the segment uz lies in the center of the cone).
    (e) Scan the list $LN(u)$ and remove each node in the interior of C.
  until $LN(u)$ is empty
end for

Definition 5 *Let G be a UDG with node set S. The directed Displaced Apex Adaptive Yao subgraph, D-DAAY(G, α, s), is defined to be the graph with node set S whose edges are obtained by applying the Displaced Apex Adaptive Yao(G, α, s) algorithm, Algorithm 1, on the graph G using cone angle $\theta = \alpha \cdot \theta_m(s, |uz|)$ and apex displacement parameter s. The undirected graph DAAY(G, α, s) is obtained by ignoring the*

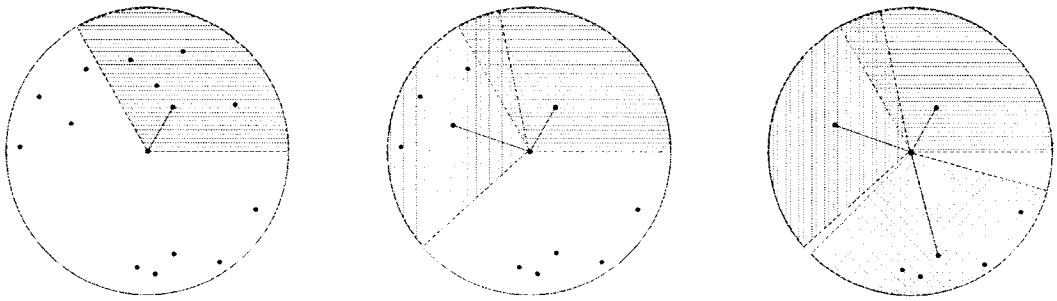

Figure 12: Applying the Displaced Apex Adaptive Yao($UDG, 1, 0$) graph algorithm on the node u of a UDG: (a) the nearest neighbor is first chosen; (b) the second nearest node out of the rest of the nodes is chosen. Note that its associated cone overlaps with the first cone; and (c) the third nearest neighbor is chosen from the list $LN(u)$.

direction of the edges in D-DAAY(G, α, s).

When $s = 0$, we simply refer to the resultant graph as the *Adaptive Yao* graph. Note that the directions of the cones used in the Displaced Apex Adaptive Yao(G, α, s) algorithm only depend on the relative directions of the selected nearest neighbors. Therefore, the resultant subgraph is the same regardless of the orientation of the point set S. Hence the $DAAY(G, \alpha, s)$ is orientation-invariant. See Fig. 12. The running time of Algorithm 1 per node is $O(n)$ where n is the number of nodes and results in a subgraph that may still have crossing edges. We show in [FAESH07] that $DAAY$ subgraphs are strongly connected, have bounded out-degree, are t-spanners with bounded stretch factor, contain the $EMST$ as a subgraph, and are orientation-invariant. We omit the details of the proofs of these properties for the DAAY subgraphs since the main focus of this thesis is on routing in sensing-coverage networks.

After determining some properties of cone angles, we define the maximum cone angle $\theta_m(s, |uz|)$.

Lemma 1 *Consider a node u and neighbor z of u. Consider an arbitrary point $k = (1-s)u+sz$ where the parameter s has a value in the range $[0, 1]$. Define L to be the line perpendicular to the line segment uz and that intersects uz at its midpoint m (corresponding to $s = 0.5$ in the above line equation). Define a cone with cone angle θ, with its apex at k, oriented such that z is in its interior and the segment uz lies in*

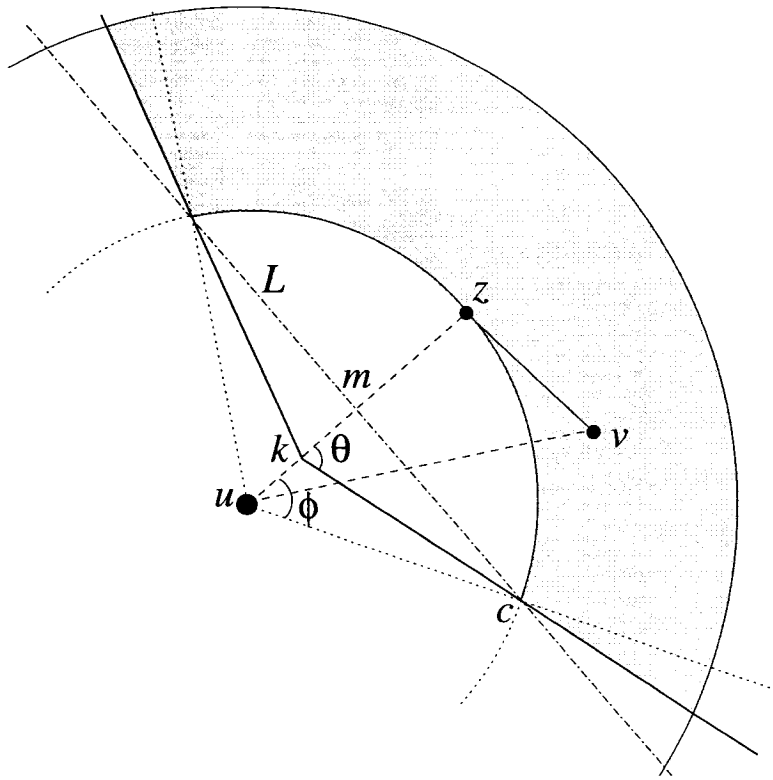

Figure 13: *DAAY* subgraph when s <0.5

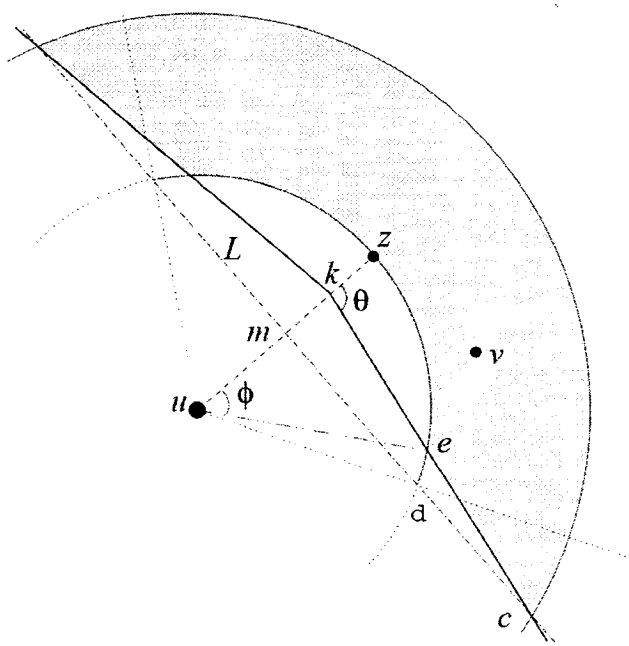

Figure 14: *DAAY* subgraph when s >0.5

the center of the cone. See Figs. 13 and 14, respectively. Consider the boundary of this cone intersecting the line L at a point c. Then the cone angle θ satisfies

$$\frac{\sin(\theta - \theta_0)}{\sin(\theta)} = \frac{s|uz|}{|uc|}, \qquad \textit{where } \cos(\theta_0) = \frac{1}{2}\frac{|uz|}{|uc|}.$$

Definition 6 *Using the same definitions as in Lemma 1, define the maximum cone angle $\theta_m(s, |uz|)$ as a function of the parameter s and the distance $|uz|$. It is as follows:*

$$\frac{\sin\left(\theta_m(s,|uz|) - \frac{\pi}{3}\right)}{\sin(\theta_m(s,|uz|))} = s \qquad \textit{if } 0 \le s < 0.5$$

$$\frac{\sin\left(\theta_m(s,|uz|) - \cos^{-1}\left(\frac{1}{2}\frac{|uz|}{r}\right)\right)}{\sin(\theta_m(s,|uz|))} = \frac{s|uz|}{r} \qquad \textit{if } 0.5 \le s \le 1$$

Note that when $0 \le s \le 0.5$, then $\theta_m(s, |uz|)$ is only a function of s such that θ is a fixed angle for fixed values of s and α. When $s = 0.5$, $\theta_m(0.5, |uz|) = \pi/2$ and, if $\alpha = 1$ such that $\theta = \theta_m(0.5, |uz|)$, we obtain the Half Space Proximal subgraph [CDK+05].

3.2 Simulation Results for *DAAY* in 2-D

In our experiments we use randomly chosen connected unit disk graphs on an area of 100 × 100. By connected we mean that the entire graph is a single connected component. We vary the number of nodes, N, between 65, 75, 85, 95, and 105 nodes. For all the results reported here, the results have been averaged over 25 graphs for each value of N. For all the graphs tested, the transmission radius R_c used is 15 units.

The stretch factors presented here are for experimentally measuring the quality of graphs and are the same as the definitions 1 and 2 mentioned in Sec. 2.1.1. In addition to the maximum stretch factors (Euclidean or hop number) over the 25 graphs, we also calculate the average stretch factors (Euclidean or hop number) in our simulations.

For each *UDG*, an Adaptive Yao subgraph (equivalent to a Displaced Apex Adaptive Yao subgraph with $s = 0$), Displaced Apex Adaptive Yao subgraphs with

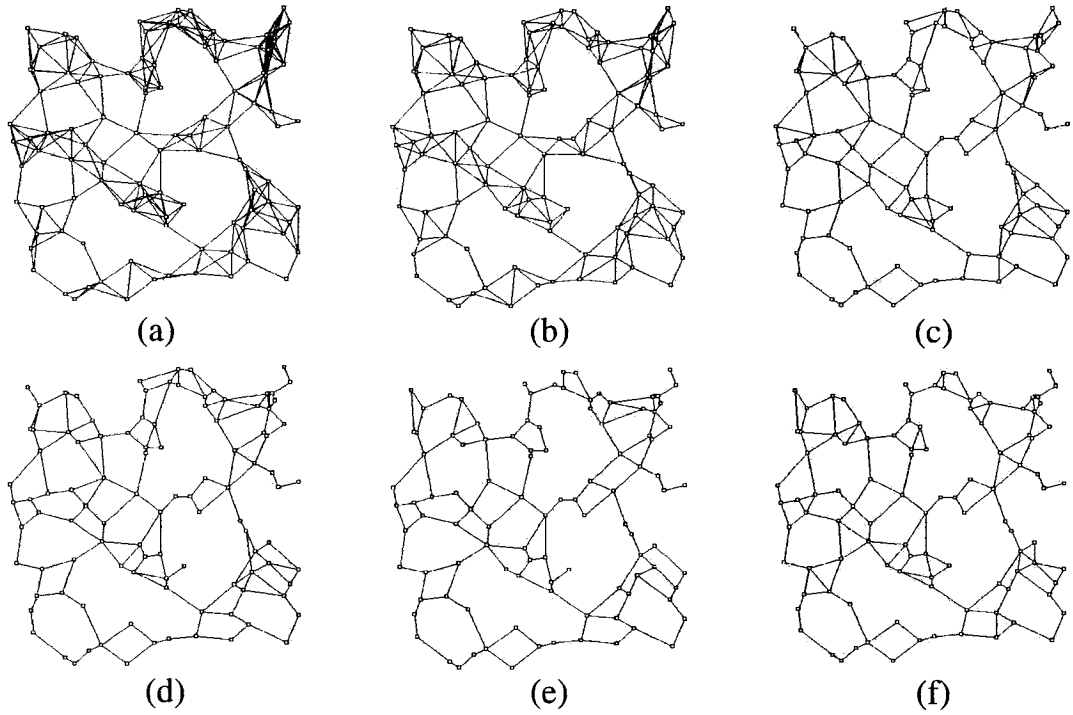

Figure 15: Original *UDG* and related subgraphs: (a) *UDG*; (b) *Yao* Graph with k=6; (c) Adaptive Yao Graph; (d) Displaced Apex Adaptive Yao Graph with s=0.25; (e) *HSP* Graph; and (f) Displaced Apex Adaptive Yao Graph with s=1.0;

$s = 0.125$ and $s = 0.25$, Half Space Proximal subgraph (equivalent to a Displaced Apex Adaptive Yao subgraph with $s = 0.5$), and Displaced Apex Adaptive Yao subgraphs with $s = 0.75$ and $s = 1.0$ are generated. For each Displaced Apex Adaptive Yao subgraph we use $\alpha = 1$ such that $\theta = \theta_m(s, |uz|)$ (recall, for $s > 0.5$, θ is a function of the distance to the chosen neighbor defining the cone). For comparison, we also generate the original Yao subgraph with $k = 6$ for each *UDG*. An example of an *UDG* and related subgraphs is given in Fig. 15. For some of the degree properties studied, we consider the directed versions of the subgraphs.

The following tests are done on the undirected version of each of the graphs mentioned above: 1) average Degree (Fig. 21); 2) maximum path stretch factor, in terms of both hop number and Euclidean distance (Fig. 25 and Fig. 27); 3) average path stretch factor, in terms of both hop number and Euclidean distance (Fig. 26 and Fig. 28); 4) weight of each graph (Fig. 18); 5) number of crossing edges of each graph (Fig. 19); and 6) degree distribution of each graph (percentages of each

degree is averaged over 25 graphs) (Fig. 16 and Fig. 17). For directed versions of the Adaptive Yao subgraph, Displaced Apex Adaptive Yao subgraphs with s=0.125 and s=0.25, Half Space Proximal subgraph, and Displaced Apex Adaptive Yao subgraph with s=0.75, we also measure 1) maximum in-degree (Fig. 24); 2) average in-degree (Fig. 23); 3) maximum out-degree (Fig. 22); and 4) average out-degree (Fig. 23).

As s approaches 0.5, from Fig. 13, Fig. 21 and Fig. 20, the average and maximum node degrees monotonically decrease until s=0.5 when we have the *HSP* graph. Then as s continues to increase to 1, the node degrees begin to increase again. This holds true across all values of N. We can see this trend reflected in the histograms of the node degrees in Fig. 16 and Fig. 17. Also, as we can see in Fig. 18, although the weights of the graphs follow the same trend as the node degrees, the number of crossing edges as s goes from 0.5 to 1 increases only slowly.

In terms of the stretch factors of the graphs, the Adaptive Yao graph (s=0) has consistently the lowest maximum and average (hop number or Euclidean length) stretch factors. For our simulations, the stretch factor for the Adaptive Yao graph was about halfway between that of the *HSP* and the Yao graph with k=6. As s increases to 0.5, the stretch factors increase to a maximum for s=0.5. Then, mirroring the trends for node degree and the weights of graphs, the stretch factor again decreases as s approaches 1. In particular, for the average and maximum Euclidean stretch factors, the drop is more significant.

The reason for this behavior is that for $s > 0.5$ we obtain a graph that maintains many of the properties of the *HSP* graph but with additional, predominantly short, edges. These edges are added since the inverted cone leaves a couple of gaps on either side of the directed edge to the chosen nearest neighbor (e.g., within the region defined by *dce* in Fig. 14) where additional close neighboring nodes may be selected. Although these additional edges are added and the node degrees of the graphs go up, the weights and the number of crossing edges increase more slowly.

The values in Figs. 29 - 34 are averaged over 25 graphs with $N = 75$ nodes. In these figures, we study the dependence of the Displaced Apex Adaptive Yao subgraphs on s and α. For all the plots, it is obvious that there is a stronger dependence on α than

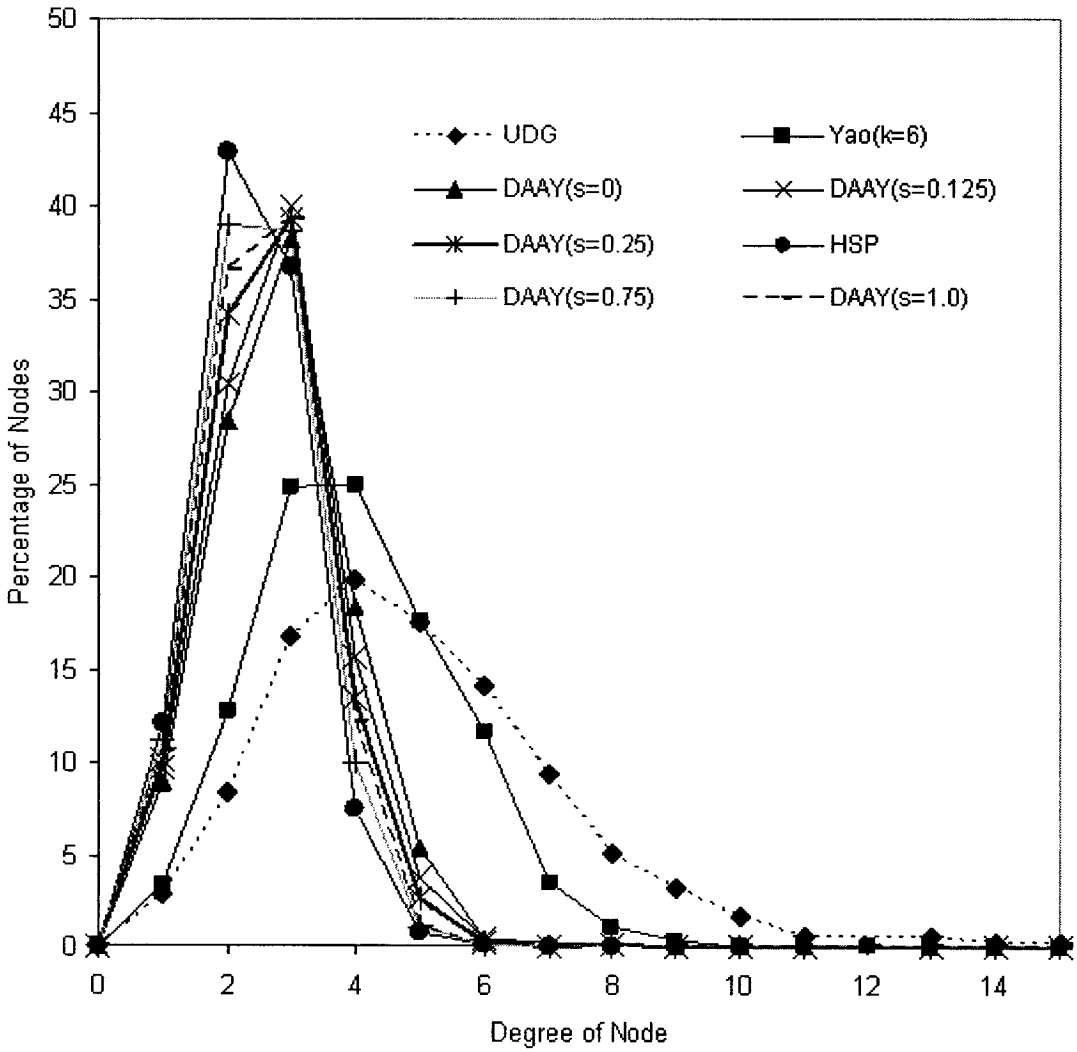

Figure 16: Histogram of degrees of nodes for a graph with 75 nodes.

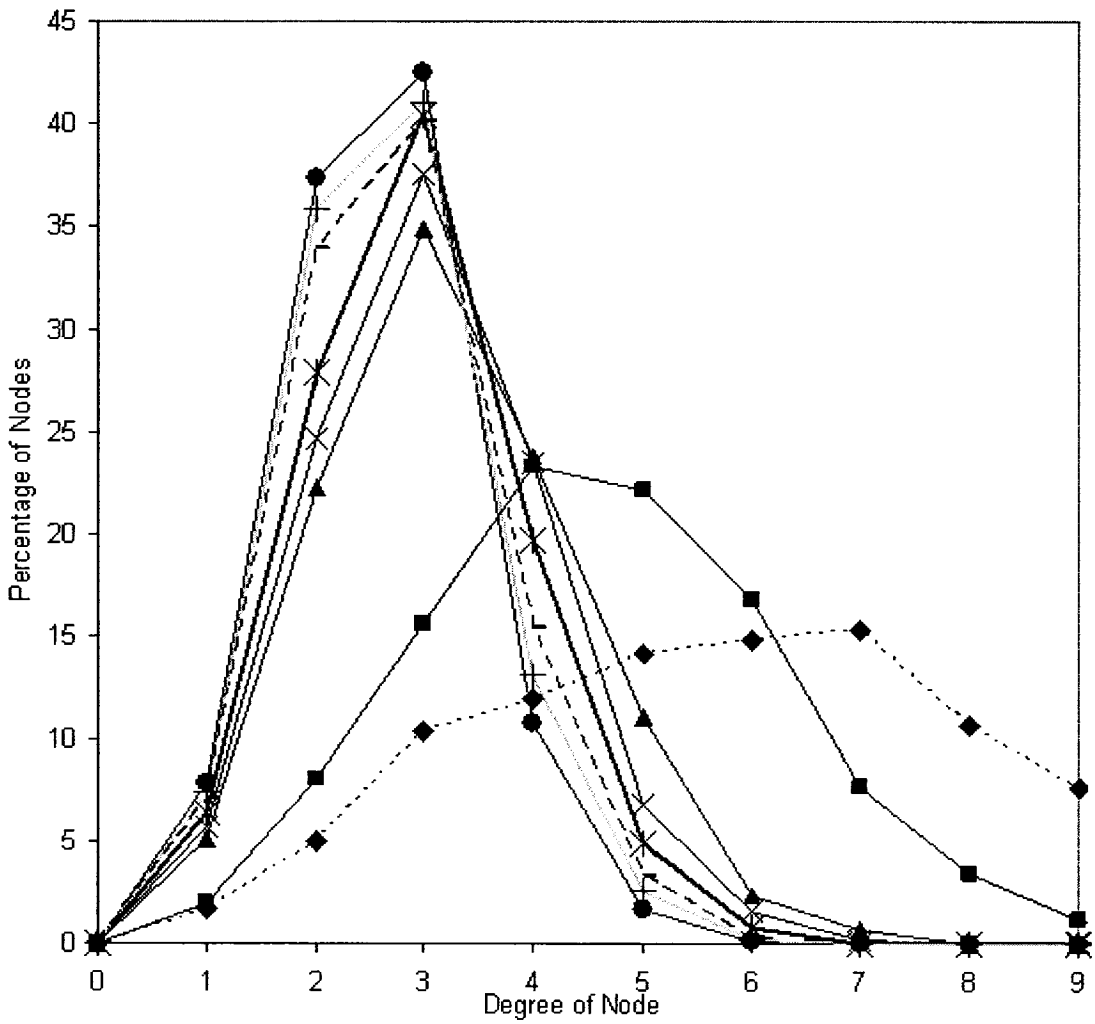

Figure 17: Histogram of degrees of nodes for a graph with 95 nodes. Same legend as Fig. 16.

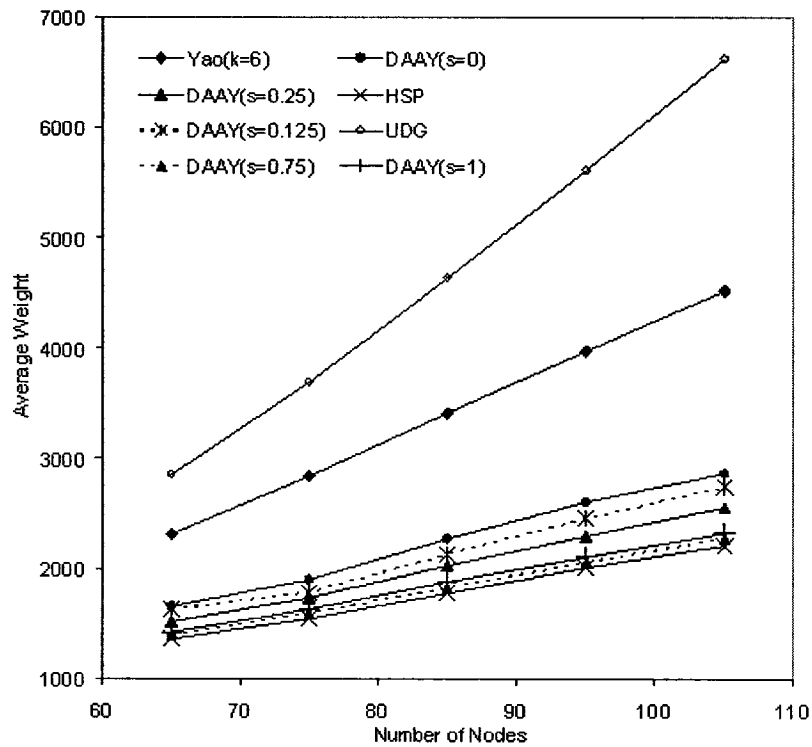

Figure 18: Average weight of graphs.

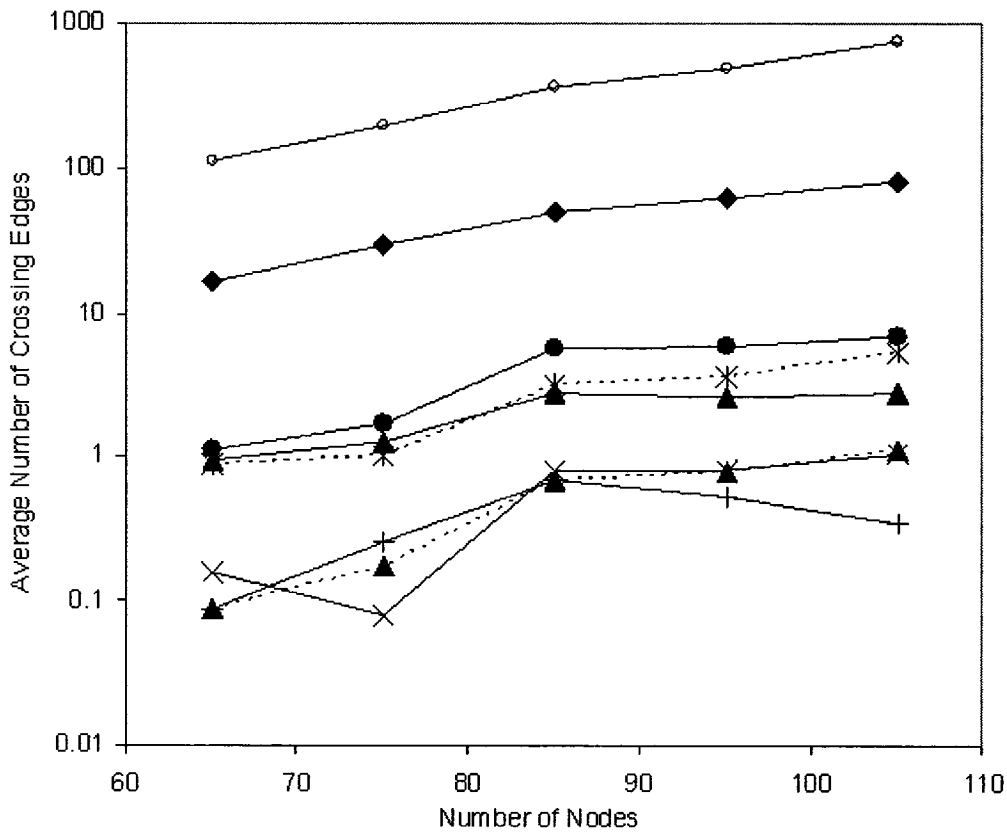

Figure 19: Number of crossing edges of graphs (log scale for number of crossing edges). Same Legend as Fig. 18.

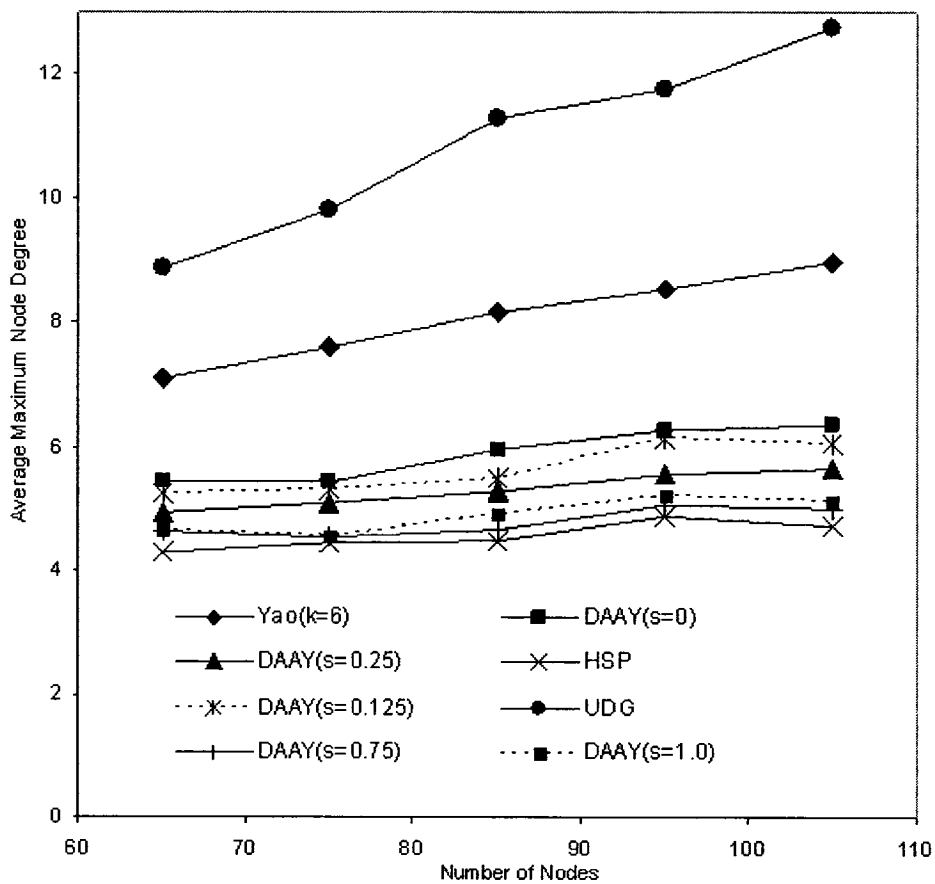

Figure 20: Average maximum node degrees for each graph with various numbers of nodes.

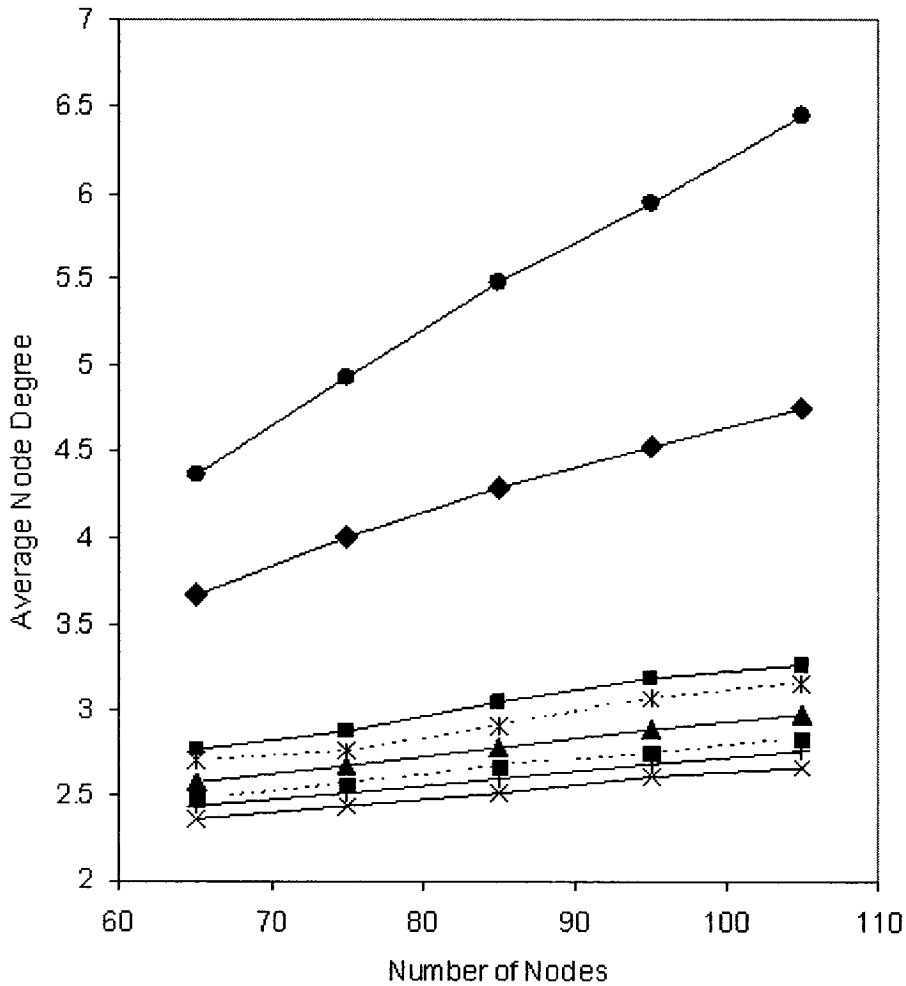

Figure 21: Average Node Degrees. Same Legend as Fig. 20.

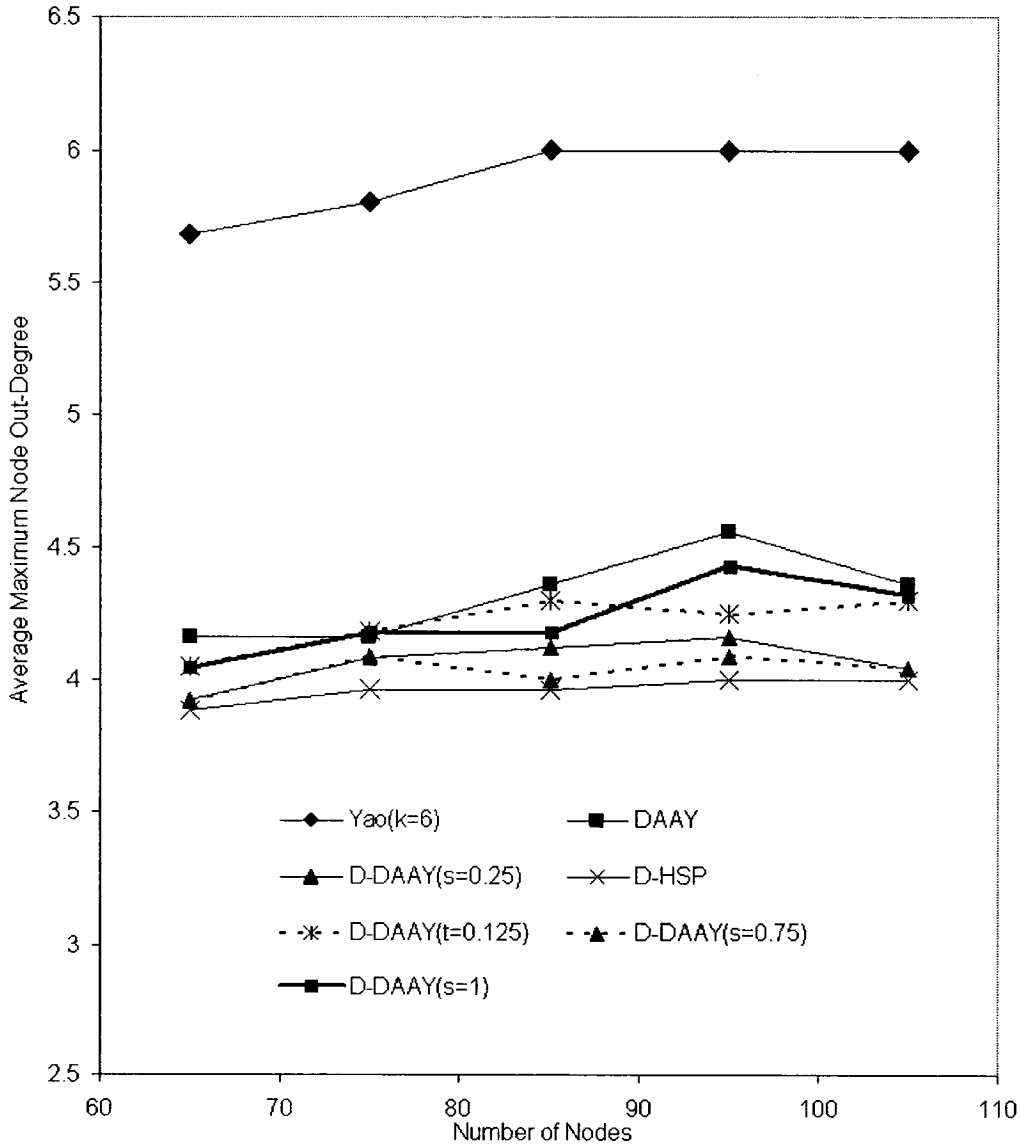

Figure 22: Average maximum node out-degrees.

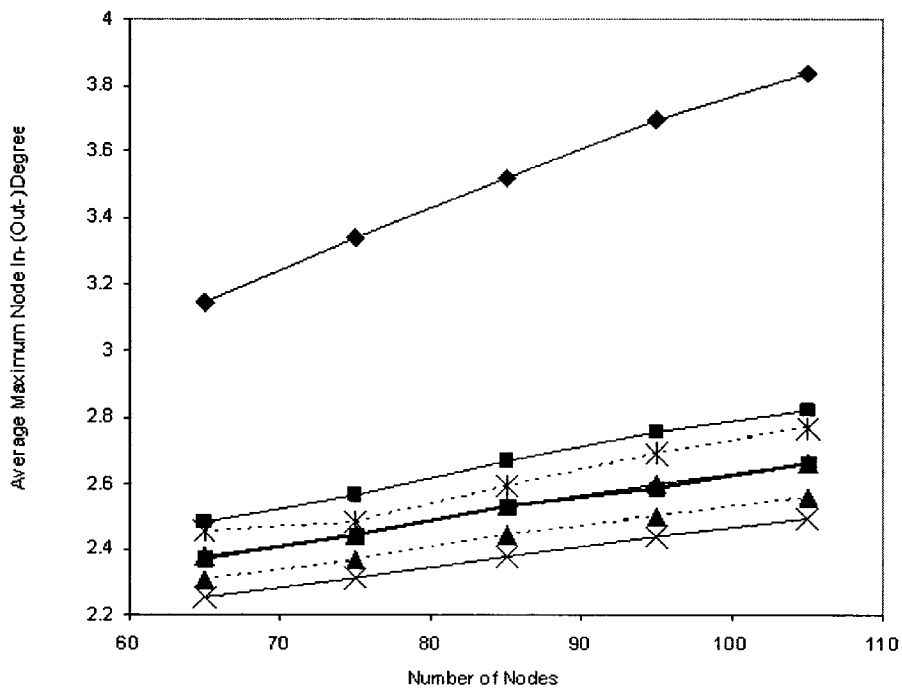

Figure 23: Average Node in-degrees (out-degrees). Same legend as Fig. 22.

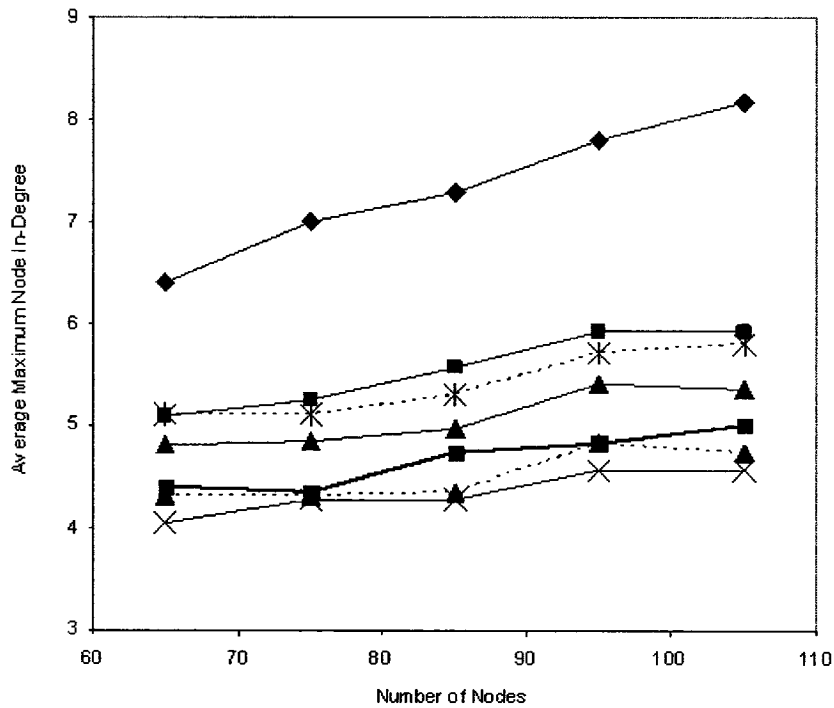

Figure 24: Average Maximum Node In-degree. Same legend as Fig. 22.

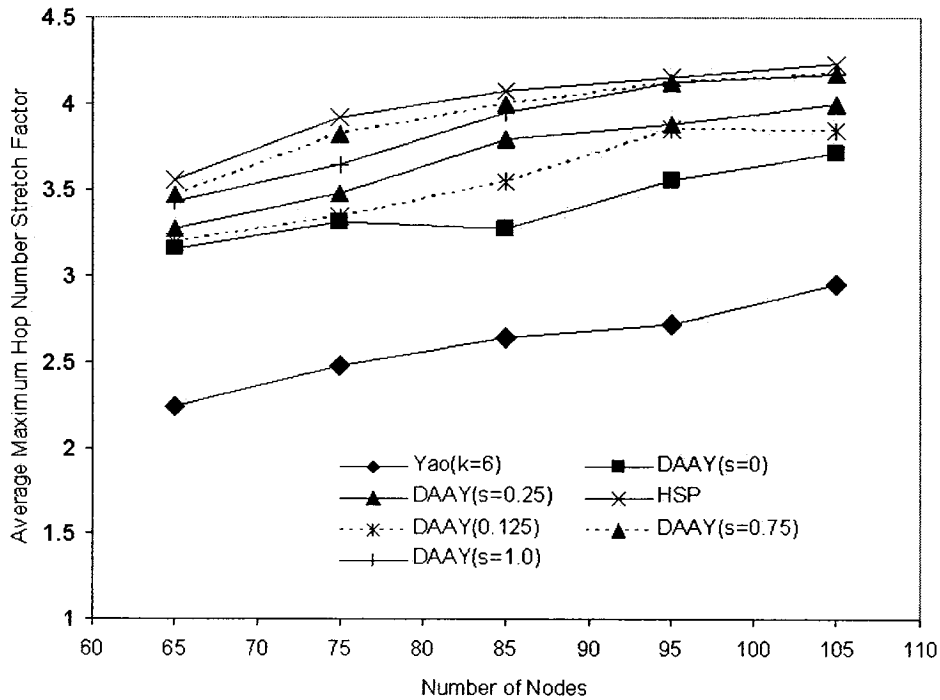

Figure 25: Average maximum hop number stretch factor for each graph with various numbers of nodes.

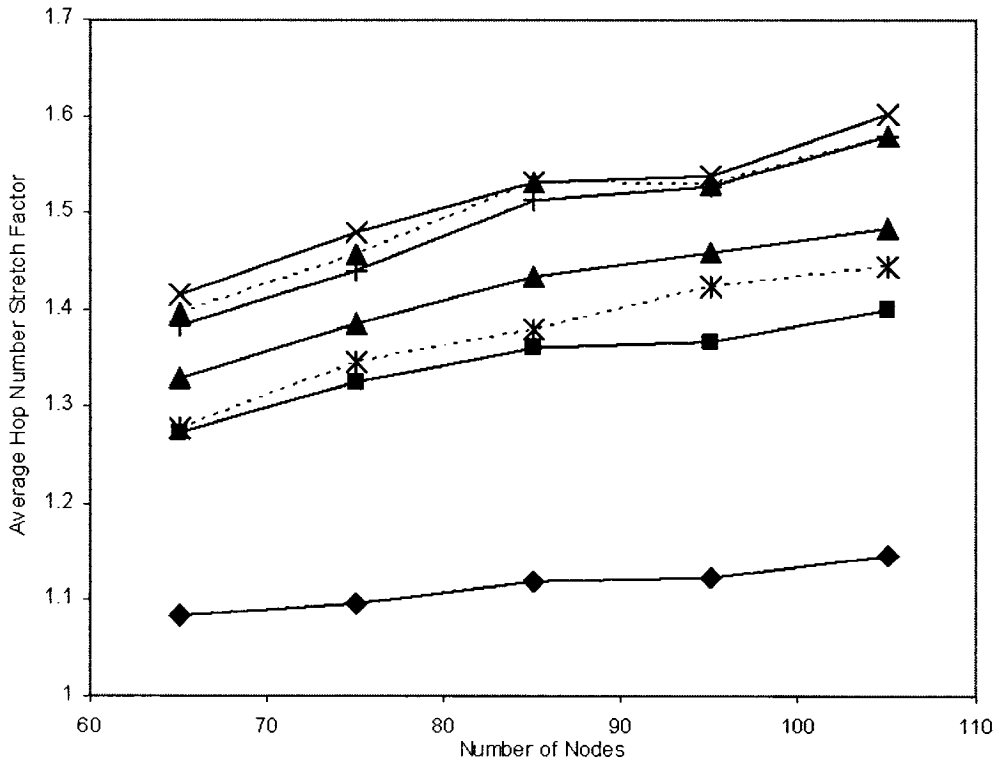

Figure 26: Average hop number stretch factor for each graph with various numbers of nodes. Same legend as Fig. 25.

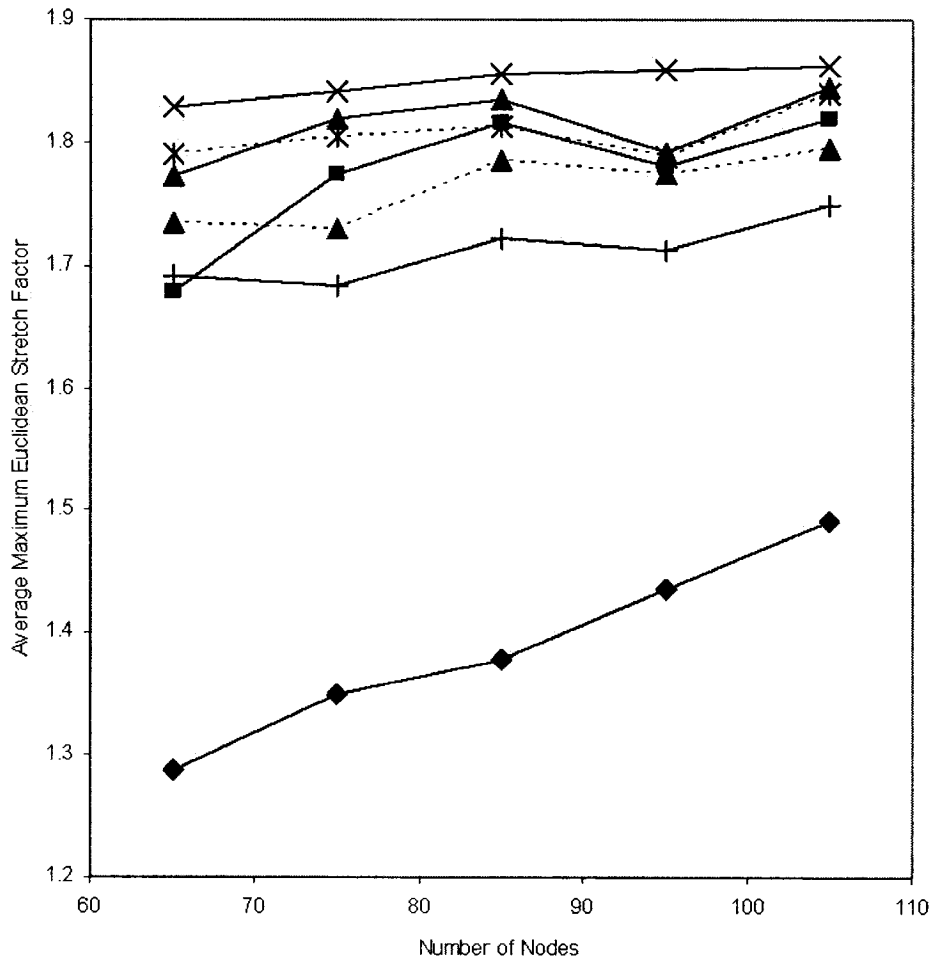

Figure 27: Average maximum Euclidean stretch factor for each graph with various numbers of nodes. Same legend as Fig. 25.

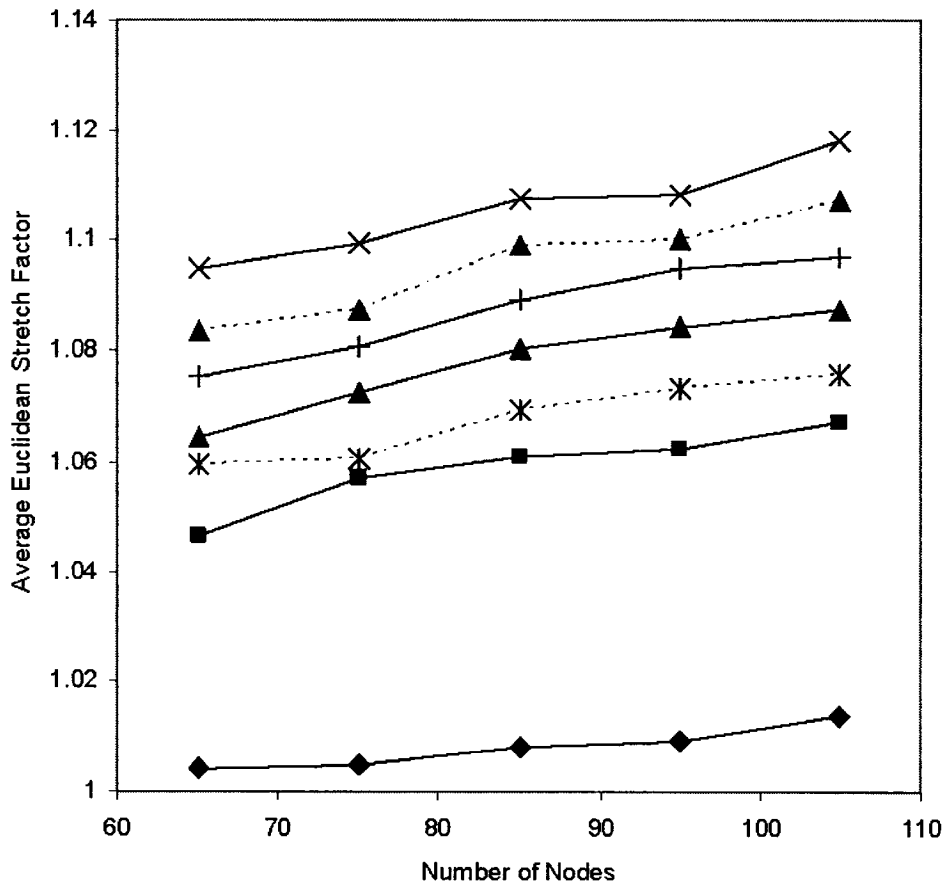

Figure 28: Average Euclidean stretch factor for each graph with various numbers of nodes. Same legend as Fig. 25.

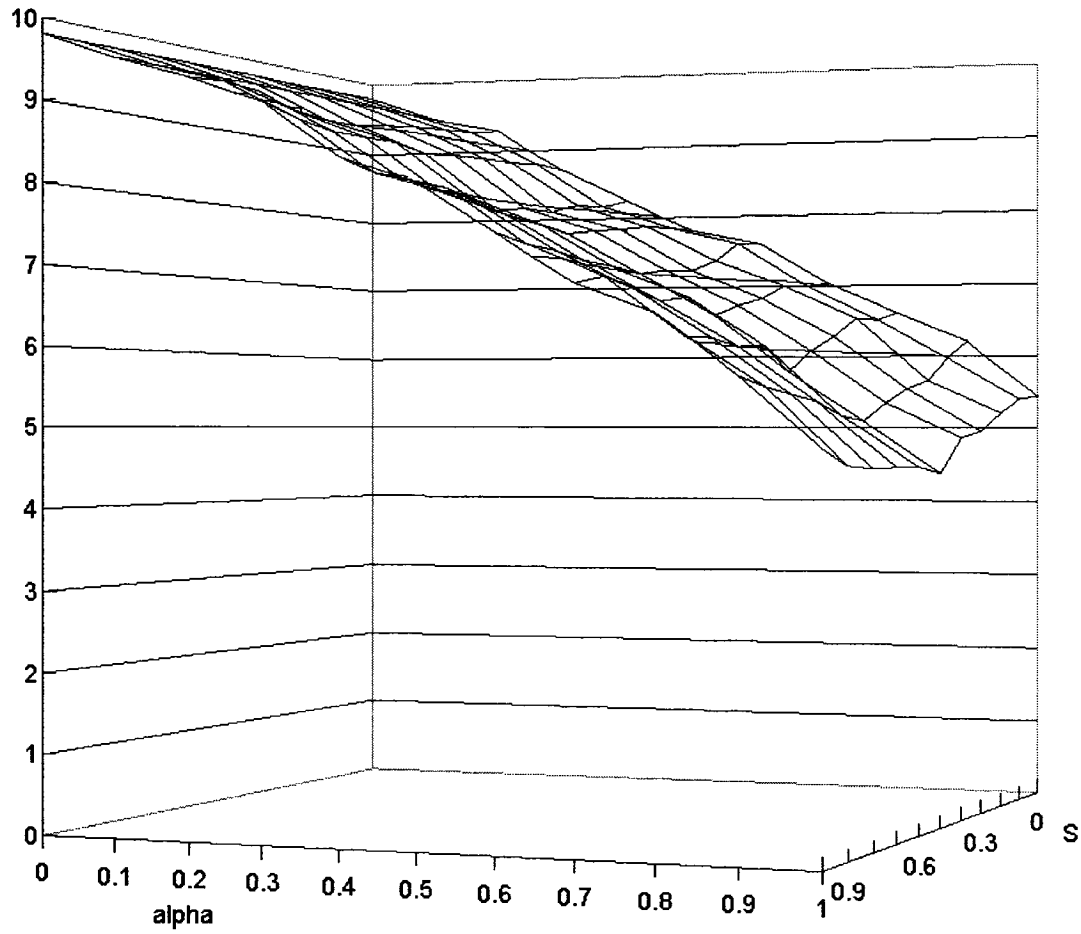

Figure 29: Maximum node degree for $DAAY$

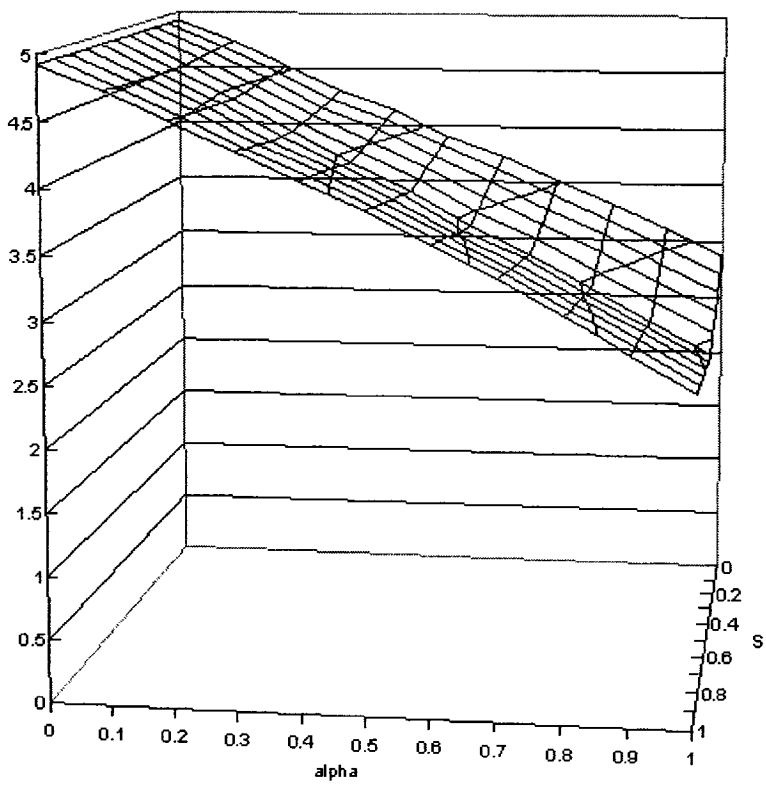

Figure 30: Average node degree for $DAAY$ as function of s and α.

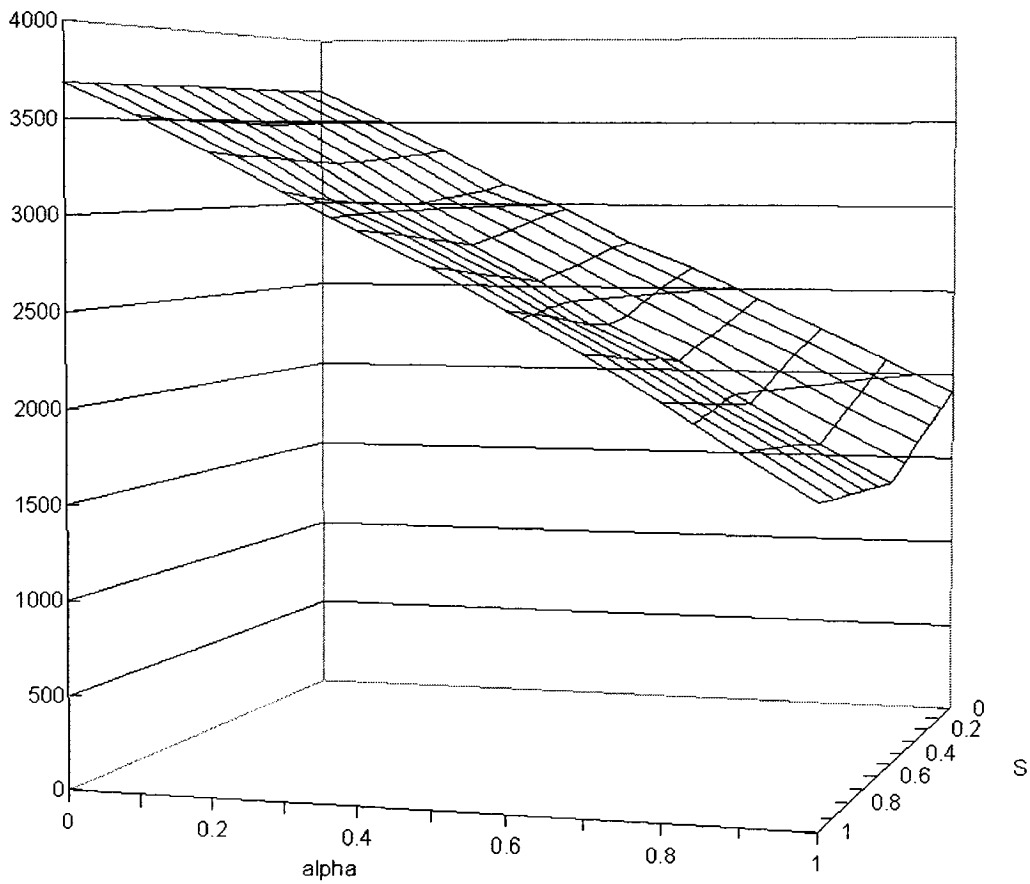

Figure 31: Average weight for $DAAY$ as function of s and α.

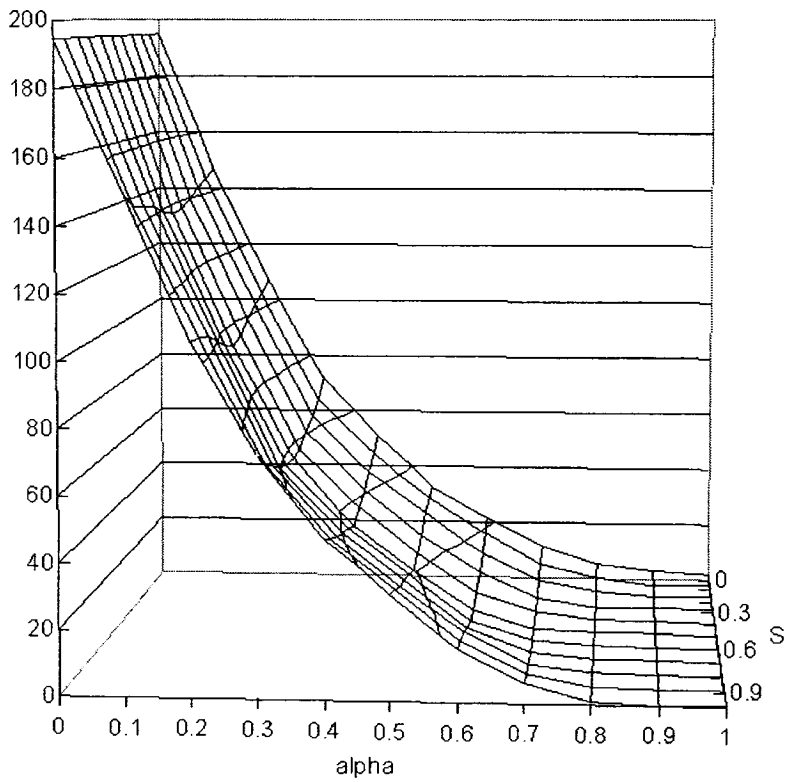

Figure 32: Average number of edge crossings for $DAAY$ as function of s and α.

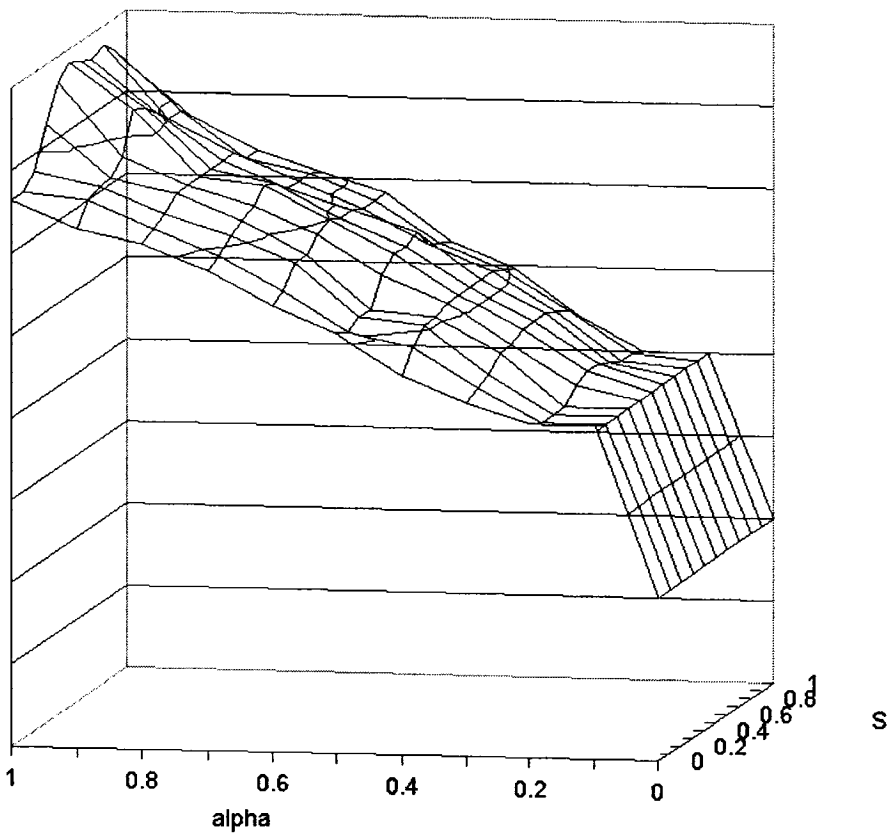

Figure 33: Maximum path stretch factor (hop) for $DAAY$ as function of s and α.

s and no matter the angle changes at a particular s value, the conclusion will be the same to the conclusion drawn for the same s when α=1 in the previous 2-D plots. In addition, for the maximum node degree, average node degree, average number of edge crossings, and average weight, there appears to be symmetry about the $s = 0.5$ value, which appears more pronounced for larger α values. As is apparent from the figures, for these latter four graph properties, about the minimum at $s = 0.5$, the values are larger as s goes to 0 as compared to the values as s goes to 1.

The trends for the node degree, weight and number of edge crossings are reflected in the stretch factor plots in Figs. 33 and 34 except that for large α values, the stretch factor values are maximum at $s = 0.5$.

3.3 Displaced Apex Adaptive Yao in 3-D

After presenting the 2-D *DAAY* in Sec. 3.1, we will present here only the simulation results for the 3-D *DAAY*. The reason is that *DAAY* graph in 3-D is just an extension of the 2-D version since the line half-way becomes a half-plane, and the cone becomes a three dimensional cone. All the angles are symmetric about the axis of the cone so the $\theta_m(s, |uz|)$ definition is essentially the same. The properties and the algorithm of the 2-D version are similar to the 3-D version. In fact even the simulation results presented for the 3-D *DAAY* are also similar to the 2-D version.

3.4 Simulation Results for *DAAY* in 3-D

In our experiments we use randomly chosen connected unit disk graphs on in a cube of 100×100×100. We vary the number of nodes, N, between 65, 75, 85, 95, and 105 nodes. For all the results reported here, the results have been averaged over 25 graphs for each value of N. For all the graphs tested, the transmission radius R_c used is 25 units. The definitions for both Euclidean and hop number stretch factors are the same as the definitions mentioned in Sec. 2.1.1. Additional to the maximum stretch factors (Euclidean or hop number), we also calculate the average stretch factors (Euclidean

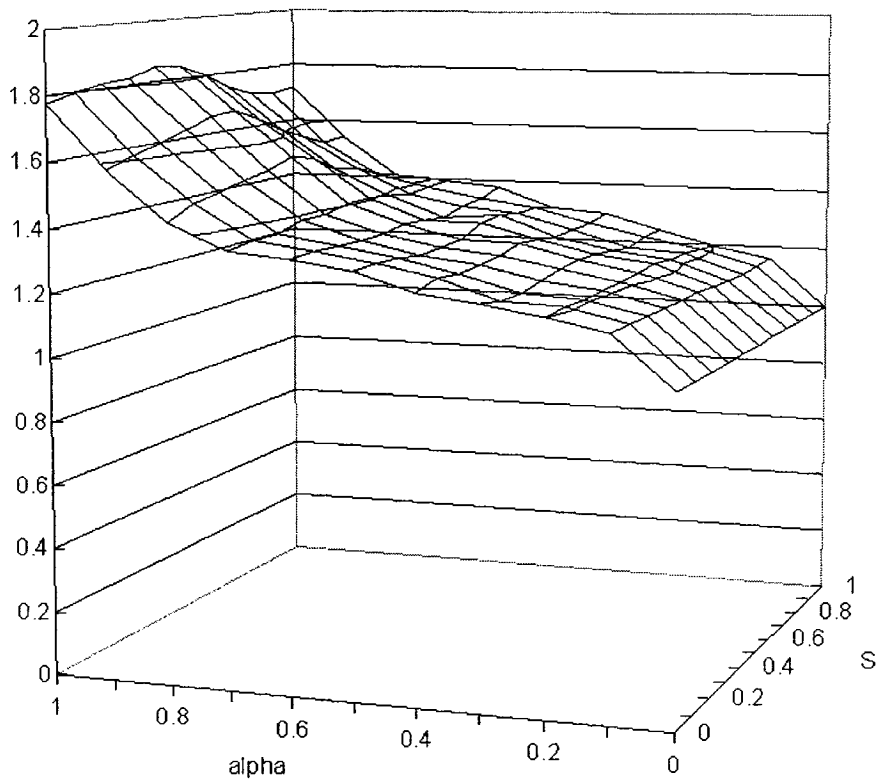

Figure 34: Maximum path stretch factor (Euclidean) for $DAAY$ as function of s and α.

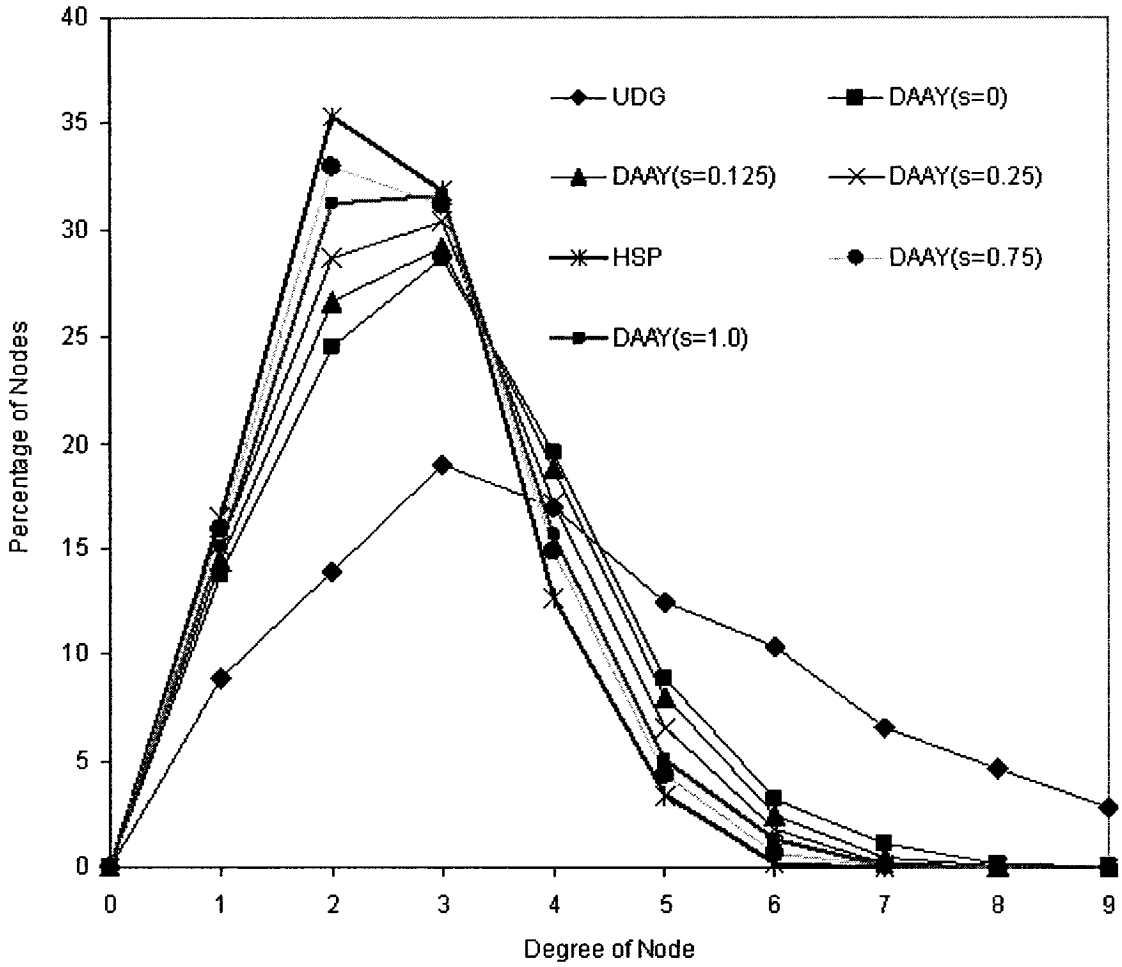

Figure 35: Histogram of degrees of nodes for a 3-D graph with 75 nodes.

or hop number) in our simulations.

For each 3-D UDG, an 3-D Adaptive Yao subgraph (equivalent to a 3-D Displaced Apex Adaptive Yao subgraph with $s = 0$), 3-D Displaced Apex Adaptive Yao subgraphs with $s = 0.125$ and $s = 0.25$, 3-D Half Space Proximal subgraph (equivalent to a Displaced Apex Adaptive Yao subgraph with $s = 0.5$), and 3-D Displaced Apex Adaptive Yao subgraphs with $s = 0.75$ and $s = 1.0$ are generated. For each 3-D Displaced Apex Adaptive Yao subgraph we used $\alpha = 1$ such that $\theta = \theta_m(s, |uz|)$ (recall, for $s > 0.5$, θ is a function of the distance to the chosen neighbors).

The following tests were done on the undirected version of each of the graphs mentioned above: 1) average Degree (Fig. 39); 2) maximum path stretch factor, in

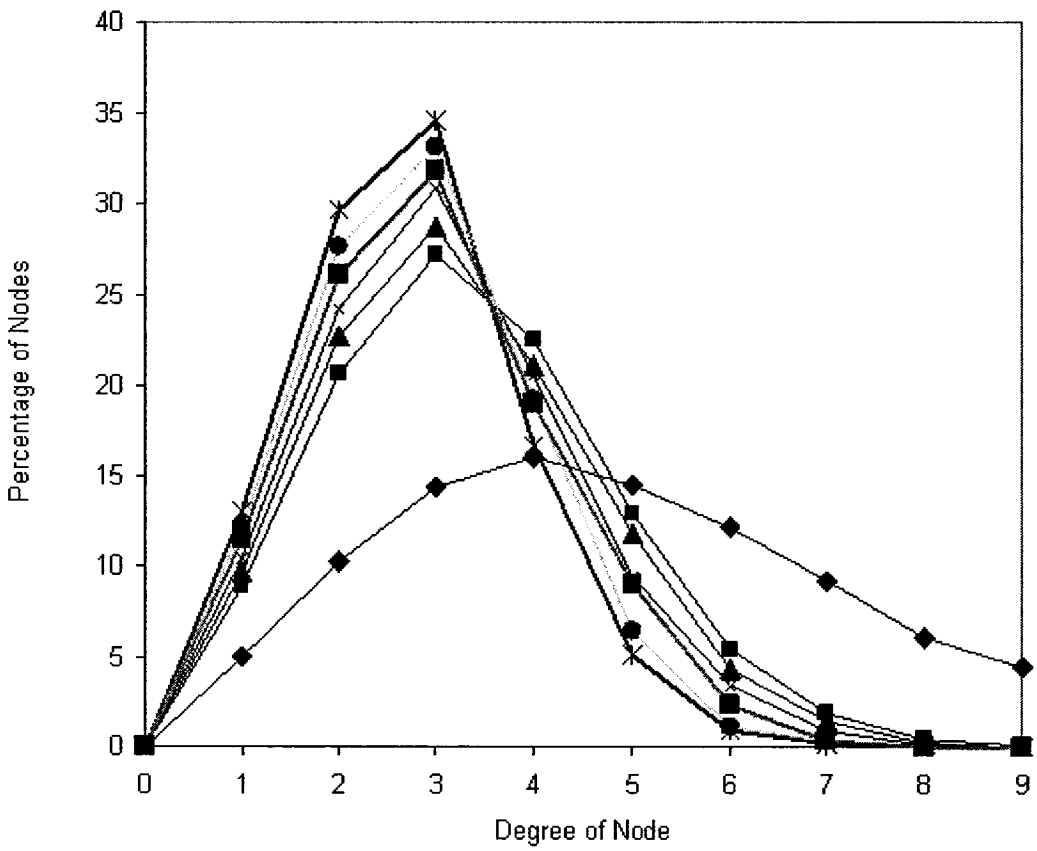

Figure 36: Histogram of degrees of nodes for a 3-D graph with 95 nodes. Same legend as Fig. 35.

terms of both hop number and Euclidean distance (Fig. 43 and Fig. 45). 3) average path stretch factor, in terms of both hop number and Euclidean distance (Fig. 44 and Fig. 46); 4) weight of each graph (Fig. 38); and 5) degree distribution of each graph (The percentages of each degree is averaged over 25 graphs) (Fig. 35 and Fig. 36). The stretch factor for a pair nodes u, v, $u \neq v$, is the ratio of the shortest length path between u and v in the subgraph over that for the original *UDG*. The path length is computed in terms of the number of hops along the path or the sum of the Euclidean lengths of the edges of the path. The maximum is taken over all distinct pairs u, v in the graph. For directed versions of the Adaptive Yao subgraph, Displaced Apex Adaptive Yao subgraphs with s equal to 0.125, 0.25, 0.75, we also measured 1) maximum in-degree (Fig. 42); 2) average in-degree (Fig. 41); 3) maximum out-degree (Fig. 40); and 4) average out-degree (Fig. 41).

As s approaches 0.5, from Fig. 39 and Fig. 37, the average and maximum node degrees monotonically decrease until s=0.5 when we have the *HSP* graph. Then as s continues to increase to 1, the node degrees begin to increase again. This holds true across all values of N. We can see this trend reflected in the histograms of the node degrees in Fig. 35 and Fig. 36.

In terms of the stretch factors of the graphs, the Adaptive Yao graph (s=0) has consistently the lowest maximum and average (hop number or Euclidean length) stretch factors. For our simulations, the stretch factor for the Adaptive Yao graph was about halfway between that of the *HSP* and the Yao graph with k=6. As s increases to 0.5, the stretch factors increase to a maximum for s=0.5. Then, mirroring the trends for node degree and the weights of graphs, the stretch factor again decreases as s approaches 1. In particular, for the average and maximum Euclidean stretch factors, the drop is more significant.

The reason for this behavior is that for $s > 0.5$ we obtain a graph that maintains many of the properties of the *HSP* graph but with additional, predominantly short, edges. These edges are added since the inverted cone leaves a couple of gaps on either side of the directed edge to the chosen nearest neighbor where additional close neighboring nodes may be selected (similar to the 2-D case). Although these additional

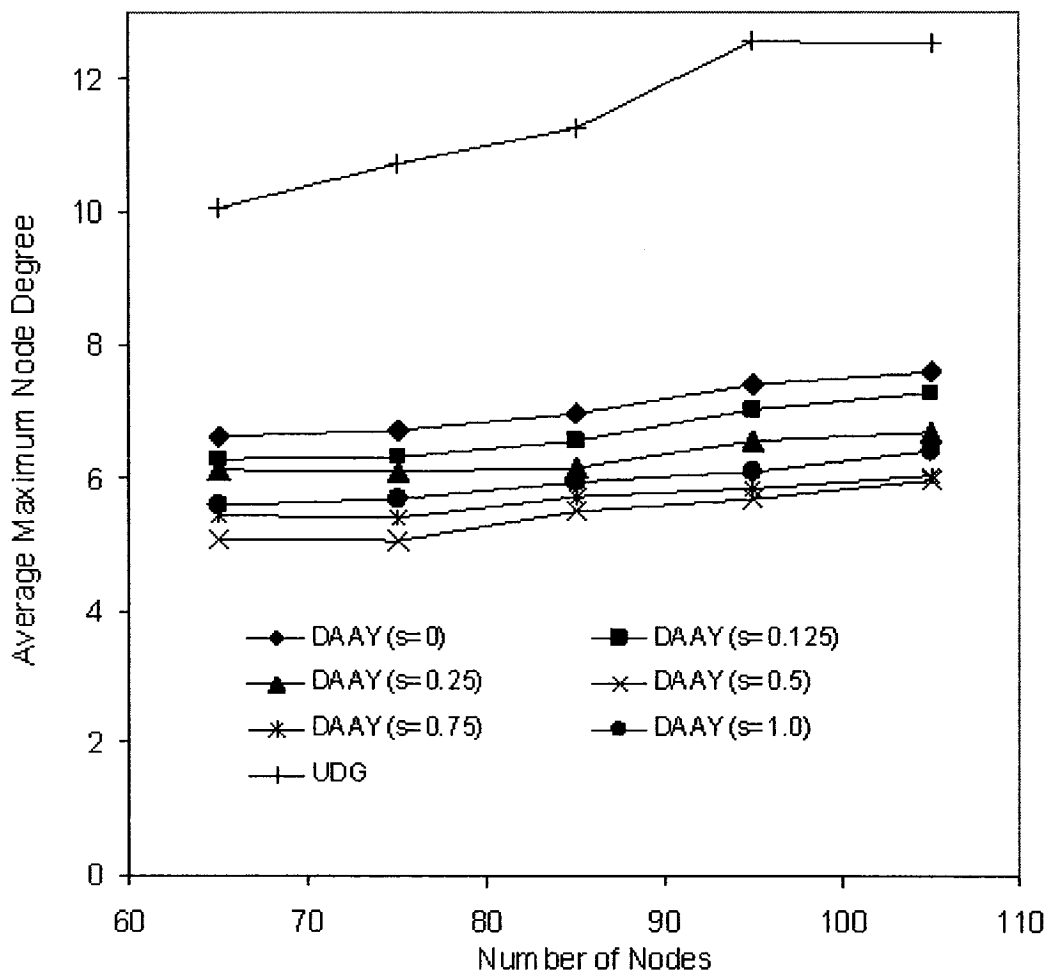

Figure 37: Average maximum node degrees for each 3-D graph with various numbers of nodes.

edges are added and the node degrees of the graphs go up, the weights increase more slowly.

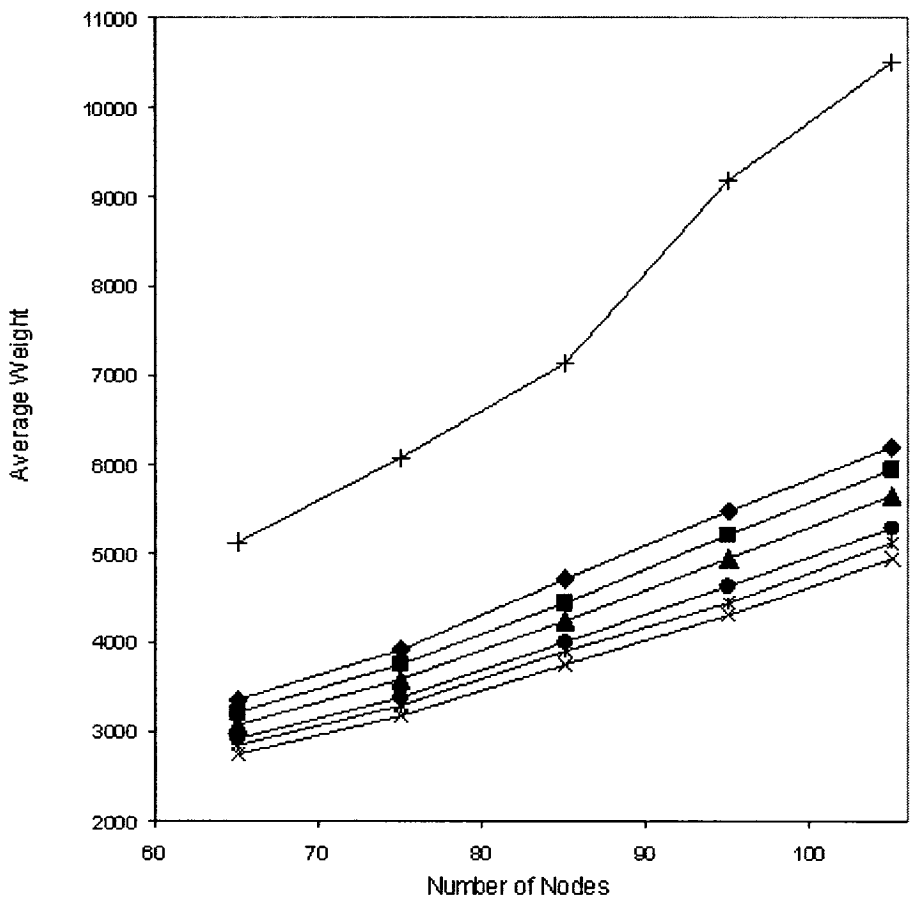

Figure 38: Weights of 3-D graphs. Same legend as Fig. 37.

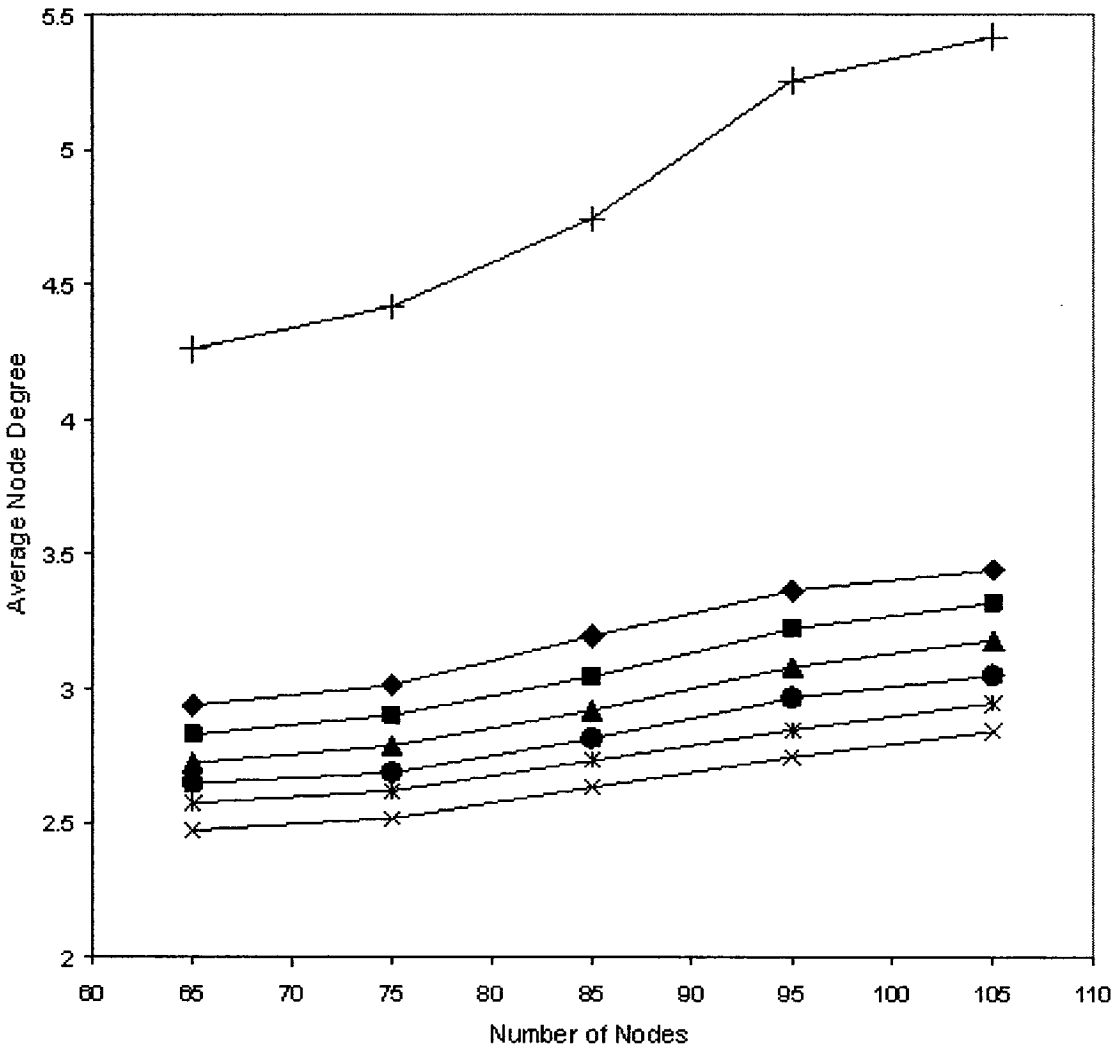

Figure 39: Average Node Degrees for 3-D graphs. Same legend as Fig. 37.

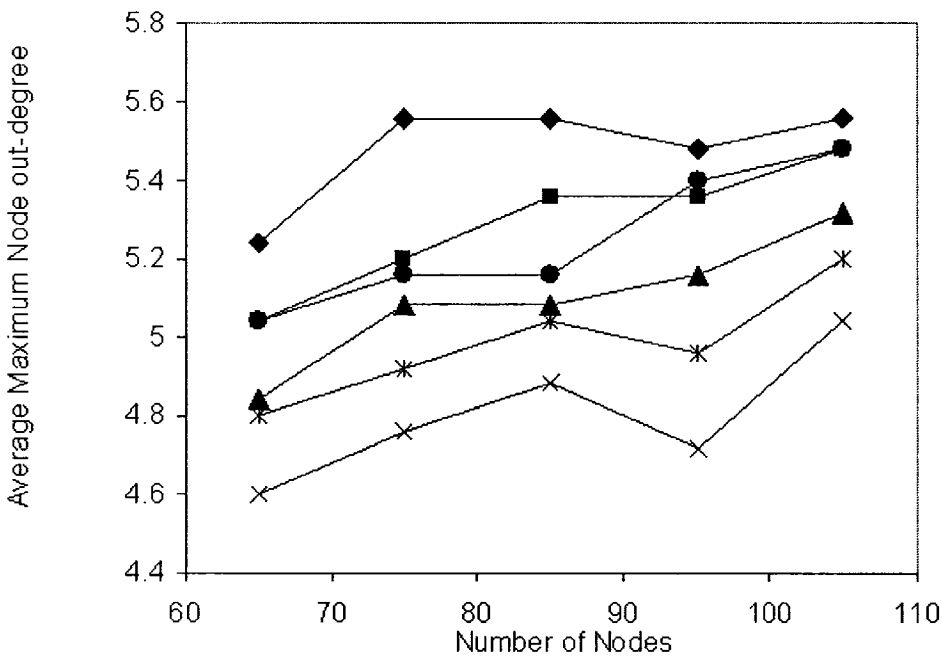

Figure 40: Average Maximum node out-degrees for 3-D graphs. Same legend as Fig. 37.

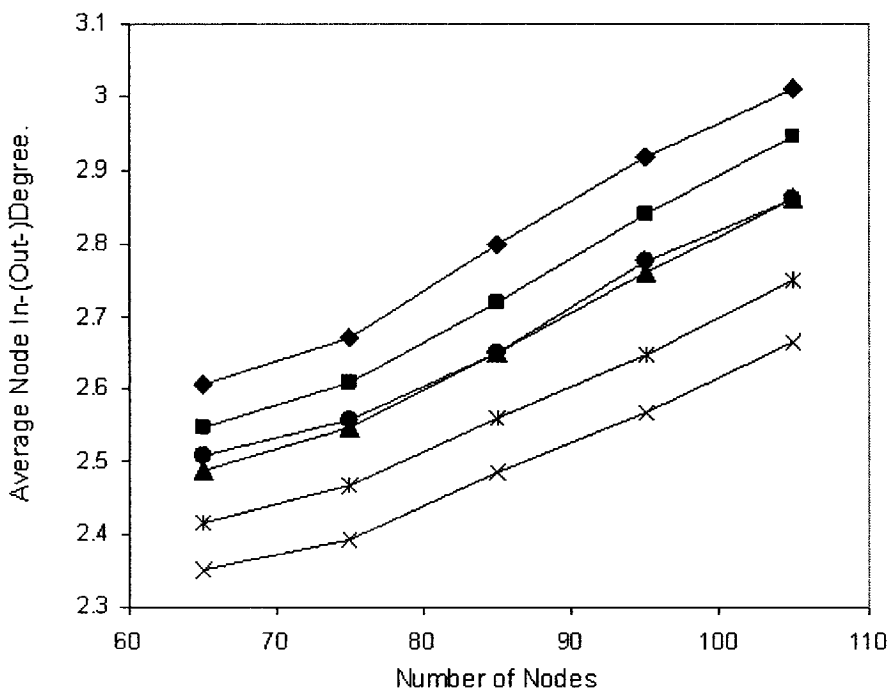

Figure 41: Average Node in-degrees(out-degrees) for 3-D graphs. Same legend as Fig. 37.

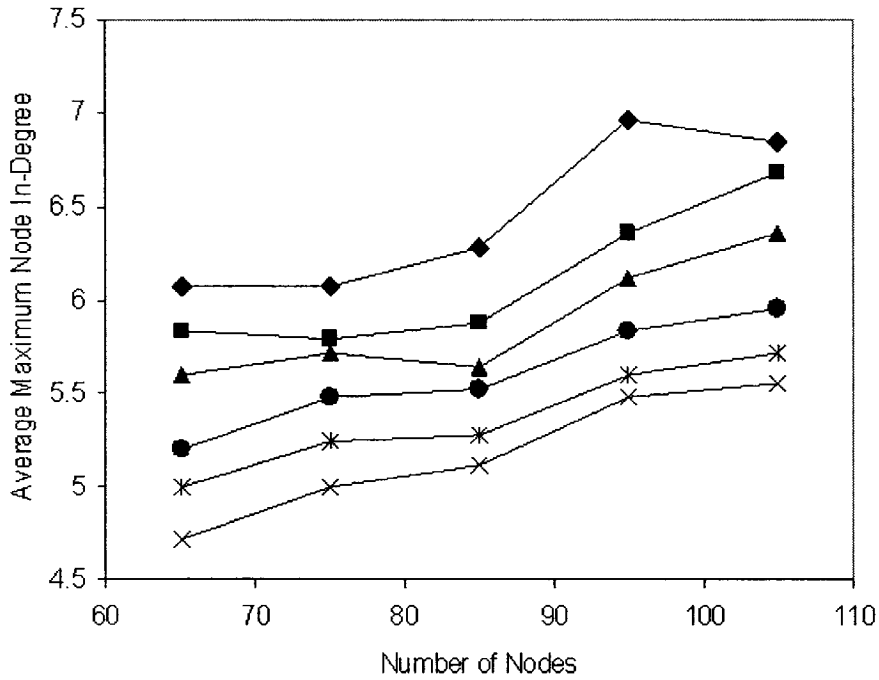

Figure 42: Average Maximum node in-degrees for 3-D graphs. Same legend as Fig. 37.

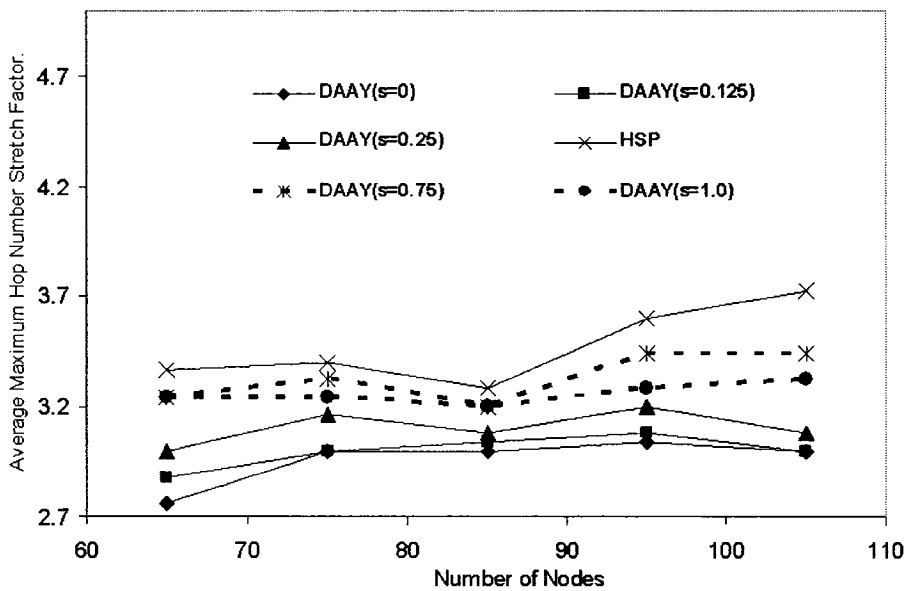

Figure 43: Average Maximum Hop Number stretch factor for each 3-D graph with various numbers of nodes.

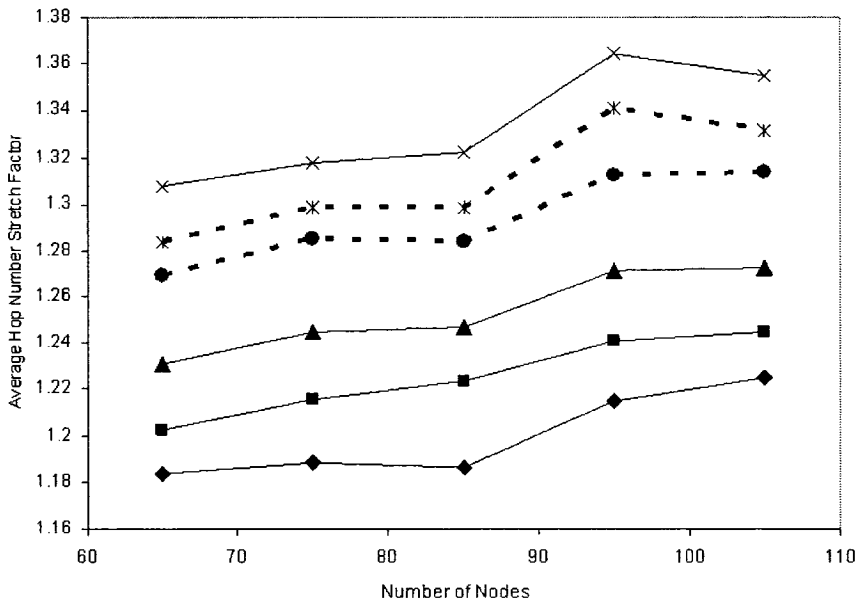

Figure 44: Average Hop Number stretch factor for each 3-D graph with various numbers of nodes. Same legend as Fig. 43.

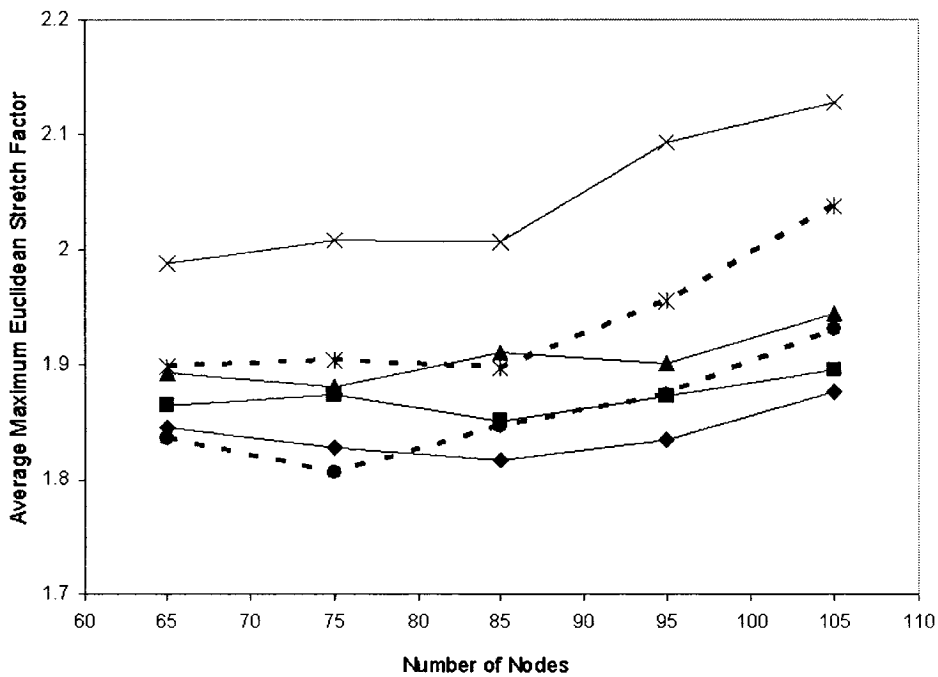

Figure 45: Average Maximum Euclidean stretch factor for each 3-D graph with various numbers of nodes. Same legend as Fig. 43.

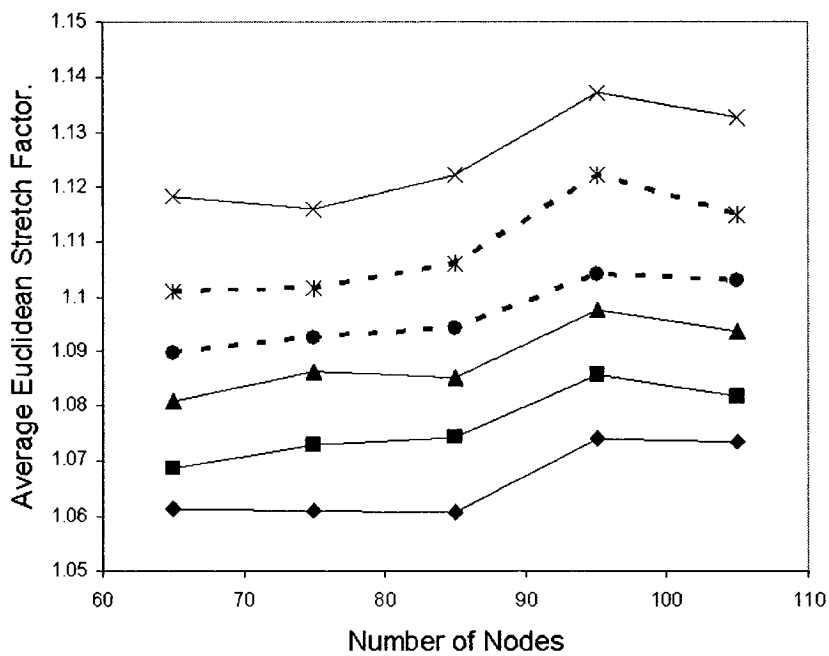

Figure 46: Average Euclidean stretch factor for each 3-D graph with various numbers of nodes. Same legend as Fig. 43.

Chapter 4

Model of a Sensing-Covering Network

Here we will present a new model of a sensing-covering network based on the *UDG* graph. As mentioned in Sec. 2.1.3, there are some approaches that ensure a network is fully covered and fully connected. Some of these approaches handle the power usage issue. The previous works are based on the decimation approach assuming a very dense set of nodes. In this Chapter, we will present a new approach that handles coverage, connectivity, and power usage issues. Our approach is based on an incrementation approach.

4.1 Assumptions of our New Model

The assumptions we make include the following:

- The nodes locations are based on 2-D space.
- Sensing range is 20m. Therefore, we can have a node with a sensing circle, and this node is the center of its sensing circle.
- All the nodes have the same sensing range. Thus any point in the field should be covered by at least one node. This leads to the definition of a sensing-covered network.

- The nodes are within a square of 500×500 meters.

These assumptions are the same as those presented by Xing *et al.* [XLPH06]. They are used as topology-based versions of some position-based routing algorithms presented in Chapter 5. The assumptions are deployed in the stochastic model. In this model, two nodes, u and v, can communicate with each other if and only if $|uv| \leq R_c$. In this model also, the network is based on the unit disk graph $G(V,E)$, where V is the set of all the nodes in the network, and E is the set of all the edges in the network. The edge$(u,v) \in E$ if and only if $|uv| \leq R_c$.

4.2 Double Range Property

The double range property means that there is a relationship between the sensing range (R_s) and the communication range (R_c) as mentioned by Xing *et al.* [XLPH06]. This relationship is presented as a ratio between R_c and R_s. It refers to the range ratio [XLPH06]. We start with $R_c/R_s = 2$. When the ratio augments, our sensor network becomes denser, achieving a much better dilation. Thus, the double range property can be generalized as follows:

$$R_c/R_s \geq 2, \textit{the more the ratio increases, the denser the network becomes.}$$

The geometric analysis by Xing *et al.* [XWZ+05] used this assumption to prove that the network is strongly connected. In other words, if the network is a sensing-covered network, this network would be strongly connected if the double range property has been taken into consideration. This property supports the routing protocols for finding a routing path to the destination. Xing *et al.* proved this theorem using the Voronoi diagram, as illustrated in Fig. 47. A Voronoi cell of node u is represented by $Vor(u)$. They first proved that if the Voronoi cells of two given nodes are adjacent, then these nodes can communicate. As shown in the Fig. 47, the vertex p is the common Voronoi vertex for the three adjacent Voronoi cells $Vor(u)$, $Vor(v)$, $Vor(w)$. The nodes u, v, and w are equal in their distance from p and they are the closest to

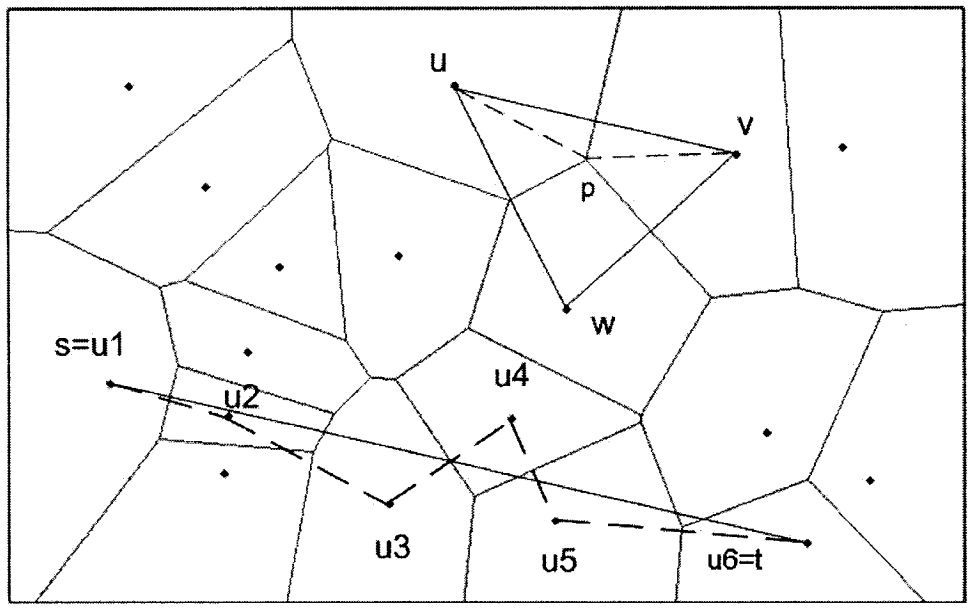

Figure 47: The Voronoi diagram of the nodes that cover the region

p among others. Therefore, p must be covered by the nodes u, v, and w, if not then it won't be covered by any nodes in the diagram. Based on the triangle inequality, they had:

$$|uv| \leq |pu| + |pv| < 2R_s \leq R_c.$$

After that, Xing *et al.* [XWZ+05] proved that the network is connected. They showed that there is a path between any two nodes in the network. Suppose the line segment st does intersect the successive Voronoi cells $Vor(s)$=$Vor(u_1)$, $Vor(u_2)$ until $Vor(u_n)$=$Vor(t)$. Since the Voronoi cells are adjacent to each other, any two consecutive nodes in the series u_1 to u_n can communicate according to the first part of the proof. Therefore, there is a communication path from s to t. This path is illustrated in the dashed line.

4.3 Constructing a Sensing Covered Network

In this design, we ensure that we have a fully covered network using the sensing coverage property of the nodes. Once every single space is covered, we can then ensure that the delivery rate for our routing protocols achieve 100% using the double

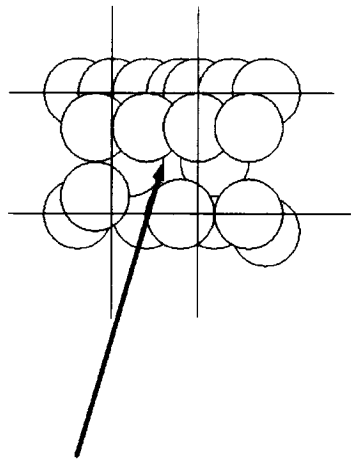

Figure 48: The blue colors are the sensing circles. Each sensing circle has its corresponding node.

range property. To construct this network, we need to demonstrate specific techniques in the upcoming sections.

4.3.1 Constructing a Grid

We assume that we have a region of $h \times h$ units in which the nodes are randomly located. In this region, a grid is constructed with $h/2+1$ columns and $h/2+1$ rows, including grid lines on the boundary of the region. Now, each row or column is represented a linked list. Initially, each row stores a space of its actual length in its corresponding linked list. In other words, there is a space on the row and its length is the same as the row length. The linked list stores initially one node which is the length of the actual row. The same applies for the column's linked list. The grid can ensure that most of the spaces have been filled by sensing circles. We mentioned the word 'most', since even if all the lines (rows and columns) have been filled, there will be some cases in which a space exists between the lines. One of those cases is demonstrated in Fig. 48. These cases are referred to as blind points [TG02]. At the beginning, as we mentioned before, the grid would be empty. See Fig. 49.

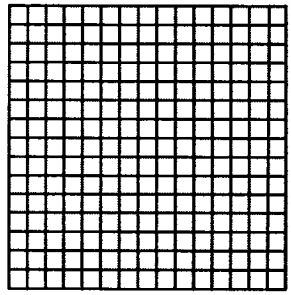

Figure 49: A sample of the empty lines in the grid of 16 rows and 16 columns. They are marked by bold black color.

4.3.2 Filling out The Spaces

In this section, we will show how we fill out all the lines in the grid, even the spaces between the lines. In other words, a single line (row or column) indicates that there is one space or multiple space fragments on it. Filling out the lines means that we are filling out the spaces.

The sensing range is responsible for checking and eliminating the spaces. We place one node at a time (as mentioned previously, each node is centered in its corresponding sensing circle), and based on this we will know what line(s) are intersected by the sensing circle. After that, we will remove the space(s) where the circle has already covered. We continue like this until every single space is covered. To clarify more about this idea, let us see Fig. 50. Note in this figure, we increased the width of the lines only for clarification purposes - actually they do not have any width.

We will be interested in the colored (other than the black) lines in Fig. 50. These colours indicate the sizes of the spaces which are on the lines; although the other lines have black bold colours (also indicating the sizes of the spaces). We place one node at a time. After that, we check if this area (it represents the place where the sensing circle intersects the lines) is fully covered by previous sensing circle(s). This can be done by checking which line is currently being intersected by the new added sensing circle. If the area of the intersected line is already covered by some previous sensing circles, then we check whether the other intersected lines are being covered or not. If these lines are completely covered, then we do not add this node since it is considered

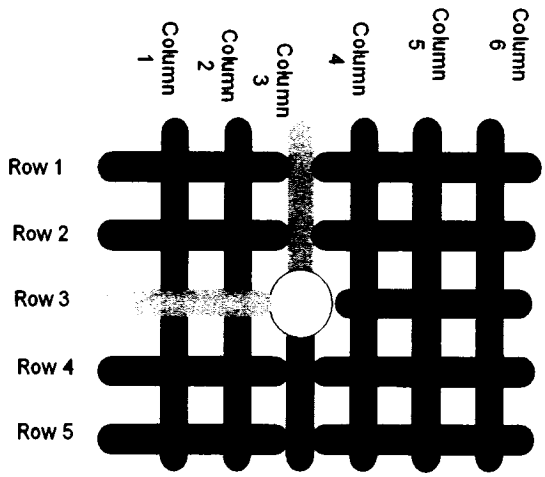

Figure 50: Shows the new sizes of the spaces in different colors (other than black). This happens after adding a sensing circle to the grid.

as a duplicate node. 'Duplicate' refers to the fact that the new node would duplicate the covering behavior of previous nodes. In other words, we do not place it at the covered area but we use it for other areas. This is referred to as 'Duplication' and will be explained in detail in Sec. 4.3.3. If these lines are partially covered, then we take the new sensing circle into our consideration. We turn on the new added node. Once the node becomes active, immediately it will split the space into new spaces or if there is no space left after adding this node, then the covered space(s) would be removed, see Figs. 51 and 52, respectively. The intersected lines are colored by other colours in order to show that we have different sizes of the new remaining spaces. In other words, the space on the intersected row (row number 3) becomes shorter than before as well as for the intersected column (column number 3). The node that represents this space in the linked list for this row will be removed, and two new nodes will be added in this row's linked list. Each new added node has a new remaining space size after placing a sensing circle at that row. The same applies for the column. We place another circle again as shown in Fig. 53, and then we will have new remaining spaces and so on. See Fig. 54. Note that the second sensing circle will split one of the remaining spaces that occurred by the first sensing circle into two new smaller lines.

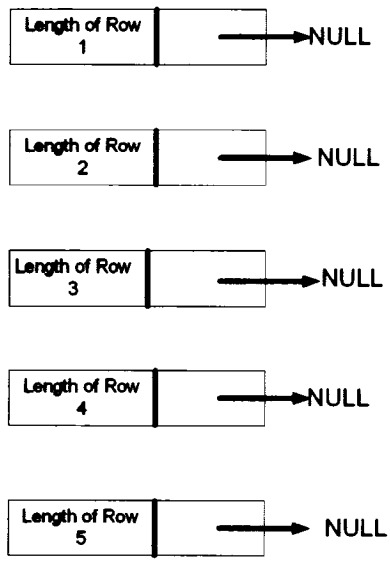

Figure 51: Each row represents a space. At the beginning, each row stores its actual length in the corresponding linked list. The same applies for columns.

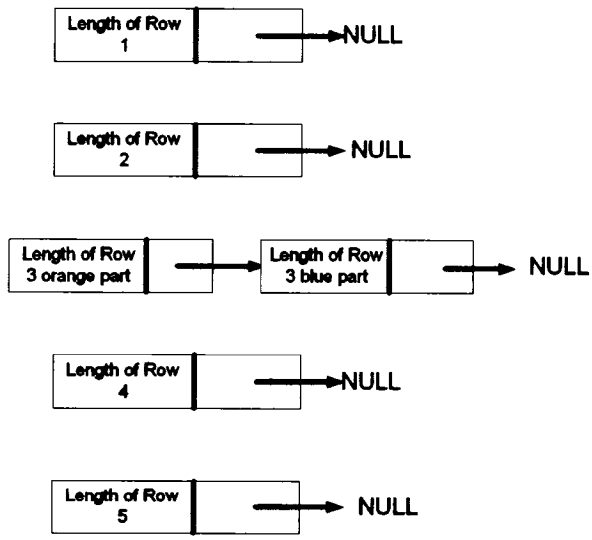

Figure 52: A demonstration of how the spaces are stored in the linked list for the rows, for example in Fig. 50. The same applies for the columns. The space becomes shorter at row 3. This happens after adding the new sensing circle.

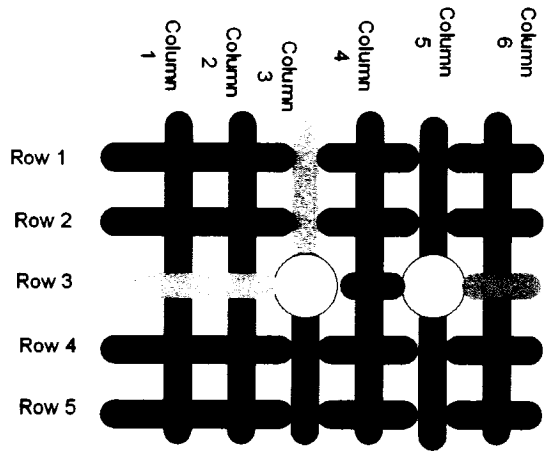

Figure 53: Shows the new sizes of the spaces in different colors. This happens after adding the second new sensing circle to the grid. The size of the space on row number 3 became smaller than before.

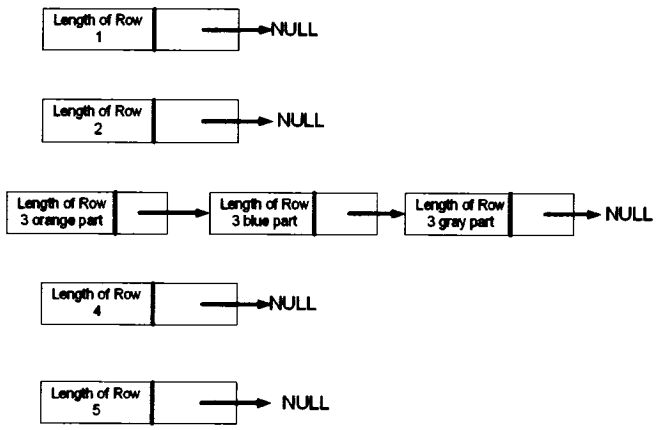

Figure 54: A demonstration of how the spaces are stored in the linked list for the rows, for example in Fig. 53. The same applies for the columns. The space became shorter at row 3. This happens after adding the second new sensing circle.

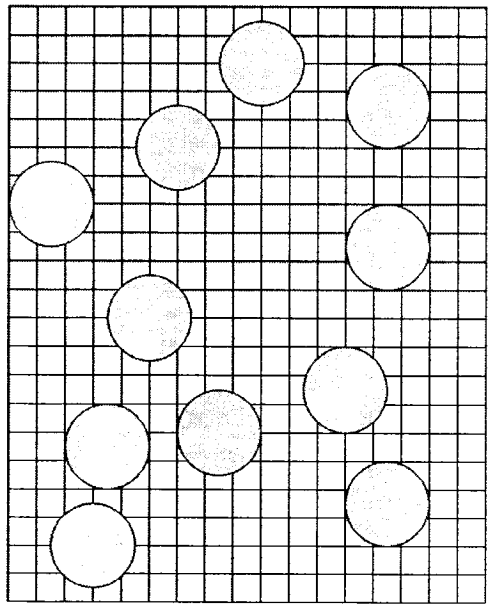

Figure 55: The remaining spaces on the grid after placing several nodes.

4.3.3 Duplicate Nodes

A demonstration for the duplication issue is shown in Fig. 56. In this figure, we identify that the sensing circle (red) is unnecessary to be added to the grid and it is considered as a duplicate node. Refer to the previous section for the coverage checking procedure. If the duplicate node were to be placed, then it would become a redundant node. Redundant nodes could increase the routing path unintentionally. They could also waste unnecessary energy. To clarify more about the duplication issue, let us take the Fig. 56 as an example. The new sensing circle is being placed randomly at column1 and column2 and at row2 and row3. We see that row2 in the selected area is already covered by some sensing circles, the same applies for row3. Now, we see that at column1, the intersected area is already covered by other sensing circles, the same applies for column2. In this case we do not place the node. This sensing circle can be used for other spaces on the grid later on.

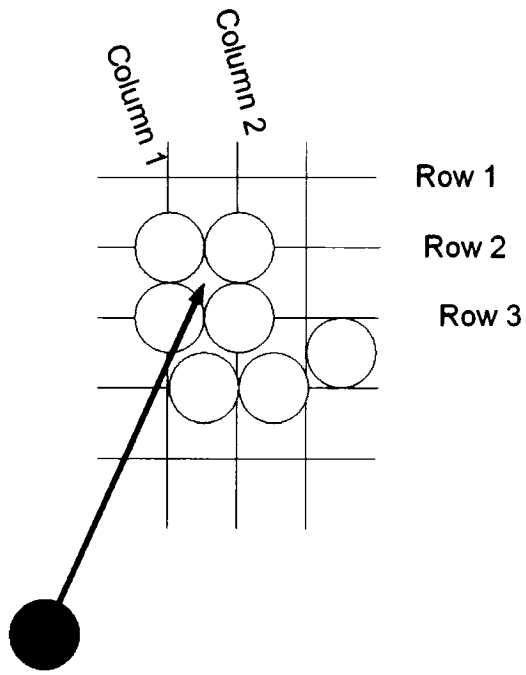

Figure 56: A sensing circle that will be placed at a location which is already filled out by other sensing circles.

4.3.4 Removing the Remaining Uncovered Spaces in the Region

We mentioned previously that in the worst case, we could have some space(s) left even if we covered all the rows and columns. Refer to Fig. 48 for one of the worst cases (blind points). To overcome this case, we represent a fact in Fig. 57.

In the above figure, we see that we have $h/2+1$ rows in a grid of h×h meters. Thus, it leads to the fact that $h/(h/2+1)=2$. This result means that between each row there is a space of two units. The same applies for the columns. Therefore, this is the only space left that we can have after filling out all the lines. We start placing nodes with a sensing radius of 18 units. After filling out all the rows and columns, we increase the radius by 1 unit to eliminate the possibility of having a space left in the grid. This means that every single point is being covered by at least $C(u, R_s)$, where R_s is the sensing radius of node u which makes a node u with a sensing circle C of radius R_s.

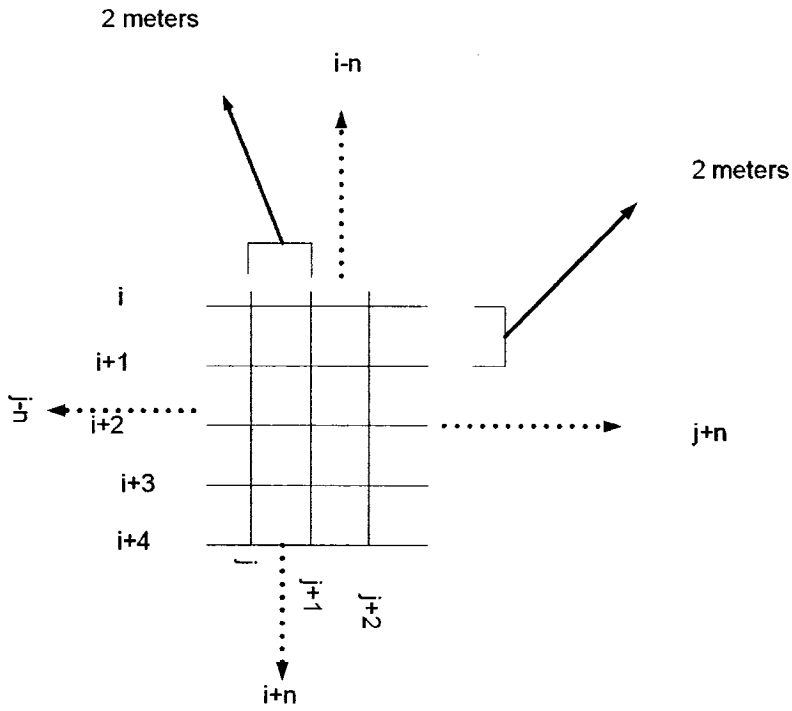

Figure 57: A representation for the rows and columns in the grid of 500X500 meters. The number of rows and columns is 251.

In [XLPH06], the *BVGF* and *GF* routing protocols are based on R_s equal to 20. Because of this, our aim is to make R_s equal to 20 meters, so that we can evaluate our routing protocols performance with the *BVGF* and *GF* routing protocols. Thus, we increase the sensing radius by 2 units instead of 1 unit.

4.3.5 The Algorithm for Covering the Spaces on the Grid and Removing Duplicate Nodes

The whole procedure in the previous sections for filling out the spaces on the grid and removing duplicate nodes are summarized in Algorithm 2. For clarity, see Figs. 50 - 53, respectively.

The algorithm assumes that the number of rows and the number of columns are denoted as $K1$ and $K2$ respectively. Each time we scan the array of lists for 'Column' and the array of lists for 'Row' to check if the sensing circle is covering a space in these two types of lists. This can be known by checking the lists that represent the

Algorithm 2 Throw_fill

Input: *An empty node set V, an array of lists for the spaces in rows 'Row', an array of lists for the spaces in columns 'Column' and R_s.*
Output: *returns a list of nodes in V and R_c .*
repeat
Let u is a random node (centered in its sensing circle) within the region of $h \times h$ meters.
for I=1 to $K1$ **do**
Scan Row I, Remove the covered area (space) by the new added sensing circle from the list of Row I.
if the list size of Row I changed and $u \notin V$ **then**
Insert u in V.
end if
end for
for I=1 to $K2$ **do**
Scan Column I, Remove the covered area (space) by the new added sensing circle from the list of Column I.
if the list size of Column I changed and $u \notin V$ **then**
Insert u in V.
end if
end for
until the array of lists 'Row' and 'Column' are empty
Let R_s=R_s+1.
R_c=2R_s

intersected lines. If the sensing circle covers a space, this space will be removed from that list. We mean by removing the space that sometimes the sensing circle covers the whole given space; thus we should remove the whole node from the linked list. We mean also that sometimes the sensing circle covers part of the space; therefore we reduce the size of the space into two new smaller spaces than the original one.

The algorithm can be further improved since it does not need to go through all columns and all rows. This improvement can be as follows. The algorithm will focus on either rows or columns since covering one of them would be sufficient. Assume that we are focusing on the rows. The algorithm should also locate the sensing circle on the grid upon placing it. Because of this information, the algorithm should know the rows that are intersected by the sensing circle. Having this kind of information will let the algorithm to only pass through these specific rows. Thus, making the algorithm much faster than before.

4.3.6 Simulation Results

As we mentioned before, the number of nodes deployed would be around one thousand nodes. Sensing radius is set to 20m. The region is 500×500m and $R_c = 2 * R_s$.

In the simulation, we measure the shortest path lengths for *UDG* graph as well as for *DAAY* with various angles. See Fig. 58. As you can see from the figure, the *UDG* graph has the highest number of short edges. The *DAAY* with θ=0 is equivalent to *UDG* graph. The more the θ increases, the less number of short edges will be in the *DAAY* subgraphs. This is due to the fact that when θ increases, the *DAAY* will eliminate more edges. Thus ending up with longer paths than before. This can be seen as we move along the figure. This is illustrated further in Fig. 59. As can be seen, the more θ increases, a smaller number of edges will be in the subgraphs due to the elimination of edges.

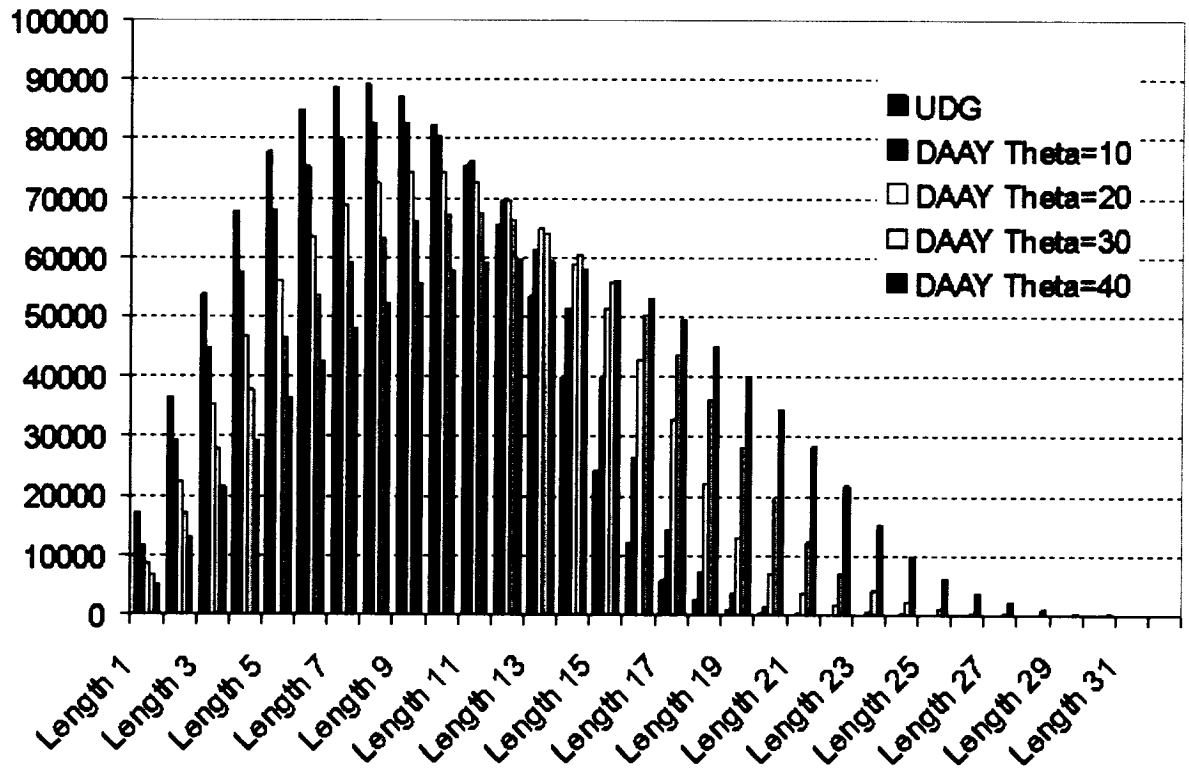

Figure 58: Lengths of shortest paths between all distinct pairs of nodes.

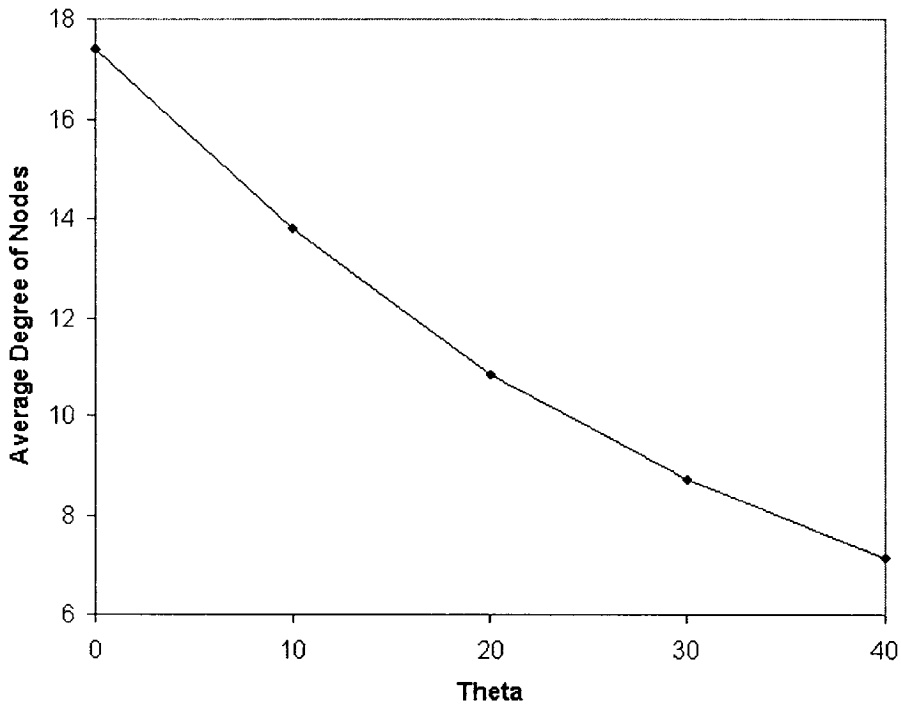

Figure 59: Average degree of nodes for $DAAY$ with various θ. Degrees are averaged over five graphs.

Chapter 5

New Position-Based Routing Algorithms

As mentioned in Chapter 3, the 2-*D DAAY* subgraph was introduced to solve some issues existed in *UDG* like the high degree of nodes. In Chapter 4, we introduced a new model that resolved some issues in sensor networks like coverage, connectivity and power usage issues. In this Chapter, the 2-*D DAAY* subgraph and the grid technique are combined together to form a graph model of our network for our new position-based routing protocols. The idea of using 3-*D DAAY* with the grid technique is out of the scope of this Thesis and needs further investigations. This Chapter lists in detail these routing protocols and how they perform on top of this topology.

As mentioned in Sec. 2.2.2, Xing *et al.* [XLPH06] claim that the *BVGF* routing protocol guarantees packet delivery. In this Chapter, we present a counter-example that shows that this protocol does not deliver the packet all the time on a general sensing covered network. To solve this issue, we propose new hybrid routing algorithms which combine the *BVGF* routing protocol with other protocols. This leads to two new enhanced versions of *BVGF* routing protocol such that these two new versions guarantee delivery. We also introduce a simplified version of *BVGF*, called Sensing Circles Close to the Line Routing Algorithm (*SCL*) replacing the Voronoi regions with the sensing ranges, and similar hybrid versions of *SCL*. After creating these versions, we compare them to each other and the *Greedy Forwarding* routing protocol

in terms of hop dilation, Euclidean dilation and time. All the algorithms used here guarantee the delivery of packets. The time measurement is proposed here for the first time for gauging the performance of the *BVGF* routing protocol (two enhanced versions) since it was not discussed by Xing *et al.* [XLPH06]. These measurements show that the two versions of the *BVGF* routing protocol are much slower than the other routing protocols as well as *GF* routing protocol. Both the hop dilation and the Euclidean dilation show some interesting properties for the all mentioned routing protocols. All these measurements are given in Secs. 5.6.1 and 5.6.2 for the directed and undirected versions of *DAAY*, respectively.

The routing algorithms implemented in the following sections are based on *UDG* and Adaptive Yao (θ=10°, 20°, 30°, 40°) subgraphs. The reason for limiting the θ is that having a θ>40°, when running *GF* and our new routing protocols on these subgraphs, will lead to a failure in packet delivery. This is because the *DAAY* will start loosing more edges which contradicts the fact that we can always have a next node with a positive progress. This may lead to local minima.

5.1 A Counterexample for *BVGF* Routing Protocol

In [XLPH06], Xing *et al.* mentioned in Lemma 1 that the Voronoi regions for the nodes are always inside their corresponding sensing circles. A proof was associated with their lemma in their paper. There is a missing case in which the neighbors of a current node can't have both criteria (positive progress and intersection with the line). Thus, *BVGF* routing protocol will fail in delivering the packet to the destination.

Lemma 2 *A current node can't always find a neighbor that has positive progress towards the destination and whose sensing circle intersects the line joining the source and the destination (The proof is illustrated in Fig. 60 as a counterexample for Theorem 5 in [XLPH06])* .

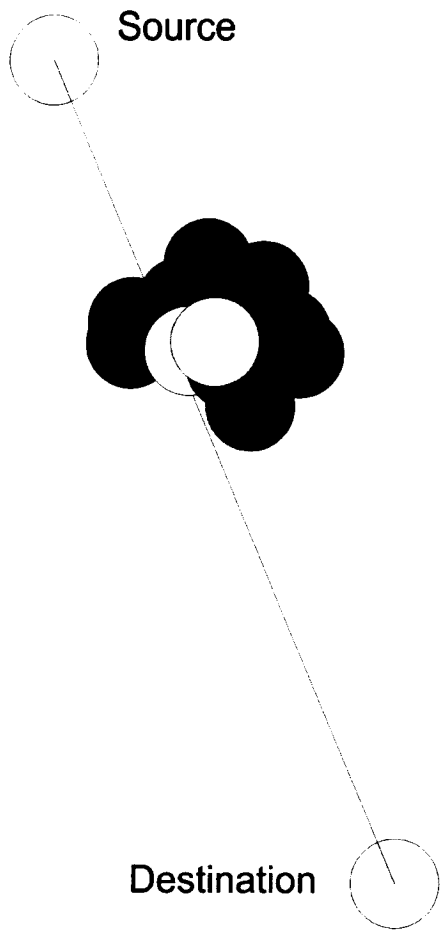

Figure 60: A counterexample for *BVGF*. This is based on *UDG* graph. A current node (white blue) is surrounded by its neighbors.

As you can see from Fig. 60, the current node (white blue) is surrounded by its neighbors with different colours. It barely crosses the source-destination line. Now even though its neighbors (gray) have positive progresses towards the destination, their sensing circles do not intersect the the source-destination line. It is obvious that the sensing circles of some nodes (magenta) do not have positive progresses towards the destination in spite of their sensing circles intersections with the line joining the source and the destination. It is interesting to note that even though the sensing circle of the neighbor (yellow) intersects the source-destination line, this neighbor has very small negative progress.

The purpose of using the sensing circles in the proof is to show that you can not have both criteria (intersection with the line and a positive progress towards the

destination) all the time. Because of this, and the fact presented in Lemma 1 in [XLPH06], definitely *BVGF* will fail sometimes to satisfy the mentioned two criteria. Then, we won't be able to move forward. Hence *BVGF* routing protocol will have a failure in the packet delivery.

5.2 GreedyClose2 routing algorithm (*GC2*)

We introduce the *GC2* routing protocol since it helps some hybrid routing protocols (mentioned later in this Chapter) recover from situations where packets are stuck. The aim here is to have a positive progress, in addition to staying as close as possible to the line between source and destination.

In this algorithm, a current node may have two best options (neighbors) towards the destination. In other words, a current node considers the first closest one to the destination and the second closest one to the destination. Among these, the current node will pick the one which has the shortest vertical projection on the line segment joining the source and the destination. Here the projection refers to the line from the neighbor of a node to the line joining the source and the destination. See Fig. 61. In this figure, the yellow node (neighbor of a current node) has the shortest projection on the line segment joining the source and the destination. The algorithm continues like this until reaching the destination. Even though *GC2* routing protocol is similar to *GF* routing protocol, it always tries to stay close to the line joining the source and the destination.

This algorithm guarantees that the packet is always delivered to the destination. This can be proven by contradiction as follows. See Figs. 62 and 63. Assume a node a (current node) covers some parts of a region. This node wont have a neighbor that has a positive progress towards the destination. Now assume that the node b has a positive progress towards the destination and is the closest node to node a as in Fig. 63. It is clear from the figure that there is a space between node a and node b. Because of the fact that every single point in a region is covered by at least one sensor node, there must be at least one sensor node that covers this space. This contradicts

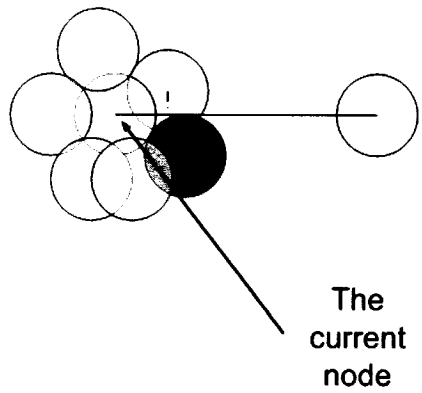

Figure 61: GreedyClose2.

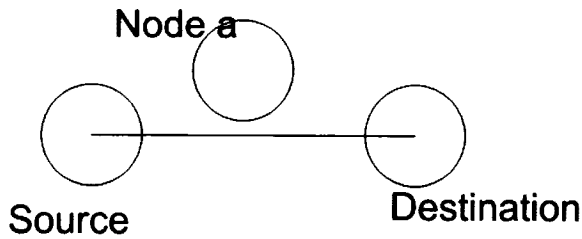

Figure 62: Node a is the current node between the source and the destination

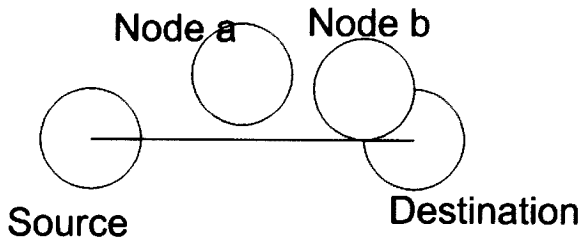

Figure 63: Node b is the closest node to node a

the fact that the node a does not have a neighbor with a positive progress towards the destination. Thus, *GC2* routing protocol will always find a sensor node that has a positive progress towards the destination. The same proof applies for the *GF* routing protocol.

The *GC2* routing protocol will be used as a recovery mode in hybrid protocols discussed later in this Chapter instead of the *GF* routing protocol. This is because even though *GF* routing protocol always picks a neighbor that is the closest neighbor to the destination, this neighbor might be far away from the line joining the source and the destination.

5.3 Smallest Angle To The Line Routing Algorithm (*SAL*)

We introduce the *SAL* routing protocol since it also helps some hybrid routing protocols (mentioned later in this Chapter) recover from places where packets are stuck. The aim here is to have a positive progress, in addition to staying close to the line between source and destination.

The idea behind this protocol is to choose the neighbor whose angle to the line joining the source and the destination is the smallest possible. The angle for one neighbor is between the line joining this neighbor and the source node, and the line joining the source and the destination. See Fig. 64. The Compass routing protocol mentioned in Sec. 2.2.1 is similar to this protocol except for the fact that the angle measured in Compass routing is between the line joining the current node and the destination node and the line joining the current node and the neighbor.

In *SAL* algorithm, the chosen neighbor that has the smallest angle, should also have positive progress towards the destination. The angle might not be the smallest angle since this algorithm picks the neighbor that has the smallest angle among all the neighbors that have positive progresses towards the destination. So there might be a neighbor that has the smallest angle but it has a negative progress, thus this

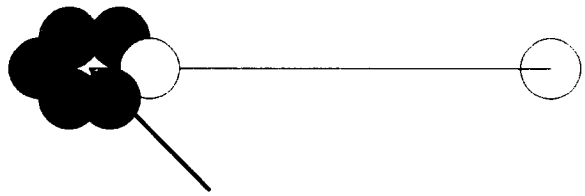

Figure 64: Smallest Angle To The Line Routing Algorithm.

algorithm will ignore this neighbor. Therefore, this protocol always stays as close as possible to the line joining the source and the destination and at the same time it will have positive progress towards the destination.

The *SAL* routing algorithm guarantees that the packet is always delivered to the destination. This proof is similar to the proof presented in Sec. 5.2.

The *SAL* routing protocol will be used as a recovery mode in hybrid protocols discussed later in this Chapter instead of *GF* routing protocol. This is because even though *GF* routing protocol always picks a neighbor that is the closest neighbor to the destination, this neighbor might be far away from the line joining the source and the destination.

5.4 Two Hybrid Versions of *BVGF* Routing Algorithm

As mentioned in Sec. 2.2.2, the *BVGF* routing algorithm chooses the next hop if this hop's Voronoi region intersects the line joining the source and the destination; and that has a positive progress towards the destination.

To overcome the counterexample presented in Sec. 60, *BVGF* routing protocol is combined with either *SAL* algorithm or *GC2* algorithm as hybrid versions of the original *BVGF* routing protocol. These hybrid versions are presented in the following subsections:

5.4.1 *BVGF:SAL* Routing Algorithm

When *BVGF* encounters a failure in the packet delivery at a particular node, *BVGF* switches to *SAL* algorithm once and then it continues as before. As mentioned in Sec. 5.3, *SAL* algorithm running on one node attempts to choose among all its neighbors, the neighbor that has the smallest angle possible. *SAL* mode allows *BVGF* to always be as close as possible to the line segment joining the source and the destination in order to avoid any failure in the packet delivery. Since *BVGF:SAL* routing algorithm depends on *SAL* routing protocol as a recovery mode, it will guarantee that the packet is always delivered to the destination. This can be proven by cases as follows.

Case 1: The *BVGF* routing protocol guarantees, by Theorem 5 in [XLPH06], that the packet is always delivered to the destination if there is always at least one neighbor that has a positive progress towards the destination and its Voronoi region intersects the line joining the source and the destination.

Case 2: The *BVGF* routing protocol switches to the *SAL* routing protocol if it does not have any neighbor such that its Voronoi region intersects the line segment joining the source and the destination, and has positive progress towards the destination. The proof that *SAL* routing algorithm guarantees packet delivery was presented in Sec. 5.3.

Thus, the *BVGF:SAL* routing protocol guarantees that the packet is always delivered to the destination.

5.4.2 *BVGF:GC2* Routing Algorithm

Another option when *BVGF* encounters a failure is to switch to *GC2* algorithm. As mentioned in Sec. 5.2, *GC2* algorithm running on one node finds the first closest neighbor to the destination and the second closest neighbor to the destination. Among these, the algorithm picks the one which has the shortest projection on the line segment joining the source and the destination. This option would make *BVGF*

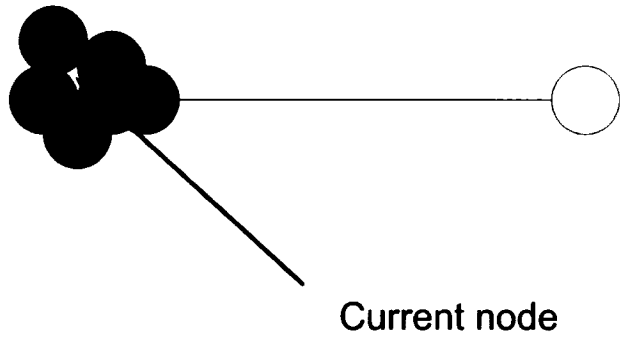

Figure 65: *SCL* algorithm.

to have less failure since it is trying to move forward towards the destination while maintaining the closeness to the line condition. Since *BVGF:GC2* routing algorithm depends on *GC2* routing protocol as a recovery mode, it will guarantee that the packet is always delivered to the destination. This can be proven by cases and the proof is similar to the proof presented in Sec. 5.4.1 but instead of *SAL* as a recovery mode, *GC2* will be used. The proof that *GC2* routing algorithm guarantees packet delivery was presented in Sec. 5.2.

5.5 Sensing Circles Close to the Line Routing Algorithm (*SCL*)

As mentioned by Xing *et al.* [XLPH06], the Voronoi region of a node is contained in its sensing circle. Because of this fact, we introduce a simplified version of *BVGF* routing protocol called *SCL* routing protocol by replacing the Voronoi regions with the sensing ranges of the nodes.

The routing algorithm works as follows. A current node checks if the sensing circles of its neighbors intersect the line segment joining the source and the destination. After having this set of nodes, the current node picks the one which is the closest to the destination. See Fig. 65. The algorithm continues like this until reaching the destination.

The *SCL* routing protocol in general fails less frequently than *BVGF* routing protocol. This is because each Voronoi region (including its boundary) is contained

within its corresponding sensing circle, there will be a greater possibility for the nodes that do have a positive progress towards the destination to have an intersection between the line segment and their sensing circles rather than using the Voronoi regions of the nodes.

In spite of the fact that this algorithm uses the sensing ranges instead of the Voronoi regions, this algorithm still does not guarantee packet delivery. This is because the counterexample (Fig. 60) mentioned for *BVGF* also applies to *SCL* routing algorithm. To overcome this problem, *SCL* routing protocol is combined with either *SAL* algorithm or *GC2* algorithm as hybrid versions of the original *SCL* routing protocol. These hybrid versions are the following.

5.5.1 *SCL:SAL* Routing Algorithm

When *SCL* encounters a failure in the packet delivery at a particular node, it switches to the *SAL* algorithm once and then continues as before with *SCL*. *SAL* algorithm running on one node attempts to choose among all its neighbors that have positive progresses, the neighbor that has the smallest angle possible. *SAL* mode allows *SCL* to always be as close as possible to the line segment joining the source and the destination in order to avoid any failure in packet delivery. Although *SCL:SAL* is similar to *BVGF:SAL* in using the same recovery mode, *SCL:SAL* routing protocol performs much faster than *BVGF:SAL* routing protocol since it does not need to calculate the Voronoi regions for the sensor nodes.

The *SCL:SAL* routing algorithm guarantees packet delivery. This can be proven by cases as the following:

Case 1: The *SCL* routing protocol guarantees that the packet is always delivered to the destination if there is always at least one neighbor that has a positive progress towards the destination and its sensing circle intersects the line joining the source and the destination.

Case 2: The *SCL* routing protocol switches to the *SAL* routing protocol if it does not have any neighbor such that its sensing circle intersects the line segment joining

the source and the destination and has positive progress towards the destination. The proof that *SAL* routing algorithm guarantees packet delivery was presented in Sec. 5.3. Thus, the *SCL:SAL* routing protocol guarantees that the packet is always delivered to the destination.

5.5.2 *SCL:GC2* Routing Algorithm

Another option when *SCL* routing algorithm encounters a failure is to switch to the *GC2* routing algorithm. As mentioned in Sec. 5.2, *GC2* algorithm running on one node finds the first closest neighbor to the destination and the second closest neighbor to the destination. Among these, the algorithm picks the one which has the shortest projection on the line segment joining the source and the destination. This option would lead *SCL* to fail less often since it is trying to move forward towards the destination while maintaining the closeness to the line condition. Even though *SCL:GC2* is similar to *BVGF:GC2* in using the same recovery mode, *SCL:GC2* routing protocol performs much faster than *BVGF:GC2* routing protocol for the same reason mentioned in Sec. 5.5.1.

The *SCL:GC2* routing algorithm always guarantees packet delivery. The proof is similar to the proof mentioned in Sec. 5.5.1 but instead of using *SAL* as a recovery mode, the *GC2* routing algorithm is used.

5.6 Evaluating the Performance of the New Routing Algorithms in 2-D Environment.

The experiments are done based on the stochastic communication model. In this model, we place around 1000 nodes randomly in a 500m×500m region in 2-D. This region is covered by a set of active nodes. The transmission radius of each host is assumed to be of a fixed size (20 m). The largest connected component is exactly equal to the number of nodes since the graph is strongly connected. We repeat this simulation five times. Each time, every single node sends a packet to every other node

in the network. We perform the routing protocols based on *UDG*, directed *DAAY*, and undirected *DAAY* subgraphs. The θ in *DAAY* in both directed and undirected versions has the values 10°, 20°, 30°, and 40°. The routing protocols running on these graphs achieve 100% delivery rate.

5.6.1 The Performance of the New Algorithms on Directed Adaptive Yao Subgraph (D-*DAAY*)

In this section, we evaluate the performance of our new routing protocols and compare their performance with *GF* and each other. The experiments show the Euclidean dilation, the hop dilation, and the time for the routing algorithms. The experiments are done based on *D-DAAY*. We vary the value of θ for *D-DAAY*. See Figs. 66 - 70, respectively.

You can see from the Fig. 66 that even though *GF* routing protocol outperforms other routing protocols, our new routing protocols are better than *BVGF:GC2* and *BVGF:SAL* routing protocols. The more θ increases, the more the average dilation increases for all routing algorithms. The reason is that when θ increases, the network will have smaller node degrees. Thus, we will start loosing some edges which might be part of shorter routing paths to some destination nodes. This loss of node degree will force the routing protocols to take longer paths than before. Another thing, when θ increases, the number of switches to other routing protocols (*SAL* and *GC2* routing protocols) increases, since a node will start loosing some edges with neighbors which may have had intersection with the line segment by their Voronoi regions or their sensing circles. You can see from the figure also, that the two hybrid versions of *BVGF* are almost the same in their behavior. The same applies for the two hybrid versions of *SCL* routing protocol.

Now let us see Fig. 67. This figure shows the Euclidean dilation for various routing protocols. It is clear that even though the enhanced versions of *BVGF* routing protocol have lower Euclidean dilation compared to other routing protocols; *SCL:GC2* and *SCL:SAL* routing protocols are close to *BVGF* versions especially as the angle of

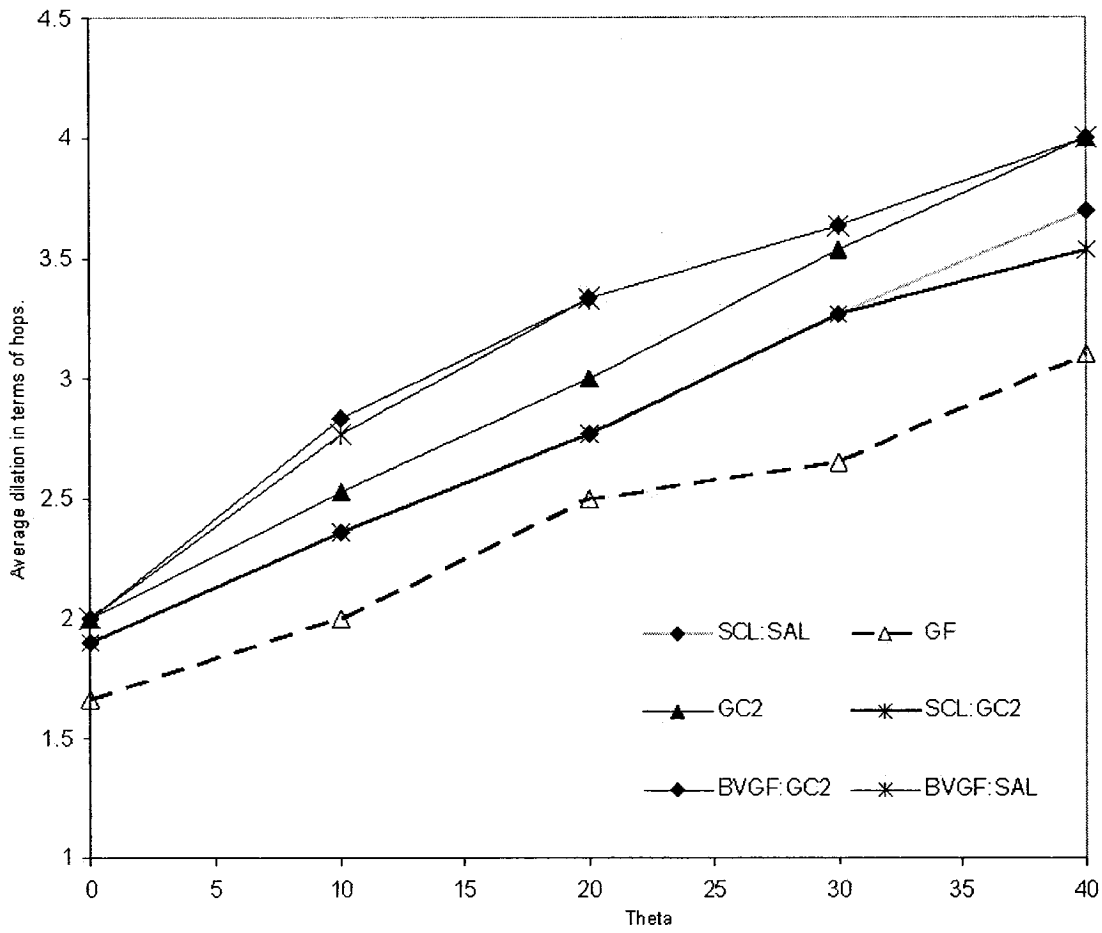

Figure 66: The hop dilation for different routing algorithms.

D-DAAY graph increases. Our routing protocols achieve a much better performance than the *GF* routing protocol. As you can see, the Euclidean dilation increases when θ increases. The number of switches also increases with the increase of θ. The reasons for these two behaviors are the same reasons mentioned above. You can see from the figure also, that the two enhanced versions of *BVGF* are almost the same in their behavior. The same applies for the two enhanced versions of *SCL* routing protocol.

Now let us look at the time measurement. See Figs. 68 and 69. In these figures we measure the time in hours for the routing algorithms on *D-DAAY* for different values of θ. From these figures, you can see that our routing protocols and *GF* routing protocol outperform the two enhanced versions of *BVGF* routing protocol.

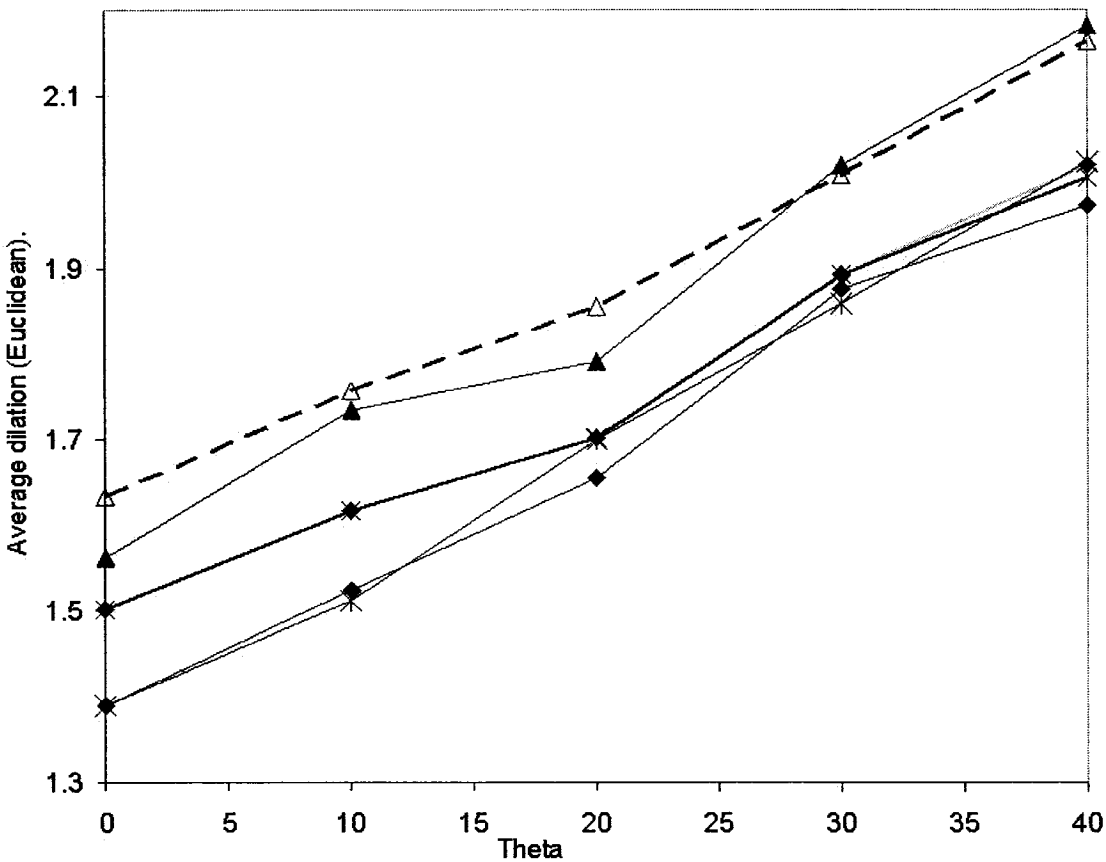

Figure 67: The Euclidean dilation for different routing algorithms. The legend is the the same as Fig. 66.

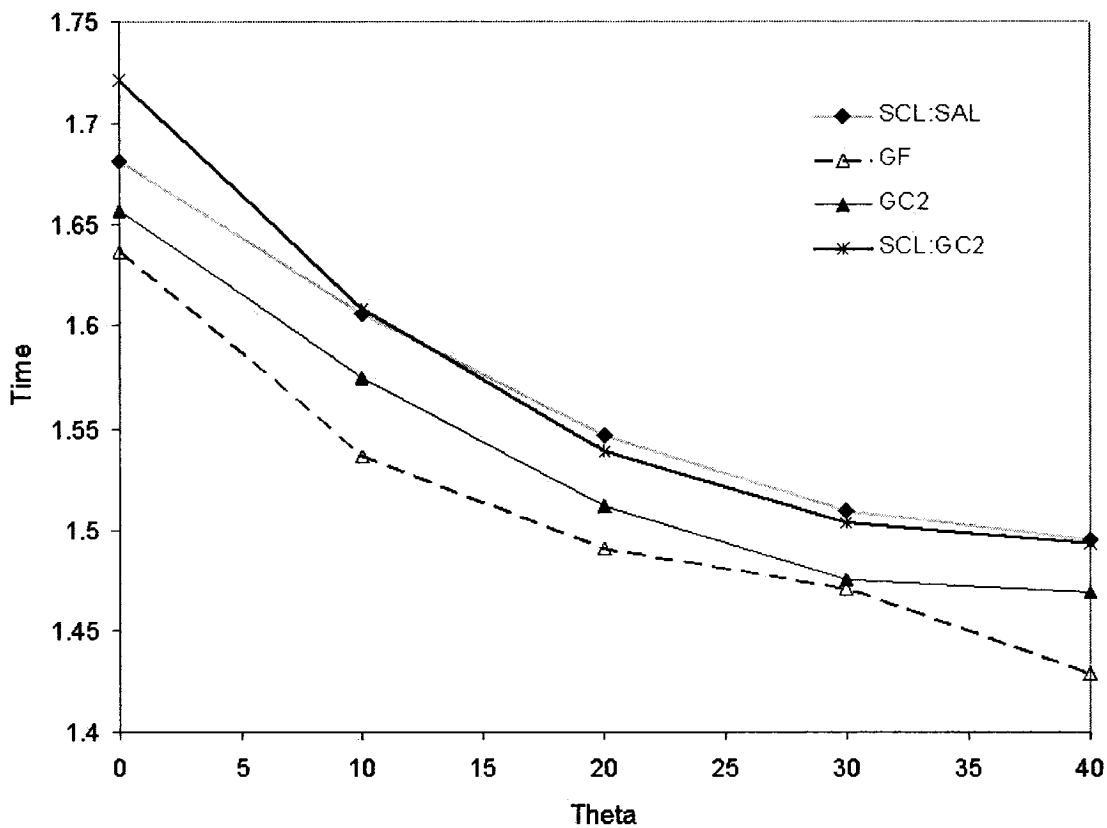

Figure 68: The time for various routing algorithms.

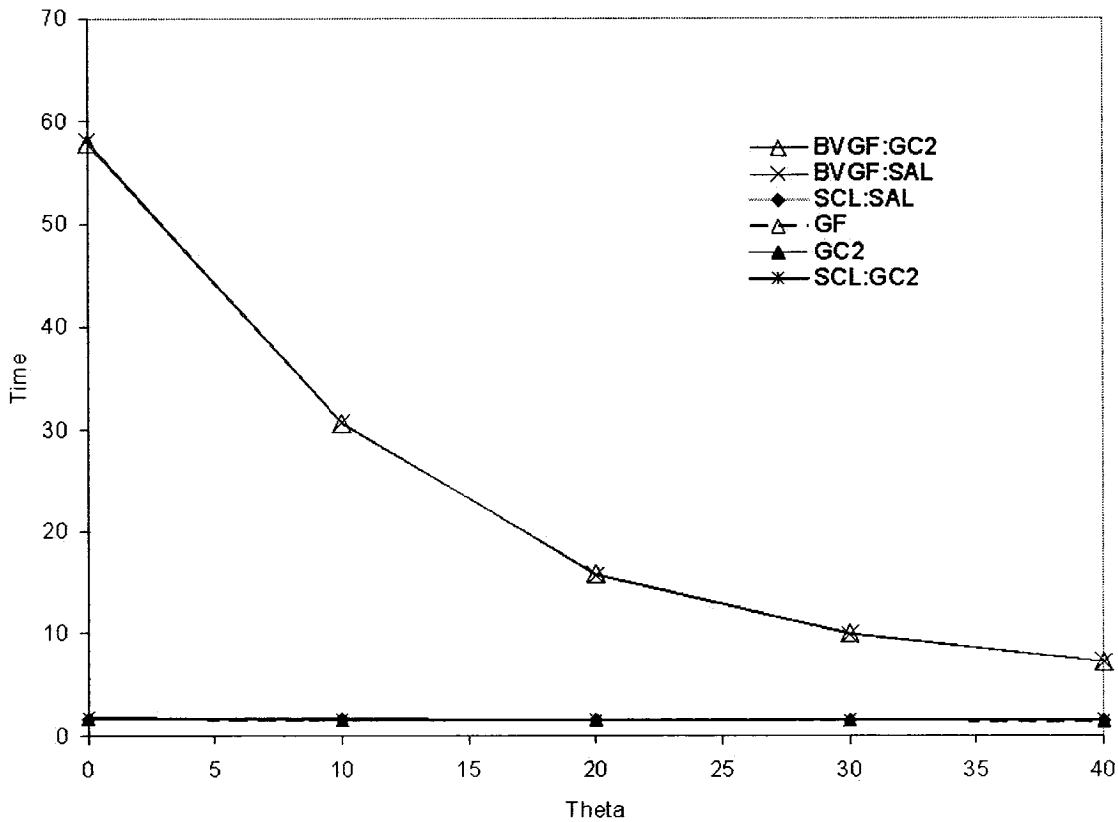

Figure 69: The time for *BVGF:SAL* and *BVGF:GC2* compared to the other routing protocols.

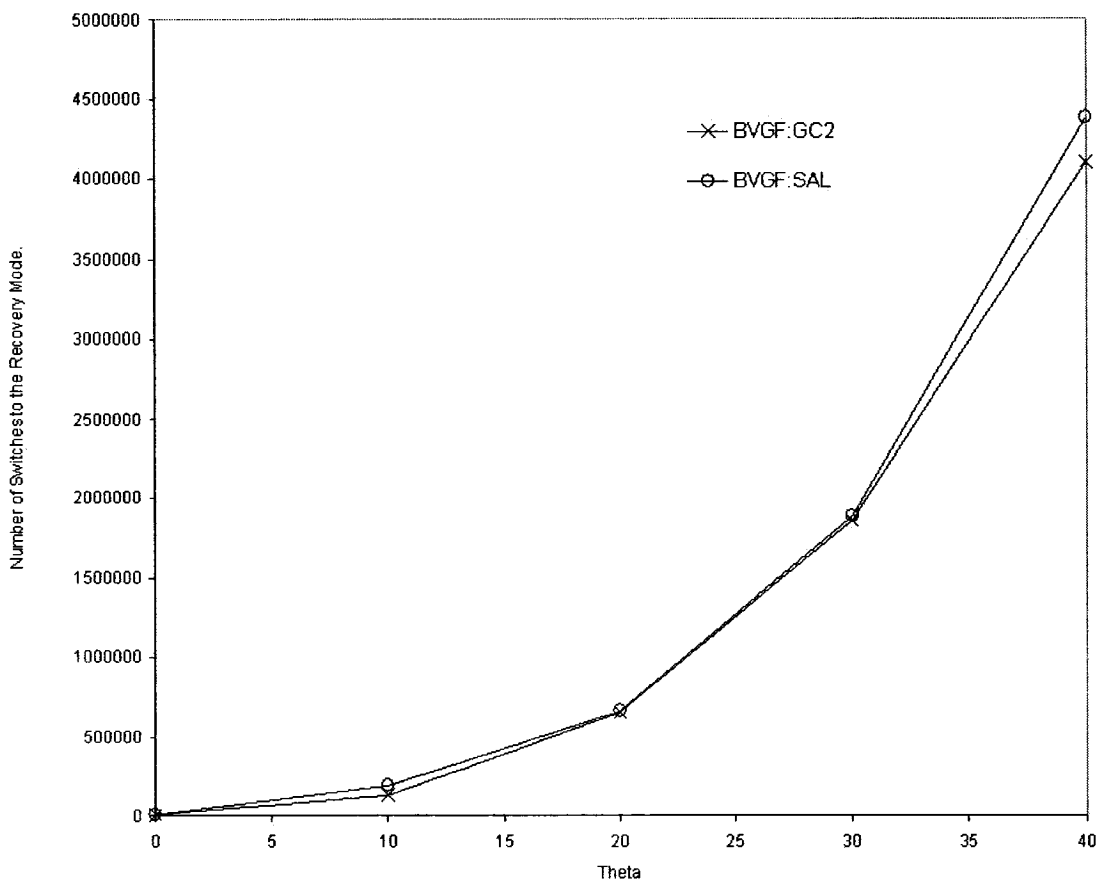

Figure 70: The number of switches to recovery modes that *BVGF* can make.

In other words, our routing algorithms and *GF* routing protocol are much faster than the two versions of *BVGF* routing protocol. You can see from the Fig. 69 that the two hybrid versions of *BVGF* routing protocol are very close to each other in their time performance. From Fig. 70, the hybrid versions of *BVGF* routing protocol are almost equal in the number of switches to recovery modes. Note that when θ=0, the *BVGF:GC2* switches 4564 times to *GC2* routing protocol, and the *BVGF:SAL* switches 4624 times to *SAL* routing protocol. Also from Fig. 68, our routing protocols are close to each other in their time measurement. They are also close to the *GF* routing protocol.

5.6.2 The Performance of the New Algorithms on Undirected Adaptive Yao Graph (*DAAY*)

In this section, we evaluate the performance of our new routing protocols and compare their performances to *GF* and each other. The experiments show the Euclidean dilation, the hop dilation, and the time measured in hours for the routing algorithms. The experiments are done based on *DAAY*. We vary the value of θ for *DAAY*. See Figs. 71 - 75, respectively.

The previous analysis for the figures in the directed *DAAY* section is almost the same for these figures. Note from Fig. 75 that when θ=0, the *BVGF:GC2* switches 4564 times to *GC2* routing protocol, and the *BVGF:SAL* switches 4624 times to *SAL* routing protocol. The idea here is to show that there is no big difference between the mentioned routing algorithms on *D-DAAY* and *DAAY* in terms of Euclidean dilation, hop dilation, and the time in terms of hours. This is because as we mentioned before, every point in the region in 2-*D* is covered by at least one node's sensing circle, and $R_c = R_s * 2$. In other words, our wireless network is a strongly connected and it is considered a dense network. The small differences are discussed in the following.

The Euclidean and hop dilation for the routing protocols in *D-DAAY* shown in Figs. 71 and 72, respectively, will be a little bit higher compared to the routing protocols on *DAAY*. This is because the more θ increases, the more edges are thrown

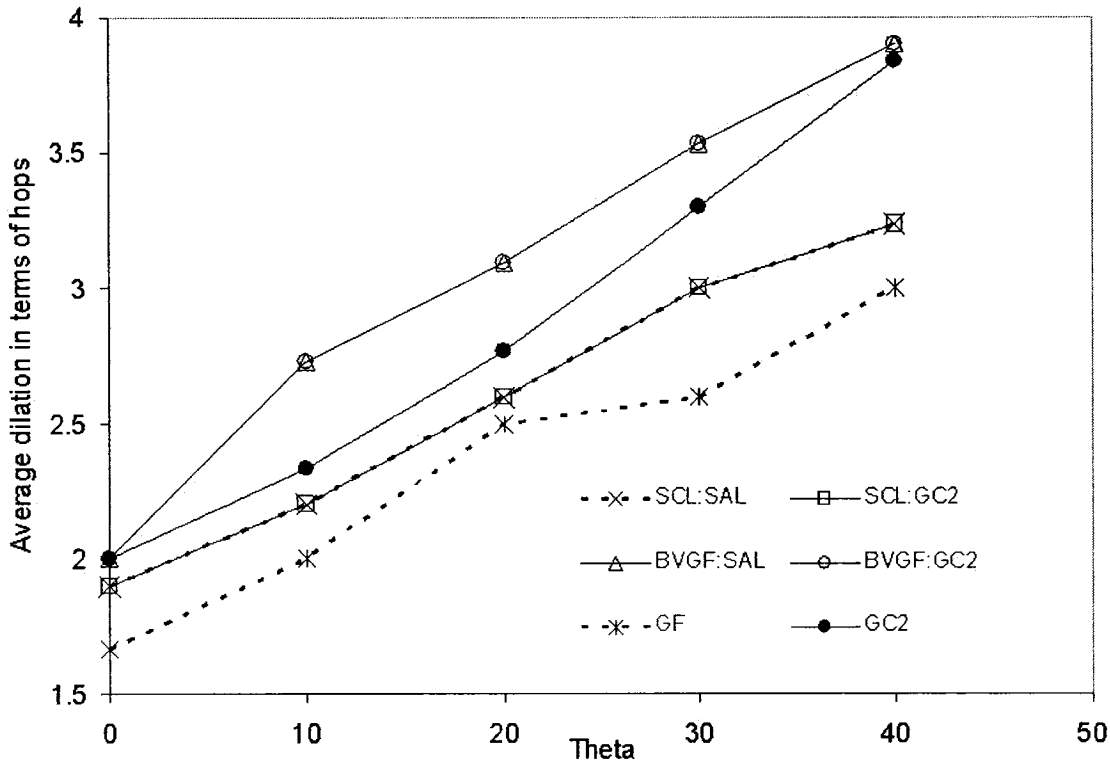

Figure 71: The hop dilation for different routing algorithms.

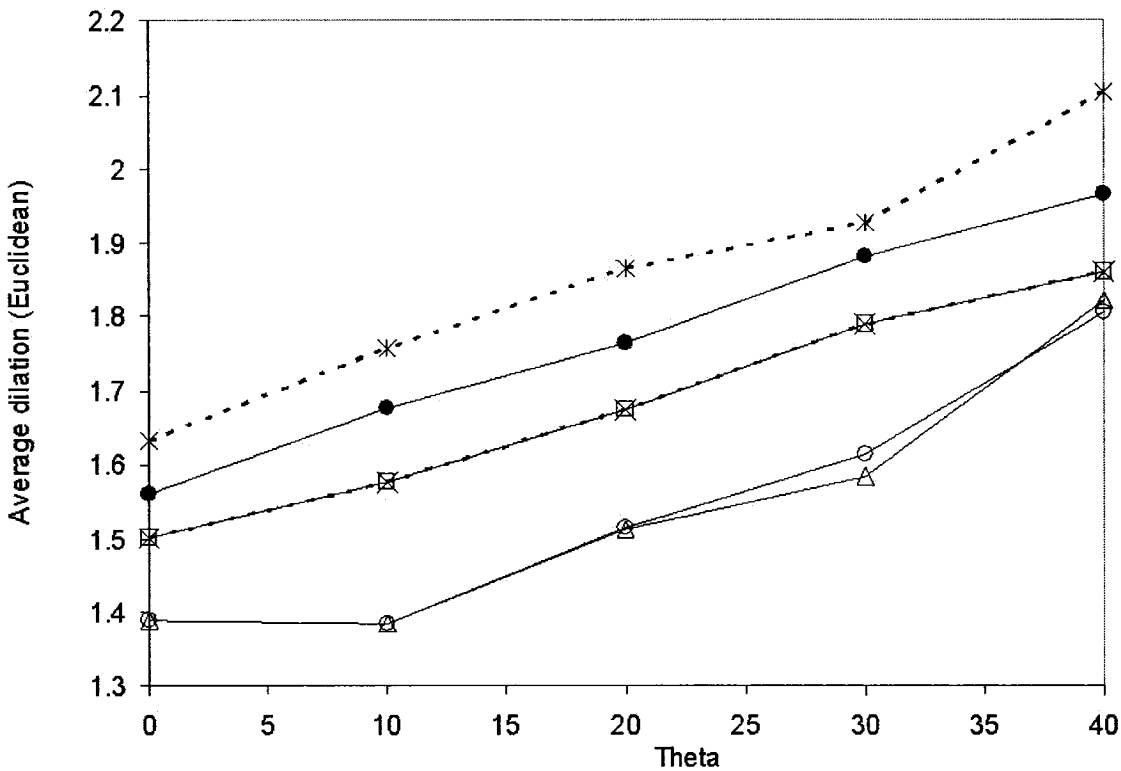

Figure 72: The Euclidean dilation for different routing algorithms. The legend is the the same as Fig. 71.

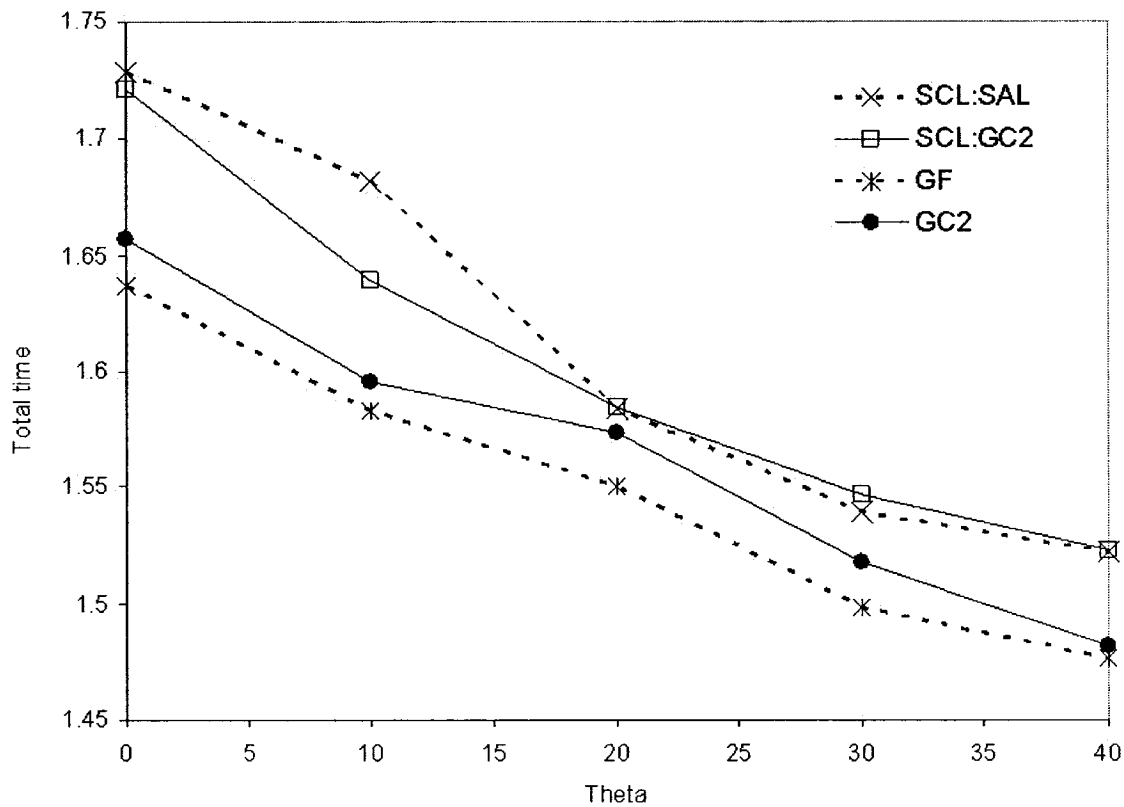

Figure 73: The time for different routing algorithms.

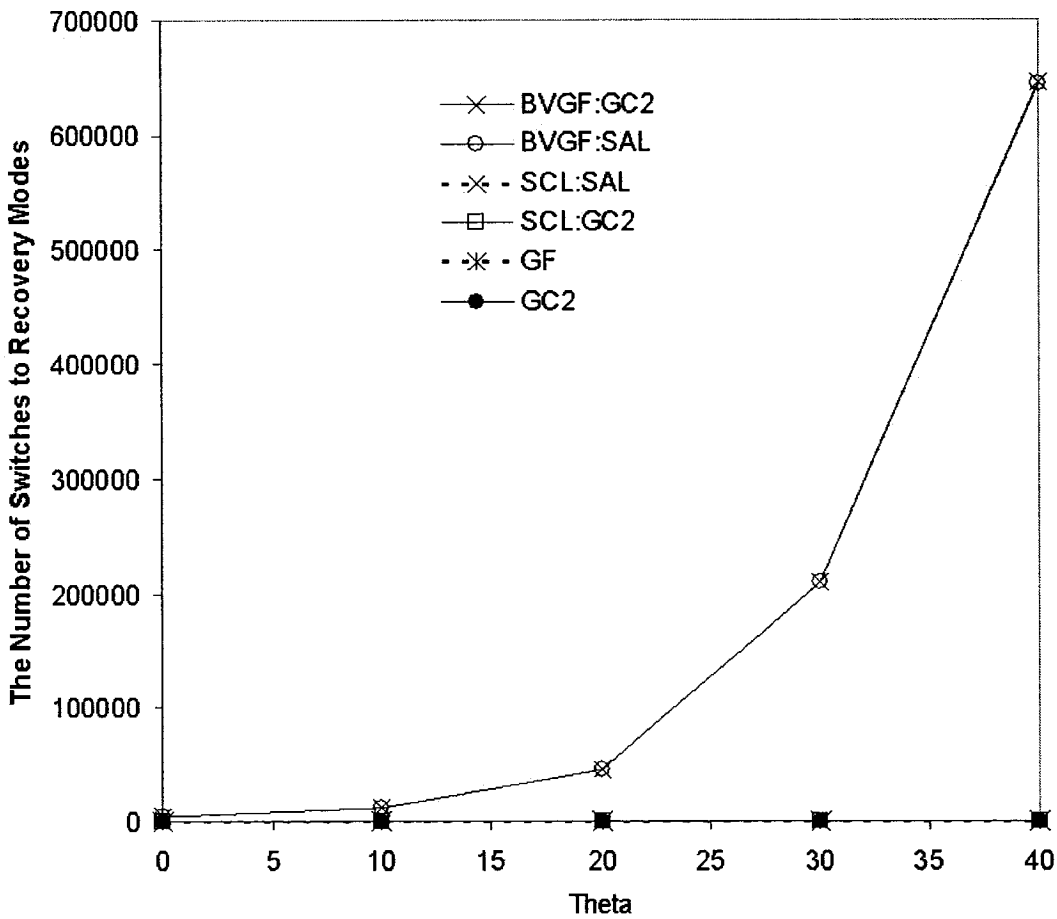

Figure 74: The time for *BVGF:SAL* and *BVGF:GC2* compared to the other routing protocol.

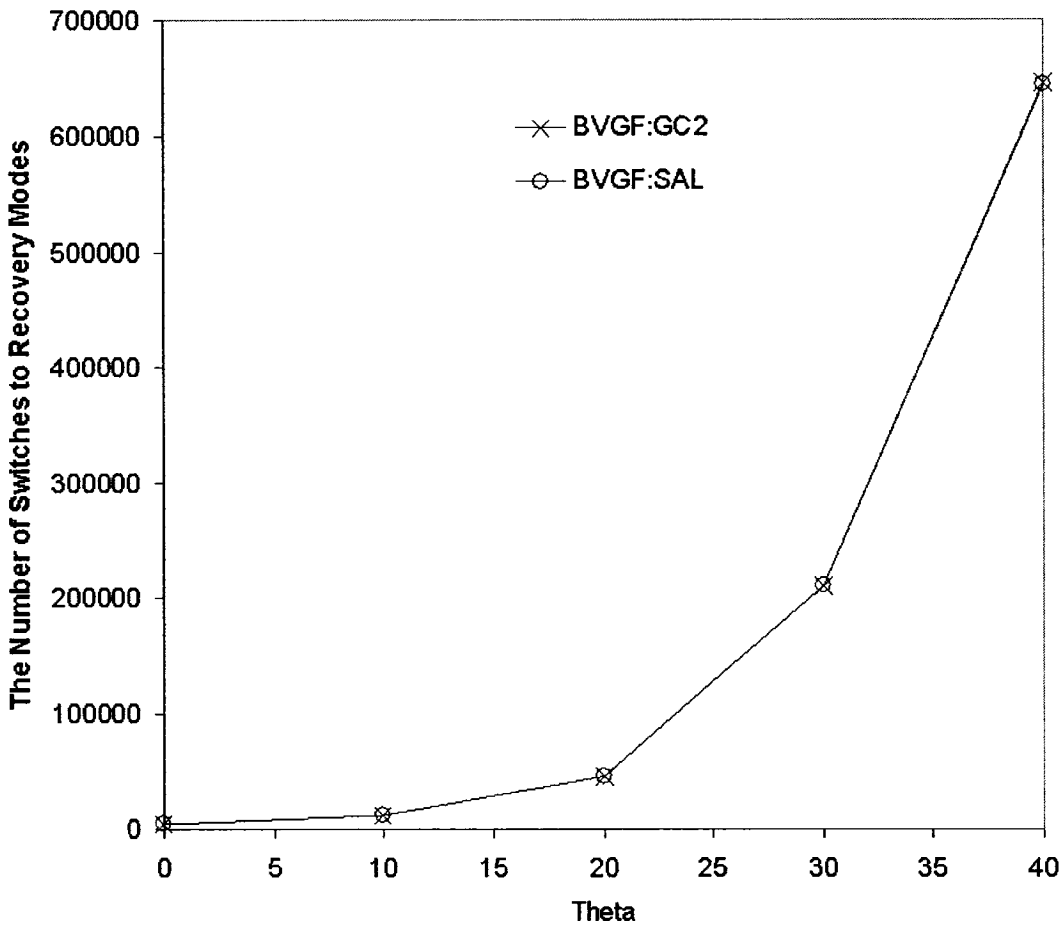

Figure 75: The number of switches to recovery modes that *BVGF* can make.

away, which make the routing paths longer than before. In addition, to this, for the directed version, there will be fewer out-edges from a current node to its neighbors than the same node in *DAAY*. In addition, even though the undirected version of *DAAY* throws away some edges like *D-DAAY* as θ increases, the remaining edges are two-way edges. According to this, *DAAY* will have a higher possibility for shorter paths than *D-DAAY*, which improve the Euclidean/hop dilation.

Another difference is the time measurement in Figs. 73 and 74. Even though the routing protocols on *DAAY* outperform the routing protocols on *D-DAAY* in terms of Euclidean and hop stretch factors, the time spent by these protocols on *DAAY* is a little bit higher than the time spent by these routing protocols on *D-DAAY*. This is because the routing protocols (whose running times are proportional to the degree of the current node.) running on *DAAY* will take a longer time compared to the routing protocols on *D-DAAY*, since a current node on *DAAY* graph has more neighbors compared to the same current node's neighbors on *D-DAAY* graph.

Chapter 6

Conclusion and future work

In this Thesis, we presented a class of *Yao*-type graphs that combine the advantages of both the *HSP* subgraph and the *Yao* subgraph by permitting control over the degree of the subgraph while being orientation-invariant. Indeed, the degree control is continuous since any cone angle less than $\theta_m(s, |uz|)$ can be used as differs from the *Yao* graph where only a discrete set of cone angles (π/k) are possible. In addition, unlike the *Yao* subgraph, *DAAY* subgraph can be easily extended to three dimensional *UDG*s. The 2-*D* and 3-*D* versions of *DAAY* come in two types: directed and undirected graphs. The experimental results presented here evaluated both 2-*D* and 3-*D* *DAAY* graphs based on several metrics, some of these are: Euclidean stretch factor, hop stretch factor and degree of nodes.

We introduced a new approach for obtaining a sensing-covered network in 2-*D* environment. This approach is referred to the grid technique and it is based on the Unit Disk Graph (*UDG*). This technique makes sure that a region is fully covered by the least number of sensor nodes. Thus, creating a fully connected network and saving up some energy. This saving would prolong the lifetime of the network. In order to create variations of this graph, we implemented the mentioned 2-*D* *DAAY* based on this graph.

Based on the above topology (*DAAY* combined with the grid technique), we proposed new position-based routing protocols and compared them to *GF* routing protocol. Among these routing protocols, two new versions of *BVGF* and *SCL* routing

protocols were created. The reason for these versions is that our experiments discovered a strong case which shows that *BVGF* and *SCL* routing protocols do not always guarantee packet delivery. We compared our new position-based routing protocols to *GF* routing protocol in term of hop stretch factor, Euclidean stretch factor, and the time taken for packet delivery. The experimental results show that even though *GF* routing protocol outperforms our routing protocols in term of hop stretch factor, the two enhanced versions of *BVGF* routing protocol do not outperform the other routing protocols. The experiments also show that even though the two versions of *BVGF* routing protocol outperform the other routing protocols in term of Euclidean stretch factor, these other protocols still outperform *GF* routing protocol. Some of our routing protocols become very similar in their performance (Euclidean stretch factor) to our two hybrid versions of the *BVGF* routing protocol as θ increases in *DAAY* subgraphs, in both directed and undirected versions.

The future work would consider several things. Finding a much more efficient way than the grid technique to cover the entire network which is based on the incrementation approach. Moving the cone along the line segment between the node and its nearest neighbor in 2-*D DAAY*, when dealing with sensor networks. The grid technique still needs to be implemented in 3-*D* so that it can be combined with the 3-*D DAAY*. Power metric should also be measured for our 2-*D* sensor network and its extension to 3-*D* environment. Also an interesting work would include a comparison between incrementation and decimation approaches in their performance. This work can be extended by combining both of these approaches.

Bibliography

[BCC+07] P. Bose, P. Carmi, M. Couture, M. Smid, and D. Xu. On a family of strong geometric spanners that admit local routing strategies. *10th Workshop on Algorithms and Data Structures (WADS)*, 2007.

[BCSW98] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward. A distance routing effect algorithm for mobility (DREAM). *In Proc. of 4th ACM/IEEE Conference on Mobile Computing and Networking (Mobicom 98)*, pages 76–84, 1998.

[BDEK01] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick. On the spanning ratio of gabriel graphs and beta-skeletons. *SIAM Journal on Discrete Mathematics*, pages 479–493, 2001.

[BFNO03] L. Barrière, P. Fraigniaud, L. Narayanan, and J Opatrny. Robust position-based routing in wireless ad-hoc networks with irregular transmission ranges. *Wireless Communications and Mobile Computing Journal*, pages 141–153, 2003.

[BGM04] P. Bose, J. Gudmundsson, and P. Morin. Ordered theta graphs. *Computational Geometry: Theory and Applications*, vol. 28(1):11–18, 2004.

[BMSU01] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. *Wireless Networks*, vol. 7(6):609–616, 2001.

[CDK+05] E. Chavez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, H. Tejeda, and J. Ur-rutia. Half-space proximal: A new local test for extracting a bounded dilation spanner. *In Proc. of the International Conference On Principles of Distributed Systems (OPODIS 2005)*, vol. 3974 of LNCS:235–245, 2005.

[Epp96] D. Eppstein. Spanning Trees and Spanners. *Technical Report ICS-TR-96-16*, 1996.

[Epp00] D. Eppstein. Spanning trees and spanners. *In Handbook of Computational Geometry, J.-R. Sack and J. Urrutia, Eds. Amsterdam: Elsevier Science Publishers B.V. North-Holland*, pages 425–461, 2000.

[FAESH07] T. Fevens, A.E. Abdallah, T. El Salti, and L. Harutyunyan. A Class of Orientation-Invariant Yao-type Subgraphs of a Unit Disk Graph. *Dial M-POMC Accepted*, 2007.

[Fin87] G. Finn. Routing and Addressing Problems in Large Metropolitan- Scale Internetworks. *Technical Report ISI Research Report ISU/RR-87-180, Inst. for Scientific Information*, 1987.

[For92] S. Fortune. Voronoi diagrams and Delaunay triangulations. *Computing in Euclidean geometry. D.Z. Du, F. Hwang eds, World Scientific*, pages 193–233, 1992.

[GS69] K.R. Gabriel and R.R. Sokal. A new statistical approach to geographic variation analysis. *Syst. Zoology*, vol. 18:259–278, 1969.

[GSB03] S. Giordano, I. Stojmenovic, and L. Blazevic. Position based routing algorithms for ad hoc networks: A Taxonomy. *Ad Hoc Wireless Networking*, pages 103–136, X. Cheng, X. Huang and D.Z. Du (eds.), Kluwer, 2003.

[HP98] Z. Haas and M. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. *In Proceedings of the ACM SIGCOMM'98*

Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pages 167–177, 1998.

[HT03] C.-F. Huang and Y.-C. Tseng. The Coverage Problem in a Wireless Sensor Network. *In WSNA 03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 115–121. ACM Press, 2003.

[HTW07] C.-F. Huang, Y.-C. Tseng, and H.-L. WU. Distributed protocols for ensuring both coverage and connectivity of a wireless sensor network. *ACM Transactions on Sensor Networks*, vol. 3(1), Article no. 5, 2007.

[JMQ+01] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Clausen. Optimized Link State Routing Protocol. *Internet draft, draftietf-manet-olsr-04.txt,work in progress*, 2001.

[KFO05] G. Kao, T. Fevens, and J. Opatrny. Position-Based Routing on 3-D Geometric Graphs in Mobile Ad Hoc Networks. *Proceedings of the 17th Canadian Conference on Computational Geometry (CCCG05)*, pages 88–91, 2005.

[KG92] J.M. Keil and C.A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete and Computational Geometry*, 7:13–28, 1992.

[KK00] B. Karp and H.T. Kung. GPSR:Greedy Perimeter Stateless Routing for Wireless Networks.Proc. Sixth ACM Int'l Conf. Mobile Computing and Networking (MobiCom '00). pages 243–254, 2000.

[KSU99] E. Kranakis, H. Singh, and J. Urrutia. Compass Routing on Geometric Networks. *Proc. 11th Canadian Conf. Computational Geometry*, pages 51–54, 1999.

[KV00] Y.B. Ko and N.H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks, 6*, pages 307–321, 2000.

[LWW02] X.-Y. Li, P.J. Wan, and Y. Wang. Power efficient and sparse spanner for wireless ad hoc networks. *In Proc. of IEEE Int. Conf. on Computer Communications and Networks (ICCCN01)*, pages 564–567, 2002.

[MKPS01] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. *In IEEE INFOCOM*, pages 1380–1387, 2001.

[MKQP01] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. *In ACM Intl Conf. on Mobile Computing and Networking (MobiCom)*, pages 139–150, 2001.

[MS80] D.W. Matula and R.R. Sokal. Properties of Gabriel Graphs Relevant to Geographic Variation Research and the Clustering of Points in the Plane. *Geographical Analysis*, 12(3):205–222, 1980.

[MWH01] M. Mauve, J. Widmer, and H. Hartenstein. A Survey on Position-Based Routing in Mobile Ad Hoc Networks. *IEEE Network*, vol. 15(6):30–39, 2001.

[PB94] CE Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *Computer Communication Review*, vol. 24(4):234–244, 1994.

[PC97] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *In IEEE INFOCOM*, vol. 3:1405–1413, 1997.

[SL01] I. Stojmenovic and X. Lin. Loop-Free Hybrid Single-Path/ Flooding Routing Algorithms with Guaranteed Delivery for Wireless Networks. *IEEE Trans. Parallel and Distributed Systems*, vol. 12(10):1023–1032, 2001.

[Sto99] I. Stojmenovic. Voronoi diagram and convex hull based geocasting and routing in wireless networks. *SITE, University of Ottawa, TR-99-11*, 1999.

[TG02] D. Tian and N.D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. *In ACM Intl Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[TK84] H. Takagi and L. Kleinrock. Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. *IEEE Trans. Comm.*, vol. 32(3):246–257, 1984.

[Tou80] G. Toussaint. The relative neighborhood graph of finite planar set. *Pattern Recognition*, vol. 12(4):261–268, 1980.

[Var96] P. Varshney. Distributed Detection and Data Fusion. *New York:Springer-Verlag*, 1996.

[XLPH06] G. Xing, C. Lu, R. Pless, and Q. Huang. Impact of Sensing Coverage on Greedy Geographic Routing Algorithms. *IEEE Transactions on Parallel and Distributed Systems*, vol. 17(4):348–360, 2006.

[XWZ+05] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C.D. Gill. Integrated Coverage and Connectivity Configuration for Energy Conservation in Sensor Networks. *ACM Trans. Sensor Networks*, vol. 1(1):36–72, 2005.

[Yao82] A.C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing 11*, vol. 4:721–736, 1982.

[YZLZ02] F. Ye, G. Zhong, S. Lu, and L. Zhang. Energy efficient robust sensing coverage in large sensor networks. *Technical report, UCLA*, 2002.

[YZLZ03] F. Ye, G. Zhong, S. Lu, and L. Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. *In 23nd International Conference on Distributed Computing Systems (ICDCS)*, 2003.

[ZH04] H. Zhang and J. C. Hou. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. *In NSF International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, 2004.