POSITION-BASED ROUTING ALGORITHMS FOR THREE-DIMENSIONAL AD HOC NETWORKS

Alaa Eddien Awad Abdallah

A thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Concordia University

Montréal, Québec, Canada

January 2009

Your file Votre référence
ISBN: 978-0-494-63398-4
Our file Notre référence
ISBN: 978-0-494-63398-4

Abstract

Position-Based Routing Algorithms for Three-Dimensional Ad Hoc Networks

Alaa Eddien Awad Abdallah, Ph.D.

Concordia University, 2009

In *position-based* routing algorithms, the nodes use the geographical information to make routing decisions. Recent research in this field addresses such routing algorithms in two-dimensional ($2D$) space. However, in real applications, the nodes may be distributed in three-dimensional ($3D$) space. Transition from $2D$ to $3D$ is not always easy, since many problems in $3D$ are significantly harder than their $2D$ counterparts. This dissertation focuses on providing a reliable and efficient position-based routing algorithms with the associated pre-processing algorithms for various $3D$ ad hoc networks.

In the first part of this thesis, we propose a generalization of the Yao graph where the cones used are adaptively centered on the nearest set of neighbors for each node, thus creating a directed or undirected spanning subgraph of a given unit disk graph (UDG). We show that these locally constructed spanning subgraphs are strongly connected, have bounded out-degree, are t-spanners with bounded stretch factor, contain the Euclidean minimum spanning tree as a subgraph, and are orientation-invariant. Then we propose the first local, constant time algorithm that constructs an independent dominating set and connected dominating set of a Unit Disk Graph in a $3D$ environment. We present a truncated octahedral tiling system of the space to assign to each node a class number depending on the position of the node within the tiling system. Then, based on the tiling system, we present our local algorithms for constructing the dominating sets. The new algorithms have a constant time complexity and have approximation bounds that are completely independent of the size of the network.

In the second part of this thesis, we implement $3D$ versions of many current $2D$ position-based routing algorithms in addition to creating many new algorithms that are specially designed for a $3D$ environment. We show experimentally that these new routing algorithms can achieve nearly guaranteed delivery while discovering routes significantly closer in length to a shortest path. Because many existing position-based routing algorithms for ad hoc and sensor networks use the maximum transmission power of the nodes to discover neighbors, which is a very power-consuming process. We propose several localized power-aware $3D$ position-based routing algorithms that increase the lifetime of a network by maximizing the average lifetime of its nodes. These new algorithms use the idea of replacing the constant transmission power of a node with an adjusted transmission power during two stages. The simulation results show a significant improvement in the overall network lifetime over the current power-aware routing algorithms.

Acknowledgments

I could not have reached the end of this long, challenging path without the support and help of many people. I would like to express my sincere gratitude to my advisers, Professor Thomas Fevens and Professor Jaroslav Opatrny, for their help, support, guidance and for allowing me the freedom to pursue my own interests. Despite difficult times, they were always there when I needed them. I would also like to thank Professor Lata Narayanan and Professor Ivan Stojmenovic for their direct and indirect contributions throughout this investigation.

I owe everything to my family: To my father, Awad Abdallah, for his constant support. Without him, I would not have applied to the Ph.D. program in the first place, my mother, Rabiha, my brothers Emad and Mohammad, and my sister Hana'a. Without their endless love and support, I would not have been here. I would also like to thank my aunts and uncles: Ndawah, Aminah, Hiam, Ahmad, Kaleb, Taleb, Tyseer, Hassan, Husain, Husni, Tahsin and Naser for their support.

I would like to thank my friends back in Jordan, Majdi, Shamsi, Amjad, Masri, Samer and Yousef who constantly tried to make me feel not so away from home. Special thanks to my friend Samer Atawneh. I would also like to thank my friend Jonah Flanagan for being my English proofreader. Finally, I would like to thank my friends here in Montreal who gave me very beautiful memories: Edi, Tarek, George, Islam, Adnan, Malek, Talin, Faraz and Shahab.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless networks have become increasingly popular in the computer industry. This is particularly true within the past decade, which has seen wireless networks being adapted to enable mobility. There are currently two variations of wireless networks. The first are known as *infrastructure networks*, e.g., those networks with fixed transmitters, known as base stations. A mobile unit within these networks connects to, and communicates with, the nearest base station that is within its communication radius. As the mobile unit travels out of range of one base station and into the range of another, a "handoff" occurs from the old base station to the new, and the mobile unit is able to continue communication throughout the network. The second type of wireless network are *Mobile Ad Hoc Networks* (MANETs). A MANET consists of a collection of wireless mobile hosts that can communicate with each other without a fixed infrastructure. The term ad hoc networks can be applied to any network where there is no communication infrastructure or the existing infrastructure is expensive or inconvenient to use. Ad hoc networking allows the devices to maintain connections to the network as well as add and remove devices to and from the network with ease.

MANETs have the following unique properties:

- There is no centralized authority for network control, routing or administration. The control and management of the network must be distributed among the

mobile nodes.

- Network devices, including routers, are free to move rapidly and arbitrarily in time and space.
- All communication, such as user data and control information, is carried out over the wireless medium. There are no wired communication links. This wireless medium is subject to noise, fading and interference.
- Resources, including energy, bandwidth, processing capacity and memory, that are relatively abundant in wired environments, are typically strictly limited and must be preserved.
- Each mobile may function as both a host and a router. In other words, besides the basic processing ability to originate and receive data transmissions as a host, the mobile nodes must also perform routing functions. Usually the mobile nodes and routers are indistinguishable in a MANET.

The set of applications for MANETs include military tactical networks, battlefields, radar networks, conferences and other similar cases. MANETs can also be used to provide crisis management service applications, such as in disaster recovery, where the communication infrastructure is damaged and restoring communication quickly is crucial. In addition to the set of emergency applications, there is a group of possible everyday implementations. For example, Bluetooth, an industrial specification for wireless personal area networks (PANs), is designed to support a personal area network by eliminating the need of wires between various devices, such as printers and personal digital assistants.

Wireless Sensor Networks are a special class of ad hoc networks. Their particularities are that the topology is usually fixed, but the size and power of the nodes are restrictive. Sensor networks are composed of a large number of sensor nodes that are positioned within a limited and defined geographical area [10]. Their applications

may include [89]:

- wireless military sensor networks (to detect and gain information about enemy movements, explosions and other phenomena of interest)
- wireless sensor networks to monitor environmental changes in plains, forests, oceans, etc.
- sensor networks to detect and characterize chemical, biological, radiological, nuclear, and Explosive (CBRNE) attacks and material
- wireless traffic sensor networks to monitor vehicle traffic on highways or in congested parts of a city
- wireless surveillance sensor networks for providing security in shopping malls, parking garages and other facilities.

Akyildiz *et al.* [10] present the following differences between sensor networks and ad hoc networks. First, the number of sensor nodes in a sensor network can be several orders of magnitude higher than the nodes in an ad hoc network. Second, sensor nodes mainly use a broadcast communication paradigm, whereas most ad hoc networks are based on point-to-point communications. Third, sensor nodes are limited in power, computational capacities and memory, while ad hoc network nodes may have bigger batteries that can be recharged at any time. Also, some ad hoc network nodes have more powerful processors than sensor network nodes.

1.1 The Problem of Routing in Ad Hoc Networks

Two nodes can communicate in a bidirectional manner if and only if the distance between them is at most the minimum of their transmission ranges. If one node wishes to communicate with another node outside its transmission range, multi-hop routing is used utilizing intermediate communicating nodes. Since mobile ad hoc

networks may change their topology frequently and without notice, routing in such networks is a challenging problem.

The routing algorithms are required to route data packets effectively and efficiently to the mobile destination in order to support different types of multimedia applications. Therefore, the design of routing algorithms for MANETs should consider the following factors, which add more difficulties and complexities to routing protocol design [101]:

- **Localization:** The routing algorithm is called local if each node of the network makes decisions based on the information obtained uniquely from the nodes located no more than a constant number of hops (usually 1) from it. The algorithm is considered global if the node makes its decisions potentially based on the information of every other node in the network. Local routing algorithms are certainly preferable if they can approximately match the performance of global routing algorithms.

- **Delivery Rate:** The delivery rate is the ratio of the number of messages received by the destinations to the number sent by senders. The primary goal of every routing scheme is to deliver a message, so a practical objective while designing routing algorithms is to guarantee message delivery.

- **Shortest Path:** One of the objectives of routing algorithms is to deliver a message with a path length close to minimum. If all nodes have the same fixed transmission power, then routing schemes may use hop count as a metric for the path length, where hop count is the number of transmissions on a route from a source to a destination. However, if nodes can adjust their transmission power (knowing the location of their neighbors), the constant per hop metric can be replaced by a power metric that depends on distance between nodes.

- **Power Awareness:** Since the wireless hosts that we are modeling are commonly powered by a limited power supply like a battery, energy efficiency needs

to be an important design consideration in any routing algorithm.

- **Memorization:** Routing algorithms that require nodes to memorize the path routes are sensitive to node memory size. If the network size increases, the average path length increases. This results in the need to expand the size of the memory for mobile nodes. Thus it is better to avoid memorizing the past routes at any node during the routing process.

- **Scalability:** Because of the multi-hop nature of ad hoc networks, the scalability is directly related to the routing protocol. Thus the performance of the well-designed routing algorithm should adapt well to large-scale ad hoc networks [77]. In general, if the algorithms are local, memoryless and energy efficient, they are usually scalable.

- **Preprocessing:** Many routing algorithms require the nodes to do some pre-processing algorithms (e.g. compute a sub graph of UDG) before they actually route the packets. Thus it is important to minimize the preprocessing requirements at any node during the routing process.

1.2 Motivation and Research Focus

Position-based routing algorithms use the position information of nodes to forward the packet in the geographical direction of the destination. In this type of routing, the node forwards the message based on the position of the node itself, the position of the destination and the position of the nodes with which it can communicate directly. The main motivation behind investigating position-based routing is to make wireless ad hoc networks more efficient. Recent research in position-based routing usually addresses such routing algorithms in two-dimensional ($2D$) networks [76, 85, 102, 104]. However, in real applications, nodes may be distributed in three-dimensional ($3D$) space. For example, underwater networks that perform ocean column monitoring

would require nodes to be positioned at different depths in the water, creating a $3D$ network [9, 38].

Recently, Durocher *et al.* [38] show that there can be no local position-based routing algorithm that guarantees delivery in $3D$ space if the thickness of one of three dimensions is more than $r/\sqrt{2}$, where r is the transmission range.

To solve the routing problem, nodes in the network execute distributed routing protocols. A routing protocol may be defined as being comprised of two parts [15, 43]: (i) a pre-processing algorithm. Given the initial connection graph G, the pre-processing algorithm is in charge of creating whatever information is necessary to improve the performance of the routing algorithm (e.g., the dominating set or subgraph of G). And (ii) the routing algorithm, e.g., a distributed algorithm running at each node, which is mainly responsible for determining, for every packet entering the node, the neighbor node to which the message has to be forwarded. In this thesis, we focus on the two parts of routing protocols.

1.3 Thesis Contributions

Our goals in this thesis are to utilize $3D$ position information (of any thickness) to provide more efficient and reliable position-based routing algorithms for various $3D$ scenarios such as urban rescue, city landscape, hilly terrain, airborne and ocean sensor networks. Our contributions below are mentioned in the order of their appearance in the body of the thesis.

1.3.1 Pre-processing Algorithms

First, since many routing strategies use a subgraph of the unit disk graph such that only the edges in the subgraph are used for routing, we introduce a generalization of the Yao graph [39] where the cones used are adaptively centered on the nearest set of neighbors for each node, thus creating a directed or undirected spanning subgraph of

a given unit disk graph (UDG). We show that these locally constructed spanning subgraphs are strongly connected, have bounded out-degree, are t-spanners with bounded stretch factor, contain the Euclidean minimum spanning tree as a subgraph, and are orientation-invariant.

Second, there is currently a growing interest in the research and development of the dominating set or connected dominating set in MANET or wireless sensor network, where several routing algorithms use a dominating set of the nodes for routing. We use the node position information to propose the first local, constant time algorithm that constructs an independent dominating set and connected dominating set of a Unit Disk Graph in a $3D$ environment [5](also called unit ball graph [38]). We present a truncated octahedral tiling system of the space to assign to each node a class number depending on the position of the node within the tiling system. Then, based on the tiling system, we present our local algorithms for constructing the dominating sets. The new algorithms have a constant time complexity and have approximation bounds that are completely independent of the size of the network.

1.3.2 Routing Algorithms

First, we created several new $3D$ position-based routing algorithms, [1, 2, 3, 84]. The coordinate face routing, *CFace*, is a heuristic that is based on $2D$ Face routing [27, 67] by adapting the algorithm to $3D$ environment. AB3D is an extension of some randomized routing algorithms from $2D$ to $3D$ space. *ABLAR* is a restricted directional flooding-based algorithm that chooses m neighbors in the direction of the destination according to a space-partition heuristic and forwards the message to all these nodes. *T-ABLAR-T* is a group of routing algorithms that combine some $3D$ deterministic progress-based routing algorithms with restricted directional flooding-based algorithms. *AB3D-ABLAR* is a group of algorithms that combine the randomized routing algorithms (AB3D) with restricted directional flooding-based algorithms. *AB3D-CFace(1)-AB3D* and *AB3D-CFace(3)* are groups of algorithms that combine

the randomized AB3D routing algorithms with a deterministic CFace algorithm. All the new algorithms are evaluated and compared with current routing algorithms. The simulation results on unit disk graphs (UDG) show a significant improvement in delivery rates (up to 99%) and a large reduction of the traffic or path dilation.

Second, most of the existing position-based routing algorithms for ad hoc and sensor networks use the maximum transmission power of the nodes to discover neighbors, which is a very power-consuming process. We propose several local *power-aware* $3D$ position-based routing algorithms [4, 6] that increase the lifetime of a network by maximizing the average lifetime of its nodes. These new algorithms use the idea of replacing the constant transmission power of a node with an adjusted transmission power during two stages: first, a lower power while discovering the neighboring nodes and second, if needed, a higher transmission power during the routing process. We evaluate our algorithms and compare their power savings with the current power-aware routing algorithms. The simulation results show a significant improvement in the overall network lifetime.

The rest of the thesis is organized as follows. Chapter 2 contains a brief review of essential concepts and definitions which we will refer to throughout the thesis. It also contains a survey of the current routing algorithms with an emphasis on the related position-based routing algorithms. Chapter 3 introduces our new geometric subgraph, which is called *Displaced Apex Adaptive Yao Graphs*, to model a MANET. Chapter 4 presents our new algorithm for constructing dominating sets in the $3D$ environment. Chapters 5 and 6 present several new position-based routing algorithms that are specially designed for $3D$ environment. Chapter 7 presents three groups of power-aware position-based routing algorithms for the $3D$ MANETs. Finally, we conclude the thesis in Chapter 8.

Chapter 2

Definitions and Background

2.1 MANETs Model and Notation

A graph $G = (V, E)$ consist of a finite set $V = v_1, v_2, ..., v_n$ whose elements are called vertices and a subset E of the Cartesian product $V \times V$, the elements of which are called edges. We will use the following conventions for notation: For node u, the set of its neighbors is denoted by $N(u)$. Let $N2(u)$ and $N3(u)$ to be the set of nodes that are 2 and 3 hops away from u respectively. A path from the node s to the node d is a sequence of nodes $s = v_1, v_2, ..., v_k = d$, such that v_i and v_{i+1} are neighbors, where $1 \leq i \leq k - 1$.

A subset V' of V is called *dominating* if every vertex from $V - V'$ is adjacent to some node in V'. A minimal dominating set of V, denoted by MINL-DS, is a dominating set of V such that no subset has this property. A dominating set is called a *connected dominating set* (CDS) if the subgraph $P(G)$ induced by V' is connected. The smallest subset of vertices that is both connected and dominating is called a *minimum connected dominating set* (MIN-CDS). A subset of vertices in a graph G is an independent set if no two vertices are connected by an edge. An independent set is maximal (MIS) if it cannot be extended by the addition of any other vertex from the graph without violating the independence property [59].

A *geometric graph* is a graph $G = (V, E)$ such that the vertices are geometric objects in R^d, where d is the dimension, and the edges are geometric objects connecting pairs of vertices.

Let $dist(u, v)$, which will be denoted occasionally by $|uv|$, be the Euclidean distance between the nodes u and v. $dist(u, v)$ is defined as follows:

$$dist(u, v) = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2 + (u_z - v_z)^2}. \quad (1)$$

A *Unit Disk Graph* (UDG) is a specific type of geometric graph usually used to model MANETs, an edge exists between two vertices (nodes) u and v if and only if the Euclidean distance between between u and v is at most 1. In the context of ad hoc networks, the vertices in the UDG represent network nodes. An edge exists between two nodes if the Euclidean distance between the two nodes is less than or equal to a node's transmission range r [75]. Here it is assumed that all nodes have transmission range equal to r, which is represented as a circle of radius r in $2D$ and a sphere volume of radius r in $3D$.

Clearly, this is a simplification of reality, since, even if all network nodes are homogeneous, this model does not account for the presence of obstacles, such as walls, buildings, mountains or weather conditions, which might obstruct signal propagation. Barrière *et al.* [19] have studied a graph model that is considerably closer to reality, proposing a generalization of the unit disk graph by considering the transmission range of the mobile host, which varies between $(1 - \epsilon)r$ and r, where $\epsilon > 0$.

2.2 Geometric Subgraphs

Although creating economical routing schemes is very important, ensuring good performance is not less important. Many routing strategies [27, 40, 67] use a subgraph of the UDG such that only the edges in the subgraph are used for routing. Therefore, much research effort has gone into the development of algorithms for subgraphs of

UDG in ad hoc networks (see [116, 82, 92] for surveys). An example of an UDG and related subgraphs is given in Figure 20.

Since the wireless hosts that we are modeling are commonly powered by a limited power supply like a battery and contain a limited amount of memory, may be mobile and the topology of the whole network is usually not available and may be variable, local algorithms are typically preferred. These algorithms are designed to achieve various objectives such as:

- **Locality [36, 5]**: A distributed algorithm is called *local* if each node of the network only uses information obtained uniquely from the nodes located no more than a constant (independent of the size of the network) number of hops from it. Thus, during the algorithm, no node is ever aware of the existence of the nodes of the network further away than this constant number of hops.

- **Low stretch factor [79, 80]**: A subgraph of G, $\mathrm{P}(G)$, is called a t-spanner of G if the length of the shortest path between any two nodes in $\mathrm{P}(G)$ is not more than t times longer than the shortest path connecting them in G. The length of the path is the sum of the lengths of the edges along the path. t is known as the *stretch factor*. Typically the aim is to find a spanner with as small t as possible, because shorter paths will be available in the subgraph. t-spanners is known to be a power-efficient strategy [79, 80].

- **Geometric Planarity [23, 107]**: A graph is called *planar* if the straight edges between the neighboring nodes do not intersect.

- **Low weight [81]**: A subgraph is called *low weight* if its total edge length is within a constant factor of the total edge length of the *Euclidean minimum spanning tree* (EMST). For a geometric graph G, a Euclidean minimum spanning tree *EMST*(G) is a minimum weight spanning tree of G.

- **Bounded degree [92, 108]**: We define a degree m bounded subgraph to be a

subgraph having no vertices of degree larger than m.

- **Orientation-invariant [39]**: If UDG is rotated by an angle, and the calculated subgraph is the same all the time, then the subgraph is orientation-invariant.

In the following we present some of the best-known geometric spanning subgraphs of UDGs that are commonly used in position-based routing algorithms.

2.2.1 Gabriel Graph

Let $Disk(u, v)$ be defined as the circle centered at the midpoint between the points u and v with a diameter $|uv|$. Then the Gabriel graph [45], denoted as $GG(G)$, is defined as follows: Given any two adjacent nodes u,v in G, the edge uv belongs to $GG(G)$ if, and only if, no other node $w \in G$ is located in $Disk(u, v)$. See Figure 1(a). Bose *et al.* [24] prove that *GG(G)* is connected, planar and a $\left(4\pi\sqrt{2n-4}/3\right)$-spanner of G. A $3D$ definition of GG is straightforward and given in [64].

2.2.2 Relative Neighborhood Graph

Consider the $lune(u, v)$ as the interior and the boundary of the region formed by the intersection of two disks of radius r, one of the disks being centered at u and the other at v. Figure 1(b) shows the lune of the two points u, v. The *Relative Neighborhood Graph* of G [60], denoted $RNG(G)$, is defined as follows: Given any two adjacent nodes u, v in G, the edge uv belongs to $GG(G)$ if, and only if, no other node $w \in G$ is located in $lune(u, v)$. In other words, an edge $uv \in RNG(G)$ if, and only if, there is no node w such that $(u, w) \leq (u, v)\, and\, (v, w) \leq (u, v)$. It is proved that $RNG(G)$ is connected, planar, and a $(n-1)$-spanner of G. A $3D$ definition for this graph is given in [2, 64, 105].

2.2.3 Yao Graph

For a geometric graph G, a *Yao Graph* (also called a *Theta Graph* [25]) $YG_k(G)$ with an integer parameter $k \geq 6$ is defined as follows [113]. First, we will define a directed Yao graph, $\overrightarrow{YG_k}(G)$, for G. At each node u in G, k equally-separated rays originating at u define k cones, as seen in Figure 1(c). In each cone, only the directed edge uv to the nearest neighbor v, if any, is part of $\overrightarrow{YG_k}(G)$. Ties are broken arbitrarily.

Let $YG_k(G)$ be the undirected graph obtained if the direction of each edge in $\overrightarrow{YG_k}(G)$ is ignored, yielding a subgraph that may have crossing edges if G is a UDG. The graph $YG_k(G)$ is a $1/(1 - 2sin(\pi/k))$-spanner of G [69, 80], has an out-degree of at most k, and contains the $EMST(G)$ as a subgraph [113]. One drawback of the $YG_k(G)$ graph is that it is not orientation-invariant. That is, if G is rotated by an angle to give G' then the resulting $YG_k(G')$ subgraph is not necessarily a rotation of $YG_k(G)$.

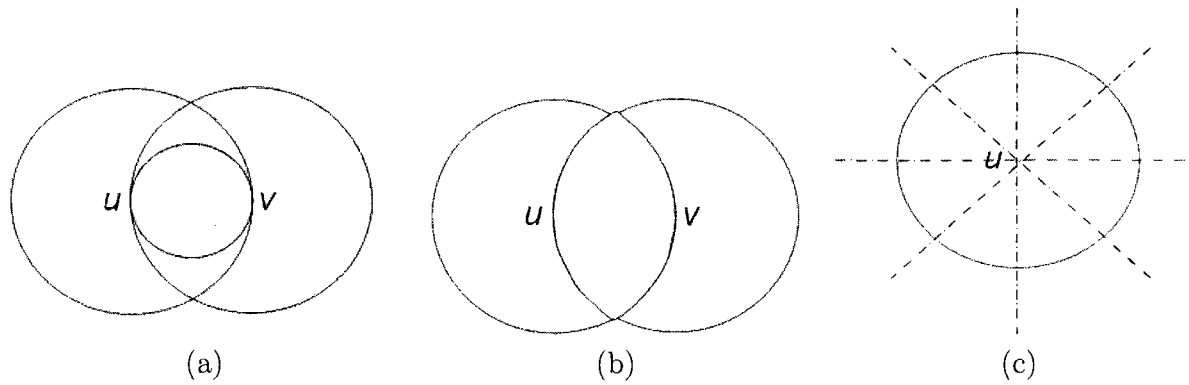


Figure 1: (a) the edge $uv \in GG(G)$ if there are no nodes in the shaded area; (b) the edge $uv \in RNG(G)$ if there are no nodes in the shaded area; (c) the sectors around a node u that are used by the Yao graph

2.2.4 Half Space Proximal Graph

For a geometric graph G, a *Half Space Proximal Graph* $HSP(G)$ is defined as follows [31]. As with the Yao Graph, first a directed $\overrightarrow{HSP}(G)$ 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. A directed edge uv 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 until all the nodes have been discarded. The selected directed edges determine the $\overrightarrow{HSP}(G)$. An illustration of the $HSP(G)$ test applied to a node in a UDG is given in Figure 2, which is taken from [31].

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 [31] for the HSP subgraph, it is strongly connected, has an out-degree of at most six, has a stretch factor of at most $2\pi + 1$, contains the $EMST(G)$ as its subgraph and is orientation-invariant.

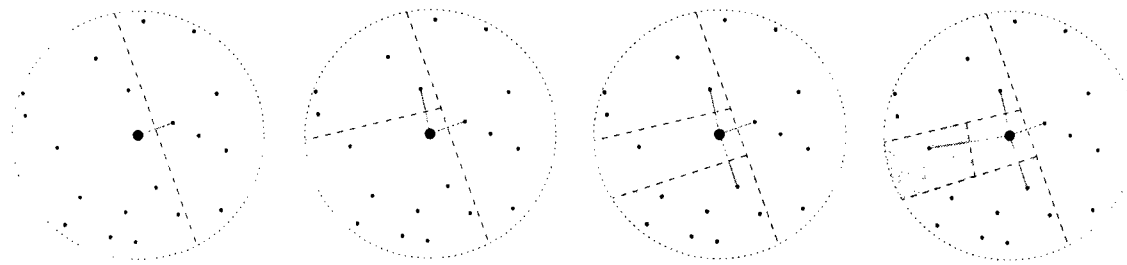


Figure 2: $HSP(G)$ step-by-step for a node

2.3 Dominating Sets

One of the basic problems in a MANET is to broadcast messages in the network, where a message is sent from one node to all nodes in the network [98]. Broadcasting has an unacceptable communication overhead, which leads to a waste of the

rare resources of wireless nodes. One effective way to decrease the communication overhead in broadcasting is to use a CDS as a virtual backbone of the nodes in the routing algorithms [13, 37, 110, 111]. In these routing algorithms, only nodes of the connected dominating set (dominators) act as routers; all other nodes communicate via a neighbor in the dominating set. Clearly, the efficiency of this approach depends largely on the process of finding the dominating sets and the size of the corresponding sub-networks [111]. Finding a MIN-CDS is NP-complete in general [35, 48, 55, 68].

Several algorithms have been previously proposed to construct an independent dominating set and a CDS for UDG, but none of these algorithms has the following 3 characteristics:

1. construction in a constant time (local according to the definition above)
2. a constant approximation bound
3. capable of working in a $3D$ environment

2.3.1 Current Research for Determining Connected Dominating Sets

The problem of finding a dominating set is an area in graph theory with extensive research activity. In 1998, a book was published that listed 1200 papers in the area of domination [52]. As mentioned before, given a complete network topology, the problem of finding the minimum dominating sets and MIN-CDSs is known to be NP-hard. In addition, due to the nature of MANETs, practical IDS and CDS construction protocols for MANETs need to be fully distributed in constant time. In the following, we will survey some of the related algorithms.

The distributed protocols in [12, 13] by Alzoubi *et al.* construct the CDS in a linear time by expanding the maximal independent set. These protocols consist of two phases. In the **first phase**, an independent dominating set is constructed as follows:

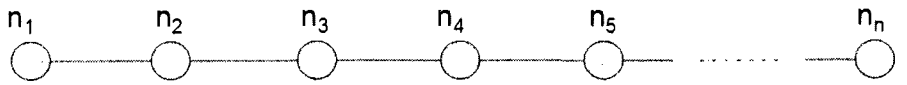


Figure 3: Worst-case example for [11, 12, 13] and many other algorithms

If the node unique ID is minimum among its neighbors, it adds itself to the dominating set and removes all its neighbors from the consideration of the set members. This process is repeated at each node such that the resulting set is a maximal independent dominating set (MIS), which is also a (non-connected) dominating set. In the **second phase**, the nodes in the set use local topology information for a node, up to 3 hops away, to add gateway nodes to the set until the set becomes a CDS. The main disadvantage of this algorithm is construction time of the independent set, which can be proportional to the number of nodes; thus it is a non-local algorithm. Consider the example in Figure 3: let $n_1 < n_2 < n_3 < .... < n_n$, n_1 takes its decision at time 0, which will dominate n_2; n_3 has to wait until n_2 to send a messages saying that it has been dominated (I am Dominated), and then it will make its decision; n_4 has to wait for the results coming from n_3, and so on. Thus n_n has to wait until all the nodes in the network make their decisions, which makes the time complexity for this algorithm $O(n)$ in the worst-case scenario. This is not efficient in an ad hoc network environment. A similar algorithm was proposed by Baker *et al.* [17, 18]. In the example above and all our analysis in Chapter 4 we ignore the local computation time in the time complexity calculations.

In [11], Alzoubi *et al.* propose an integration between the connected dominating set CDS and the local Delaunay graph to construct a geometric planar and spanner backbone of the wireless network. The distributed algorithm starts by constructing a CDS using a technique similar to those used by Alzoubi *et al.* [12, 13] or by Baker *et al.* [17, 18] as described above. The next step is to build the local Delaunay graph [79] on top of the constructed CDS. Alzoubi *et al.* [11] prove that the constructed graph is planar and has bounded degree. Because this algorithm uses the same idea as in

[12, 13] for implementing the IDS, which leads to the CDS, the worst-case example in Figure 3 is applicable to this algorithm too. Thus the time complexity is $O(n)$.

The Greedy algorithm [34, 62, 99] for constructing a dominating set is a global algorithm where the run-time depends on the number of nodes. The Greedy algorithm picks a node that covers the biggest number of uncovered nodes and puts it into the dominating set; it repeats the same algorithm as long as there are uncovered nodes.

In [83], Liang *et al.* propose an implementation of the Greedy algorithm in a distributed manner; the algorithm is called a *Distributed Database Coverage Heuristic* (DDCH) and can be summarized as follows: Each node u calculates its span, which is the number of uncovered nodes that u covers, and sends its span and *id* to all nodes within 2 hop neighbors; next, the node u adds itself to the dominating set if its ordered pair of span and *id* is higher than that of any node within 2 hop neighbors. The distributed time complexity of this algorithm is also linear in the number of nodes. See a worst-case example in [61]. A randomized version of DDCH, called *a Local Randomized Greedy algorithm* (LRG), has been proposed in [61]. This algorithm has $O(\log n \log \Delta)$ time complexity, with Δ being the maximum node degree, and $O(\Delta)$ approximation ratio. In [47], another distributed randomized algorithm was proposed by Gao *et al.* that maintains the dominating sets for mobile nodes. It is shown that it has a constant approximation ratio with a high probability, but the constant approximation ratio is quite large. As with the other algorithms discussed before, a drawback is the time complexity that is $O(n \log n)$.

For a class of general graphs, Kuhn *et al.* [72, 73, 74] have given approximation lower bounds for covering problems as a function of the size of the neighborhood through which each message may be propagated. They show that with k communication rounds, a dominating set cannot be approximated better than by a factor $\Omega(\frac{n^{c/k^2}}{k})$ for some constant $c \geq 1/4$.

None of the algorithms mentioned above has both a constant approximation bound and a constant worst-case time bound. One approach to achieve these bounds is

to use the underlying geographic information. The first algorithm to determine a dominating set in $2D$ within a constant approximation of the optimal dominating set in a constant time (which depend on the degree if we consider the local computation time) was proposed by Czyzowicz *et al.* [36]. Assuming that each node knows its geometric location in a plane, the algorithm starts by associating each node with a class number that depends on the position of the nodes within a regular hexagonal tiling of the plane. After the nodes determine their class number, they acquire the class numbers of all their neighbors. In each hexagon, the dominators are determined on the basis of unassigned neighbors with the minimum class number closest to the center of the hexagon (or some similar local heuristic) under consideration. Since this algorithm assumes that the nodes have a geometric location in two-dimensional space, it is not directly applicable for all wireless applications where the nodes may be located in three-dimensional space.

2.4 Routing Protocol Quality

There are several quantitative, independent metrics for judging the performance of MANETs routing protocols. Desirable quantitative properties include:

- *Path dilation:* The spanning-ratio of a subgraph is only a bound to the performance of a routing scheme. We still need to develop routing schemes that select paths that are close to the shortest path. The path dilation is used as an accurate measure of the quality of the routing scheme and is defined as the average ratio of the length of the path returned by the routing algorithm, even when routing on a subgraph, to the length of the shortest path in the UDG.

- *Routing traffic:* If the algorithm uses some sort of flooding, then it should try not to employ a lot of nodes during the routing process because the more nodes participate in the routing process, the more overhead and collision may happen in the network; also, there is more of a chance for nodes to fail from running

out of batteries. Traffic is used as a measure here and is defined as follows: the average ratio of the number of nodes the packet visits during the routing process to the number of nodes in the shortest path in the UDG.

- *Network Survivability:* Network survivability may be defined as the remaining power in the maximally used node, assuming each node starts off with the same power, during a set of consecutive routing messages.

2.5 Classification of Routing Protocols

Finding improved routing algorithms is a challenging problem for MANETs. Several routing protocols for ad hoc networks have been proposed to solve the multi-hop routing problem. Each is based on different assumptions and concepts. In general, Mauve *et al.* [85] classify the routing algorithms in MANETs as being of two basic types: topology-based [20, 28, 94, 95, 100] and position-based [19, 49, 85]. Figure 4 shows our view of the general classification of ad hoc routing algorithms.

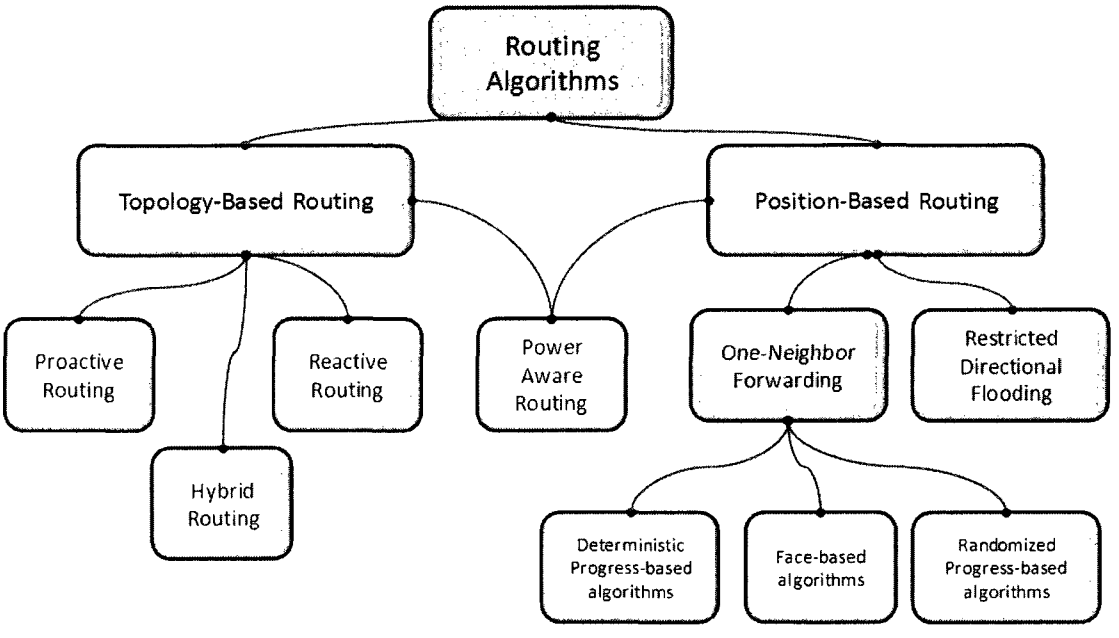


Figure 4: Classification of routing algorithms

2.5.1 Topology-based Routing Protocols

Topology-based routing protocols define an explicit route among nodes using the information about the links that exist in the network. Those protocols can be further divided into three main categories: proactive, reactive and hybrid protocols.

Proactive Routing Protocols

Proactive routing protocols use a periodic information exchange to maintain an understanding of the network topology. The whole network should, in theory, be known to all nodes. The advantage of a proactive protocol is that routes are readily available when one node wishes to send a message to another node. The *destination sequence distance vector* (DSDV) [90] routing protocol, the *wireless routing protocol* (WRP) [86] and the *cluster-head gateway switch routing* protocol (CGSR) [32] are all types of proactive routing protocols. R-DSDV [33] is an example of a randomized version of a proactive protocol.

These protocols can suffer from a high volume of control packets overhead because of the need to distribute network topology and route path maintenance information even if a network path is unused.

Reactive Routing Protocols

Reactive protocols seek to set up routes on-demand. If a node wants to initiate communication with a node to which it has no route, the routing protocol will try to establish such a route. This means that it maintains only the routes that are currently in use. Reactive protocols typically use less bandwidth in terms of control packets to discover topology information, but even so, packets to discover new routes must sometimes be flooded through the network, which consumes immense bandwidth. The *dynamic source routing* protocol (DSR) [20, 63], and the *ad hoc on demand distance vector* routing protocol (AODV) [91] are some examples of reactive routing protocols.

Hybrid Routing Protocols

Hybrid routing algorithms, such as the Zone Routing Protocol (ZRP) [50], integrate local proactive routing and global reactive routing to achieve a higher level of efficiency and scalability. However, route maintenance is still required. The border between local region and global region limits the distribution efficiency of information about network topology changes.

Topology-based routing can usually find the shortest path, in terms of the number of hops, between a pair of nodes. However, it can be difficult for these routing methods to handle large ad hoc networks with many nodes or with frequently changing connectivity among nodes [57].

2.5.2 Position-based Routing

Position-based routing [49, 85], or online routing [26], algorithms eliminate some of the limitations of topology-based routing by using geographical information about the mobile nodes to make decisions about routing packets. In general, a position-based routing algorithm has the following characteristics:

- Each node in the network has the means to determine its coordinates. This can be obtained through a GPS receiver or another such mechanism [29, 66].
- Each node can find the position of a node with which it wishes to communicate by making use of a location service [54, 78], receiving the position from a previous packet from that node or some other mechanism.
- There is no need for nodes to store routing tables. Nodes maintain only the information about their neighbors at most a fixed number of hops (usually one hop) away.
- A node in the network knows the positions of the nodes with which it can communicate directly, simply by using a periodical broadcast beacon containing

information such as node identifier and geographic coordinates.

- The routing decision at each hop in the route can be made based on the locations of the current node, its neighbors and the destination node.

Position-based routing scales to a large number of network nodes and is efficient when nodes move frequently. There are two main types of packet-forwarding strategies for position-based routing [85]: one-neighbor forwarding and restricted directional flooding.

One-neighbor Forwarding

With one-neighbor forwarding, the algorithm forwards the packet in every step to exactly one of its neighbors. We can consider three types of these algorithms: *Deterministic progress-based*, *Randomized progress-based* and *Face-based* routing algorithms.

Deterministic Progress-based Algorithms:

With deterministic progress-based routing algorithms [42, 56, 71, 104, 106], the current node (the node holding the packet) forwards the packet at every step to one of its neighbors that makes progress to the destination. These algorithms are known to fail in delivering the packet in certain situations that are called the *local minimum* phenomena, in which a packet may get stuck at a node that does not have a neighbor that makes a progress to the destination, even though the source and destination are connected in the network. The following algorithms belong to the progress-based strategy:

- ***Greedy* [42]:** For this algorithm, the current node c forwards the packet to the neighbor node u that minimizes the remaining distance to the destination node d. See Figure 5(a). Formally, $Gdy(c, N(c), d) = u \in N(c) : dist(u, d) \leq dist(w, d)$ for all $w \in N(c)$. The same procedure is repeated until the destination node is reached. If the packet reaches a local minimum, then the algorithm

fails. Greedy routing is a loop-free algorithm.

- ***Compass* [71]:** In this algorithm, the current node c forwards the packet to the neighbor node u that minimizes the angle $\angle ucd$, where d is the destination. See Figure 5(b). Formally, $Cmp(c, N(c), d) = u \in N(c) : \angle ucd \leq \angle wcd$ for all $w \in N(c)$. If the packet reaches a local minimum, then the algorithm fails. It was proved in [103] that Compass routing is not a loop-free algorithm.

- ***Ellipsoid algorithm*[112]:** In this algorithm, the current node c forwards the message to its neighbor node u that minimizes the summation of the distance from c node to u and the distance from u to the destination node d. See Figure 5(c). Formally, $Elp(c, N(c), d) = u \in N(c) : (dist(c, u) + dist(u, d)) \leq (dist(c, w) + dist(w, d))$ for all $w \in N(c)$. The same procedure is repeated until the destination node is reached. If the packet reaches a local minimum, then the algorithm fails.

- ***Most Forward Routing (*MFR*)* [106]**: This algorithm maximizes the progress towards the destination by forwarding the packet to the neighboring nodes whose projection onto the line between the current node and the destination is closest to the destination. See Figure 5(d). Formally, $MFR(c, N(c), d) = u \in N(c) : dist(proj(u, cd), d) \leq dist(proj(w, cd), d)$ for all $w \in N(c)$, where $proj(u, cd)$ is the projection of the node u on the line cd. Since MFR is a progress-based algorithm, it fails if the packet reaches a local minimum node. In most cases, MFR selects the same path as the Greedy algorithm.

$3D$ extensions for the previous deterministic progress-based algorithms: Purely deterministic progress-based algorithms can be considered in $3D$ with little modification. Kao *et al.* [64] and Nanda [87] propose the $3D$ extensions for the Greedy and Compass algorithms. Extensions for the Ellipsoid and Most Forwarding algorithms are provided in [64].

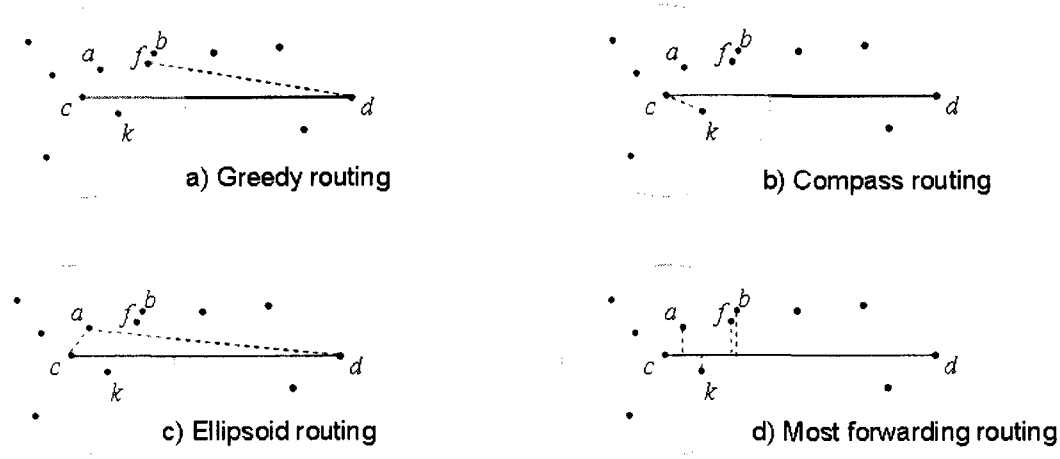


Figure 5: Various routing algorithms taken from [64]

For further illustration of the operation of each of the above mentioned algorithms, consider the example given in Figure 6 where the source and the destination are s and d_1, respectively. In this example, Greedy chooses a as the next node and the whole path is s, a, r, o, d_1; MFR chooses e as the next node and the whole path is s, e, o, d_1; and with Compass protocol s chooses b as the next node and the whole path is s, b, r, o, d_1. In the example in Figure 6, if the destination is changed to d_2, all the above algorithms will fail to deliver the message. Greedy, MFR and Compass will reach the node w, which is a local minimum because there is no neighbor for w that makes a progress to d_2.

Many algorithms attempt to deal with the local minimum problem. Fin [42] proposes to flood all ℓ-hop neighbors (nodes at distance at most ℓ hops from current node, where ℓ is a network-dependent parameter) until a node closer to the destination than c is found. Takagi and Kleinrock [106] propose countering the local minimum problem by forwarding the packet to the node with the least backward (negative) progress. However, this raises the problem of looping packets. Stojmenovic and Lin [103] alternately suggest dropping the packet if the best choice for the current node is to return the message to the node that packet came from.

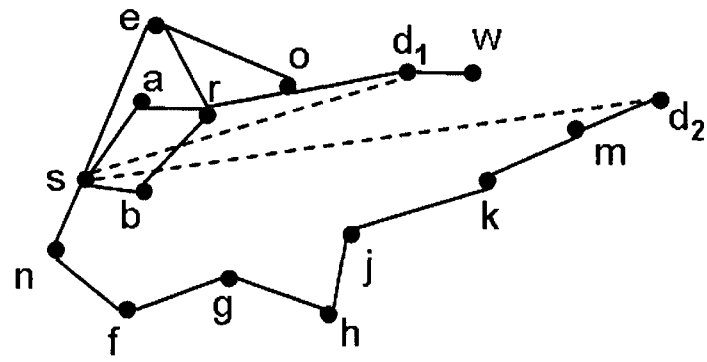


Figure 6: A sample network topology to illustrate the operation of the algorithms

Randomization-based Algorithms

Randomization-based routing algorithms [26, 41, 88] try to solve the local minimum problem described above by choosing the next node randomly from a subset of the current node's neighbors. These strategies minimize the accuracy of information needed about the position of the neighbors. In general, they have higher delivery rate than the deterministic algorithms at the price of a higher stretch factor. The following are some examples of randomization-based algorithms:

- **Random progress method [88]:** In this algorithm the current node c forwards the packet to a randomly selected neighbor closer to the destination.

- **AB algorithms [41]:** The AB (above/below) algorithm can be described as follows: Each algorithm has two attributes: AB(R,S) where **R** is one of **CM** (as in Compass), **GR** (Greedy) or **ELP** (Ellipsoid-Based), and **S** is one of **U**, **A** or **D**. Each routing algorithm is based on initially determining two candidate neighbors, one neighbor of c from above the cd line, n_1, and, similarly, one neighbor of c below the cd line, n_2. Out of all the possible neighbors from above (below) the cd line, n_1 (n_2) is the one that would be chosen by the **R** protocol. Which of these two candidate neighbors is actually chosen depends on the symbol for **S**. If the symbol is **U**, then the next node is chosen uniformly at

random from n_1 and n_2. If the symbol is **A**, then the next node is chosen from n_1 and n_2 with probability $\theta_2/(\theta_1+\theta_2)$ and $\theta_1/(\theta_1+\theta_2)$ respectively, where $\theta_1 = \angle n_1cd$ and $\theta_2 = \angle n_2cd$. Finally, if the symbol is **D**, then the next node is chosen from n_1 and n_2 with probability $dis_2/(dis_1+dis_2)$ and $dis_1/(dis_1+dis_2)$, respectively, where $dis_1 = dist(n_1, d)$ and $dis_2 = dist(n_2, d)$. If either n_1 or n_2 is not defined, then the other neighbor is chosen by default.

In the example given in Figure 6, if the source node s wants to send a packet to d_1, the AB algorithm at s will choose a or b randomly as the next node for its packet. If the destination is changed to d_2, the AB algorithm will fail to deliver the message.

$3D$ extension for randomized algorithms: We provide a $3D$ extension for the randomized algorithm in [2]. This algorithm will be explained in detail in Chapter 5.

Face-based Algorithms

To guarantee the delivery of the packets, position information can be used to extract a planar subgraph so that routing can be performed on the faces of this subgraph, known as Face routing or perimeter routing [27, 67]. The advantage of this approach is that the delivery of packets can always be guaranteed. The original Face routing algorithm was called Compass Routing II in [71]. An optimization of this algorithm is given in [27] and called Face2. In [14], Face routing is adapted to guarantee delivery on restricted classes of non-planar graphs. In the following we will explain in detail how Face routing works and how a combination of Face and the above progress-based algorithms helps to decrease the path dilation.

- **Face2 algorithm [27]:** This algorithm starts by extracting the GG from the UDG. Then the packets are routed over the faces of GG, which are intersected by the line between the source and the destination, sd, using the right-hand rule. That is, the boundary of f is traversed in the counterclockwise direction, unless

the current edge crosses sd at an intersection point closer to the destination than any previously discovered intersection point. In this case, the algorithm switches to the next face sharing the edge and continues with the right-hand rule. This algorithm is repeated until the node arrives to the destination. The Face routing algorithm guarantees the delivery only over a $2D$ planar geometric graph [44]. Figure 7 shows an example how the algorithm works.

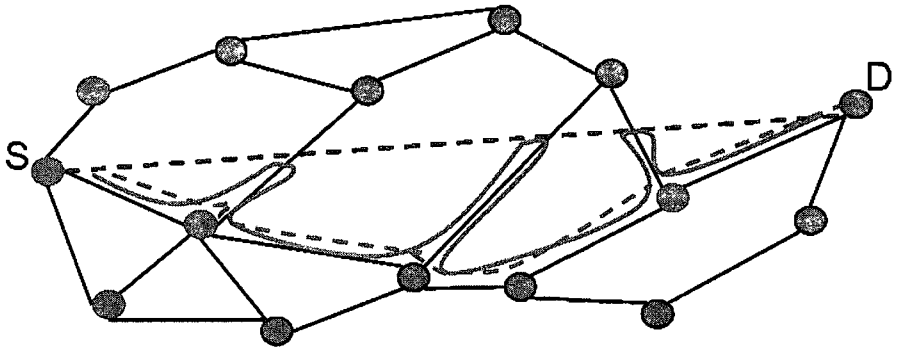


Figure 7: Face routing algorithm

- **Greedy Perimeter Stateless Routing (GPSR):** Greedy Perimeter Stateless Routing (GPSR) [67] is another position-based routing algorithm. This algorithm combines Greedy routing and Face routing. GPSR makes Greedy forwarding decisions using the local information. Packets are forwarded to the next-hop node, which moves the packet the most "toward" the position of the destination. If the packet reaches a region where Greedy forwarding is impossible, the algorithm enters into recovery mode by routing around the perimeter of the region. GPSR traverses the perimeter (Face in [27]) of the region on the planar graph to the destination in the recovery mode. Once the packet reaches a node closer than the previous local minimum, the packet switches back to Greedy forwarding again. GPSR guarantees the packet delivery. This algorithm is also termed *GFG* (*Greedy-Face-Greedy*) [27]. A combination of Face2 and the AB algorithm has been provided by Abdallah *et al.* [40].

$3D$ extension of Face routing algorithm: Face routing, GFG and GPSR algorithms guarantee delivery only over a $2D$ planar geometric graph. In a three-dimensional network, extracting a straight-line planar graph is not an option since the notions of planarity and routing about the perimeters of faces do not exist. Therefore, we cannot directly perform the Face routing protocol on a $3D$ network.

- **Projective Face Routing Algorithm:** To enable routing on a $3D$ MANET based on the Face routing protocol, Kao *et al.* [64] propose mapping the $3D$ network to a projection plane where the $2D$ Face routing algorithm can then be applied. Specifically, their Projective Face algorithm uses two orthogonal planes that intersect along the line between the source and the destination, changing to the second projection plane if the first plane leads to failure. This algorithm gives significantly better delivery rate than the other deterministic $3D$ routing algorithms such as the Greedy, Compass, Ellipsoid and Most Forward routing algorithms, although the algorithm also leads to a very large hop path dilation.

  To enhance the performance of the projective Face routing algorithm, Kao *et al.* [65] propose three heuristics to modify the projective Face routing algorithm. The resulting $3D$ routing algorithm, called Adaptive Least-Squares Projective (ALSP) Face routing, gives nearly certain delivery rate (e.g., nearly always 100% by simulation), while the hop stretch factor is relatively high. The heuristics are described as follows:

  1) Least-Squares Projection (LSP) Plane: To determine the initial projection plane, the least-squares mathematical optimization technique is used to find the best-fitting plane [97]. To maintain the local characteristic of the routing algorithm, only the source node s, destination d and the $N2(s)$ neighbors are selected as the set of data points for computing the least-squares projection plane. The heuristic is aimed at having a projected graph with minimal distortion so that the number of crossing edges can potentially be reduced.

2) Multi-Projection-Plane Strategy: A significant increase in the delivery rate is possible by utilizing more than one projection plane, all the planes arranged at certain angles from the original LSP plane. All projection planes have a common line of intersection. If we make a cross-section that is perpendicular to all the planes and look along their intersection, the dihedral angles between each pair of neighboring planes are identical. Let Ns be the number of planes. The dihedral angle between each pair of neighboring planes is thus π/Ns degrees. When switching from one order LSP plane to the next order LSP plane, this ordering is strictly followed.

3) Adaptive Behavior Scale (ABS): During the routing process, the current node can be viewed as an alternate source node during the routing process. Kao *et al.* define a fixed parameter called Adaptive Behavior Scale (ABS). The ABS is used to determine when to recalculate the LSP plane. After an ABS number of hops have been preformed on the current order LSP plane, the LSP plane is recalculated with the current node c, $N2(c)$ and the destination d. This heuristic makes the Projective Face routing algorithm more dynamic and robust by having the LSP plane better reflect the local graph structure about the current node and potentially lead to fewer crossing edges.

These approaches do not guarantee delivery, as a connected planar graph cannot be extracted from the projected graph (see Figure 8). Experiments show that the delivery rate is significantly higher than the other $3D$ progress-based routing algorithms. Li *et al.* [84] study the performance of the combination of Projective Face Routing and progress-based routing in $3D$ mobility environment.

- **Coordinate Face Routing Algorithm, CFace(3)** [2] This algorithm may be summarized as follows. The $3D$ nodes are first projected onto the xy plane. Then Face routing is performed on this projected graph. If the routing fails, e.g. a loop is detected, the nodes are then reprojected onto the second plane, the

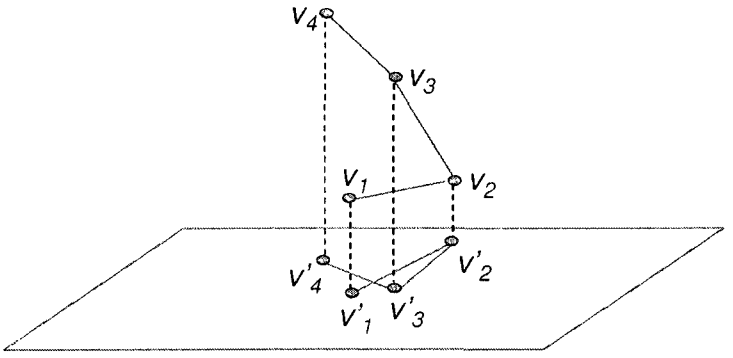


Figure 8: Projective Face routing algorithm. The neighboring nodes are preserved after projection. This figure is taken from [64]

yz plane. Then Face routing is performed again. If the routing fails again, the nodes are projected onto the third plane, the xz plane. Face routing is again performed. See Figure 9. A simplified version of CFace(3), called *CFace(1)*, attempts Face routing with the nodes projected once only onto one of the xy, yz or xz planes, randomly chosen. Abdallah *et al.* [2] propose a combination between CFace and Randomization-based routing algorithms.

Restricted Directional Flooding

In restricted directional flooding, the current node forwards the packet to more than one neighbor that is located closer to the destination than the forwarding node itself. This partial flooding can be used only for path discovery [70] or for packet forwarding [21]. The following algorithms are examples of position-based routing algorithms using restricted directional flooding.

- **Distance routing effect algorithm for mobility (DREAM) [70]:** For this algorithm, the current node c forwards the packet to all neighbors in the direction of the destination d. A node is considered to be in the direction of d if it is located in the cone shown in Figure 10. In order to determine that cone, c calculates the region around d, called the expected region. It is the circle around d of radius equal to $v_{max} * (t_1 - t_0)$ where t_1 is the current time, t_0 is

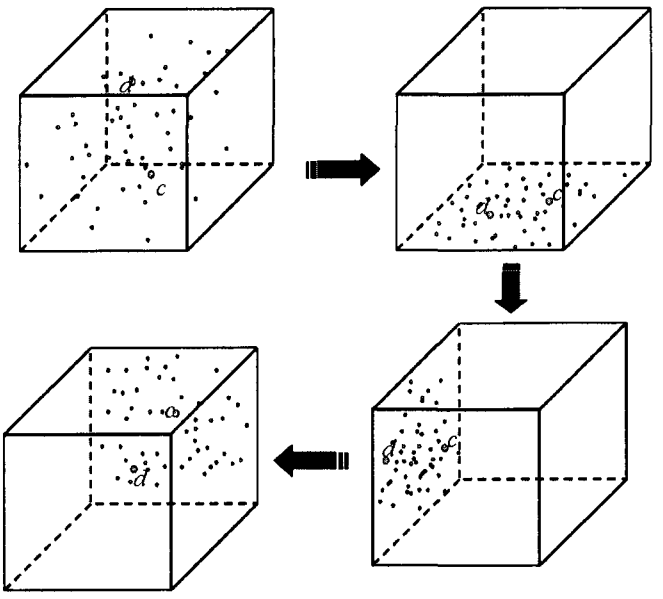


Figure 9: CFace(3) routing algorithm. The algorithm attempts 2D Face routing, cyclically until success, with the nodes projected onto the xy plane, the yz plane, and then the xz plane

the time stamp of the position information that c has about d, and v_{max} is the maximum speed of the node in the network. The neighbor nodes repeat the same procedure.

- **The geocasting based Location-Aided Routing (LAR) [21]:** LAR limits flooding of the route discovery packets to a small group of nodes that belong to the request zone. The request zone is defined as the rectangle with the source s in one corner and the expected zone in the opposite see Figure 10. The expected zone is defined exactly as for DREAM. The procedure for route discovery in LAR is as follows: First, the source puts the location information of itself and the destination in the routing request packet; second, the routing request packet is broadcasted within the request zone. In other words, the nodes within the request zone forward the message, while others discard the message; third, after receiving the route request, the destination sends back a route reply

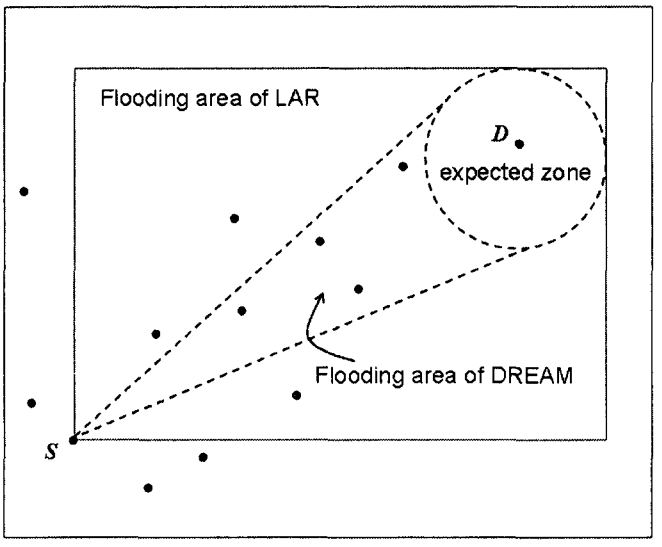


Figure 10: To route from s to d, with DREAM a current node will forward the packet to all the neighbors' nodes inside the cone, while with LAR it will forward the packet to all neighbors' nodes inside the rectangle

packet that contains its current location.

- **GEcho:[75]** This algorithm uses a combination of greedy routing and flooding-based algorithms. In this algorithm the message is forwarded in Greedy mode as long as possible. If the message arrives to a local minimum, the algorithm switches to Echo mode (flooding mode).

  The flooding phase is initiated by the local minimum node c by sending a flooding message containing a Time To Live (TTL) counter to all its neighbors. Each node receiving the flooding message for the first time decrements the TTL counter by one and retransmits the message to all its neighbors (with the exception of the neighbor it received the message from). In the synchronous model, this flooding phase constructs a Breadth First Search (BFS) tree. From the leaves of this tree, the nodes where the TTL counter reaches 0, echo messages are sent back to c along the BFS tree constructed during the flooding phase. An inner node in the BFS tree can decide locally when to send an echo message to its parent in the tree by awaiting the receipt of an echo message from all of

its children.

The algorithm returns back to Greedy mode when the message arrives at a node closer to the destination than the previous local minimum. The main drawback of this algorithm is the high message complexity, $O(\delta)$ where δ is the number of edges in the network.

$3D$ versions of the above restricted directional flooding algorithms:

LAR3D [1] This algorithm is a straightforward $3D$ extension of LAR. With the available information of the destination node d, the source node s computes the expected zone for d, which is a sphere around d of radius equal to $v_{max} * (t_1 - t_0)$ where t_1 is the current time, t_0 is the time stamp of the position information that s has about d. the node uses this zone to define the flooding area, which is defined as a rectangular box with the two opposite corners s and $s+(1+\sqrt{3}|r|/|d-s|)*(d-s)$ (the minimum-size rectangular box enclosing node s and the sphere of radius r around d). See Figure 11.

GEcho can be used in $3D$ environment without any modification. A $3D$ extension of DREAM is straightforward, where the flooding area would be the $3D$ cone shown in Figure 11.

2.5.3 Power and Cost Awareness Routing

A crucial problem in multi-hop routing is to find an efficient and correct route between a source and a destination; however, for many networks, an important problem in multi-hop routing is providing an energy-efficient routing protocol because of the limited battery life of the wireless nodes. Transmission power management, which selects the optimized power level of nodes, is one of the primary means of increasing the lifetime of the nodes. The power consumption at each node in an ad hoc network can be divided into three phases, according to functionality [16]:

1. The power consumed for transmitting the message

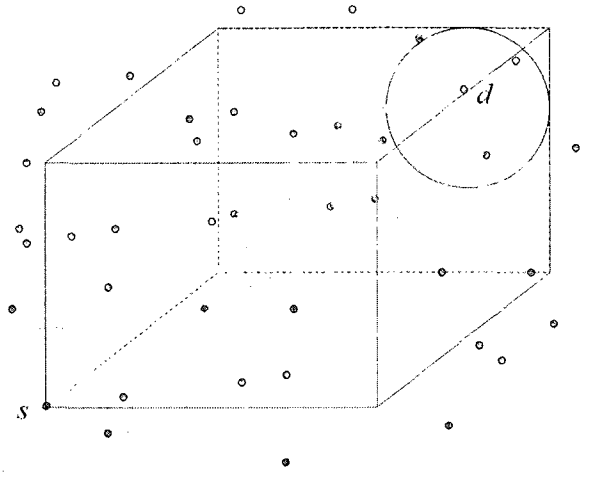


Figure 11: To route from s to d: with 3DDREAM, a current node will forward the packet to all the neighbors' nodes inside the $3D$ cone, while with LAR3D, it will forward the packet to all neighbors' nodes inside the rectangular box

2. The power consumed during message reception

3. The power consumed while the node is idle.

In ad hoc networks, for each ordered pair of nodes (u, v), there is an associated transmission power threshold, denoted by $P(u, v)$, which indicates the transmission power needed by u so that its signal can be received by v. The transmission power threshold for a pair of nodes depends on a number of factors including the distance between the transceivers, interference, noise, environment, etc. [93].

In previous work [7, 94, 104, 114, 115], two main metrics have been used to optimize power routing for a sequence of messages.

The first metric, called the power metric, tries to minimize the energy consumed for each message. If the transmission range is fixed for all the nodes, then the number of nodes in the route path is used as the energy required for the routing task. This metric can be optimized if the nodes can adjust their transmission range. Then the constant metric can be replaced by a power metric that depends on the distances between nodes. In formal terms [51], let e_j be the energy required by the packet

j to traverse a sequence of nodes $n_1, n_2, \ldots, n_k$, where n_1 is the source and n_k is the destination. If $p(n_i, n_{i+1})$ is the power needed to forward j over one hop from n_i to n_{i+1}, then the aim of the power metric is to minimize e_j over all j, where $e_j = \sum_{i=1}^{k-1} p(n_i, n_{i+1})$. A drawback of the power metric is that some nodes may be repeatedly chosen over many routes, which quickly leads to their failure. In many cases this may result in the loss of network connectivity.

The second metric, called network survivability [30, 96] or cost metric [104], tries to maximize the lifetime of the nodes. Given alternative routing paths, select the one that will result in the longest network operation time. One way of optimizing this metric is by choosing the nodes with plenty of energy as relaying nodes. Formally, let the current node be c and $N(c)$ be the set of its neighbors' nodes. Let $g(x)$ be the remaining energy at the node x. Then the next node a is chosen such that $a = x \in N(c) : g(x) < G(y)$ for all $y \in N(c)$. We will focus on the second metric for our routing algorithms in Chapter 7.

Power Consumption Wireless Model

A wireless model has been proposed in [93] in which the power consumption between two nodes at distance ω is expressed as $u(\omega) = \omega^{\alpha} + \beta$, where α is the path loss exponent in the power consumption model and β is a constant that represents the energy consumed in computer-processing and encoding-decoding at both transmitter and receiver. It has been proven in [104] that $u(\omega)$ is optimal if $\omega \leq (\beta/(1-2)^{1-\alpha})^{1/\alpha}$. If ω is greater than that, then the greatest power savings are obtained when ω is divided into $n > 1$ equal length subintervals of size $(\beta/(\alpha - 1))^{1/\alpha}$.

Existing Power-aware Routing Algorithms

Several power-aware routing algorithms that try to minimize the total energy consumed by the packet and also increase the average network lifetime have been proposed [76, 102, 104]. Let the current node be c, a be a neighbor of c, and the

destination be d. Let $h = dist(c, a)$, $p = dist(c, d)$ and $q = dist(a, d)$, where $q < p$. Let the cost of transmitting a packet between two nodes at distance ω be $\ell\omega^{\alpha} + \beta$, where ℓ, α, β are constants that depend on the wireless model. Let $\bar{h}$ be the average length of all edges out from the source s. Let $f(a) = \frac{1}{g(a)}$, where $g(a)$ is the remaining lifetime for the node a. Also, $\bar{f}(a)$ is the average value of $f(x)$ for a and all neighbors x of a, $\bar{g}(a)$ is the average value of $g(x)$ for a and all neighbors x of a; r is the transmission radius. The algorithms are summarized as follows [104]:

- **Power Algorithm:** This algorithm tries to minimize the total energy consumed by the packet in the routing process, regardless of the available energy at the nodes. With the Power algorithm, the current node c chooses as a next node a, which minimizes the expression: $P(c, a) + P(a, d)$, where $P(c, a) = \ell h^{\alpha} + \beta$ is the cost to reach a and $P(a, d) = q\beta(\ell(\alpha - 1)/\beta)^{1/\alpha} + q\ell(\ell(\alpha - 1)/\beta)^{(1-\alpha)/\alpha}$, which is an assumption that the cost for the rest of the routing process are optimal.

- **Cost-*i* Algorithm:** This algorithm uses the cost metric. It tries to maximize the network lifetime by carefully choosing the next node from the set of neighbors with plenty of energy. In this algorithm the current node chooses a next node a, which minimizes the equation: $cost(a) = f(a) * t/r$, where $t = \bar{f}(a)$.

- **Cost-*ii* Algorithm:** Since the factor t is network dependent, there are different versions of the previous algorithm. One of those algorithms is called *Cost-ii*. In this algorithm, the current node chooses one of its neighbors, say a, which minimizes the equation: $cost(a) = f(a) * t/r$, where $t = 1/\bar{g}(a)$.

There are two ways to combine power and cost metrics into a single metric, based on the product or sum of the two metrics.

- **Power*Cost:** In this algorithm, the current node chooses one of its neighbors, say a, which minimizes the following equation: Power*Cost(a)= *Power* *

$Cost(c,a) + Power * Cost(a,d)$, where $Power * Cost(c,a) = f(a) * (\ell h^{\alpha} + \beta)$ and $Power * Cost(a,d) = (q\beta(\ell(\alpha-1)/\beta)^{1/\alpha} + q\ell(\ell(\alpha-1)/\beta)^{(1-\alpha)/\alpha}) * \bar{f}(a)$.

- **Power+Cost:** In this algorithm the current node chooses one of its neighbors a, which minimizes the equation: Power+Cost(a)= $(Power + Cost(c,a)) + (Power+Cost(a,d))$, where $Power+Cost(c,a) = [\bar{f}(s)*(\ell h^{\alpha}+\beta)]+[f(a)*(\ell\bar{h}^{\alpha}+\beta)]$ and $Power+Cost(a,d) = (q\beta(\ell(\alpha-1)/\beta)^{1/\alpha}+q\ell(\ell(\alpha-1)/\beta)^{(1-\alpha)/\alpha})*\bar{f}(a)$.

Kuruvila *et al.* [76] propose another set of power and cost-awareness routing algorithms that choose the next node so to guarantee progress to the destination, assuming such a node exists. In the *Power Progress algorithm*; the current node forwards the packet to one of its neighboring nodes that is closer to the destination than itself and minimizes $(h^{\alpha}+\beta)/(p-q)$. Similarly, in the *Cost Progress algorithm*, the next node is the one that is closer to the destination than the forwarding node and minimizes $f(a)/(p-q)$.

Since all the above algorithms are deterministic algorithms that suffer from the local minimum problem, they do not guarantee the delivery of the message in a connected graph. Stojmenovic *et al.* [102] propose guaranteed delivery algorithms in $2D$ space, which combine Power (P), Cost (C) and Power*Cost (PC) algorithms with Face routing algorithm, similar to the way the Greedy algorithm is combined with Face to define the GFG algorithm. Those algorithms start with P, C or PC forwarding decisions. Once a packet reaches a local minimum, the Face routing starts. If the message arrives to a node closer to the destination than the local minimum node, the algorithm switches back to P, C or PC forwarding again. These algorithms have been called *PFP*, *CFC* and *PCFPC* respectively; because these combined algorithms use face routing, they are not applicable in $3D$ environment.

Chapter 3

Displaced Apex Adaptive Yao Graphs

In this chapter, we introduce a new class of orientation-invariant Yao-type subgraphs of a UDG, that is a generalization of the Yao graph where the cones used 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.

3.1 Displaced Apex Adaptive Yao Graphs

Let V 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.

Let the parameter p be the closed line segment between u and v: $(1-p)u + pz$, $0 \leq p \leq 1$. Any particular choice of p represents the position of the apex of the cone. We will use as a second parameter α, $0 \leq \alpha \leq 1$, to determine θ as a fraction of a maximum cone angle, $\theta_m(p, |uz|)$, which we define shortly, which is a function of p

and the distance from the current node u to the nearest neighbor z for which the cone is determined.

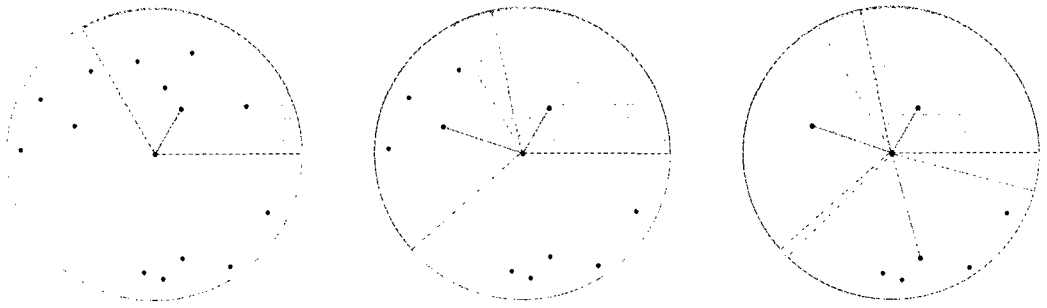
Algorithm 3.1 Displaced Apex Adaptive Yao(G, α, p) graph algorithm

Input: *A graph G with the node set V, an angle parameter α, and a parameter p.*
Output: *A list of directed edges L for each node $u \in V$ which represent the Displaced Apex Adaptive Yao subgraph of G, $\overrightarrow{DAAY}(G, \alpha, p)$.*

for all $u \in V$ **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 $\theta_m(p, |uz|)$.
    (c) Let $r = (1-p)u + pz$ be a point on the line segment uz.
    (d) Consider the cone C with its apex at r with a cone angle $\theta = \alpha \cdot \theta_m(p, |uz|)$ and z in its interior, such that the line uz bisects the cone C into two equal halves.
    (e) Scan the list $LN(u)$ and remove each node in the interior of C.
  until $LN(u)$ is empty
end for

Definition 3.1. *Let G be a UDG with node set V. The directed Displaced Apex Adaptive Yao subgraph, $\overrightarrow{DAAY}(G, \alpha, p)$, is defined to be the graph with node set V whose edges are obtained by applying the Displaced Apex Adaptive Yao(G, α, p) algorithm, Algorithm 3.1, on the graph G using cone angle $\theta = \alpha \cdot \theta_m(p, |uz|)$ and apex displacement parameter p. The undirected graph $DAAY(G, \alpha, p)$ is obtained by ignoring the direction of the edges in $\overrightarrow{DAAY}(G, \alpha, p)$.*

When $p = 0$, 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, p) 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 V. Hence the $DAAY(G, \alpha, p)$ is orientation-invariant. See Figure 12.



(a) (b) (c)

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 then chosen. Note that its associated cone overlaps with the first cone; and (c) the third nearest neighbor is chosen from the list $LN(u)$.

3.2 Displaced Apex Adaptive Yao Properties

Lemma 3.1. *Consider a node u and neighbor z of u. Consider an arbitrary point $k = (1-p)u + pz$ where the parameter p has a value in the range $[0, 1]$. Define L to be the line perpendicular to the line segment uz that intersects uz at its midpoint m (corresponding to $p = 0.5$ 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. Consider the boundary of this cone intersecting the line L at a point c. Then the cone angle θ is defined by*

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

Proof. Consider the triangle $\triangle uck$. Let θ_0 be the interior angle at u. Then the interior angle at c is $\theta - \theta_0$. The interior angle can be determined from the right triangle $\triangle muc$, $\cos(\theta_0) = \frac{1}{2}\frac{|uz|}{|uc|}$. Also, the interior angle at k is $\pi - \theta$. By the sine law,

$$\frac{\sin(\theta - \theta_0)}{\sin(\theta)} = \frac{p|uz|}{|uc|}.$$

□

Corollary 3.1. *Consider a node u and neighbor z of u. Consider an arbitrary point $k = (1-p)u + pz$ where the parameter p has a value in the range $[0,1]$. Define L to be the line perpendicular to the line segment uz that intersects uz at its midpoint m (corresponding to $p = 0.5$ 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. Consider the boundary of this cone intersecting the line L at a point c. If $|uc| = |uz|$ then $\theta_0 = \pi/3$ and*

$$\frac{\sin(\theta - \pi/3)}{\sin(\theta)} = \frac{p|uz|}{|uz|} = p.$$

Definition 3.2. *Consider a node u and neighbor z of u. Consider an arbitrary point $k = (1-p)u + pz$ where the parameter p has a value in the range $[0,1]$. Define L to be the line perpendicular to the line segment uz that intersects uz at its midpoint m (corresponding to $p = 0.5$ 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. Consider the boundary of this cone intersecting the line L at a point c. Define the maximum cone angle $\theta_m(p, |uz|)$ as a function of the parameter p and the distance $|uz|$ as follows:*

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

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

Note that when $0 \le p \le 0.5$, then $\theta_m(p, |uz|)$ is only a function of p such that θ is a fixed angle for fixed values of p and α. When $p = 0.5$, $\theta_m(0.5, |uz|) = \pi/2$ and, if $\alpha = 1$ so that $\theta = \theta_m(p, |uz|)$, we obtain the Half Space Proximal graph [31].

3.2.1 Connectivity

Theorem 3.1. *Consider a node set V and $UDG(V)$ defined on V. If $UDG(V)$ is connected and the cone angle θ is less than or equal to $\theta_m(p, |uz|)$ then $\overrightarrow{DAAY}(UDG(V), \alpha, p)$ is strongly connected.*

Proof. Consider an edge $uv \in UDG(V)$. To show that $\overrightarrow{DAAY}(UDG(V), \alpha, p)$ is strongly connected we will show that there is a directed path from u to v in $\overrightarrow{DAAY}(UDG(V), \alpha, p)$. Proof is done by contradiction. Assume that there is at least one edge $uv \in UDG(V)$ such that there is no directed path from u to v in $\overrightarrow{DAAY}(UDG(V), \alpha, p)$. Let uv be the shortest such edge in $UDG(V)$. This implies that there is an edge $uz \in \overrightarrow{DAAY}(UDG(V), \alpha, p)$ such that $|uz| \leq |uv|$, because the edge uv should be in the cone of uz selected by the Displaced Apex Adaptive Yao$(UDG(V), \alpha, p)$ algorithm.

Now consider the triangle $\triangle uzv$ in Figure 13. The choice of maximum cone angles in Definition 3.2 is based on the idea that v could be placed anywhere in the open half-plane H containing z defined by the line perpendicular to the line segment uz in the middle of uz (the point corresponding to $p = 0.5$ in our parametrization of the uv; labeled as m in Figure 13). Consider the two cases defined by the value of p. First, assume $0 \leq p < 0.5$ (for example, the apex of the cone would be at the point labeled as k in the figure). Then to keep v in the interior of H, the maximum cone angle θ would define a cone that intersects the boundary of H at a point at distance $|uz|$ from u (such a point is labeled as c in the figure). By Corollary 3.1, $\theta_m(p, |uz|)$ is as defined in Definition 3.2.

Now, assume $0.5 \leq p \leq 1$. To keep v in the interior of H, the maximum cone angle θ would define a cone that intersects the boundary of H at a point at distance r from u (such a point is labeled as c in the Figure 13). By Lemma 3.1 and noting that $|uc| = r$, $\theta_m(p, |uz|)$ is as defined in Definition 3.2.

In either case, since the cone angle is less than or equal to $\theta_m(p, |uz|)$, then any position of the node v inside the cone for z such that $|uz| \leq |uv|$ would give $|zv|$ strictly

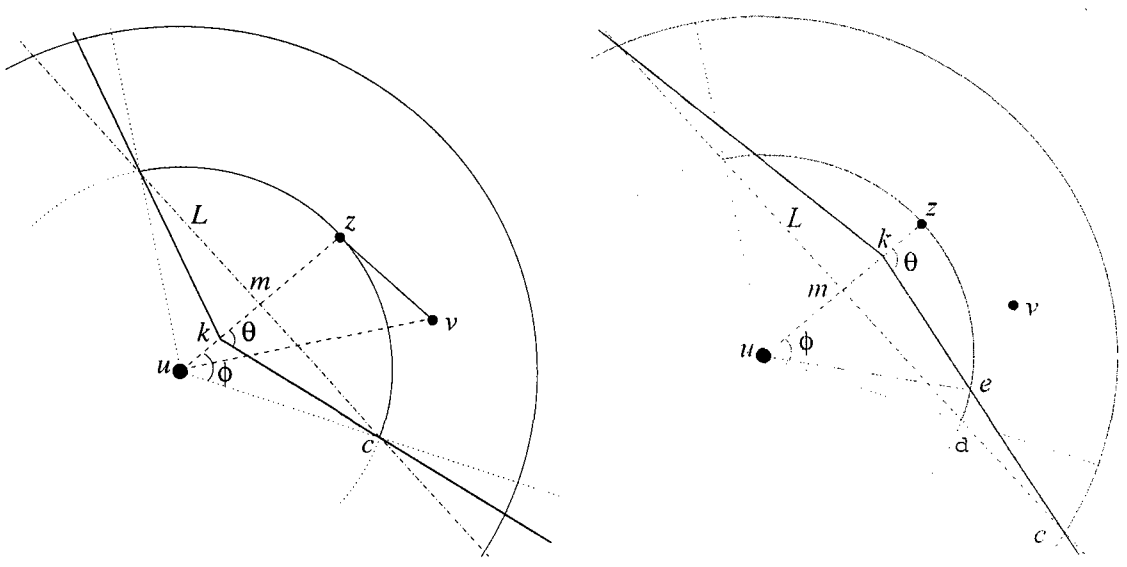


Figure 13: The left diagram is for $0 \leq p < 0.5$. The bottom diagram is for $0.5 \leq p \leq 1$. In both diagrams, the dark shaded area is the forbidden region where any other neighboring nodes are excluded

less than $|uv|$. Since uv is an edge in $UDG(V)$, then zv is also an edge in $UDG(V)$. Therefore, there exists a directed path from z to v in $\overrightarrow{DAAY}(UDG(V), \alpha, p)$, and so there is a directed path from u to v in $\overrightarrow{DAAY}(UDG(V), \alpha, p)$. □

Note that when $p > 0.5$, the further away the nearest neighbor z, the larger the cone angle limit $\theta_m(p, |uz|)$. If a fixed cone angle $\hat{\theta}$ was used, since as $|uz| \to 0$ the cone angle approaches $\pi/2$, it would have to be $\hat{\theta} \leq \pi/2$ to ensure connectedness. Using $\hat{\theta} = \pi/2$ would then give a variation of the Half Space Proximal subgraph with the forbidden zone half-plane intersecting the line segment uz at the point corresponding to $p > 0.5$ rather than at the midway point.

3.2.2 Bounding the Node Out-degree

Theorem 3.2. *The out-degree of any node in* $\overrightarrow{DAAY}(UDG(V), \alpha, p)$, $0 \leq \theta \leq \theta_m(p, |uz|)$, *is at most* $\left\lfloor \frac{2\pi}{\phi} \right\rfloor$ *where* ϕ *is defined by*

$$\frac{\sin(\theta - \phi)}{\sin(\theta)} = p. \tag{2}$$

Proof. From the definition of Displaced Apex Adaptive Yao(G, α, p), the smallest angle between any two edges is θ because any nearest neighbor selected forms an edge that will be outside, or on the boundary of, the cone for any other neighbor. Consider a node u. Let z be the nearest neighbor that defines a cone. For $0 \leq p < 0.5$, the smallest angle between z and another nearest neighbor w is ϕ if w is placed at the intersection c of the cone boundary and the circle of radius uz centered on u (the point labeled as c in Figure 13). By Corollary 3.1, the angle between c and z about u is defined by Eq. 2. If $\theta = \theta_m(p, |uz|)$, $\phi = \pi/3$. Similarly, for $0.5 \leq p \leq 1$, the smallest angle between z and another nearest neighbor w is ϕ if w is placed at the intersection e of the cone boundary and the circle of radius uz centered on u (the point labeled as e in Figure 13). It is straightforward to show, using a proof similar to that for Lemma 3.1 and noting that $|ue| = |uz|$, that the angle ϕ between e and z about u is also defined by Eq. 2. Therefore, the angle between any two selected edges will be greater than or equal to ϕ, which is a function of p. So, the maximum out-degree for any node will be $\frac{2\pi}{\phi}$. Any fraction of a cone overlapping in the worst-case will not add to the out-degree of the node, so the maximum integer out-degree of any node is be $\left\lfloor \frac{2\pi}{\phi} \right\rfloor$.

□

Consider the worst-case example in Figure 14, in (a) the nearest neighbor z is first chosen, and the forbidden area is defined as a cone of 2ϕ angle; (b) the next node w_1 in the worst-case will be exactly on the border of the cone, which will define its'

cone that overlap the first cone; (c) third point w_2 will be on the border of the second cone; (d) after doing all the points we can see that the circle around the node u is divided in the worst-case into $\left\lfloor \frac{2\pi}{\phi} \right\rfloor$ cones.

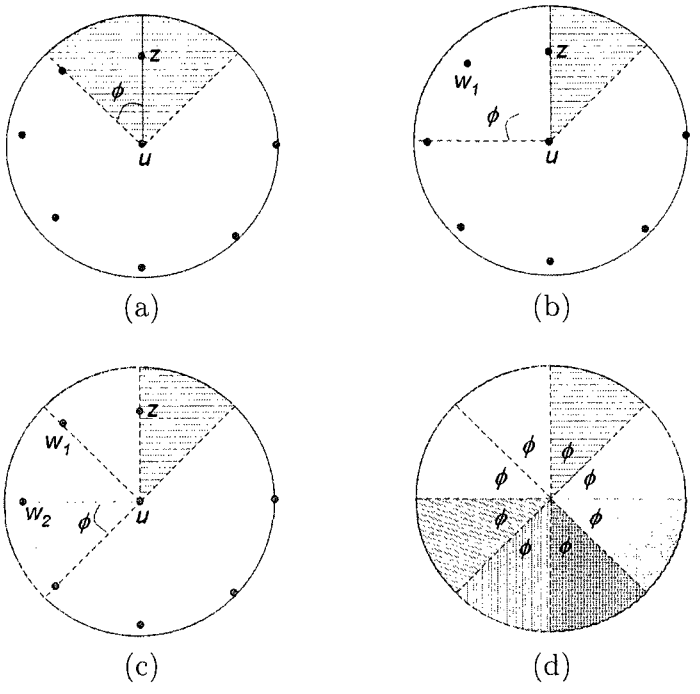


Figure 14: Worst-case example of the node degree in $DAAY(UDG(V), \alpha, p)$

3.2.3 Stretch Factor

Theorem 3.3. *Let $V \subseteq \mathbb{R}^2$ be a set of n points and let $\theta < \pi/3$ be the cone angle. Then $DAAY(UDG(V), \alpha, p)$ is a spanner with stretch factor $\frac{1}{1 - 2\sin(\frac{\theta}{2})}$.*

Proof. Let uv be an edge in UDG that is not selected by Displaced Apex Adaptive Yao(G, α, p) algorithm. Since, by Theorem 3.1, $DAAY(UDG(V), \alpha, p)$ is connected, then there is a shortest path from u to v. Let a "worst" such path from u to v in $DAAY(UDG(V), \alpha, p)$ be $u_0 = u, u_1, u_2, \ldots, u_n = v$. See Figure 15. By the Displaced Apex Adaptive Yao(G, α, p) algorithm, the angle $\angle u_{i+1} u_i v \leq \eta$ (which we will determine) and $|u_i u_{i+1}| < |u_i v|$ since otherwise $u_i v$ would be part of $DAAY(UDG(V), \alpha, p)$

and part of the path. Also, by Theorem 3.1, $|u_{i+1}v| < |u_iv|$ since we can always decrease the distance to v from each u_i along the path.

Now consider the triangle $\triangle\ u_iu_{i+1}v$ (see Figure 15). Let a be the point on u_iv such that $|u_ia| = |u_iu_{i+1}|$. By the triangular inequality $|u_{i+1}v| \leq |u_{i+1}a| + |av|$. Note that $|u_{i+1}a| = (2sin\frac{\eta}{2})|u_iu_{i+1}|$, and $|av| = |u_iv| - |u_iu_{i+1}|$. Applying these two latter equations to the triangular inequality, we obtain

$$|u_{i+1}v| \leq |u_iv| - |u_iu_{i+1}|(1 - 2sin\frac{\eta}{2}).$$

Applying the previous analysis iteratively on the entire path, we have

$$\sum_{0\leq i<m} |u_{i+1}v| \leq \sum_{0\leq i<m} \left(|u_iv| - \left(|u_iu_{i+1}|(1 - 2sin\frac{\eta}{2})\right)\right).$$

Therefore,

$$\sum_{0\leq i<m} (|u_iu_{i+1}|) \leq \left(\frac{1}{1 - 2sin\frac{\eta}{2}}\right) \sum_{0\leq i<m} (|u_iv| - |u_{i+1}v|) \leq \left(\frac{1}{1 - 2sin\frac{\eta}{2}}\right) |u_0v|.$$

Note that for the stretch factor to be bounded by this inequality, then η must be restricted by $\eta < \pi/3$, other wise $1 - 2sin\frac{\eta}{2} = 0$ if $\eta = \pi/3$

To determine the value of η, first consider the largest angle possible between u_i, v, and u_{i+1}. The larger the angle, the larger the stretch factor along the path. If u_{i+1} is a nearest neighbor of u_i defining a cone during the execution of the Displaced Apex Adaptive Yao(G, α, p) algorithm, then placing a node f at the intersection of the cone boundary and the boundary of the circle of radius r centered at u_i would give the largest angle. Defining a triangle $\triangle u_ifu_{i+1}$ and using a similar analysis as used in the proof for Lemma 3.1, the internal angle η at u_i is

$$\frac{\sin(\theta - \eta)}{\sin(\theta)} = \frac{p|u_iu_{i+1}|}{|u_if|}.$$

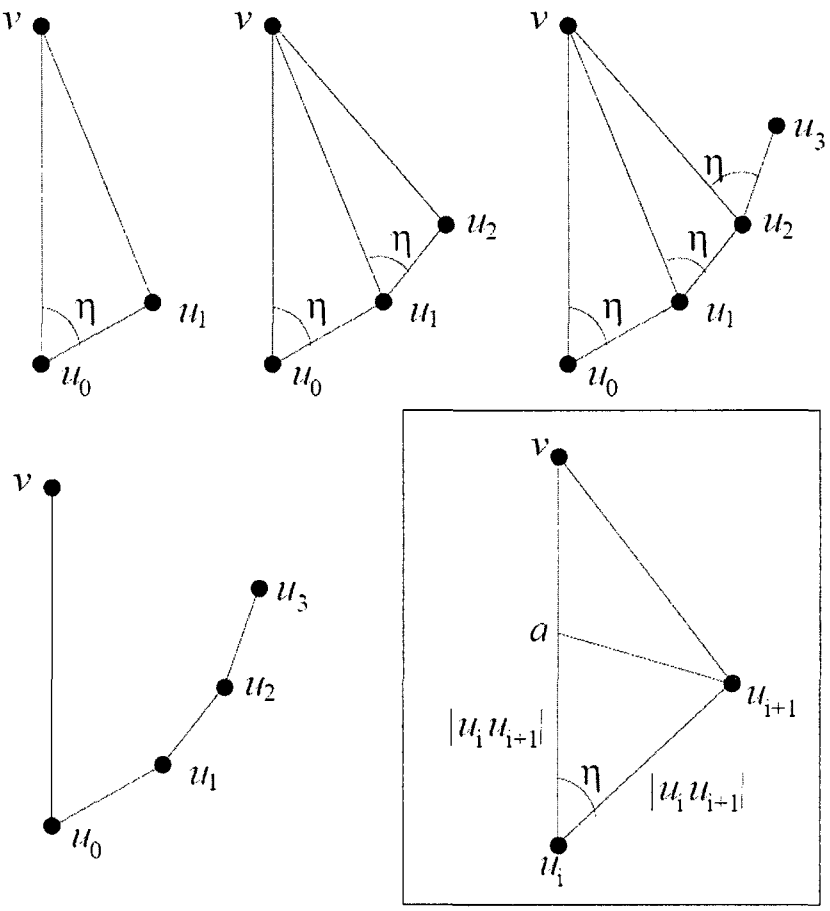


Figure 15: Left: A scenario for the worst shortest path that could be selected in Displaced Apex Adaptive Yao(G, θ, p). The edge (u_0, v) is not selected by the Displaced Apex Adaptive Yao(G, θ, p) algorithm since (u_0, u_1) is shorter than (u_0, v), and (u_0, v) is inside the cone of (u_0, u_1). The same occurs for $(u_1, v), \ldots, (u_n, v)$; Right: One step of the iterative sequence of the path

Note that as $|u_i u_{i+1}|$ approaches 0, the angle η is maximized. For $0 \leq p < 0.5$, this maximum angle is θ, and for $0.5 \leq p \leq 1$, this maximum angle is $\pi/2$. To ensure that $\eta < \pi/3$, in both cases, we must restrict $\theta < \pi/3$ to bound the stretch factor. □

3.2.4 Containing Euclidean Minimum Spanning Tree

Theorem 3.4. *Consider a node set V and $UDG(V)$ defined on V. Assume that $UDG(V)$ is connected. Then DAAY$(UDG(V), \alpha, p)$, $\theta = \alpha \cdot \theta_m(p, |uz|)$, $0 \leq \alpha \leq 1$, contains the Euclidean Minimum Spanning Tree EMST$(UDG(V))$ as a subgraph.*

Proof. Let $EMST(UDG(V))$ be an Euclidean Minimum Spanning Tree of $(UDG(V))$ that contains the maximum number of edges of $DAAY(UDG(V), \alpha, p)$.

We do a proof by contradiction. Assume there is an edge uv in $EMST(UDG(V))$ that is not in $DAAY(UDG(V), \alpha, p)$. This implies that there is an edge $uz \in DAAY(UDG(V), \alpha, p)$ such that $|uz| \leq |uv|$, because the edge uv should be in the cone of another shorter edge selected by the Displaced Apex Adaptive Yao$(UDG(V), \alpha, p)$ algorithm, and $|vz| < |uv|$ (otherwise, by Theorem 3.1,
the $\overrightarrow{DAAY}(UDG(V), \alpha, p)$ would not be strongly connected).

Since $EMST(UDG(V))$ is a spanning tree, there is a path from v (or u) to z. If the path is from v to z, then removing uv from the graph and adding the edge uz we obtain a spanning tree with equal or less weight with an additional edge from $DAAY(UDG(V), \alpha, p)$, a contradiction. If the path is from u to z, then removing uv from the graph and adding the edge vz we obtain a spanning tree with less weight, again a contradiction. □

In [39] experimental results is given to explore the new subgraphs properties in comparison with the existing subgraphs.

3.3 Empirical Results

In our experiments we used randomly chosen connected unit disk graphs on an area of 100×100. We varied 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 23 graphs for each value of N. For all the graphs tested, the transmission radius r used was 15 units.

For each UDG, an Adaptive Yao subgraph (equivalent to a Displaced Apex Adaptive Yao subgraph with $p = 0$), Displaced Apex Adaptive Yao subgraphs with $p = 0.125$ and $p = 0.25$, Half Space Proximal subgraph (equivalent to a Displaced

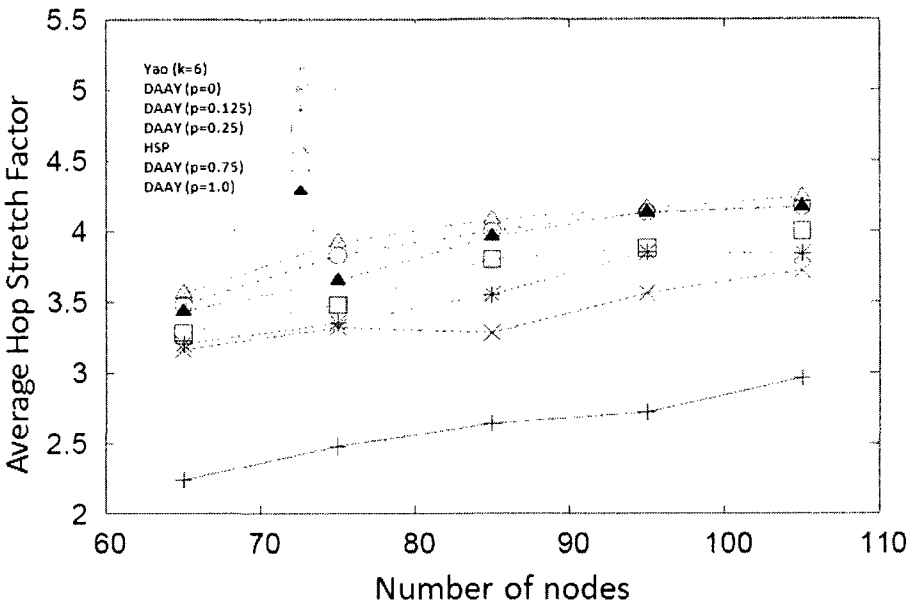


Figure 16: Average hop number stretch factor for each graph with various number of nodes

Apex Adaptive Yao subgraph with $p = 0.5$), and Displaced Apex Adaptive Yao subgraphs with $p = 0.75$ and $p = 1.0$ are generated. For each Displaced Apex Adaptive Yao subgraph we used $\alpha = 1$ such that $\theta = \theta_m(s, |uz|)$ (recall, for $p > 0.5$, θ is a function of the distance to the chosen neighbors). For comparison, we also generate the original YAO subgraphs with $K = 6$ for each UDG. An example of an UDG and related subgraphs is given in Figure 20.

In Figure 16 and Figure 17, we show the average stretch factor in terms of both hop number and Euclidean distance. It is clear that the Adaptive Yao graph ($p = 0$) has consistently the lowest average (hop number or Euclidean length) stretch factor. For our simulations, the average stretch factor for the Adaptive Yao graph was about halfway between that of the HSP and the Yao graph with k = 6. As p increases to 0.5, the average stretch factor increase to a maximum for $p = 0.5$. The stretch factor again decreases as p approaches 1.

In Figure 18 and Figure 19, we show the average node degree and the average in-degree (out-degree). As p approaches 0.5, from Figure 18, the average node degrees

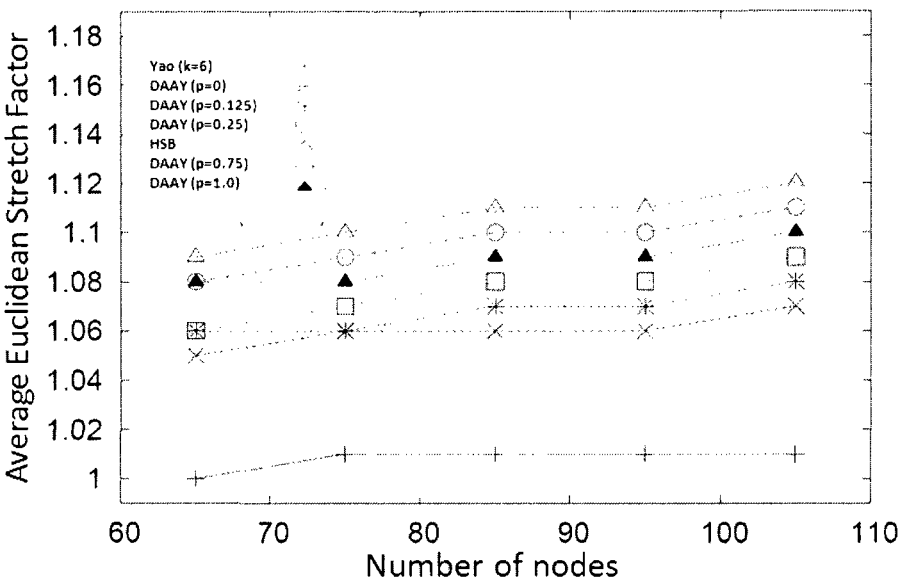


Figure 17: Average Euclidean stretch factor for each graph with various number of nodes

monotonically decrease until $p = 0.5$ when we have the HSP graph. Then as p 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 on the in-degree (out-degree) in Figure 19.

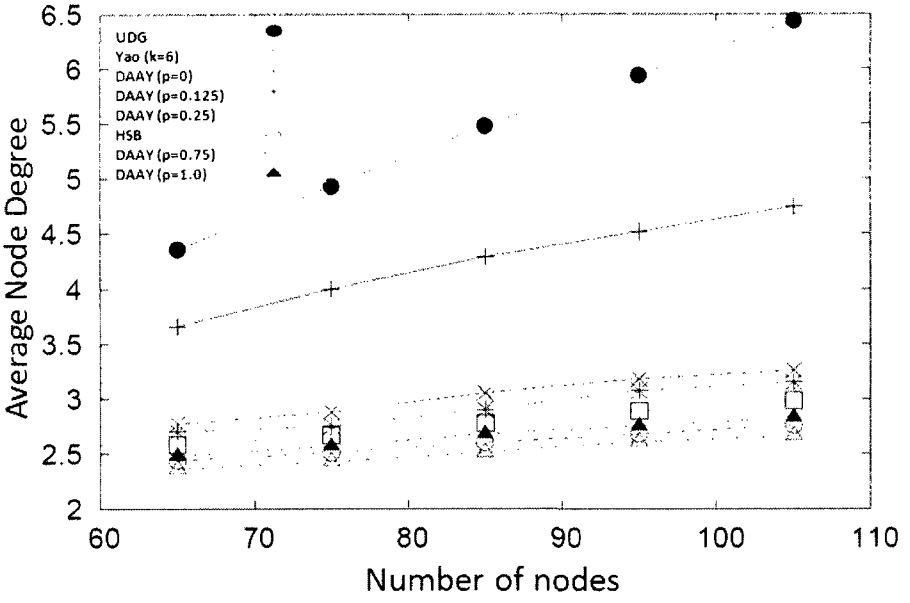


Figure 18: Average node degrees

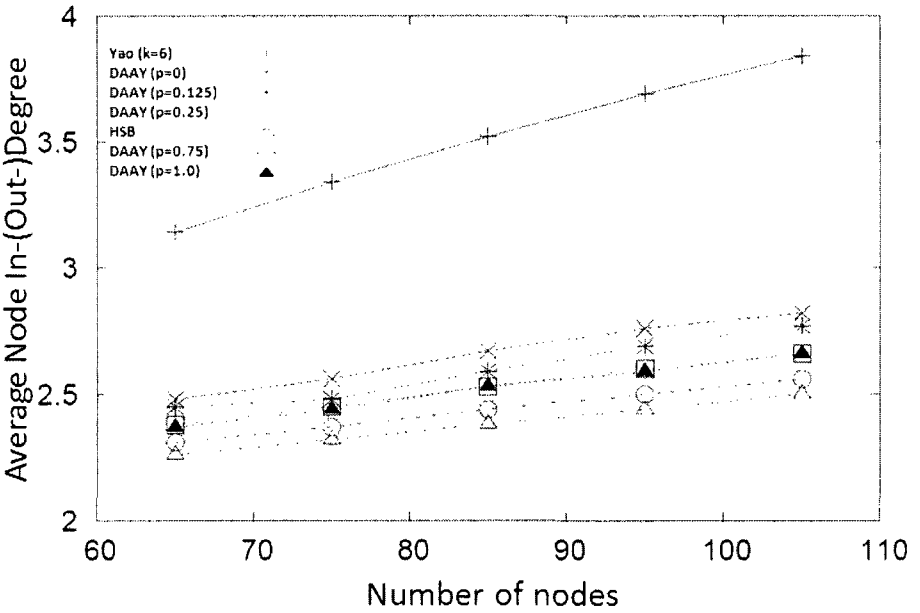


Figure 19: Average in-degree (out-degree)

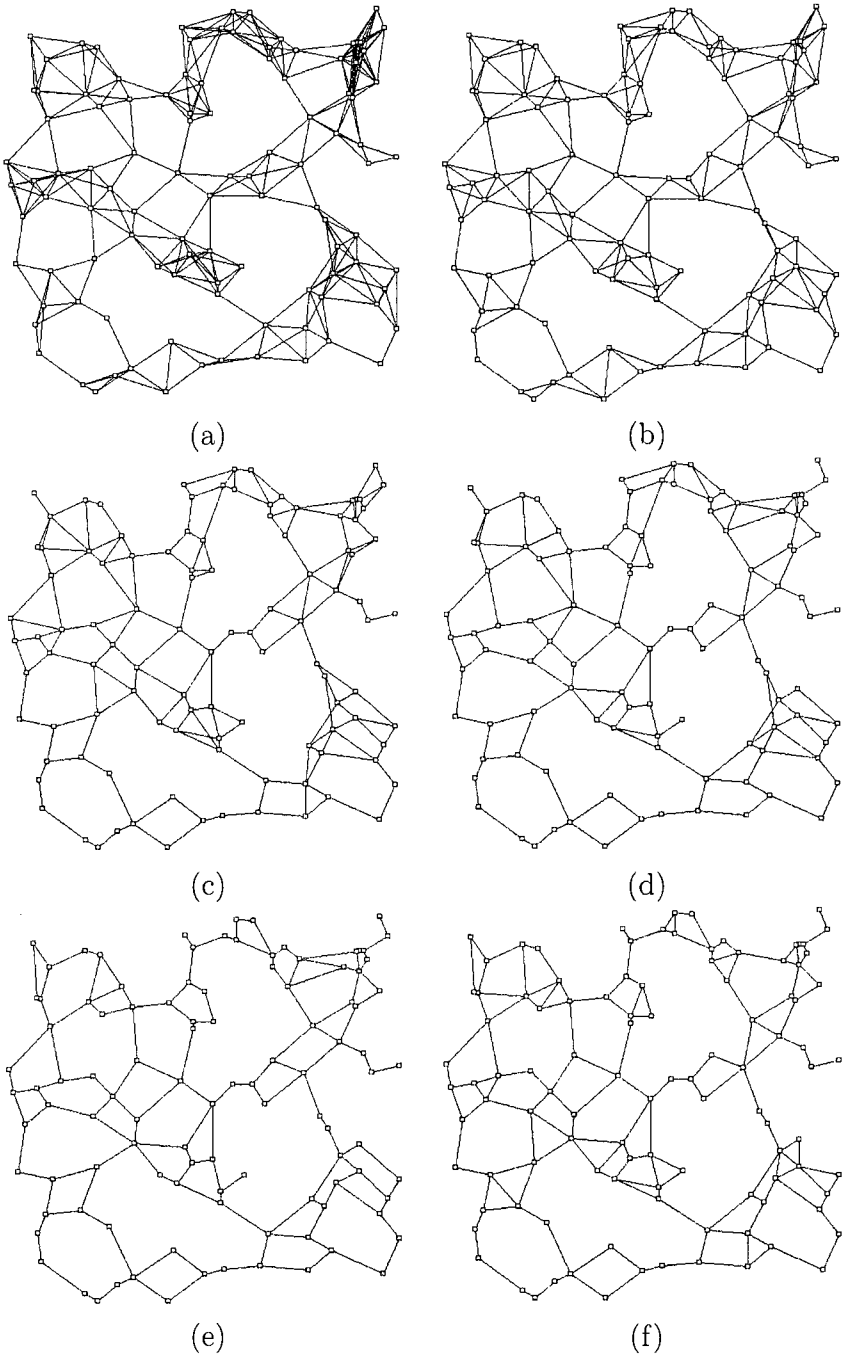


Figure 20: Original UDG and related subgraphs: (a) UDG; (b) Yao Graph with k = 6; (c) Adaptive Yao Graph; (d) Displaced Apex Adaptive Yao Graph with p = 0.25; (e) HSP Graph; (f) Displaced Apex Adaptive Yao Graph with p = 1.0

Chapter 4

A Virtual Backbone Technique to Improve Routing Algorithms

In this chapter we present a truncated octahedron tiling system of the $3D$ space to assign to each node a class number depending on the position of the node within the tiling system. Then, based on this tiling classification system, we present generalizations of the algorithms from [36] for constructing dominating sets and CDSs in $3D$. Theoretical lower bounds on the size of these sets are given and an empirical study of the algorithms is done.

4.1 Tiling System for $3D$ Space

Consider a truncated octahedron of unit diameter as shown in Figure 21 which consists of 14 faces, 6 squares of edge length equal to α, and 8 hexagons of edge length equal to α. Since the truncated octahedron has a diameter equal to 1, then $\alpha = \frac{1}{\sqrt{10}}$.

The tiling system is based on a two level subdivision of the three-dimensional space. At the highest level, the space is tiled with identically shaped tiles that fill the entire space, with no gaps or overlaps. Each tile used in our tiling system consists of 65 truncated octahedra which occupy the entire volume of the tile with no gaps

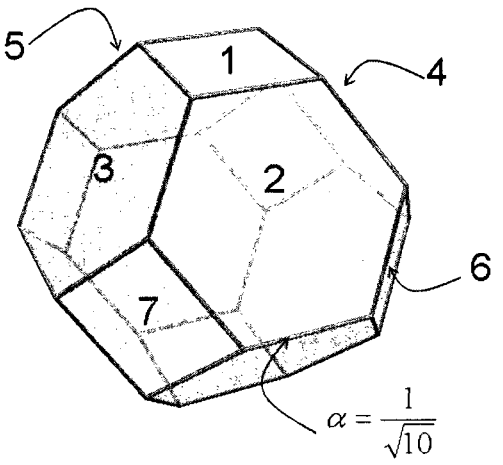


Figure 21: Unit diameter truncated octahedron, the faces labeled $1, 2, ..6$ belong to the class represented by this truncated octahedron

or overlaps. Each truncated octahedron in the tile represents one class which has a unique integer. For any truncated octahedron only faces $1, 2, 3, 4, 5, 6$ and 7 belong to it, see Figure 21. In other words, if a node located exactly on the shared face between two truncated octahedra T_1 and T_2, the node is considered of class 1 if according to T_1 this face is $1, 2, 3, 4, 5, 6$ or 7 otherwise it will be considered to be of class 2. Assume that the first truncated octahedron, class 1, is centered at the coordinates (x_1, y_1, z_1) (ie. the z-axis passes through the center of face 1, the x-axis passes through the center of the edge between face 5 and the face opposite to face 2, and y-axis passes through the center of the edge between face 4 and the face opposite to face 3. We will call this orientation as the *centering orientation*), then the coordinates of the centers of the classes from 2 to 65 are shown on Table 1. They all have the same orientation as class 1. See Figure 22 for an example of the tile used, showing the placement of the truncated octahedra in the tile with the associated classes labels.

Assume that the tiling starts by placing the center of one tile, T_1, at the coordinate (x_1, y_1, z_1), with orientation equal to the centering orientation. To cover all the faces of T_1 we need 14 other adjacent tiles that are in contact with T_1 in the positions

Table 1: Coordinates of the 65 truncated octahedra for a tile T_1 centered at (x_1, y_1, z_1), where $\alpha = \frac{1}{\sqrt{10}}$.

Class 1	(x_1, y_1, z_1)	Class 34	$(x_1 + 2\alpha, y_1 + 4\alpha, z_1 + \alpha\sqrt{2})$
Class 2	$(x_1, y_1, z_1 + 2\alpha\sqrt{2})$	Class 35	$(x_1 - 2\alpha, y_1 + 4\alpha, z_1 + \alpha\sqrt{2})$
Class 3	$(x_1, y_1, z_1 - 2\alpha\sqrt{2})$	Class 36	$(x_1 + 2\alpha, y_1 - 4\alpha, z_1 + \alpha\sqrt{2})$
Class 4	$(x_1 + 2\alpha, y_1, z_1 + \alpha\sqrt{2})$	Class 37	$(x_1 - 2\alpha, y_1 - 4\alpha, z_1 + \alpha\sqrt{2})$
Class 5	$(x_1 + 2\alpha, y_1, z_1 - \alpha\sqrt{2})$	Class 38	$(x_1 + 2\alpha, y_1, z_1 - 3\alpha\sqrt{2})$
Class 6	$(x_1 - 2\alpha, y_1, z_1 + \alpha\sqrt{2})$	Class 39	$(x_1 - 2\alpha, y_1, z_1 - 3\alpha\sqrt{2})$
Class 7	$(x_1 - 2\alpha, y_1, z_1 - \alpha\sqrt{2})$	Class 40	$(x_1, y_1 + 2\alpha, z_1 - 3\alpha\sqrt{2})$
Class 8	$(x_1, y_1 + 2\alpha, z_1 + \alpha\sqrt{2})$	Class 41	$(x_1, y_1 - 2\alpha, z_1 - 3\alpha\sqrt{2})$
Class 9	$(x_1, y_1 + 2\alpha, z_1 - \alpha\sqrt{2})$	Class 42	$(x_1 + 2\alpha, y_1 + 2\alpha, z_1 - 2\alpha\sqrt{2})$
Class 10	$(x_1, y_1 - 2\alpha, z_1 + \alpha\sqrt{2})$	Class 43	$(x_1 + 2\alpha, y_1 - 2\alpha, z_1 - 2\alpha\sqrt{2})$
Class 11	$(x_1, y_1 - 2\alpha, z_1 - \alpha\sqrt{2})$	Class 44	$(x_1 - 2\alpha, y_1 + 2\alpha, z_1 - 2\alpha\sqrt{2})$
Class 12	$(x_1 + 2\alpha, y_1 + 2\alpha, z_1)$	Class 45	$(x_1 - 2\alpha, y_1 - 2\alpha, z_1 - 2\alpha\sqrt{2})$
Class 13	$(x_1 + 2\alpha, y_1 - 2\alpha, z_1)$	Class 46	$(x_1 + 4\alpha, y_1, z_1 - 2\alpha\sqrt{2})$
Class 14	$(x_1 - 2\alpha, y_1 + 2\alpha, z_1)$	Class 47	$(x_1 - 4\alpha, y_1, z_1 - 2\alpha\sqrt{2})$
Class 15	$(x_1 - 2\alpha, y_1 - 2\alpha, z_1)$	Class 48	$(x_1, y_1 + 4\alpha, z_1 - 2\alpha\sqrt{2})$
Class 16	$(x_1, y_1, z_1 + 4\alpha\sqrt{2})$	Class 49	$(x_1, y_1 - 4\alpha, z_1 - 2\alpha\sqrt{2})$
Class 17	$(x_1, y_1, z_1 - 4\alpha\sqrt{2})$	Class 50	$(x_1 + 4\alpha, y_1 + 2\alpha, z_1 - \alpha\sqrt{2})$
Class 18	$(x_1 + 2\alpha, y_1, z_1 + 3\alpha\sqrt{2})$	Class 51	$(x_1 + 4\alpha, y_1 - 2\alpha, z_1 - \alpha\sqrt{2})$
Class 19	$(x_1 - 2\alpha, y_1, z_1 + 3\alpha\sqrt{2})$	Class 52	$(x_1 - 4\alpha, y_1 + 2\alpha, z_1 - \alpha\sqrt{2})$
Class 20	$(x_1, y_1 + 2\alpha, z_1 + 3\alpha\sqrt{2})$	Class 53	$(x_1 - 4\alpha, y_1 - 2\alpha, z_1 - \alpha\sqrt{2})$
Class 21	$(x_1, y_1 - 2\alpha, z_1 + 3\alpha\sqrt{2})$	Class 54	$(x_1 + 2\alpha, y_1 + 4\alpha, z_1 - \alpha\sqrt{2})$
Class 22	$(x_1 + 2\alpha, y_1 + 2\alpha, z_1 + 2\alpha\sqrt{2})$	Class 55	$(x_1 - 2\alpha, y_1 + 4\alpha, z_1 - \alpha\sqrt{2})$
Class 23	$(x_1 + 2\alpha, y_1 - 2\alpha, z_1 + 2\alpha\sqrt{2})$	Class 56	$(x_1 + 2\alpha, y_1 - 4\alpha, z_1 - \alpha\sqrt{2})$
Class 24	$(x_1 - 2\alpha, y_1 + 2\alpha, z_1 + 2\alpha\sqrt{2})$	Class 57	$(x_1 - 2\alpha, y_1 - 4\alpha, z_1 - \alpha\sqrt{2})$
Class 25	$(x_1 - 2\alpha, y_1 - 2\alpha, z_1 + 2\alpha\sqrt{2})$	Class 58	$(x_1 + 4\alpha, y_1 + 4\alpha, z_1)$
Class 26	$(x_1 + 4\alpha, y_1, z_1 + 2\alpha\sqrt{2})$	Class 59	$(x_1 + 4\alpha, y_1 - 4\alpha, z_1)$
Class 27	$(x_1 - 4\alpha, y_1, z_1 + 2\alpha\sqrt{2})$	Class 60	$(x_1 - 4\alpha, y_1 + 4\alpha, z_1)$
Class 28	$(x_1, y_1 + 4\alpha, z_1 + 2\alpha\sqrt{2})$	Class 61	$(x_1 - 4\alpha, y_1 - 4\alpha, z_1)$
Class 29	$(x_1, y_1 - 4\alpha, z_1 + 2\alpha\sqrt{2})$	Class 62	$(x_1 + 4\alpha, y_1, z_1)$
Class 30	$(x_1 + 4\alpha, y_1 + 2\alpha, z_1 + \alpha\sqrt{2})$	Class 63	$(x_1 - 4\alpha, y_1, z_1)$
Class 31	$(x_1 + 4\alpha, y_1 - 2\alpha, z_1 + \alpha\sqrt{2})$	Class 64	$(x_1, y_1 + 4\alpha, z_1)$
Class 32	$(x_1 - 4\alpha, y_1 + 2\alpha, z_1 + \alpha\sqrt{2})$	Class 65	$(x_1, y_1 - 4\alpha, z_1)$
Class 33	$(x_1 - 4\alpha, y_1 - 2\alpha, z_1 + \alpha\sqrt{2})$		

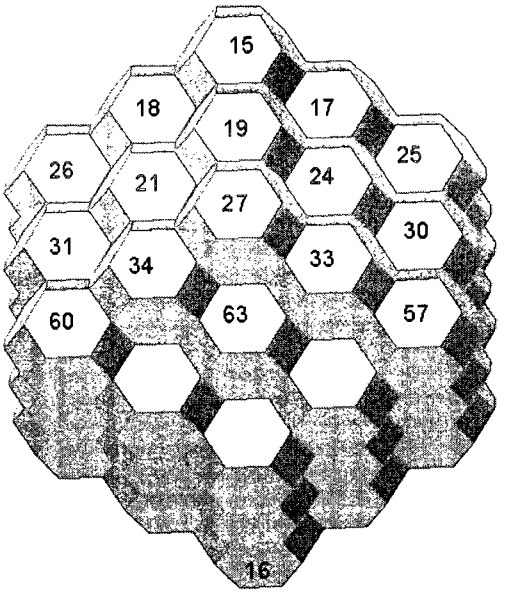
Table 2: Coordinates of the 14 tiles around T_1, with $\alpha = \frac{1}{\sqrt{10}}$.

T_j	(X_1, y_1, z_1)
T_{j1}	$(X_1, y_1, z_1 - 10\alpha\sqrt{2})$
T_{j2}	$(X_1, y_1, z_1 - 10\alpha\sqrt{2})$
T_{j3}	$(X_1 + 6\alpha, y_1 + 4\alpha, z_1 + 5\alpha\sqrt{2})$
T_{j4}	$(X_1 + 6\alpha, y_1 + 4\alpha, z_1 - 5\alpha\sqrt{2})$
T_{j5}	$(X_1 - 4\alpha, y_1 + 6\alpha, z_1 + 5\alpha\sqrt{2})$
T_{j6}	$(X_1 - 4\alpha, y_1 + 6\alpha, z_1 - 5\alpha\sqrt{2})$
T_{j7}	$(X_1 - 6\alpha, y_1 - 4\alpha, z_1 + 5\alpha\sqrt{2})$
T_{j8}	$(X_1 - 6\alpha, y_1 - 4\alpha, z_1 - 5\alpha\sqrt{2})$
T_{j9}	$(X_1 + 4\alpha, y_1 - 6\alpha, z_1 + 5\alpha\sqrt{2})$
T_{j10}	$(X_1 + 4\alpha, y_1 - 6\alpha, z_1 - 5\alpha\sqrt{2})$
T_{j11}	$(X_1 + 10\alpha, y_1 + 2\alpha, z_1)$
T_{j12}	$(X_1 + 10\alpha, y_1 - 2\alpha, z_1)$
T_{j13}	$(X_1 - 10\alpha, y_1 + 2\alpha, z_1)$
T_{j14}	$(X_1 - 10\alpha, y_1 - 2\alpha, z_1)$

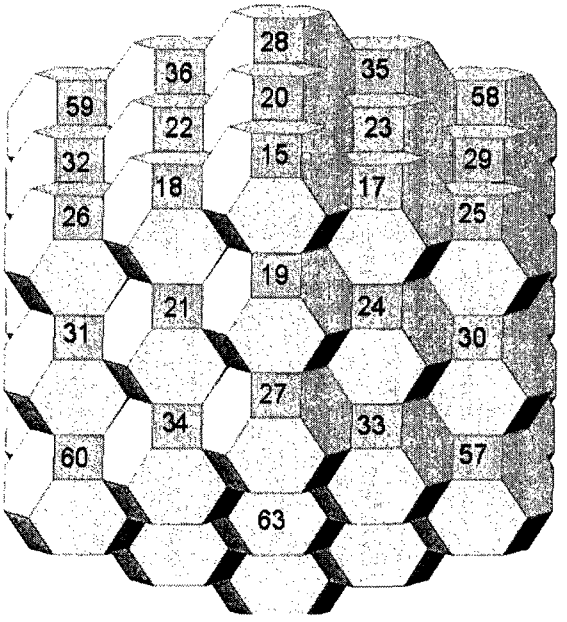
summarized on Table 2. Each tile has the same orientation as T_1. Figure 23 shows the space tiling process used in our algorithm. It is clear that any node can calculate locally its class number by determining to which tile and corresponding truncated octahedron it belongs. In the following we prove some properties of our space tiling system.

Lemma 4.1. *In the 3D space tiling system above, any two nodes that are of the same class number, but belong to two different truncated octahedra, are at Euclidean distance greater than* 2.

Proof. Consider the center of the truncated octahedron of class i at the coordinate (x_1, y_1, z_1) then there are 14 truncated octahedron of the same class i from adjacent tiles centered on the coordinates shown in Table 2. To prove this lemma, we need to prove that any node of class i is at a distance of more than 2 to any other node in the 14 truncated octahedra of the same class in the adjacent tiles. The distances are as follows:



(a)



(b)

Figure 22: The tile used in the tiling system divided into 65 truncated octahedra of diameter 1 and the class numbering associated with the truncated octahedra (a) is the side view (looking along the y-axis) of the tile; (b) is the top-side view of the tile

1. The smallest distance between the two truncated octahedra of class i from T_j and T_{j1} is equal to the distance between the two centers minus two times the distance from the center to the border of the truncated octahedron,
$\sqrt{(x_1 - x_1)^2 + (y_1 - y_1)^2 + (z_1 + 10\alpha\sqrt{2} - z_1)^2} - 2\alpha\sqrt{2} \approx 3.577$

2. The smallest distance between the two truncated octahedra of class i from T_j and T_{j2} equals

$$\sqrt{(x_1 - x_1)^2 + (y_1 - y_1)^2 + (z_1 - 10\alpha\sqrt{2} - z_1)^2} - 2\alpha\sqrt{2} \approx 3.577$$

3. The smallest distance between the two truncated octahedra of class i from T_j and from one of T_{j3}, T_{j4}, T_{j7} or T_{j8} equals
$\sqrt{(x_1 \pm 6\alpha - x_1)^2 + (y_1 \pm 4\alpha - y_1)^2 + (z_1 \pm 5\alpha\sqrt{2} - z_1)^2} - 2\alpha\sqrt{2} \approx 2.299$

4. The smallest distance between the two truncated octahedra of class i from T_j and from one of T_{j5}, T_{j6}, T_{j9} or T_{j10} equals
$\sqrt{(x_1 \pm 4\alpha - x_1)^2 + (y_1 \pm 6\alpha - y_1)^2 + (z_1 \pm 5\alpha\sqrt{2} - z_1)^2} - 2\alpha\sqrt{2} \approx 2.299$

5. The smallest distance between the two truncated octahedra of class i from T_j and from one of $T_{j11}, T_{j12}, T_{j13}$ or T_{j14} equals
$\sqrt{(x_1 \pm 10\alpha - x_1)^2 + (y_1 \pm 2\alpha - y_1)^2 + (z_1 - z_1)^2} - 2\alpha\sqrt{2} \approx 2.330.$

□

In our initial study of tiling the $3D$ space, we used a cube of unit diameter as a cell (class) instead of a truncated octahedron. We found that each tile would need at least 125 cubes to guarantee that the nodes with the same class number in different tiles are separated by a distance greater than 2. Compared to using a truncated octahedron, this number of cubes would increase the constant number of rounds required for the algorithms in Sections 4.2 and 4.4.

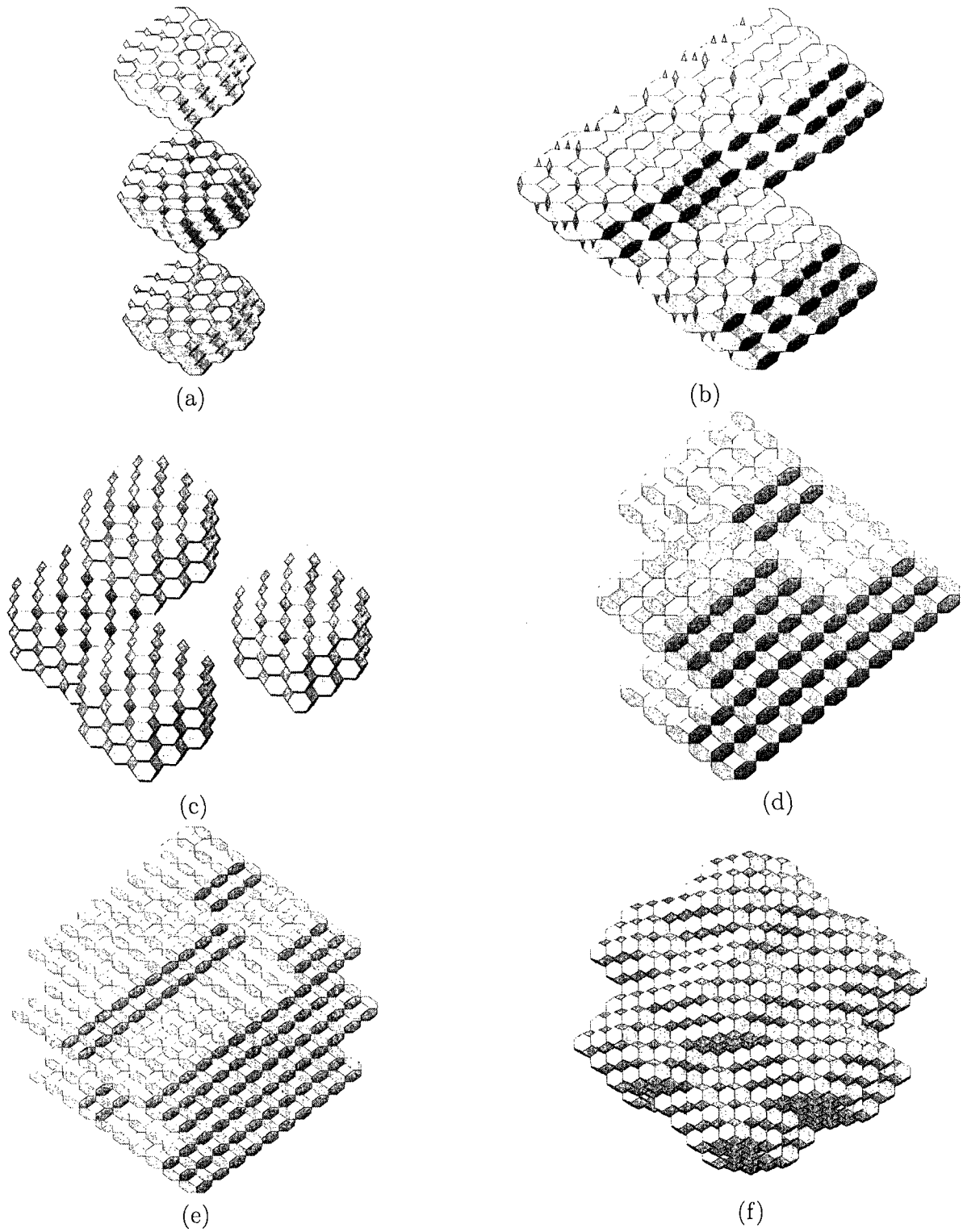


Figure 23: The tiling system used. (a) T_2 and $T3$ on top and bottom of T_1; (b) T_4 and T_5 added to the upper and lower side of T_1; (c) T_1, T_{11}, T_{12} and T_{13} side view; (d)T_1, T_{11}, T_{12} and T_{13} top view; (e) and (f) the final view of the space tiling

4.2 A Local Algorithm For 3D Independent Dominating Sets (3D-LIDS)

Using the space partition described in the previous section, each node can determine its class number locally (using a constant number of arithmetic operations). Because the nodes are aware of the locations of all their neighbors, they can also calculate the class number of each neighbor. It is clear that the nodes that are in same truncated octahedron are neighbors because the diameter of the truncated octahedron is 1.

Our local construction of the dominating sets is based on a similar algorithm proposed by Czyzowicz *et al.* [36] for $2D$. In this algorithm the dominator node m can be chosen according to different heuristics; e.g. the node with the highest degree; the node with the maximum power-level, if the power saving is an important issue for the algorithm; or the node closest to the center of the truncated octahedron. (This latter heuristic is used in our algorithm description and the simulation results in Section 4.5).

Let T_x be the truncated octahedron that contains the node x. Each node x independently does one of the following depending on its class number and two hop information:

- If x is of class 1, then a node m closest to the center of T_x in the same truncated octahedron (T_x) will be designated as a dominator.
- If x is of class other than 1, using the information about its' neighbors, x defines a set $S_1(x)$ of all nodes in the same truncated octahedron that have no neighbor of lower class, and then chooses from $S_1(x)$ a node m closest to center of T_x to be a dominator.
- If x is of class other than 1, and the set $S_1(x)$ is empty, then x requests from every neighbor i of lower class number to run the algorithm if not already running. When all nodes in T_x finish their calculations, node m from T_x that is

not dominated and closest to the center of T_x becomes a dominator.

When the node finishes its calculation, it informs all the neighbors that the dominator selection is completed in its truncated octahedron. The algorithm 3D-LIDS in detail is shown in Algorithm 4.1.

Algorithm 4.1 3D-LIDS($x, N(x), N2(x)$)

1: ***Input*** *node* x, $N(x)$, $N2(x)$.
2: ***Output:*** *Return a set of nodes added to the dominator set Dom after executing the algorithm for the node* x.
3: Let $cls(x)$ represent the class number of x
4: $Cn \leftarrow$ geometric center of T_x
5: x calculates $S_1(x)$
6: **if** $(cls(x) = 1)$ **then**
7: $m \leftarrow u \in N(x) \cup x : cls(u) = 1$ and $dist(u, Cn) < dist(w, Cn)$ for all $w \in N(x) \cup x$ and $cls(w) = 1$.
8: $Dom \leftarrow Dom \cup m$
9: **else if** ($cls(x) \neq 1$ and $S_1(x)$ is not empty) **then**
10: $m \leftarrow u \in S_1(x) : dist(u, Cn) < dist(w, Cn)$ for all $w \in S_1(x)$.
11: $Dom \leftarrow Dom \cup m$
12: **else**
13: **for** $i \leftarrow 1$ to the number of nodes in $N(x)$ **do**
14: **if** ($cls(i) < cls(x)$) **then**
15: the node x will wait the node i to finish
16: **end if**
17: **end for**
18: $m \leftarrow u \in N(x) \cup x : u$ is not dominated and $cls(u) = cls(x)$ and $dist(u, Cn) < dist(w, Cn)$ for all $w \in N(x) \cup x$ and $cls(w) = cls(x)$ and not dominated.
19: $Dom \leftarrow Dom \cup m$
20: **end if**
21: x informs all its neighbors that a dominator selection in its truncated octahedron is completed and give them the results.

4.3 Properties of 3D-LIDS

Before we bound the size of the independent dominating set resulted from 3D-LIDS, we will describe some of its properties. Let Dom be the set of dominator nodes that results from applying 3D-LIDS on each node V.

4.3.1 Locality

Lemma 4.2. *The selection of a dominator in a truncated octahedron of Class i by Algorithm 4.1 depends only on the nodes that are at most $i-1$ hops away from the nodes in the given truncated octahedron.*

Proof. We consider here two cases. **Case 1**: the selection of a dominator of a node in truncated octahedron (T_j) of class 1. This is done by checking only the nodes inside that T_j, which is not more than 1 hop away. **Case 2**: the selection of a dominator of a node in a truncated octahedron (T_j) of class $i \neq 1$. Here the algorithm waits the results that comes from neighbors nodes of lower classes, so eventually it will reaches nodes of class 1 after at most $i-1$ steps.

□

Lemma 4.2 proves that the 3D-LIDS is a local algorithm because it terminates in a constant number of steps.

4.3.2 Domination and Independence

The $2D$ version of the following two lemmas are proved in [36], and the $3D$ version of these lemmas are included here for completeness.

Lemma 4.3. *Let G be a connected UDG. Dom is a dominating set of G.*

Proof. Select a vertex v in G. We show that v is either in *Dom* or adjacent to a vertex in *Dom*. Assume that v is not in *Dom*. We consider the following two cases:

1. If v is of class 1, then one of the nodes in the truncated octahedron containing v is designated as a dominator in line 7.

2. If v is of class $i > 1$ and at least one node of its truncated octahedron is not dominated by a node of an adjacent lower class, one of the nodes in the truncated octahedron is designated as a dominator in line 18.

In both cases, since the diameter of the truncated octahedron is 1, node v is dominated by the designated node.

□

Lemma 4.4. *The Euclidean distance between any two nodes of Dom is more than one. Thus Dom is an independent set of G.*

Proof. Any truncated octahedron contains at most one element of *Dom*. According to Lemma 4.1, the distance between any two vertices of different truncated octahedra of the same class is greater than 2. Thus, the dominators selected in the truncated octahedron of class i for any fixed i are independent. In a truncated octahedron of class $i > 1$, a dominator is designated in line 18 only as a vertex that is not dominated by an adjacent element in *Dom* of lower class. Thus the distance of a dominator belonging to class i to dominators of class $j < i$ is more than 1. □

4.3.3 Other Related Properties

Lemma 4.5. *For any dominator node $u \in Dom$, there is at least another dominator $v \in Dom$ such that the hop distance between them is at most* 3.

Proof. For contradiction, assume there is no path in UDG between a node $u \in Dom$ and any other node $v \in Dom$ of length less than 3 hops. So the shortest path has at least 4 hops, u, x_1, x_2, x_3, v. By Lemma 4.2, It is clear that x_1 and x_3 are not dominators because they are neighbors to other dominators. This gives us two cases.

1. x_2 is a dominator which makes the hop distance between u and x_2 equal to 2, a contradiction.

2. x_2 is a neighbor to a dominator node w which makes the hop distance between u and w equal to 3, a contradiction.

□

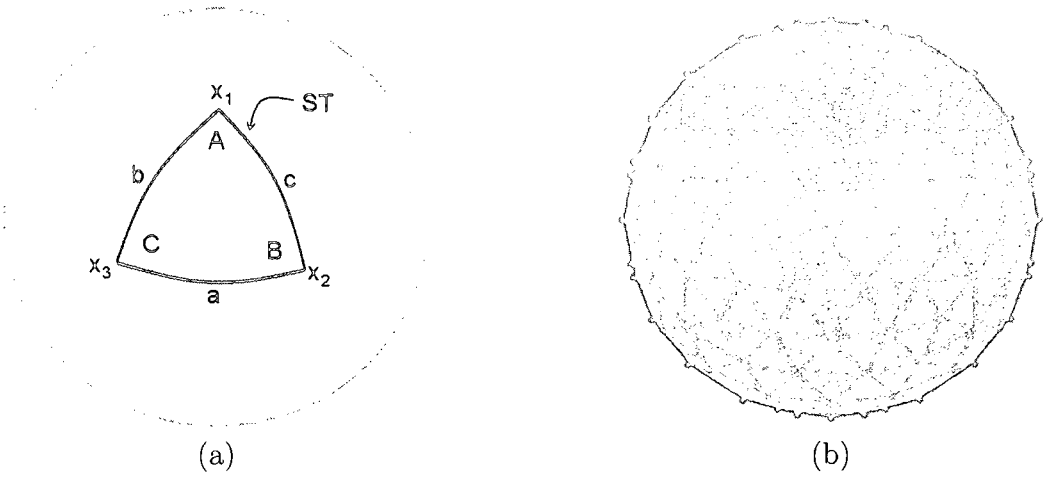


Figure 24: (a) Spherical triangle of equal length sides; (b) The worst-case of having m equal size spherical triangles on the surface of S_0

Lemma 4.6. *For any node u, the number of dominators inside the sphere centered at u with a radius of k units is bounded by a constant η_k*

Proof. The proof is similar to that given in [11] for $2D$. It is known that the distance between any two dominators is greater than one unit. Then the half-unit radius spheres centered at the dominators are disjoint. So to find how many dominators can lie inside a sphere of a radius K units centered at some node u, we have to find the ratio of the volume of the sphere of radius $k + 0.5$ to the volume of the disjoint spheres of radius 0.5. Thus

$$\eta_k = \frac{\frac{4}{3}\pi(k+0.5)^3}{\frac{4}{3}\pi(0.5)^3}. \tag{3}$$

When $k = 2$ or 3, we have $\eta_k = 125$ or 343, respectively.

□

Theorem 4.1. *Let $G = UDG(V, E)$ be a unit disk graph and Dom be a set of dominators for G calculated by Algorithm 1. For any optimal dominating set Dom^* of G, we have $|Dom|/|Dom^*| < 24$. Thus the competitive ratio of Algorithm 4.1 is*

24.

Proof. According to Lemmas 4.3 and 4.4, *Dom* is a dominating and independent set of G. Let Dom^* be the minimum dominating set of G and N be a vertex of Dom^*. Consider the elements of *Dom* that are dominated by N, e.g. the nodes of *Dom* that lie in the sphere of radius r around N. We will call this sphere S_0. Consider the following two cases:

- **Case 1:** If one of the elements in *Dom* is equal to N, then by Lemma 4.4, no other element of *Dom* can be in S_0.

- **Case 2:** If no element of *Dom* is equal to N, then by Lemma 4.4, the nodes from *Dom* are at a distance greater than r. So to maximize the number of nodes from *Dom* that are covered by N, they should be on the surface of S_0. Let x_1 be an element from *Dom* which lies on the surface of S_0; the second node, x_2, should be, in the worst-case, exactly r units far from N and x_1; the third node x_3 should be exactly r units distance from N, x_1 and x_2. The shape that results from those 3 nodes is a spherical triangle ST. See Figure 24(a). The sides of this spherical triangle have equal angular lengths.

  In the worst-case there will be m equal size spherical triangles next to each other on the surface of S_0. See Figure 24(b). Thus the number of the spherical triangles m is equal to

$$m = \left\lfloor \frac{\text{surface area of the sphere } S_0}{\text{area of the spherical triangle } ST} \right\rfloor \tag{4}$$

  The area of $S_0 = 4\pi r^2$. Let ST have the angles A, B and C (measured in radians at the vertices along the surface of the sphere), and the angular lengths of the sides of the triangle (in radians) be $a \equiv \angle x_1 N x_2$, $b \equiv \angle x_1 N x_3$ and $c \equiv \angle x_2 N x_3$. Then the area of $ST = r^2[(A + B + C) - \pi]$ (see [8, 22] for the proof). Since the distances between the 4 nodes x_1, x_2, x_3 and N are equal, then

the angles A, B and C are equal, and the sides lengths are $a = b = c = \frac{\pi}{3}$ [22]. Let $s = \frac{1}{2}(a+b+c) = \frac{\pi}{2}$. It is known that

$$\tan(\frac{1}{2}A) = \sqrt{\frac{\sin(s-b)\sin(s-c)}{\sin s \sin(s-a)}} = \sqrt{\frac{\sin^2(\frac{\pi}{6})}{\sin\frac{\pi}{2}\sin\frac{\pi}{6}}}. \quad (5)$$

Therefore, $A = 2{*}tan^{-1}(\sqrt{(0.5)}) \approx 1.231$ and the area of $ST = r^2[(3.693) - \pi] = 0.551r^2$. By substituting this value in Equation (4), we obtain

$$m = \left\lfloor \frac{4\pi r^2}{0.551r^2} \right\rfloor = 22. \quad (6)$$

Thus, in the worst-case, around one node from Dom^* there are 24 nodes from Dom: The first triangle adds 3 nodes and the rest of the triangles add one node each. This makes the competitive ratio of Algorithm 4.1 at most 24.

□

4.4 Connected Dominating Set Locally

Our local algorithm to construct a CDS consists of two phases. In the first phase, an IDS is found using Algorithm 4.1. In the second phase, each dominator creates paths connecting dominators that are at most three hops apart. There are many algorithms proposed to connect a set of dominators [11, 12, 13, 46], most of them depend on using three hops node information.

In our algorithm the connector node can be chosen between different candidates according to different heuristics; e.g. the node with the maximum power level, if the power saving is an important issue for the algorithm; the node closest to a dominator; or the node with the highest degree; (This latter heuristic is used in our algorithm description and the simulation results in Section 4.5).

Our algorithm for finding the connectors can be described as follows: each node x independently does the following:

1. For every dominator node y in $N2(x)$ with a lower class number than x. Node x chooses from $N(x)$ a node u with the highest degree that creates a path (x, u, y).

2. For every dominator node y in $N3(x)$, with a lower class number than x, or has the same class number as x but x is closer to the central class than y. Node x chooses two nodes u from $N(x)$ and v from $N2(x)$ that creates the path (x, u, v, y). u, v can be chosen according to the same heuristic above.

Algorithm 4.2 shows our version of constructing a connected dominating set using three hop information.

Algorithm 4.2 3D-LCDS

Input *Dominating set Dom resulting from Algorithm 4.1*
Output: *Return a set of connectors nodes (Conn) that connect every dominator node $x \in Dom$ with some dominator located in $N2(x)$ or $N3(x)$.*
for every dominator node $x \in Dom$ **do**
  let $N(x)$ is the set of one hop neighbors of x, $N2(x)$ is the set of neighbors that are 2 hops away from x and $N3(x)$ is the set of neighbors that are 3 hops a way from x. Let $Deg(x)$ is the number of nodes in $N(x)$.
  for every dominator node $y \in N2(x)$ **do**
    if $cls(x) < cls(y)$ **then**
      $u \leftarrow i \in N(x) : i \in N(y)$ and $Deg(i) < Deg(w)$ for all $w \in N(x)$ and $w \in N(y)$
      $Conn \leftarrow Conn \cup u$
    end if
  end for
  for every dominator node $y \in N3(x)$ **do**
    if $(cls(x) < cls(y))$ or ($cls(x) = cls(y)$ and $dist(x, \text{Central Class}) < dist(y, \text{Central Class})$) **then**
      $(u, v) \leftarrow (i, j) : i \in N(x)$ and $j \in N2(x)$ and $j \in N(i)$ and $j \in N(y)$ and $Deg(i) < Deg(w)$ for all $w \in N(x)$ and $w \in N(j)$ and $Deg(j) < Deg(k)$ for all $k \in N(y)$ and $k \in N(i)$
      $Conn \leftarrow Conn \cup u \cup v$
    end if
  end for
end for

Lemma 4.7. *If UDG is connected, then the set of dominators and connectors constructed by Algorithm 4.1 and Algorithm 4.2 is a connected dominating set.*

Proof. (By contradiction). Assume the set $CDS = Dom \cup Conn$ has at least two distinct components, $C1$ and $C2$. let $C2$ is the closest component to $C1$, Since these two components are connected in the original UDG, then there is a path connects $C1$ and $C2$. Consider the shortest path $m, x_1, x_2, x_3, ..., n$, such that $m \in C1$ and $n \in C2$. By Lemma 4.3, we have the following four cases:

1. If the path length is equal to 4, e.g. the path is m, x_1, x_2, x_3, n, see Figure 25(a), then x_1, x_2 and x_3 are not dominators, other wise we will have a closer component to $C1$ than $C2$. But by Lemma 4.3, every node should be a dominator or a neighbor to a dominator, which makes x_2 dominated by a node from another component, $C3$. Contradiction, $C3$ is closer to $C1$ than $C2$. This case is the same for any path of length greater than 4.

2. If the path length is equal to 3, e.g. the path is m, x_1, x_2, n, see Figure 25(b), then x_1, x_2 are not dominators and they are not dominated by other nodes from outside $C1$ or $C2$, other wise we will have a closer component to $C1$ than $C2$. By Lemma 4.3, we have m and n are both dominators. By Algorithm 4.2, because the hop distance between m and n is 3. Node m or node n will select connectors between them according to their class number. Contradiction.

3. If the path length is equal to 2, e.g. the path is m, x_1, n, see Figure 25(c), then x_1 is not a dominator, other wise we will have a closer component to $C1$ than $C2$. Then by Lemma 4.3, we have the following 4 subcases

   - If m is a dominated node (not dominator) and n is a dominated node, then x_1 should be dominated by another node k from from a third component $C3$ this makes $C3$ closer to $C1$ than $C2$. Contradiction.

- If m is a dominator node and n is a dominated node (not dominator), then n should be dominated by another dominator node k from from $C2$ this makes the distance between node m and k equal to 3. By Algorithm 4.2, node m or node k will select connectors between them according to their class number. Contradiction.
- If n is a dominator node and m is a dominated node (not dominator), then m should be dominated by another dominator node k from from $C1$ this makes the distance between node m and k equal to 3. By Algorithm 4.2, node n or node k will select connectors between them according to their class number. Contradiction.
- If m is a dominator node and n is a dominator node, this makes the distance between node m and n equal to 2. By Algorithm 4.2, node m or node n will select a connector between them according to their class number. Contradiction.

□

Lemma 4.8. *In the worst-case, the number of connectors added by Algorithm 4.2 is* 280 *for every dominator.*

Proof. Let *Dom* be the set of the dominators calculated from Algorithm 4.1. Then from Lemma 4.6, in the worst-case, for any dominator node x the number of dominators that are within 2 units radius sphere centered at x is 125, and the number of dominators within 3 units is 343. This means the number of dominator in $N2(x)$ is at most 125, and the number of dominators in $N3(x) = 343 - 125 = 218$.

If a dominator node x adds one node to connect x with every dominator in $N2(x)$ and 2 nodes to connect x with every dominator in $N3(x)$. Thus, in the worst-case the number of connectors added for each dominator is $125 + (2 * 218) = 561$.

But according to Algorithm 4.2, a dominator node x adds one node to connect x with the dominators in $N2(x)$ that has a lower class numbers and 2 nodes to connect

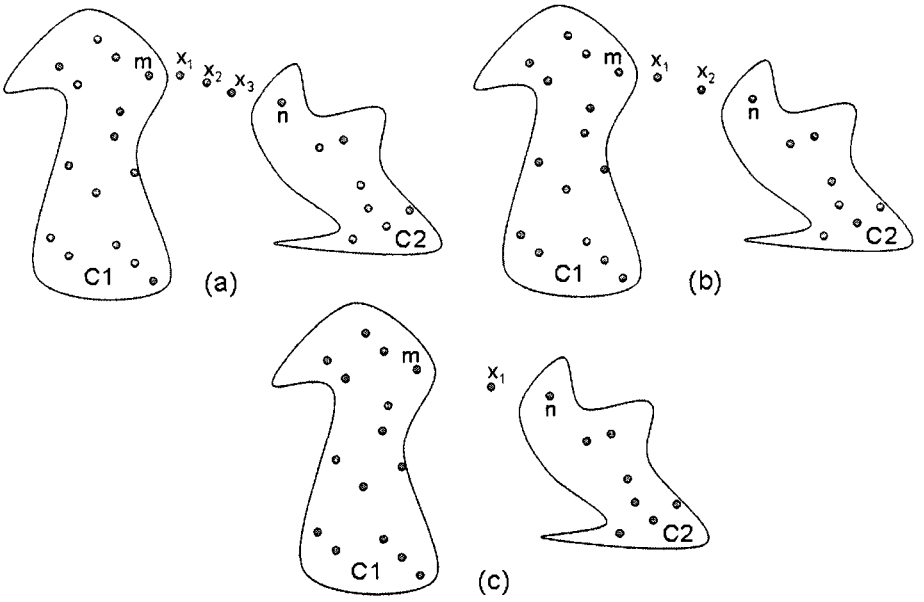


Figure 25: Proof for Lemma 4.7

x with every dominator in $N3(x)$ that has lower class number or with the same class number but with higher distance from the central node. Thus, Half of the connectors that are counted before is added to each dominator, which means the number of connectors are $\lfloor 561/2 \rfloor = 280$

□

It has been proved in [11] that the stretch factor of their connected dominating set is 3. This proof is applicable to our 3D-LCDS that results from Algorithm 4.2.

4.5 Simulations and Results

In order to evaluate the performance of our proposed algorithms, we conducted a simulation study for computing the IDS and CDS. We measured the size of the independent dominating set (IDS) generated by applying Algorithm 4.1 on each node in the network and the average number of connectors generated from Algorithm 4.2. Then we compared them with the global Greedy algorithm [34], and the Alzoubi algorithm in [11, 12, 13].

In the simulation experiments, we run the algorithms on a randomized network environment, where a UDGs with 500 random nodes generated in a box of side length 100. The transmission range r of the nodes is set to $5, 10, 15$, or 20. In this way, we can control the density of the generated graphs, since the density of the generated graphs increases as r increases. To compute the average number of dominators and the average number of connectors, we run the algorithms on 25 graphs for each transmission range.

In Figure 26, we show the number of nodes in the IDS compared with the dominating set resulted from global Greedy and Alzoubi algorithm [12] in a randomized network environment. As expected, the global algorithm has the lowest average number of nodes in IDS. But surprisingly, our algorithm gave almost the same results as the Alzoubi algorithm [12] keeping in mind that our algorithm runs in a constant time compared with the linear time Alzoubi algorithm. Figure 27 shows the number of connectors required to connect the dominating sets from Figure 26. Our algorithm has a lower number of connectors than the Alzoubi algorithm [12].

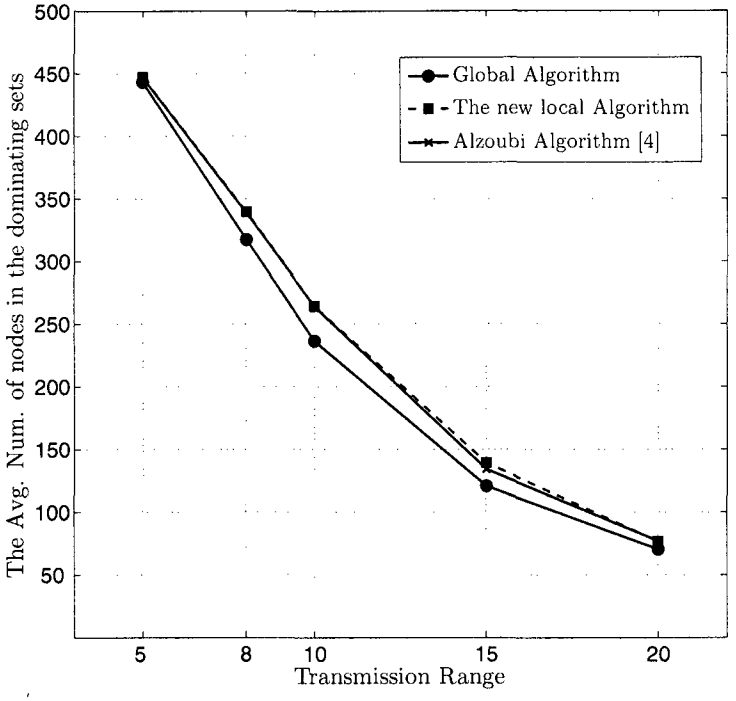


Figure 26: The average number of nodes in the Dominating Set in random environment

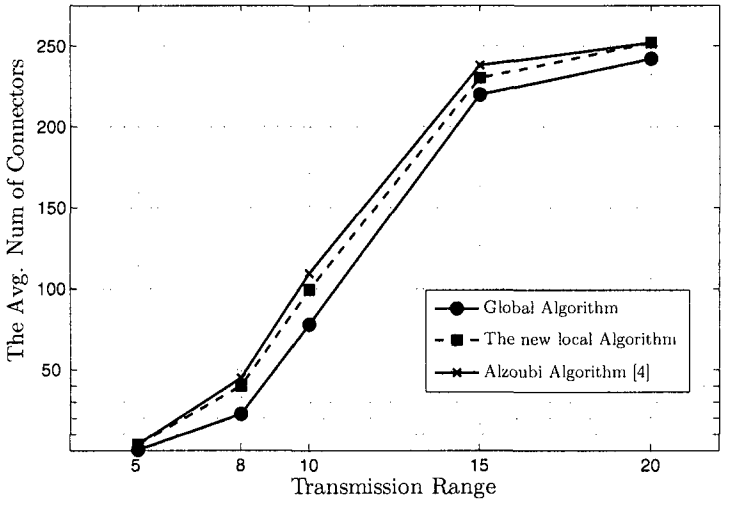


Figure 27: The average number of connectors in random environment

Chapter 5

Hybrid One-neighbor Forwarding Position Based Routing with Partial Flooding

5.1 Introduction

Deterministic progress-based routing algorithms like Greedy, Compass and MFR, in $2D$ or $3D$ UDGs, have very low delivery rates if the network is sparse (the average node degree is small). In this chapter four groups of algorithms are proposed. AB algorithms increase the delivery rate for $2D$ but if we extend them to $3D$, it is not obvious what is the best way to choose the candidate neighbors because there is no way to determine if a node is above or below the line passing through the source and destination nodes in $3D$. Therefore, the first group is an extension of AB algorithm from $2D$ to $3D$ called AB3D. The second, third and the fourth groups are combinations of AB3D and other progress-based routing algorithms from one side and LAR3D algorithms from the other side. In the routing process we assume that the current node is c, the source node is s and the destination node is d.

The $3D$ extension of the forwarding zone of LAR is defined as a rectangular box

Q with the two opposite corners s and $s + (1 + \sqrt{3}|r|/|d - s|) * (d - s)$ (the minimum size rectangular box enclosing node s and the sphere of radius r around d). In the following descriptions of the new routing algorithms, any progress-based algorithm mentioned, such as Greedy, Compass and MFR, will refer to the $3D$ version of the algorithm.

To help motivate the development of these new routing algorithms in the next sections, we will mention for initial comparison purposes the delivery rates (and in some cases the routing traffic) determined by representative simulations for unit disk graphs of 75 nodes, with transmission ranges of 25 units, randomly distributed in a cube with sides of length 100 units.

5.2 Group 1: *AB3D*(*m*,R,S)

In the following definitions, as in the $2D$ AB algorithm definition the symbol R will be used to represent one choice of several possible progress-based routing algorithms, that is, R is one of CM (as in Compass), GR (Greedy) or MFR. Similarly, the symbol S in the naming of the routing algorithms will be used to represent the probability weighting when randomly choosing between more than one candidate neighbors, where S is one of U, A, or D. Suppose that there are m possible candidate neighbors (a subset of all neighbors) to choose from, $n_1, \ldots, n_m$. If the symbol S is U, then the next node x is chosen uniformly at random from n_i. If the symbol S is A or D, then the next node x is chosen from n_i with probability $(1 - p_i)/\sum_{k=1}^{m} p_k$, where $p_i = \theta_i = \angle n_i cd$ if $S = A$, or $p_i = dist(n_i, d)$ if $S = D$. If any of the nodes n_i is not defined, then the algorithm uses the remaining available nodes.

Assume that n_1 is the closest point to d from $N(c)$ (n_1 could be chosen according to a Compass-based measure, but in the simulations this does not lead to better results). Define the plane Pln_1 that passes through c, d, and n_1. Define the second plane Pln_2 that is perpendicular to Pln_1 such that the intersection line between the

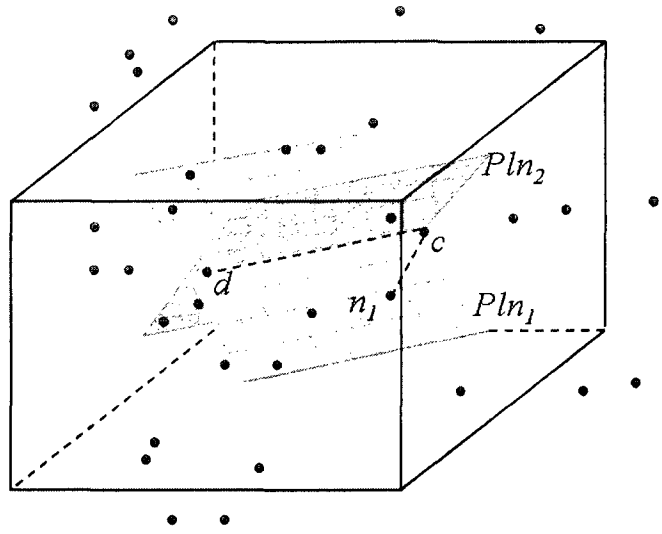


Figure 28: Plane Pln_1 passes through c, d and n_1, plane Pln_2 perpendicular to Pln_1 and both planes contain the line cd

two planes is the line cd. See Figure 28. Each algorithm has three attributes, which is reflected in our naming convention: AB3D(m,R,S) where m is 3 or 5 which represents the number of candidate neighbors, R and S are defined as above. When $m = 3$, the two candidate neighbors in addition to n_1 are chosen as follows. One neighbor n_2 of c is chosen from the half-space above the plane Pln_1 according to the R protocol. Similarly, one neighbor n_3 of c is chosen from the half-space below the plane Pln_1 according to the R protocol. If m is 5, in addition to n_1 c uses the protocol R to choose from $N(c)$ four neighbors n_2, n_3, n_4, n_5 each one in one side of the four regions that result for the intersection between Pln_1 and Pln_2. Once the set of candidate neighbors are determined, c forwards the packet to one of those candidates as chosen by the probability weighting determined by the symbol for S. Algorithm 5.1 gives pseudocode for this procedure for choosing the next node in AB3D(3,R,S). Using the representative simulation results for 75 nodes mentioned near the end of Section 5.1, we find that this set of algorithms increases the delivery rate to around 80% compared to about 63% for the $3D$ versions of the regular progress-based routing algorithms. See Section 5.6 for detailed simulation comparison results.

Algorithm 5.1 One step of AB3D(3,R,S)

Input *position of the current node c, the destination node d and* $N(c)$
Output: *a node from* $N(c)$ *called Next, which represents the next step of the packet.*
(1) $n_1 \leftarrow gdy(c, N(c), d)$;
(2) c computes the plane Pln_1;
(3) c computes the list *above* which contains all the nodes above Pln_1 from $N(c)$, and the list *below* which contains all the nodes from $N(c)$ below Pln_1.
if (R=GR) **then**
  $n_2 \leftarrow gdy(c, \text{above}, d)$
  $n_3 \leftarrow gdy(c, \text{below}, d)$
else if (R=CM) **then**
  $n_2 \leftarrow cmp(c, \text{above}, d)$
  $n_3 \leftarrow cmp(c, \text{below}, d)$
else
  $n_2 \leftarrow MFR(c, \text{above}, d)$
  $n_3 \leftarrow MFR(c, \text{below}, d)$
end if
if (S=U) **then**
  Next←Choose uniformly at random one of n_1, n_2, n_3
else if (S=D) **then**
  $sum\_p \leftarrow 0$;
  for $i \leftarrow 1$ to 3 **do**
    $p_i \leftarrow dist(n_i, d)$
    $sum\_p \leftarrow sum_p + p_i$
  end for
  Next←Choose one of n_i with probability equal $(1 - p_i)/sum\_p$;
else if (S=A) **then**
  $sum\_p \leftarrow 0$;
  for $i \leftarrow 1$ to 3 **do**
    $p_i \leftarrow \theta_i \leftarrow \angle n_i cd$
    $sum\_p \leftarrow sum_p + p_i$
  end for
  Next←Choose one of n_i with probability equal $(1 - p_i)/sum\_p$;
end if

5.3 Group 2: *ABLAR*(m,R)

All the algorithms in ABLAR use the same space partition at the current node c as in the AB3D algorithms. See Figure 28. Similar to AB3D, each algorithm has two attributes, which is reflected in our naming convention: ABLAR(m,R) where m is 3 or 5 which represents the number of candidate neighbors, R is one of CM (as in Compass), GR (Greedy), or MFR, as before. Also each algorithm defines n_1, Pln_1 and Pln_2 as for the AB3D algorithms, if needed. We show two examples of this class of algorithms as follows:

ABLAR(3,CM): the current node c uses the Compass algorithm to choose from $N(c)$ one node n_2 above the plane Pln_1 and another node n_3 below Pln_1. Then c forwards the packet to all the n_i that lie inside the box Q. If any of the those candidate nodes get the message for the second time, the node just ignores it and does not forward it. Figure 29 shows an example of the forwarding process.

ABLAR(5,GR): c uses the Greedy algorithm to choose four neighbors n_2, n_3, n_4 and n_5 from $N(c)$, each one in each of the four regions that result for the intersection between Pln_1, Pln_2. And then c forwards the packet to all those selected neighbors n_i that are inside the box Q. Using the example simulation results for 75 nodes, ABLAR(5,GR) has a delivery rate of more than 99% but with traffic of around 11.

5.4 Group 3: *T-ABLAR*(m,R)-*T*

Although the delivery rate has been increased in the previously mentioned algorithms, the traffic that is caused by these algorithms is relatively high, see Section 5.6. In the following set of algorithms we try to decrease this traffic by utilizing a progress-based routing algorithm such as Greedy, Compass or MFR (the type is determined according to T), for as long as the packet can make progress towards the destination. When it reaches a local minimum, it changes to ABLAR(m,R) for one step and then progress-based routing is resumed. Note that during the ABLAR(m,R) step,

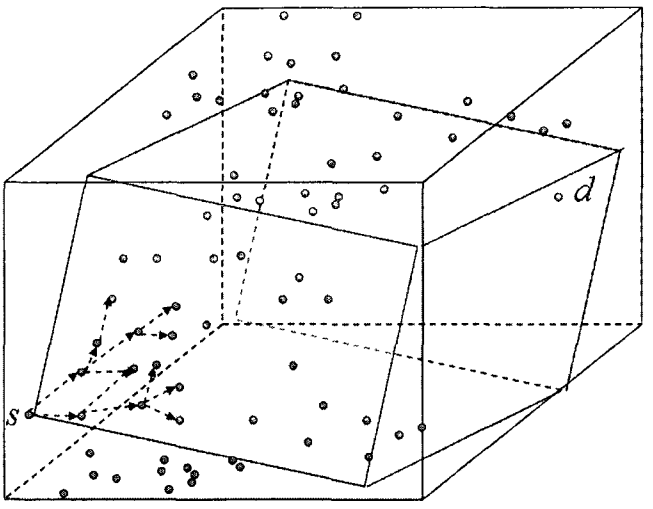


Figure 29: With ABLAR(3,R), the current node chooses up to 3 neighbors and forwards the packet to all of those inside the flooding area Q (the shaded box)

if a receiving neighboring node has seen the packet before, it will just drop it. See Figure 30 for an example of the execution of an algorithm of this group. Algorithm 5.2 gives pseudocode outlining how MFR-ABLAR(3,GR)-MFR works. Using the example simulation results for 75 nodes, the delivery rate for MFR-ABLAR(5,GR)-MFR is around 88% and the traffic is around 4.3.

5.5 Group 4: *AB3D-ABLAR*(m,R)

The previous group of algorithms decreased the traffic of ABLAR, but unfortunately it does not achieve a similar delivery rate, see Section 5.6. The fourth group AB3D-ABLAR has both the advantages of all the algorithms above: very high delivery rate while at the same time low traffic. An algorithm in this group starts with any one of the nine distinct AB3D algorithms. Once a threshold is passed in terms of the number of hops, the algorithm permanently switches to ABLAR and it does not go back again to AB3D. How the threshold value is chosen will be discussed in Section 5.6 where the experimental results are presented. Figure 31 gives an example of the path followed by the algorithm. When the packet starts at the node s, AB3D used

Algorithm 5.2 MFR-ABLAR(3,GR)-MFR algorithm

Input *Source node s, the destination node d.*
Output: *Return success if the destination is reached.*
$c \leftarrow s$
while (true) **do**
  while (!(reach d) or !(Local Minimum)) **do**
    $c \leftarrow MFR(c, N(c), d)$
  end while
  if (the packet arrives at d) **then**
    return success
  end if
  if (Local Minimum) **then**
    Use ABLAR(3,GR) to choose 3 nodes n_1, n_2, and n_3.
    for $i \leftarrow 1$ to 3 **do**
      MFR-ABLAR(3,GR)-MFR(n_i,d)
    end for
  end if
end while

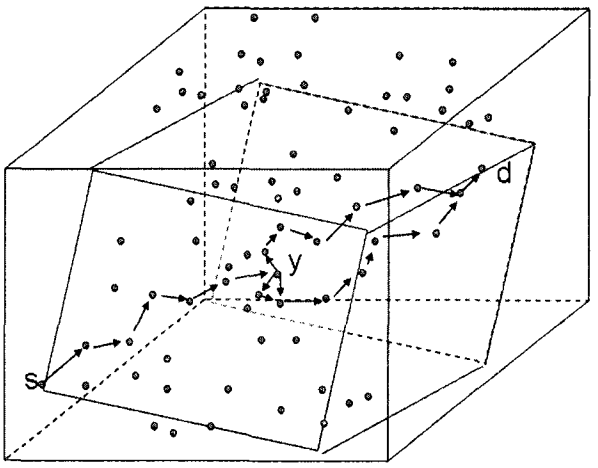


Figure 30: The packet started at s using progress-based routing like Compass until it reaches the local minimum y. ABLAR is used for one step and then progress-based routing is resumed

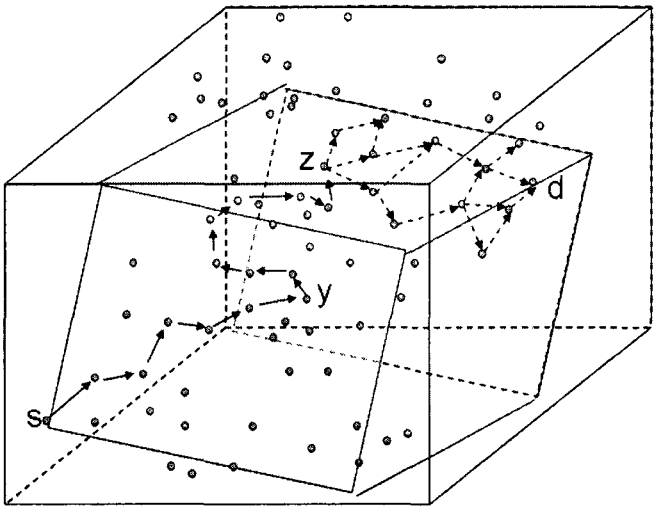


Figure 31: AB3D-ABLAR algorithm example.

until the packet reaches y, the local minimum node. The randomized character of AB3D allows the algorithm to pass y. AB3D continues until the threshold is reached at the node z, at which point the algorithm switches to ABLAR.

There are three differences between this group of algorithms and T-ABLAR-T: (i) T-ABLAR-T is a deterministic algorithm while AB3D-ABLAR is a randomized algorithm. (ii) In T-ABLAR-T the algorithm switches to ABLAR if the local minimum is reached, while in AB3D-ABLAR it switches if a threshold is reached. The reason for using a threshold is based on the algorithm proposed in [106], where although the packet reaches a local minimum it can still be forwarded to the node with the least backward (negative) progress to the destination. (iii) In T-ABLAR-T, the algorithm uses ABLAR for just one step, then it goes back to T, while AB3D-ABLAR algorithm just keeps using ABLAR.

5.6 Simulations and Results

In this section we describe the simulation environment, demonstrate and interpret the results, and compare the new algorithms with other deterministic and randomized routing algorithms.

5.6.1 Simulation Environment

In the simulation experiments, a set V of n points (where $n \in \{65, 75, 85, 95\}$) is randomly generated in a cube with sides of length 100 units. The maximum transmission radius of each host is set to 25 units. We first calculate the UDG for V. If the graph is connected, it is used in the simulation, otherwise it is discarded. We set the threshold to n (this threshold is used for the randomized AB3D(m,R,S) and AB3D-ABLAR(m,R) algorithms). The source and the destination nodes are then randomly picked from the generated graph. It is suggested in [75] to consider simulations with node density per unit disk of around 5 in $2D$ environment, which would correspond to the graph with an average node degree of around 4. Figure 32 illustrates a histogram of the node degrees for the graphs with the chosen simulation values n. Graphs with $n = 65$ are closest to the node density of interest while graphs with fewer nodes typically have very tree-like structure. When n is larger than 95, a substantial percentage of nodes have degrees larger than 6 which indicates highly connected graphs.

An algorithm succeeds if a path to the destination is found. To compute the packet delivery rate, this process is repeated with 100 random connected graphs and the percentage of successful deliveries determined. To compute the average packet delivery rate, the packet delivery rate is determined 100 times and an average taken. Additionally, out of the $10,000$ runs used to compute the average packet delivery rate an average of overall traffic is computed. Since there are more than 50 different combinations of the algorithms, it's difficult to show all of these combinations, thus we select some of the algorithms which gave the most interesting results. We provide several different analyzes: a comparison of the algorithms on UDGs with 75 nodes, a comparison of the effect of the number of nodes on algorithm performance, a study of the effect of the threshold on the randomized algorithms, and a comparison of the effect of the number of candidate neighbors in associated algorithms across UDGs with different number of nodes. In each analysis we study the delivery rate and traffic versus the node density, and the threshold.

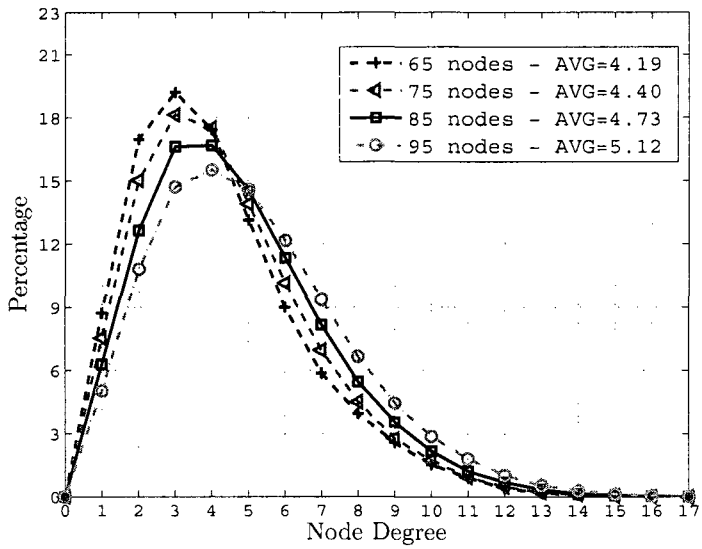


Figure 32: Histogram of the average node degrees of the 10,000 generated unit disk graphs

5.6.2 Results

We present a comparison between the different groups of algorithms in terms of packet delivery rate and overall traffic in Table 3. For comparison purposes, just two algorithms from each group with $n = 75$ are presented. It is immediately evident from the result given in Table 3 that deterministic progress-based algorithms (Greedy, Compass and MFR) have the lowest delivery rates (less than 65%) which yields the low traffic because the packets that fail to arrive to the destination are not counted in the traffic. The first group of randomized algorithms AB3D comes after that with a delivery rate over 79% and average traffic around 3.5. The delivery rate of the second group of ABLAR algorithms, with $m = 5$, reaches to 99% but these algorithms have the second worst average traffic (around 11.5 for $n = 75$) after LAR3D with traffic around 13. The third group T-ABLAR-T reaches a 88% delivery rate with traffic around 4. The fourth group AB3D-ABLAR decreases the average traffic compared to LAR3D by more than 55% while still reaching a 99% delivery rate. All our results have a 95% confidence intervals.

Table 3: Average packet delivery rate and average traffic for selected algorithms in UDG with $n = 75$.

	Algorithms	Delivery rate	Average traffic
	COMPASS	63.43	*1.04*
	GREEDY	62.60	*1.02*
	MFR	62.78	*1.03*
Group 1	AB3D(3,MF,D)	81.88	*3.04*
	AB3D(3,CM,A)	79.16	*3.40*
Group 2	ABLAR(5,CM)	99.37	*11.43*
	ABLAR(5,MF)	99.36	*11.48*
Group 3	MF-ABLAR(5,GR)-MF	88.61	*4.30*
	GR-ABLAR(5,GR)-GR	87.49	*3.20*
Group 4	AB3D(3,CM,A)-ABLAR(5,GR)	99.18	*6.97*
	AB3D(3,GR,D)-ABLAR(5,GR)	99.15	*6.24*
	LAR3D	99.62	*13.84*

Effect of the network density. Figure 33 and Figure 34 illustrate the effect of the number of nodes (network density) on the performance of the algorithms. In all the algorithms, as the number of nodes increases, the delivery rate also increases. For groups 1 and 4, this can be explained by the randomization process for the algorithms. Increasing the number of nodes means there is a greater chance for a good route to the destination. For groups 2, 3, and 4 increasing the number of nodes gives a better chance of the current node to find a candidate neighbor for the partial flooding process to succeed, which gives a better chance to reach the destination. Figure 34 shows how the traffic is affected by the network density. Because increasing the number of nodes implies increasing the possibility of long detours being discovered during randomized routing, the traffic is increased in AB3D. For groups 2, 3 and 4 there is an increase in the number of flooded nodes, which increases the traffic. All the conclusions from Table 3 apply for any node density. Therefore, the fourth group of algorithms continues to have 55% of the traffic as used by LAR3D while at the same time obtaining a nearly the perfect delivery rate.

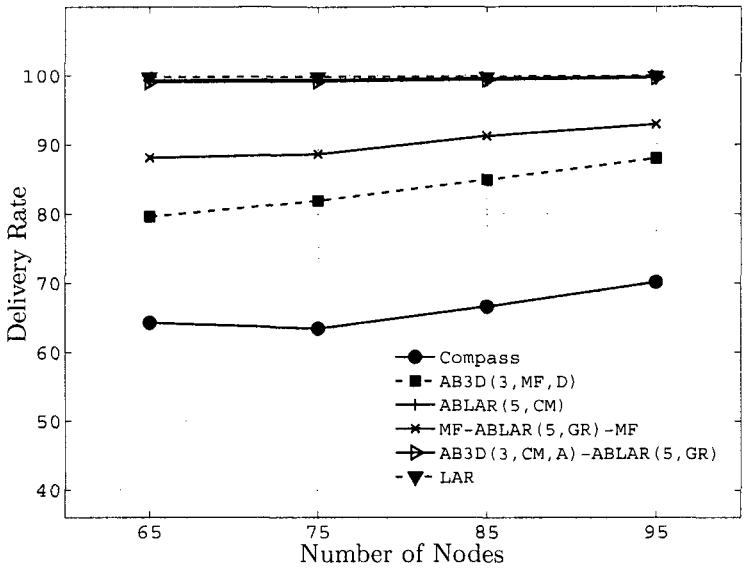


Figure 33: The packet delivery rate at different node densities (box side = 100 units, maximum transmission range = 25 units, threshold = n)

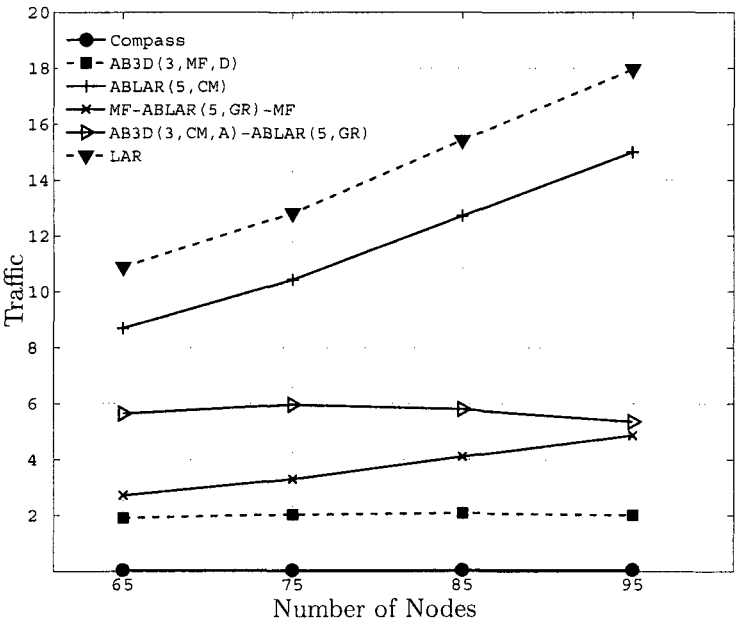


Figure 34: The average traffic at different node densities (box side = 100 units, maximum transmission range = 25 units, threshold = n)

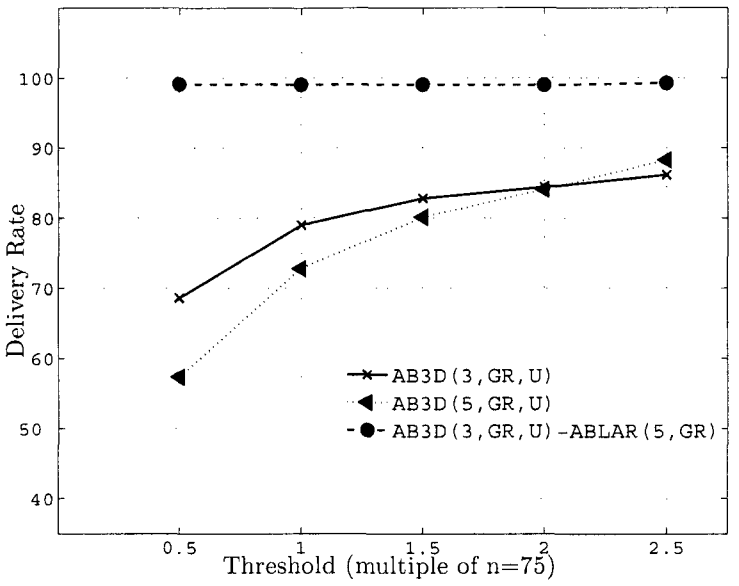


Figure 35: The packet delivery rate at different thresholds (box side = 100 units, maximum transmission range = 25 units)

Effect of the threshold. Figure 35 and Figure 36 show the effect of varying the threshold value on the average delivery rate and average traffic of some randomized algorithms from groups 1 and 4. We find that when the threshold is set to n the relative behavior of the algorithms is established and the difference between algorithms is clear. The delivery rate of all algorithms increases when increasing the threshold, and this is very clear with the AB3D algorithms, with 5 candidate neighbors. Since the increase of the delivery rate means more successfully delivered packets added to the average traffic, then the average traffic would be expected to increase. The simulation results in Figure 36 confirm this expectation, with an increase in average traffic corresponding to the increase in threshold.

Effect of the number of candidate neighbors, m. Figure 37 and Figure 38 depict the effect of the candidate neighbors m in the new algorithms with different graphs densities. We find AB3D(3,R,S) algorithms have a higher delivery rate than AB3D(5,R,S). This is because, with a larger selection of candidate neighbors from

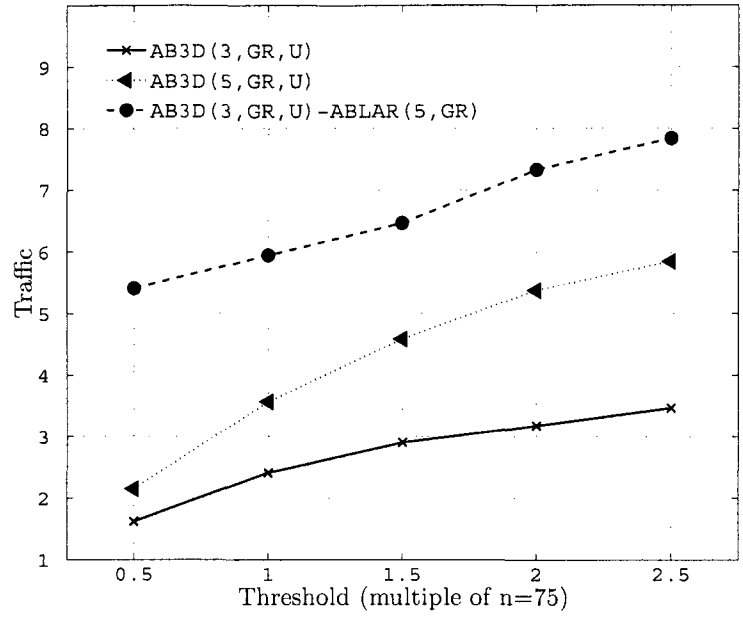


Figure 36: The average traffic at different thresholds (box side = 100 units, maximum transmission range = 25 units)

which only one is chosen randomly, there is a greater chance of the path being detoured and not delivering the packet before the threshold of n is passed. For ABLAR algorithms (and those hybrid algorithms using the ABLAR algorithms), $m = 5$ is found to give higher delivery rates with the trade-off of higher traffic. By flooding with more neighbors, there is a greater chance of discovering a successful path with the price of more traffic. The values of m for all the algorithms that give the higher delivery rates have been used for the other comparisons of the algorithms.

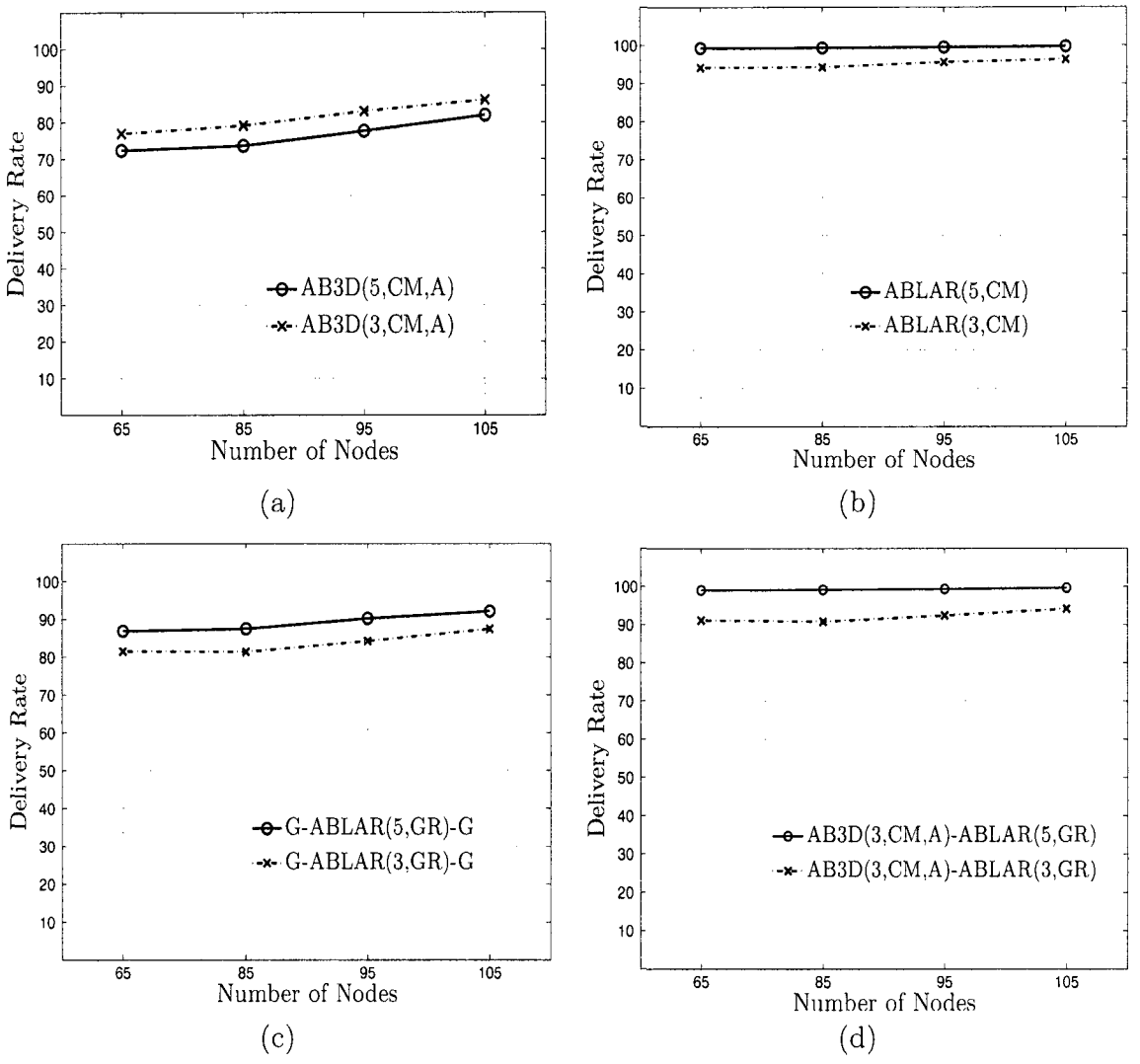


Figure 37: The average packet delivery rate with $m = 3$ and $m = 5$ for selected algorithms (box side = 100 units, maximum transmission range = 25 units)

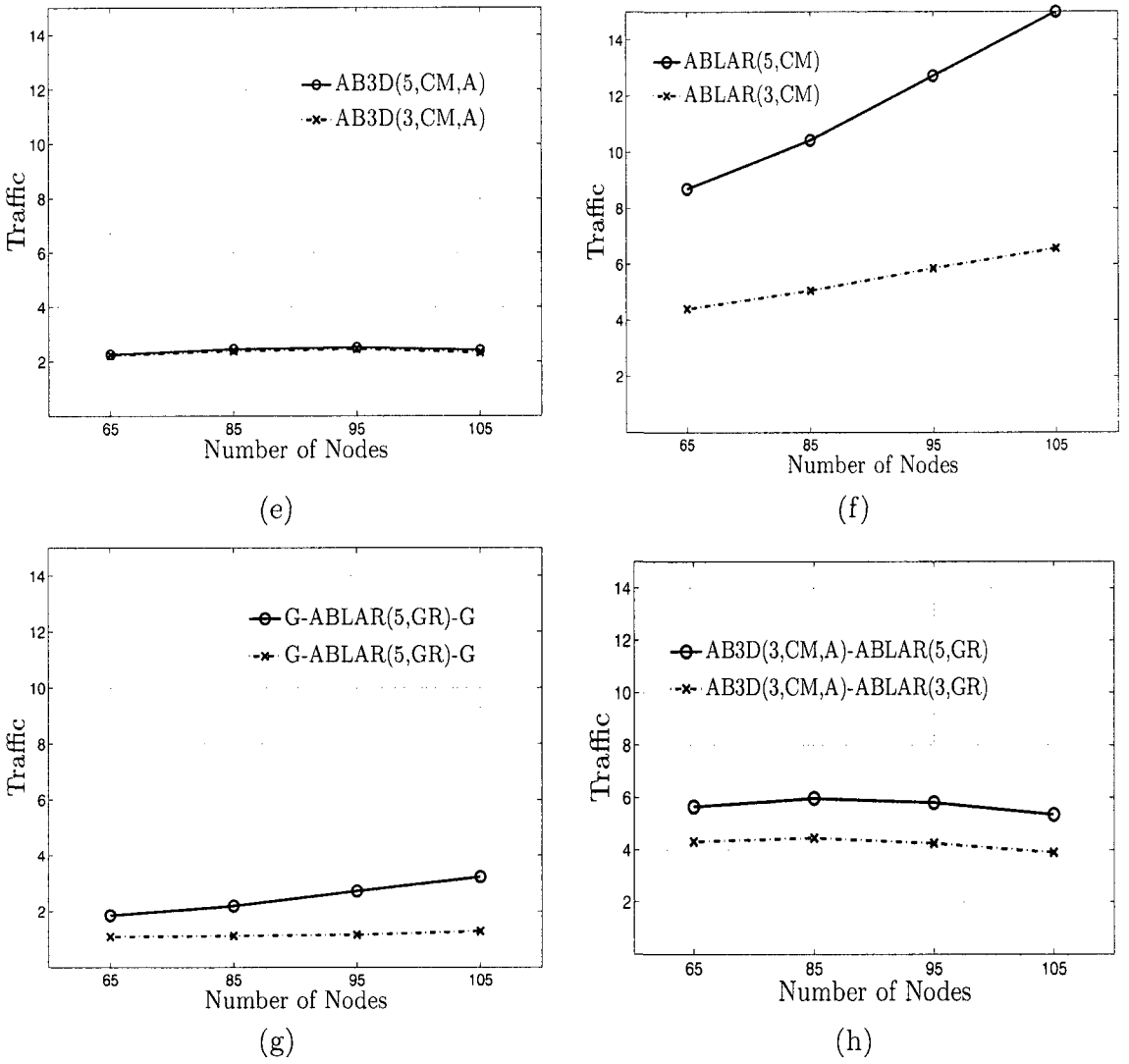


Figure 38: The average traffic with $m = 3$ and $m = 5$ for selected algorithms (box side = 100 units, maximum transmission range = 25 units, threshold = n)

Chapter 6

Randomized Position-based Routing Algorithms

In the previous chapter we showed a combination between some deterministic progress-based routing algorithms with partial flooding algorithms LAR3D. To avoid the flooding, in this chapter we show two new groups of 3D-position-based routing algorithms which combine randomized $AB3D(3, R, S)$ routing algorithms with CFace routing [2].

6.1 AB3D(3,R,S)-CFace(1)-AB3D(3,R,S)

This group of hybrid algorithms starts with any one of the nine distinct AB3D(3,R,S) algorithms. Once a local threshold is passed in terms of the number of hops and we arrive at a local minimum, the algorithm switches to CFace(1) starting from the local minimum c as the new source node. If the destination is not reached during CFace(1) and looping occurs, the algorithm goes back to AB3D(3,R,S) and the count for the local threshold restarts at 0. In this algorithm, the reason for using a local threshold is based on the algorithm in [106], where although the packet reaches a local minimum it can still be forwarded to the node with the least backward (negative) progress.

In addition to the local threshold, the algorithm uses a global threshold to drop

the packet if the total number of hops exceeds the global threshold. AB3D(3,R,S)-CFace(1)-AB3D(3,R,S) is given in Algorithm 6.1.

Algorithm 6.1 AB3D(3,R,S)-CFace(1)-AB3D(3,R,S)

1: ***Input*** *source node s, the destination node d local threshold LTH and global threshold GTH.*
2: ***Output***: *return True if the destination is reached or False other wise.*
3: **repeat**
4: $LPath\_Length \leftarrow 1$
5: Call $AB3D(3, R, S)$ routing
6: **if** the packet arrive **then**
7: return (True)
8: **end if**
9: **if** (local min) and ($LPath\_Length$ >LTH) **then**
10: Randomly choose one of the planes xy, yz or xz
11: Project all nodes on the selected plane
12: Call Face routing starting from current node c
13: During face routing
14: **if** the packet reach the destination **then**
15: return (True)
16: **else if** loop happens **then**
17: go to 3 with the current node c as new source
18: **else**
19: Face routing continues.
20: **end if**
21: **end if**
22: **until** GTH is reached
23: return (False)

6.2 AB3D(3,R,S)-CFace(3)

The main difference between this set of algorithms and the *AB3D(3,R,S)-CFace(1)-AB3D(3,R,S)* algorithms is that instead of going back to AB3D(3,R,S) if the first projective plane fails, these try other projective planes. The algorithms also start with AB3D(3,R,S). If a threshold is reached together with a local minimum, the algorithm switches to CFace(1) using the xy plane. Again if a loop happens the algorithm goes to yz plane. Finally, if yz plane fails, the algorithm switches to xz

plane. AB3D(3,R,S)-CFace(3) is summarized in Algorithm 6.2.

The key advantage of these hybrid algorithms is the improvement in performance over randomized AB3D(3,R,S) algorithms and CFace(3) algorithm, with a decrease of the large path dilation caused by CFace(3) routing algorithm. In the next section, we show the simulation results which illustrate the advantages of our algorithms.

Algorithm 6.2 AB3D(3,R,S)-CFace(3)

1: ***Input*** *source node s, the destination node d and local threshold LTH.*
2: ***Output***: *return True if the destination is reached or False other wise.*
3: Call $AB3D(3, R, S)$ routing
4: **if** the packet arrive **then**
5: return (True)
6: **end if**
7: **if** (local min) and (path length>LTH) **then**
8: $w \leftarrow c$
9: $Counter \leftarrow 1$
10: **while** ($counter < 4$) **do**
11: **switch**(counter)
12: **case** 1: $CurrentPlane \leftarrow xy$ plane
13: **case** 2: $CurrentPlane \leftarrow yz$ plane
14: **case** 3: $CurrentPlane \leftarrow xz$ plane
15: **endswitch**
16: Project all nodes on the selected plane
17: Call Face routing starting from current node c
18: During face routing
19: **if** the packet reach the destination **then**
20: return (True)
21: **else if** loop happens **then**
22: $Counter \leftarrow counter + 1$
23: go to 10 with the current node c as new source
24: **else**
25: Face routing continues.
26: **end if**
27: **end while**
28: **end if**
29: return (False)

6.3 Simulations and Results

In this section we describe our simulation environment, and then we show and interpret our results, comparing our algorithms with deterministic routing algorithms Greedy, Compass, Ellipsoid and CFace(3).

6.3.1 Simulation Environment

In the simulation experiments, a set V of n points (where $n \in \{65, 75, 95, 105\}$) is randomly generated in a cube of side length 100. The maximum transmission radius of each host is set to 25. We set the global_threshold to $2n$ and the local_threshold to n. We first calculate all components in the graph. Then we select the largest component (LC) among all the components to perform the routing algorithms. The source and destination nodes are then randomly picked from LC.

Average packet delivery rate and path dilation are computed as in Section 5.6.1. Since we have more than eighteen different combinations of the algorithms, it is difficult to show all of these combinations, thus we show the some algorithms which gave the most interesting results. We provide three separate analyzes. In these analyzes we study the delivery rate and path dilation versus the node density, the threshold and the subgraph type. In all graphs we just show the best algorithm in each proposed class.

6.3.2 Results

We present the comparison between different groups of algorithms in terms of packet delivery rate and path dilation in Tables 4, 5 and 6. For comparison purposes, we will focus on $n = 75$, and $n = 95$. It is immediately evident form the result given in Table 4 that the delivery rate of CFace(3) jumps to 94%, but this algorithm has by far the worst path dilation (around 10 for $n = 75$). Our new algorithm *AB3D(3,R,S)-CFace(1)-AB3D(3,R,S)* almost reaches the delivery rate of *CFace(3)*,

Table 4: Average packet delivery rate, DR, and average path dilation, PD, in UDG.

	$n = 75$		$n = 95$	
Algorithms	DR	PD	DR	PD
COMPASS	63.60	1.03	65.74	1.05
GREEDY	62.56	1.02	64.22	1.03
CFACE(3)	94.53	9.87	95.56	14.59
AB3D(3,CM,D)	80.37	3.02	85.68	3.22
AB3D(3,CM,A)	76.91	3.41	84.25	3.70
AB3D(3,GR,D)	79.90	2.99	84.94	3.20
AB3D(3,GR,A)	76.91	3.35	83.68	3.55
AB3D(3,CM,D):CFACE(3)	97.76	6.25	98.04	6.91
AB3D(3,CM,A):CFACE(3)	97.49	7.33	97.85	7.58
AB3D(3,GR,D):CFACE(3)	97.54	6.37	97.92	6.98
AB3D(3,GR,A):CFACE(3)	97.28	7.27	98.04	7.82
AB3D(3,CM,D):CFACE(1):AB3D(3,CM,D)	92.90	4.97	94.53	5.35
AB3D(3,CM,A):CFACE(1):AB3D(3,CM,A)	92.61	5.84	94.59	6.01
AB3D(3,GR,D):CFACE(1):AB3D(3,GR,D)	92.71	5.04	94.15	5.46
AB3D(3,GR,A):CFACE(1):AB3D(3,GR,A)	92.13	5.89	94.03	6.09

but it decreases the path dilation by 50%. The best delivery rate with over 97% is found in *AB3D(3,R,S)-CFace(3)* and also has a lower path dilation than *CFace(3)* algorithm. We find, from Tables 4 to 6, that the algorithms based on AB3D(3,C,D) and AB3D(3,G,D) have the best performance in terms of delivery rate and path dilation.

Effect of using a subgraph of UDG for routing. In Figure 39 we can see the influence of the subgraph on the delivery rate. In Figure 40 we show the influence of the subgraph over the path dilation. First, in terms of delivery rate, as expected, the deterministic and randomized AB3D(3,R,S) algorithms delivery rate decreased over both GG and RNG graphs, due to potentially fewer neighbors available in the progress direction.

Our new hybrid algorithms have roughly the same best performance on all three graphs. CFace(3) has the best delivery rate on Gabriel subgraphs, followed by RNG and then on UDG. This can be explained also by considering the number of edges;

Table 5: Average packet delivery rate, DR, and average path dilation, PD, in GG.

	$n = 75$		$n = 95$	
Algorithms	DR	PD	DR	PD
COMPASS	60.95	1.02	62.53	1.03
GREEDY	60.43	1.02	62.47	1.03
CFACE(3)	95.03	8.73	94.83	11.58
AB3D(3,CM,D)	79.96	3.60	84.27	3.86
AB3D(3,CM,A)	75.39	4.20	81.93	4.56
AB3D(3,GR,D)	79.60	3.59	84.38	3.86
AB3D(3,GR,A)	75.44	4.18	81.10	4.48
AB3D(3,CM,D):CFACE(3)	97.80	6.51	98.05	6.83
AB3D(3,CM,A):CFACE(3)	97.47	7.93	97.64	8.11
AB3D(3,GR,D):CFACE(3)	97.67	6.51	97.90	6.80
AB3D(3,GR,A):CFACE(3)	97.46	7.84	97.69	8.27
AB3D(3,CM,D):CFACE(1):AB3D(3,CM,D)	93.70	5.56	94.55	5.74
AB3D(3,CM,A):CFACE(1):AB3D(3,CM,A)	92.77	6.90	94.52	6.99
AB3D(3,GR,D):CFACE(1):AB3D(3,GR,D)	93.12	5.53	94.52	5.93
AB3D(3,GR,A):CFACE(1):AB3D(3,GR,A)	92.39	6.85	94.16	6.90

fewer edges implies fewer crossing edges in the projected face which means less chance for the packet to enter a loop. In terms of the path dilation, the algorithms depend on AB3D have the best path dilation over UDG graph, but in both Gabriel subgraph and relative neighborhood graph the path dilation is increased for the same reason mentioned above. On these subgraphs, CFace(3) algorithm the path dilation has been decreased because there is less chance for the packet to enter a loop, which means no new projective planes.

Effect of the network density. In Figure 41 we illustrate the effect of the number of nodes (network density) on the performance of the algorithms. In all the algorithms, as the number of nodes increased the delivery rate also increased. This is because most of the proposed algorithms depend on the randomized AB3D which means there is a better chance for a good route to the destination. When n is equal to 65 the delivery rate does not follow the above trend, because the LC is very small, which implies that the path between any pair of nodes is relatively short. Figure 42

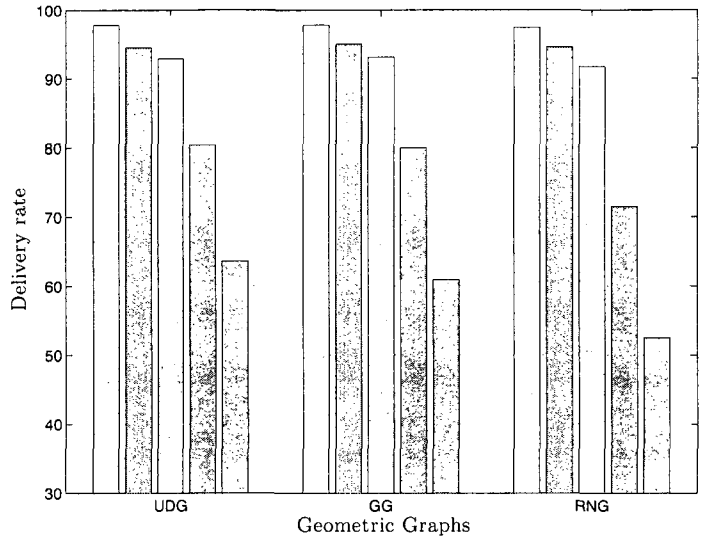


Figure 39: The packet delivery rate for different geometric graphs; see Figure 40 for legend

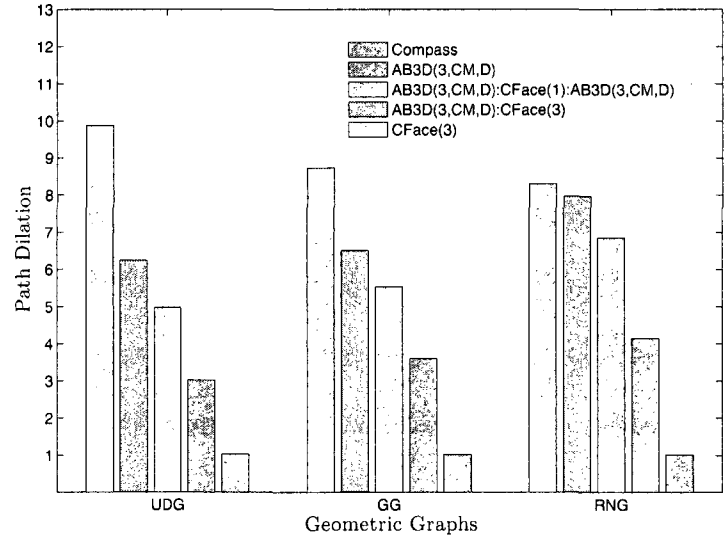


Figure 40: The average path dilation for different geometric graphs

Table 6: Average packet delivery rate, DR, and average path dilation, PD, in RNG.

	$n = 75$		$n = 95$	
Algorithms	DR	PD	DR	PD
COMPASS	52.58	1.02	51.76	1.02
GREEDY	52.81	1.01	51.94	1.02
CFACE(3)	94.57	8.30	95.40	10.85
AB3D(3,CM,D)	71.48	4.14	75.52	4.87
AB3D(3,CM,A)	61.12	4.93	65.07	5.67
AB3D(3,GR,D)	71.68	4.13	75.25	4.84
AB3D(3,GR,A)	61.75	4.96	65.82	5.75
AB3D(3,CM,D):CFACE(3)	97.41	7.96	98.02	9.07
AB3D(3,CM,A):CFACE(3)	97.21	10.26	97.32	12.01
AB3D(3,GR,D):CFACE(3)	97.71	7.89	97.71	8.88
AB3D(3,GR,A):CFACE(3)	97.18	10.29	97.47	11.82
AB3D(3,CM,D):CFACE(1):AB3D(3,CM,D)	91.71	6.85	93.24	7.70
AB3D(3,CM,A):CFACE(1):AB3D(3,CM,A)	90.40	9.01	92.24	10.30
AB3D(3,GR,D):CFACE(1):AB3D(3,GR,D)	91.94	6.89	93.67	7.74
AB3D(3,GR,A):CFACE(1):AB3D(3,GR,A)	90.43	8.90	91.86	10.48

shows how the path dilation is effected by the network density. Because increasing the number of nodes implies increasing the possibility for long detours being discovered during randomized routing, the path dilation is increased. For CFace(3) there is an increase in the number of crossing edges, which means greater chance for entering into loops, therefore increasing the probability of having to project to the second and third plane.

Effect of the threshold. Figure 43 and Figure 44 show the effect of varying the threshold value on the average delivery rate and average path dilation of all studied algorithms. We find that when the threshold is set to n the relative behavior of the algorithms is established and the difference between algorithms is clear. The delivery rate of all algorithms can been increased by increasing the threshold, and this is very clear with the AB3D(3,R,S) algorithm. Since the increasing of the delivery rate means more successfully delivered packets added to the average path dilation, then the average path dilation is expected to increase. The simulation results also

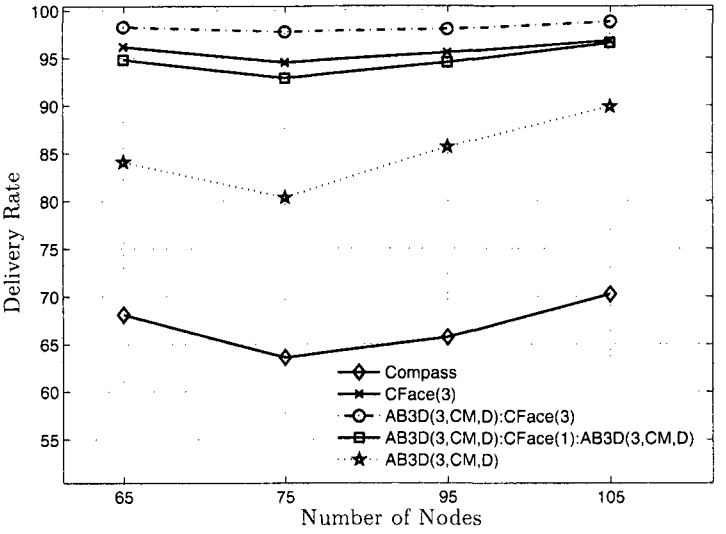


Figure 41: The packet delivery rate at different node densities

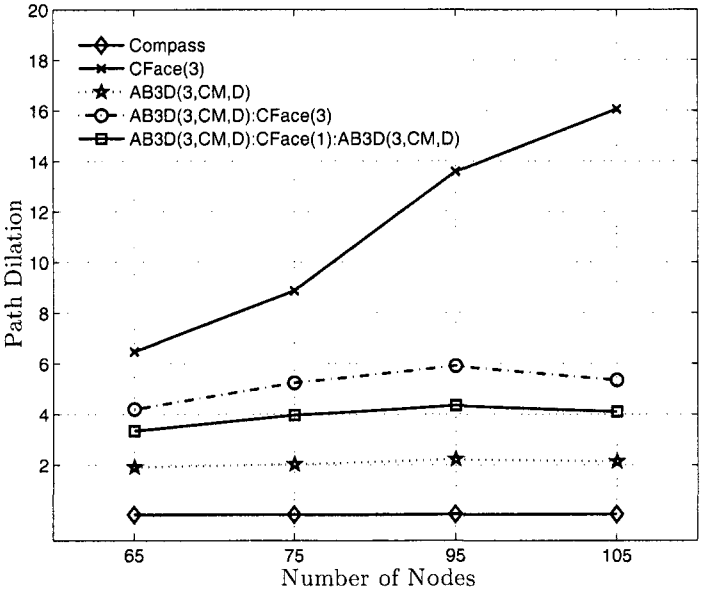


Figure 42: The average path dilation at different node densities

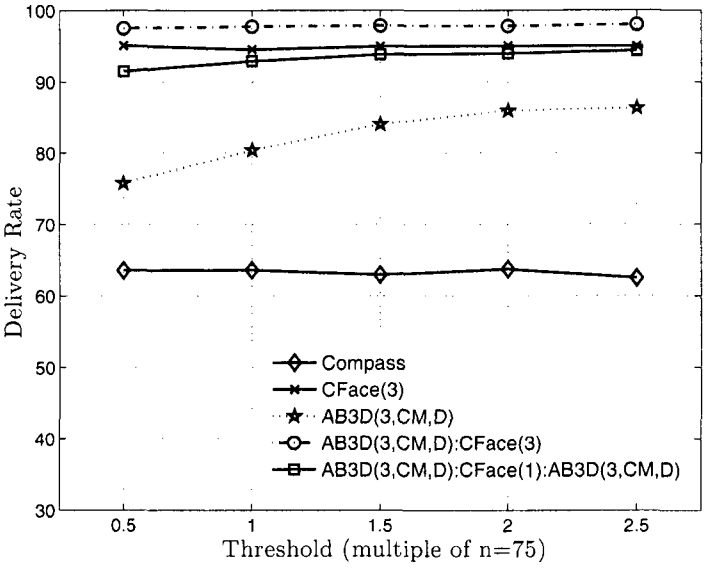


Figure 43: The packet delivery rate at different thresholds

confirm this expectation, with an increase in average path dilation corresponding to an increase in threshold.

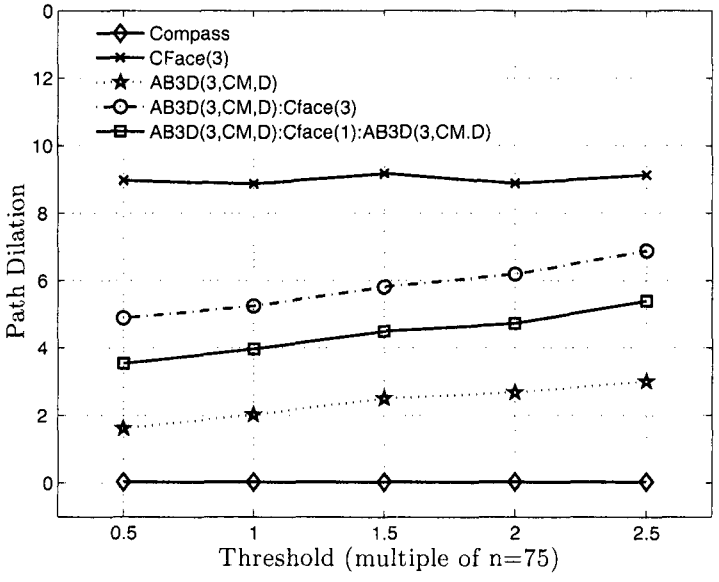


Figure 44: The average path dilation at different thresholds

Chapter 7

Power-aware Position-based Routing Algorithms using Adjustable Transmission Ranges for $3D$ Ad Hoc and Sensor Networks

In this chapter, we propose several new $3D$ position-based, local power-aware routing algorithms for ad hoc wireless network. These algorithms attempt to maximize the average network survivability. In the next Section we give a detailed description of the new routing algorithms. Experimental results to demonstrate the much improved performance of the proposed methods in comparison with existing techniques are presented in Section 7.3.

7.1 New Power-aware Position-based Routing Algorithms

7.1.1 Power-aware Greedy (PAG)

Most of the routing algorithms without power-awareness use a fixed transmission range for all the nodes, so the nodes may waste power by transmitting more than is needed for correct reception. The power-aware algorithms described in Chapter 2 use an adaptive transmission range to transmit the data messages during the routing process, but they still use a fixed (maximum) transmission range for the control messages (periodic hellos) to tell neighboring nodes about their location. Our new power-aware routing protocols are based on adjustments of the node transmission power at two stages: (i) while discovering the neighboring nodes, and (ii) during the routing process.

In PAG, the nodes use the optimal transmission range $((\beta/(\alpha-1))^{1/\alpha})$, if possible, every γ seconds for periodic discovery control messages. Whenever the degree of the node drops below ϱ, it instead uses the maximum transmission range for the periodic discovery control messages η times, and then it goes back to use the optimal transmission range, where η, ϱ and γ are network dependent.

Table 7 gives an example of how the node changes its transmission range of the periodic discovery control messages depending on the node degree. In this example let $\gamma = 3$, $\eta = 4$ and $\varrho = 4$. Since the degree at time 0 is greater than 4, the node uses the optimal transmission range at time 3, which is the next time interval. At time 6 the node finds that the degree is dropped to less than 4 so it uses the maximum transmission range in the next 4 times intervals $9, 12, 15$ and 18, then at time 21 it tests the optimal transmission range again.

Greedy routing is used for routing between the source and destination. If the packet reaches a local minimum at the low transmission level, then the current node

Table 7: example of how the node changes the transmission range of the periodic discovery control messages depending on the node degree, $\gamma = 3$, $\eta = 4$ and $\varrho = 4$

Time	0	1	2	3	4	5	6	7	8	9	10	11	12
Transmission range	O			O			O			M			M
Node degree	5	5	5	6	6	6	3	3	3	7	7	7	6
Time	13	14	15	16	17	18	19	20	21	22	23	24	25
Transmission range			M			M			O			O	
Node degree	6	6	8	8	8	7	7	7	5	5	5	4	4

increases its transmission range to its maximum and runs the neighbor nodes discovery step again. Figure 45 gives an example of this process. If a node does not discover a new neighbor that makes progress to the destination, then the algorithm fails, otherwise Greedy routing continues. The details of the algorithm are given in Algorithm 7.1.

There are two main differences between PAG and the non-power-aware Greedy algorithm. First, during the neighbor discovery phase, in PAG all the nodes exchange periodic hello messages with the optimal transmission range. In Greedy, all the nodes exchange information by transmitting and receiving using the maximum transmission range. Also, during the routing phase (data packet routing process), in PAG the current node c forwards the message with a power cost equal to $\ell h^{\alpha} + \beta$, where h is the distance between c and the next node, while Greedy forwards the message with power cost equal to $\ell r^{\alpha} + \beta$, where r is the maximum transmission range.

7.1.2 Power-aware Greedy-Cost (PAGC)

This algorithm is similar to PAG algorithm, but the main difference is in the Greedy part, it uses the Cost Progress idea from [76] as follows. When the packet arrives at some node c, instead of choosing the closest neighbor to the destination as a second node, PAGC chooses the neighbor that makes progress to the destination and has the maximum remaining energy.

Algorithm 7.1 PAG

Input *source node s, the destination node d, The time between two consecutive neighbors' discovery γ, the number of times of using the maximum transmission range η, and degree threshold ϱ.*
Output: *True if the packet arrives to d, and False otherwise.*
for each node **do**
  if (*Maximum-transmission* range $< (\beta/(\alpha-1))^{1/\alpha}$) **then**
    Optimal-transmission $\leftarrow$ *Maximum-transmission* range.
  else
    Optimal-transmission $\leftarrow (\beta/(\alpha-1))^{1/\alpha}$
  end if
end for
if (the node w with the *Optimal-transmission* range has enough neighbors, greater than ϱ at time t) **then**
  w uses the *Optimal-transmission* range for discovering neighbors in the next time interval after γ seconds (at time $t+\gamma$).
else
  w uses the *Maximum-transmission* range for discovering neighbors in the next η time intervals (at times $t+\gamma, t+2\gamma, .., t+\eta\gamma$).
end if
while (the packet does not arrive to d) **do**
  if (the current node does not have a neighbor that makes a progress to d) **then**
    if (the current node does not use the *Maximum-transmission* range) **then**
      The node uses the *Maximum-transmission* range to send a control message asking for neighbors.
      The nodes that make a progress to d inside the new sphere (Radius) send back a control message telling about their positions.
      if (there is a new neighbor that makes a progress to d) **then**
        The current node sends the packet to the node that makes the greatest progress to d
      else
        return (False)
      end if
    else
      return (False)
    end if
  else
    The current node sends the packet to the neighbor node that makes greatest progress to d.
  end if
end while
return (True).

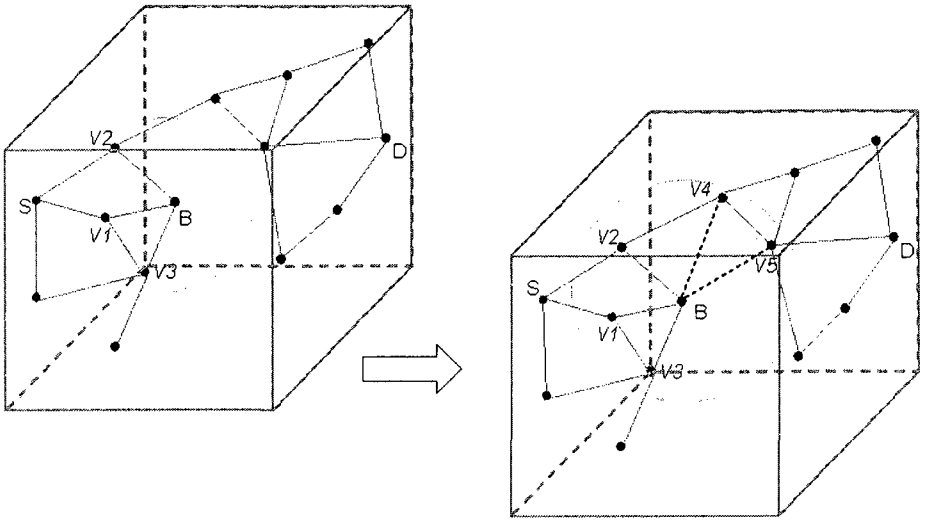


Figure 45: PAG algorithm, The packet arrives to the node B which is local minimum, so it increases the transmission range to the maximum

7.1.3 PAG:CFace(3)

The previous algorithms PAG, PAGC and their associated fixed power Greedy algorithm have a great advantage in terms of power saving. In our simulations, though, they suffer from low delivery rates if the network is very sparse. Our solution is similar to the solution used in Chapter 6 which is CFace routing if the PAG algorithm fails to deliver the message. The combination is called PAG:CFace(3) and is summarized in Algorithm 7.2.

7.1.4 PAG:CFace(1):PAG

Our second hybrid algorithm starts with PAG algorithm. Once a local minimum is reached at low transmission range, the current node holding the packet adjusts its transmission range. If it stays in the local minimum situation the algorithm extracts locally the 3D-GG graph from the UDG and projects it on one plane, which is randomly one of the yx, yz or xz planes, and then it switches to CFace(1). CFace(1) traverses that projective plane starting from the local minimum node as the new source node. When the packet reaches a node closer to the destination than the

Algorithm 7.2 PAG:CFACE(3)

1: ***Input*** *source node s, the destination node d, and a threshold TH*
2: ***Output***: *True if the packet arrives to d, and False otherwise.*
3: Let c be the current node holding the packet during the routing algorithm
4: Call PAG routing algorithm.
5: **if** (c adjusts its transmission range, and after that, it stays in the local minimum situation) **then**
6: $w \leftarrow c$
7: Extract 3D-GG from the UDG
8: $Counter \leftarrow 1$
9: **while** ($counter < 4$) **do**
10: **switch**(counter)
11: **case** 1: $CurrentPlane \leftarrow xy$ plane
12: **case** 2: $CurrentPlane \leftarrow yz$ plane
13: **case** 3: $CurrentPlane \leftarrow xz$ plane
14: **endswitch**
15: Project the 3D-GG on the $CurrentPlane$
16: Call Face routing on the projected graph starting from w
17: During face routing in 16:
18: **if** The packet arrives to d **then**
19: return (True)
20: **else if** (the length of the path on the $CurrentPlane$ is greater than TH) **then**
21: $Counter \leftarrow counter + 1$
22: **else**
23: Face routing continues.
24: **end if**
25: **end while**
26: return (False)
27: **end if**

local minimum, the algorithm switches back to the PAG routing algorithm on the UDG, and so on. The algorithm drops the packet if the path length exceeds a specific threshold.

Algorithm 7.3 PAG:CFace(1):PAG

1: ***Input*** *source node s, the destination node d and the threshold TH*
2: ***Output***: *True if the packet arrives to d, and False otherwise.*
3: Let c be the current node holding the packet during the routing algorithm
4: **if** (c is a neighbor to d) **then**
5: return (True)
6: **end if**
7: **if** (the path length $> TH$) **then**
8: return (False)
9: **end if**
10: call PAG algorithm.
11: **if** (c adjusts its transmission range, and after that, it stays in the local minimum situation) **then**
12: $w \leftarrow c$
13: Extract 3D-GG from the UDG
14: Project the 3D-GG on xy, yz or xz plane, which is chosen randomly,
15: Call Face routing on the projected graph starting from w
16: During Face routing in 14:
17: **if** ($dist(c,d) < dist(w,d)$) **then**
18: call(PAG:CFace(1):PAG) starting from the current node c
19: **end if**
20: **end if**

PAG:CFace(1):PAG is summarized in Algorithm 7.3. Figure 46 shows an example of this algorithm: when the increasing of the transmission range at the local minimum node B did not give it any new neighbor that makes progress, the algorithm extracts the 3D-GG and projects it on the xz plane and switches to CFace(1). If face routing passes the local minimum, reaching a node closer to the destination than the local minimum, it switches back to PAG.

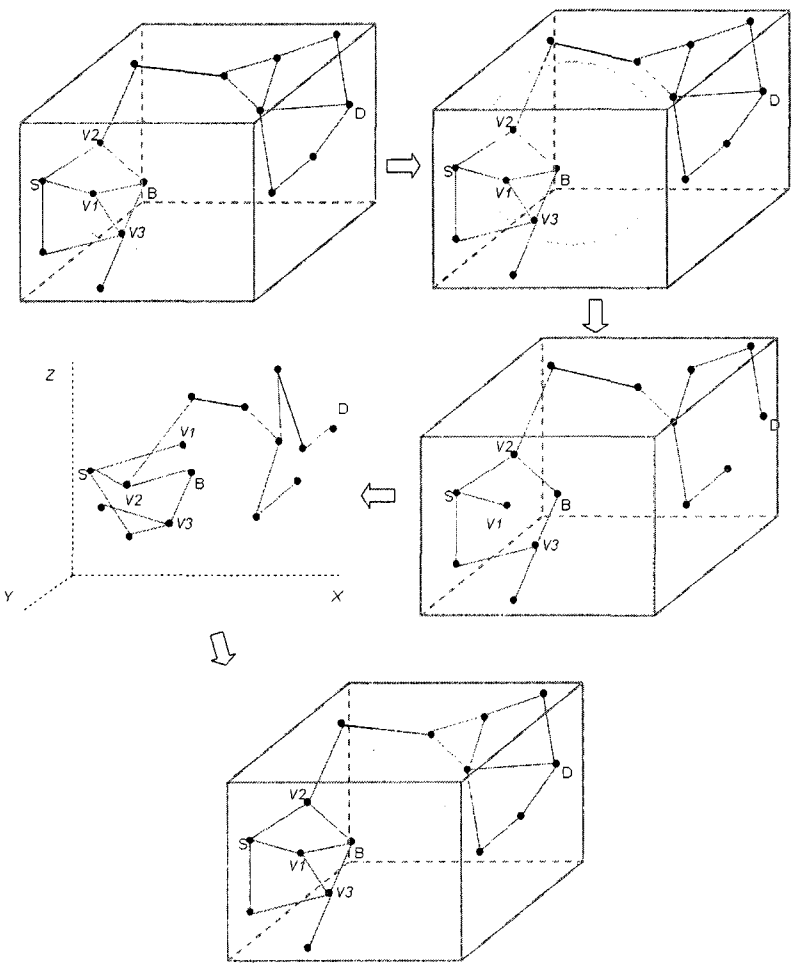


Figure 46: PAG:CFace(1):PAG algorithm steps

7.2 Power-aware Position-based Routing Algorithms using Half of the Maximum Transmission Range

In our initial study of the power-aware routing [4], we have proposed a set of algorithms that use the power adjustment of the periodic control messages. There are 3 main differences between that set of algorithms and our new algorithms:

1. In [4] the algorithms use half of the maximum transmission range, for the periodic control messages, while the new algorithms use the optimal transmission range. Both go to maximum transmission range if needed.

2. The algorithms in [4] uses the low transmission range for the periodic control messages all the time without checking the density of the node, while the new algorithms check the node density to decide whether to use the optimal transmission range or the maximum transmission range for the periodic control messages.

3. The algorithms in [4] do not recover from face routing if the local minimum is passed, while the new algorithms do.

7.3 Simulations and Results

7.3.1 Simulation Environment

We use in our simulation the wireless model that has been proposed by Heinzelman *et al.* [53] which assumes the radio dissipates $E_{elec} = 100\ nJ/bit$ power to run the transmitter or receiver circuitry. Both transmitter and receiver nodes consume E_{elec} to transmit one bit. The radio dissipates $E_{amp} = 100\ pJ/bit/m$ to run the transmit amplifier, assuming ω^2 energy loss due to channel transmission, where ω is the distance between nodes. This implies the sender consumes $(E_{amp} * \omega^2)$ power to transmit one

bit. According to the above wireless model, transmitting a $k-bit$ message at distance ω the transmitter expends $E_{Tx}(k,d) = E_{elec} * k + E_{amp} * k * \omega^2$. To receive it, the receiver node expends $E_{Rx}(k) = E_{elec} * k$. From the model above, it is clear that the final expression to transmit one bit message equal to $\omega^2 + 2000$, hence, β=2000 and $\alpha = 2$ which means the optimal transmission range $(\beta/(\alpha - 1))^{1/\alpha}$ is equal to 44.7.

In the simulation experiments, we run the algorithms in two different simulation environments. The first one is called the sensor network environment and the second one is called the ad hoc environment. For the sensor network, a UDG with $205, 600, 1300, 2450, 4200, 6450$ or 9500 random nodes is generated in different boxes of side length $200, 300, 400, 500, 600, 700$ or 800, respectively. The average degree of nodes is around 20, the degree proposed previously in [10]. The fixed maximum transmission range is set to 66 units, and the values of $\gamma = 3$, $\eta = 5$, $\varrho = 4$ are used, a threshold equal to the number of nodes is used in the hybrid algorithms PAG:CFace(3) and PAG:CFace(1):PAG. All our results have a 95% confidence intervals.

In the ad hoc network environment a UDG with random 200 nodes is generated in a box of side length 250 units. With different maximum transmission ranges and a fixed starting transmission range equal to 44.7.

Initially, for the algorithms with an adjusted transmission power, the transmission range for all nodes is set to 44.7 with the possibility that the node will increase its transmission range to its maximum. For the algorithms with a constant transmission power, their transmission range is set to the maximum. A fixed size data packet of length 16 bytes is used in addition to a 6 bytes control packet that contains the ID, position, and current battery level of the node. Initially, all the nodes have an equal energy level.

First we randomly generate a UDG. If the graph is connected, we use it in the simulation, otherwise another UDG is randomly generated. Then a set of 2000 source-destination pairs is randomly chosen for the sensor network, or a set of 1000 source-destination pairs for the ad hoc environment. All the routing algorithms are then

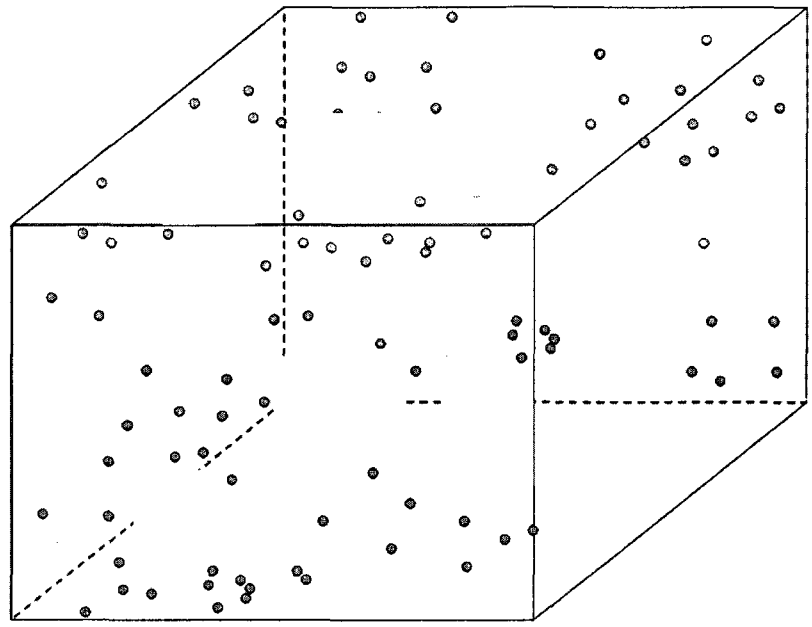


Figure 47: Example of how we simulate some empty regions in the network environment

applied in turn on the chosen source-destination pairs. Over all the source-destination pairs, we compute the average power consumed by the maximally used node after applying the algorithms.

To compute the packet delivery rate, this process is repeated with 5 random graphs for the sensor environment and 10 random graphs for the ad hoc environment, and the percentage of successful deliveries determined. Also, we simulated some holes (empty regions) in the network by placing empty spheres of volume 50 units, that do not contain any nodes inside, the summation of the all the volumes of all spheres is equal to 13% of the simulation boxes volumes. See Figure 47 for an example.

7.3.2 Observed Results - Sensor Network Environment

Because the sensor network is a very dense network, the node degree is around 20 [10, 104], the success rate for all the algorithms discussed in Section 7.1 is always around 100%, therefore the success rate will not be a variable in the measurements. Also, there is no need to test the combined algorithms PAG:CFace(3) and

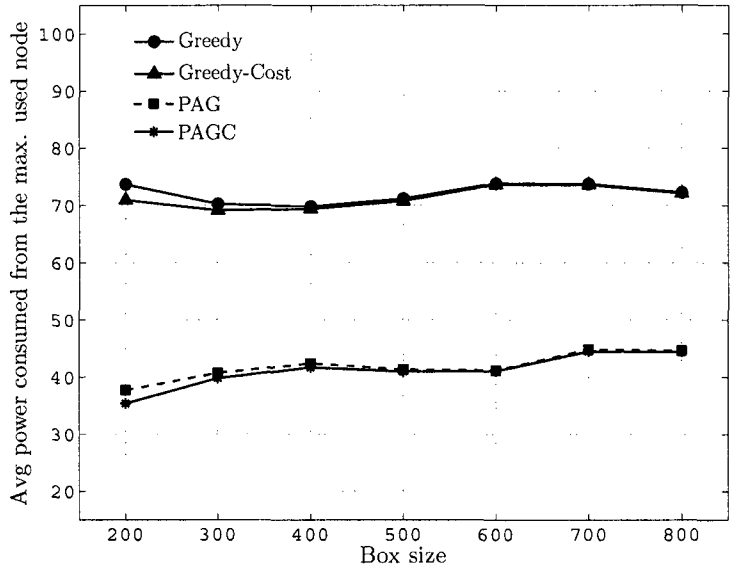


Figure 48: The average power consumption from the maximum used node after 2000 source destination pairs routing process of PAG and other power-aware algorithms

PAG:CFace(1):PAG in this environment, since the graphs are so dense, switching to face routing rarely occurs.

Figure 48 shows the average power consumed from the maximum used node in PAG, PAGC and its associated Greedy and Cost algorithms. It is clear from this figure that the average power consumed from maximum used node of the new algorithms PAG and PAGC is decreased by around 45%, which in turn increases the network lifetime to around twice that of a fixed transmission radius algorithm. Figure 49 shows the effect of holes in the network. The result is close to the result found in Figure 48. The main difference is in the algorithms PAG and PAGC, where the average power consumption of the maximum used node has been slightly decreased. This can be explained by the nodes degree. Because of the holes the nodes are closer to each other which means there are more neighbors for the low transmission range and a greater chance to use different neighbors for the different routed packets.

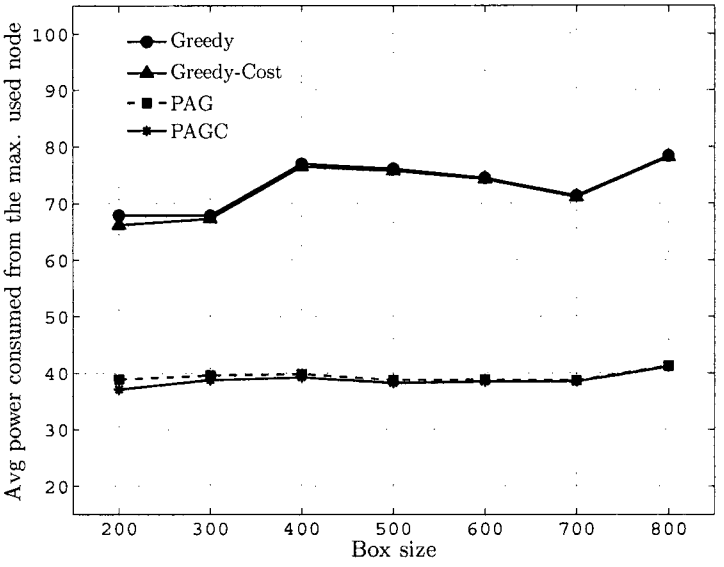


Figure 49: The average power consumption from the maximum used node after 2000 source destination pairs routing process of PAG and other power-aware algorithms

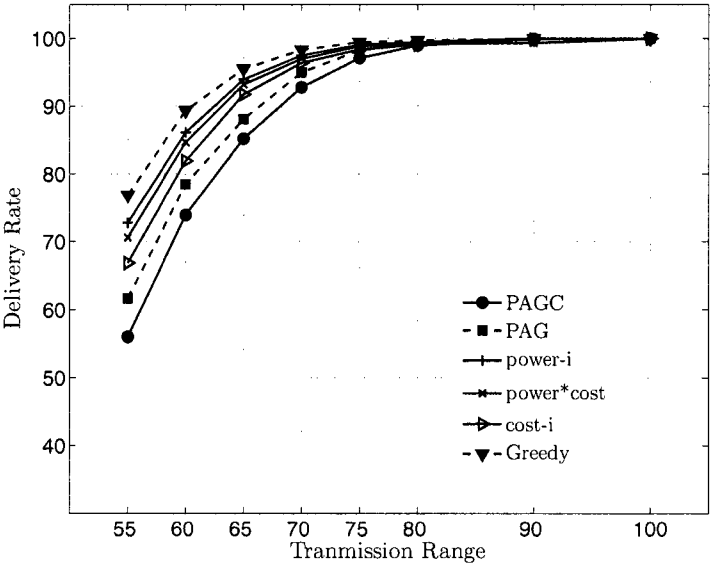


Figure 50: The average delivery rate of PAG, Greedy and other different power-aware routing algorithms

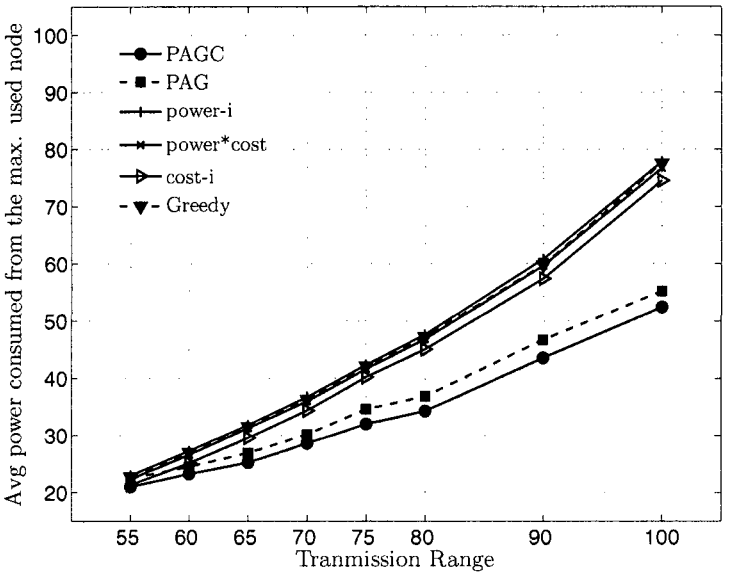


Figure 51: The average power consumption from the maximum used node after 1000 source destination pairs routing process of PAG and different power-aware routing algorithms

7.3.3 Observed Results - Ad Hoc Network Environment

Since the network is less dense in the ad hoc network environment, the delivery rate of Greedy, PAG and PAGC will be effected by the local minimum phenomena which implies that the delivery rate is a very important measurement here. Figure 50 shows the delivery rate of PAG and the other power-aware routing algorithms, the delivery rate of Greedy algorithm is shown in the same figure. It is immediately evident from this figure that the delivery rate of PAG and PAGC is slightly less than regular Greedy for the low transmission range but has almost the same delivery rate as Greedy when the transmission range increases above 70.

From Figure 51, which shows the average power consumed from the maximum used node, the new algorithms PAG and PAGC decrease the average power used by the maximum used node by around 30%, compared to those algorithms with a fixed transmission radius. The other power-aware routing algorithms do not gain more than a 4% increase in the network lifetime. From Figure 50, it can be seen that PAG,

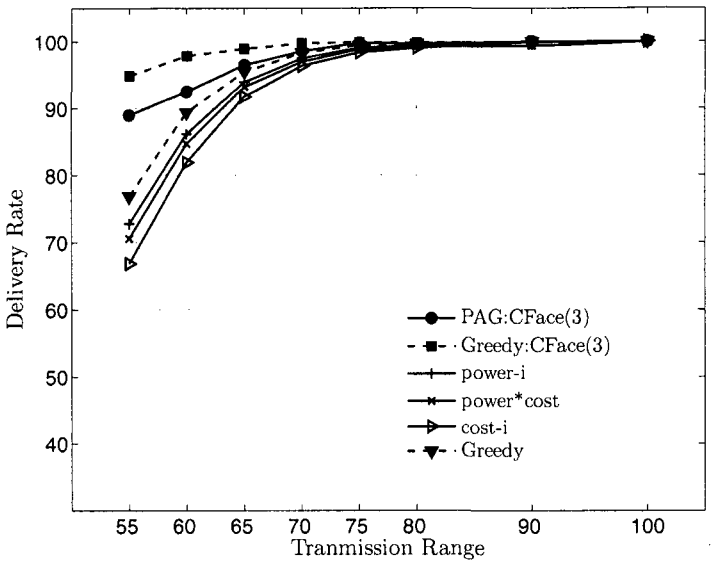


Figure 52: The average delivery rate of PAG:CFace(3), Greedy:CFace(3) and other different power-aware routing algorithms

Greedy and all other studied algorithms have a low delivery rate if the network is sparse and a 100% delivery rate in dense networks. This can be explained by the number of neighbors. Fewer neighbors implies less chance to choose a good route that makes progress to the destination.

Figures 52 and 53 show the expected results that our algorithms Greedy:CFace(3) and PAG:CFace(3) increase the delivery rate to around 90% for sparse networks and to about 100% for dense networks. The drawback is an increase of the average power from the maximum used node over Greedy and PAG. Still PAG:CFace(3) has a longer network lifetime up to 25% more than Greedy:CFace(3) if the average node degree is above 6, which means the transmission range is above 75.

The results of the PAG:CFace(1):PAG and Greedy:CFace(1):Greedy algorithms are shown in Figures 54 and 55. This algorithm tries to compromise between the two groups of algorithms Greedy and PAG on one side, and PAG:CFace(3) and Greedy:CFace(3) on the other side. First, in terms of delivery rate, as expected, PAG:CFace(1):PAG has nearly a 100% delivery rate for an average network with a

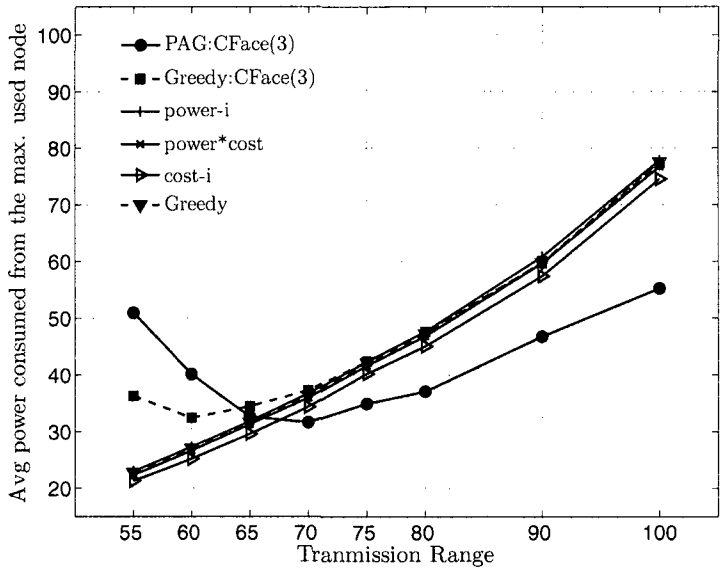


Figure 53: The average power consumption from the maximum used node after 1000 source destination pairs routing process of PAG:CFace(3), Greedy:CFace(3) and other power-aware routing algorithms

transmission range between 70 and 80 which has an average nodes degree around 6. The delivery rate for the sparse network is greater than Greedy:CFace(3) and PAG:CFace(3). This algorithm still increases the network lifetime by decreasing the power consumption of the nodes.

The most important advantage of all our new algorithms is the substantial increase of the network lifetime while preserving the delivery rates. The same simulations have been done using the Compass routing algorithm [71] in place of the Greedy algorithm, generating new routing algorithms, called PAC, PAC:CFace(3), and PAC:CFace(1):PAC. The results are nearly the same as for the Greedy-based algorithms.

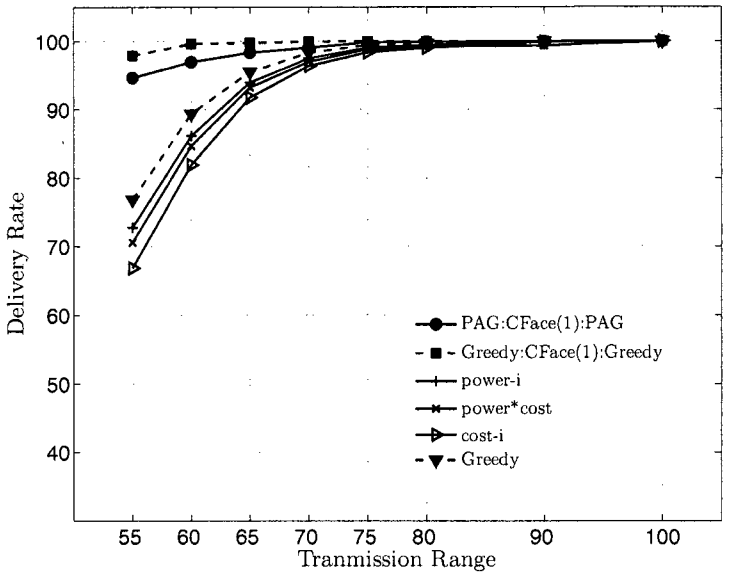


Figure 54: The delivery rate for PAG:CFace(1):PAG, Greedy:CFace(1):Greedy and the other power-aware routing algorithms

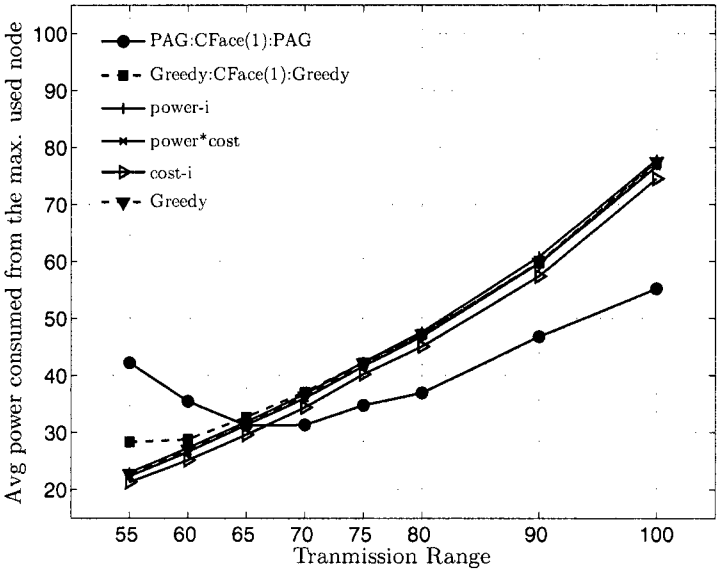


Figure 55: The average power consumption from the maximum used node after 1000 source destination pairs routing process of PAG:CFace(1):PAG, Greedy:CFace(1):Greedy and other power-aware routing algorithms

Chapter 8

Conclusion and Future Research

This chapter briefly concludes the thesis by highlighting the major contributions of this research and pointing out several future research directions.

This thesis has presented several adaptive and efficient $3D$ protocols that use the geographical position of ad hoc nodes for providing network-layer services in ad hoc networks. In particular, we have presented a UDG subgraph algorithm, dominating set algorithms and several position-based routing algorithms. The effectiveness of the proposed method is demonstrated through numerical simulations. We believe that the $3D$ results provided in this thesis will be useful in many ways for the research and implementation of future $3D$ networks.

In the next section, the contributions made in each of the previous chapters and the concluding results drawn from the associated research are presented. Suggestions for future research directions related to this thesis are provided in Section 8.2.

8.1 Contributions of the thesis

- In Chapter 3, 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 $\theta_m(p, |uz|)$ 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, the class of DAAY subgraphs easily extends to three-dimensional UDGs. The class of graphs presented also preserves the common properties of the Yao and HSP graphs such as bounded out-degree, having the EMST as a subgraph, and being spanner graphs with bounded stretch factors.

- In Chapter 4, we proposed the first fully local algorithms that construct a dominating set and a connected dominating set of UDG in $3D$ environment in a constant time. The algorithm does not rely on the spanning tree construction, which makes it practical for situations where the topology changes are frequent and unpredictable. We proved that the size of the constructed dominating set is at most 24 times the optimal. We also showed the importance of the constructed independent dominating set on maintaining a low approximation ratio for the construction of the CDS. A simulation study has been conducted to compare our proposed algorithm with the Greedy global algorithm [34, 62, 99] and the linear time algorithms by Alzoubi *et al.* [12, 13] in terms of the size of the dominating set and the number of connectors. Our algorithms have similar results as Alzoubi algorithms, though our algorithms run in a constant time compared to a linear time of the Alzoubi algorithms.

- We improved a new version of Projective Face routing algorithms (CFace) in Chapter 2. Unlike ALSP routing algorithms, [65], in which the current node holding the packet has to know the location of the nodes that are 2 hops away to calculate the projection plane, CFace does not need any information to calculate the projection plane since it projects the nodes on the coordinate planes. The simulation results show that CFace has a comparable delivery rate to ALSP, but with less path dilation.

- We proposed a new class of randomized $3D$ position-based algorithms for routing in mobile ad hoc networks, called the AB3D algorithms in Chapter 5. In this class, the current node uses a space-partition heuristic to divide the space into m regions and choose one neighbor from each region using Greedy or Compass then forwards the message randomly to one of these m nodes. Our simulation results demonstrate that these randomized algorithms, on non-planar graphs like the UDG, yield a definite improvement over all the other deterministic algorithms studied, when considering the delivery rate.

- In order to build a position-based routing algorithm that could route packets in a scalable and effective manner in $3D$ environment, we developed the ABLAR routing algorithms in Chapter 5. This class of algorithms uses the partial flooding strategy, where the current node chooses m neighbors according to a space-partition heuristic and forwards the message to all these nodes. According to the simulation results in Section 5.6; this algorithm gives a nearly certain delivery rate, but the associated traffic was high.

- In Chapter 6, we created two new hybrid routing algorithms, AB3D-CFace(1)-AB3D and AB3D-CFace(3), that combine the efficiency of randomized progress-based algorithms with the high delivery rate of Face routing. Our experiments on unit disk graphs, and their associated Yao subgraphs and Gabriel subgraphs show that these hybrid algorithms increased the delivery rate to over 97% while keeping the average dilation of the route much smaller than that of Face routing.

- To combine the efficiency of progress-based and randomized progress-based algorithms with a high delivery rate of ABLAR routing; we proposed two new groups of hybrid routing algorithms. First, T-ABLAR-T, where progress-based routing is used until a local minimum is reached. The algorithm then switches to ABLAR for one step after which the algorithm switches back to the progress-based algorithm again. Second, AB3D-ABLAR, in which AB3D is used until a

threshold is passed in terms of number of hops. The algorithm then switches to an ABLAR algorithm. Our experiments showed that the best performing algorithms in the AB3D-ABLAR group indeed increased the delivery rate to over 99% while managing to save more than 55% of the average traffic consumed by LAR3D.

- In Chapter 7, we proposed several local power-aware routing algorithms for $3D$ ad hoc and sensor networks under two concurrent constraints: maximizing the delivery rate while maximizing the lifetime of the network by minimizing the energy consumption by the nodes. Our new algorithms are based on the idea of replacing the constant transmission power of a node with an adjusted transmission power during two stages: first, a lower power while discovering the neighboring nodes and second, if needed a higher transmission power during the routing process. The simulation results demonstrated that the new routing algorithm PAG has a delivery rate near 100% with dense networks and increases the network lifetime to around twice that of the Greedy algorithm. Our second and third algorithms, PAG:CFace(3) and PAG:CFace(1):PAG, significantly increase the delivery rate for sparse networks while also increasing the average lifetime of a network.

8.2 Future research directions

Several interesting research directions motivated by this thesis are discussed next. In addition to designing scalable routing algorithms for ad hoc and sensor networks, we intend to accomplish the following projects in the near future:

- In all our simulations, we assumed that the UDG was static. Node mobility is not considered in this thesis. In real situations, nodes may change their position with time. Nodes may also become inactive after a certain time and vice versa. It would be interesting to develop and test all our routing algorithms

in mobile environments. However, we expect most of our proposed algorithms that depend on AB3D to perform well on dynamic $3D$ unit disk graphs since the randomization component of these algorithms can adjust to reasonably large changes in node positions.

- In this research, we assume the location service is available. It is desired to integrate the location service into our routing algorithms. Thus, the proposed algorithms can be implemented in a real mobile node more easily.

- The simulations were designed and implemented using C++. Although great care was taken in designing and implementing a series of network simulations that could easily be modified to test different scenarios, it became increasingly difficult to make changes to the network as ideas surfaced. The next step would be to port the simulation to the ns-2 network simulation environment.

- DAAY can be extended easily to the 3D space. Since DAAY distributes location information and performs calculations in a one-hop transmission range, it is possible to implement DAAY in $3D$ space. The hard part is to prove similar properties for the subgraph as in $2D$. Further experimentation, including routing algorithms, would show the impact of our subgraph on the routing algorithms.

- For the dominating set proposed in Chapter 4, it would be interesting to show a tighter bound on the size of the connected dominating set. The message complexity analysis is not included: trying to create our algorithms with an optimal number of control messages remains a part of future work.

- Due to some specific applications and newly developed techniques, the concept of a connected dominating set can be modified or further extended for more efficient usage. That is, constructing an energy-efficient virtual backbone in

MANETs for broadcasting applications using directional antennas may be considered [109]. As a part of future work it would be interesting to see if our algorithms can be extended to adapt such applications.

- Other observations about the connected dominating sets are as follows: Alzoubi *et al.* [11] integrate the connected dominating set and the local Delaunay graph to form a backbone of the wireless network that is planar and a spanner. However, we proved in Chapter 2 that their algorithm is not completely local: its time complexity is $O(n)$, which makes it a bad choice for ad hoc wireless networks. Moreover, the construction of a Delaunay graph assumes that the nodes are static or can be viewed as static during a reasonable period of time. This is acceptable in sensor networks, but not in a situation where nodes move frequently and unpredictably. Based on these observations, we are interested in combining our local algorithm for constructing the connected dominating set with the $3D$ version of some other spanner subgraphs, e.g. DAAY or Yao, such that node mobility is allowed.

- As in most of the research papers in the area of MANETs, in this thesis we assumed that all nodes are homogeneous. We did not consider the presence of obstacles, such as walls, buildings or weather conditions, which might obstruct signal propagation. Barrière *et al.* [19] consider irregular transmission range of the mobile nodes. This work is unique in MANET literature, and further investigation of the behavior of our routing algorithms under such a varying radio range model would be an interesting area for a future research.

- One obvious part of our future work is to test our $3D$ algorithms with the connected dominating set constructed in Chapter 4 as a backbone.

- The widely-accepted existing routing protocols designed to accommodate the needs of such self-organized networks do not address possible threats aiming at the disruption of the protocol itself. The assumption of a trusted environment

is not one that can be realistically expected; hence, several efforts have been made towards the design of a secure and robust routing protocol for ad hoc networks. It would be desirable to integrate some of the security heuristics [58] into our routing algorithms.

Bibliography

[1] A.E. Abdallah, T. Fevens, and J. Opatrny. Hybrid position-based 3d routing algorithms with partial flooding. In *Proc. of the Canadian Conference on Electrical and Computer Engineering*, pages 1135–1138, Ottawa, May 2006.

[2] A.E. Abdallah, T. Fevens, and J. Opatrny. Randomized 3-d position-based routing algorithm for ad-hoc networks. In *Proc. of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS)*, pages 1–8, San Jose, July 2006.

[3] A.E. Abdallah, T. Fevens, and J. Opatrny. High delivery rate routing algorithms for 3d ad hoc network. *To appear in Computer Communications journal on Algorithmic and Theoretical Aspects of Wireless Ad Hoc and Sensor Networks*, 2007.

[4] A.E. Abdallah, T. Fevens, and J. Opatrny. Power-aware 3d position-based routing algorithms for ad hoc networks. In *Proc. of the IEEE International Conference on Communications ICC-2007*, pages 1–6, Glasgow, June 2007.

[5] A.E. Abdallah, T. Fevens, and J. Opatrny. 3d local algorithm for dominating sets of unit disk graphs. In *submission pending*, 2008.

[6] A.E. Abdallah, T. Fevens, J. Opatrny, and I. Stojmenovic. Power-aware 3d position-based routing algorithms using adjustable transmission ranges for ad hoc and sensor networks. *Submitted for publication, Ad Hoc Networks Journal.*

[7] A.E. Abdallah, M.Hassan, G. Kao, and C. Morosan. Topology control for balanced energy consumption in emergency wireless deployments. In *Proc. of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 41–48, Montreal, September 2005.

[8] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables.* 9th printing. New York: Dover, p. 79, 1972.

[9] I. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks*, 3(3):257–279, 2005.

[10] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[11] K. Alzoubi, X. Li, Y. Wang, P. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):408–421, 2003.

[12] K. Alzoubi, P. Wan, and O. Frieder. Distributed heuristics for connected dominating sets in wireless ad hoc networks. *Journal of Communications Networks*, 4(1):141–149, 2002.

[13] K. Alzoubi, P. Wan, and O. Frieder. Message-optimal connected-dominating-set construction for routing in mobile ad hoc networks. In *Proc. of the Third ACM international Symp. Mobile Ad Hoc Networking and Computing (MobiHoc 02)*, pages 157–164, Lausanne, June 2002.

[14] S. Ansari, L. Narayanan, and J. Opatrny. A generalization of the face routing algorithm to a class of non-planar networkss. In *Proc. of the Annual International Conference on Mobile and Ubiquitous: Networks and Services (MOBIQUITOUS)*, pages 213–225, California, July 2005.

[15] F. Araujo and L. Rodrigues. Survey on position-based routing. In *Technical Report TR-01*, University of Lisbon, 2006.

[16] J. Aslam, Q. Li, and D. Rus. Three power-aware routing algorithms for sensor networks. *Wireless Communications and Mobile Computing*, 2(3):187–208, 2003.

[17] D. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Transactions on Communications*, 29(11):1694–1701, 1981.

[18] D. Baker, A. Ephremides, and J. Flynn. The design and simulation of a mobile radio network with distributed control. *IEEE Journal on Selected Areas in Communications*, 2(1):226–237, 1984.

[19] 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*, 3(2):141–153, 2003.

[20] S. Basagni, I. Chlamtac, and V. Syrotiuk. Dynamic source routing for ad hoc networks using the global positioning system. In *Proc. of the IEEE Wireless Communications and Networking Conference(WCNC'99)*, pages 21–24, New Orleans LA, September 1999.

[21] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proc. of the 4th annual ACM/IEEE international conference on Mobile computing and networking (MOBICOM)*, pages 76–84, Dallas, 1998.

[22] W. Beyer. *CRC Standard Mathematical Tables.* 28th ed. Boca Raton, FL: CRC Press, pp. 131 and 147-150, 1987.

[23] P. Boone, E. Chavez, L. Gleitzky, E. Kranakis, J. Opatrny, G. Salazar, and J. Urrutia. Morelia test: Improving the efficiency of the gabriel test and face routing in ad-hoc networks. In *Proc. of the 11th Colloquiua on Structural Information and Communication Complexity (SIROCCO)*, pages 23–34, Slovakia, June 2004.

[24] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick. On the spanning ratio of gabriel graphs and beta-skeletons. In *Proc. of the Latin American Theoretical Infocomatics (LATIN)*, pages 479–493, Mexico, April 2002.

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

[26] P. Bose and P. Morin. Online routing in triangulations. In *10th Annual International Symposium on Algorithms and Computation (ISAAC '99), volume 1741 of LNCS*, pages 113–122, India, December 1999.

[27] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks journal*, 7(6):609–616, 2001.

[28] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, Dallas TX, October 1998.

[29] S. Capkun, M. Hamdi, and J.-P. Hubaux. Gps-free positioning in mobile ad-hoc networks. *Cluster Computing Journal*, 5(2):118–124, 2002.

[30] C. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(4):609–619, 2004.

[31] E. Chaves, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, H. Tejeda, and J. Urrutia. Half-space proximal: A new local test for extractinga bounded dialation spanner of a unit disk graph. In *Proc. of the 9th International Conference on Principles of Distributed Systems (OPIDIS2005)*, pages 235–245, Italy, December 2006.

[32] C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proc. of the IEEE Singapore International Conference on Networks (SICON)*, pages 197–211, Singapore, April 1997.

[33] W. Choi and S. Das. Performance of randomized destination-sequence distance vector (R-DSDV) protocol for congestion control in ad hoc wireless network routing. In *Proc. of the applied Telecommunications symposium wireless Track (ATS)*, San Diego, April 2002.

[34] V. Chvatal. A greedy heuristic for the set covering problem. *Math. Oper. Res. 4*, pages 233–235, 1979.

[35] B. Clark, C. Colbourn, and D. Johnson. Unit disk graphs. *Discrete Math*, 86:165–177, 1990.

[36] J. Czyzowicz, S. Dobrev, T. Fevens, H. Gonzalez-Aguilar, E. Kranakis, J.Opatrny, and J. Urrutia. Local algorithms for dominating and connected dominating sets of unit disk graphs. In *Proc. of the the 8th Latin American Theoretical Informatics Symposium LATIN08*, April 2008.

[37] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *Proc. of the IEEE International Conference on Communications (ICC 97), Vol. 1*, pages 376–380, Monral, June 1997.

[38] S. Durocher, D. Kirkpatrick, and L. Narayanan. On Routing with Guaranteed Delivery in Three-Dimensional Ad Hoc Wireless Networks. In *To apper in*

the proc. of the Ninth International Conference on Distributed Computing and Networking (ICDCN 2008), India, January 2008.

[39] T. Fevens, A. Abdallah, T. Elsalti, and L. Harutyunyan. Class of orientation-invariant yaotype subgraphs of a unit disk graph. In *Proc. of the 4th ACM SIGACT-SIGOPS International Workshop on Foundations of Mobile Computing (DialM-POMC 2007)*, pages 1–8, Portland, August 2007.

[40] T. Fevens, A.E. Abdallah, and B. Bennani. Randomized ab-face-ab routing algorithms in mobile ad hoc network. In *4th International Conference on AD-HOC Networks and Wireless, volume 3738 of LNCS*, pages 43–56, Mexico, October 2005.

[41] T. Fevens, I. Haque, and L. Narayanan. A class of randomized routing algorithms in mobile ad hoc networks. In *Proc. of the 1st Algorithms for Wireless and Ad-hoc Networks (A-SWAN)*, pages 347–358, Boston, August 2004.

[42] G. Fin. Routing and addressing problems in large metropolitan-scale internetworks. In *Technical Report ISU/RR-87-180, USC ISI, Marina del Ray*, California, 1987.

[43] P. Fraigniaud and C. Gavoille. A space lower bound for routing in trees. In *In 19 th Annual Symposium on Theoretical Aspects of Computer Science (STACS) volume 2285 of LNCS*, pages 65–75, March 2002.

[44] H. Frey and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In *Proc. of the 12th annual international conference on Mobile computing and networking*, pages 390–401, Los Angeles, September 2006.

[45] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278, 1969.

[46] B. Gao, Y. Yang, and H. Ma. An efficient approximation scheme for minimum connected dominating set in wireless ad hoc networks. In *Proc. of the IEEE Vehicular Technology Conference*, pages 2744–2748, Los Angeles, September 2004.

[47] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *Proc. of the 17th annual Symposium on Computational Geometry*, pages 188–196, Medford, June 2001.

[48] M. Garey and D. Johnson. *Computers and Intractability-A Guide to the Theory of NP-Completeness.* CA:Freeman, San Francisco, 1979.

[49] S. Giordano, I. Stojmenovic, and L. Blazevic. Postion based routing algorithms for ad hoc networks for ad hoc networks: A taxonomy. In *Ad hoc wireless Networking (ed. X. Cheng, X. Huang, and D.Z. Du), Kluwer*, pages 103–136, July 2003.

[50] Z. Haas and M. Pearlman. The performance of query control schemes for the zone routing protocol. *ACM/IEEE Trans. Net.*, 9(4):427–438, 2001.

[51] I. Haque, C. Assi, and J. Atwood. Randomized energy aware routing algorithms in mobile ad hoc networks. In *Proc. of the 8th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2005)*, pages 71–78, Montreal, October 2005.

[52] T. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs.* Fast Computers. Marcel Dekker, Inc., 1998.

[53] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *Proc. of the International conference on System Sciences*, pages 3005–3014, Hawaii, January 2000.

[54] J. Hightower and Borriello G. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, 2001.

[55] D. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* PWA Publishing Company, Boston, MA, 1995.

[56] T. Hou and V. Li. Transmission range control in multihop packet radio networks. *IEEE Trans. on Communications*, 34(1):38–44, 1986.

[57] P. Hsiao and H. Kung. Gravity routing in ad hoc networks: Integrating geographical and topology-based routing. In *Proc. of the International Symposium on Parallel Architectures, Algorithms and Networks*, pages 397–403, china, May 2004.

[58] Y. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security & Privacy*, 2(3):28–39, 2004.

[59] J. Hurink and T. Nieberg. Approximating minimum independent dominating sets in wireless networks. *Technical Report, Department of Applied Mathematics, University of Twente, Enschede. ISSN 1874-4850*, 2007.

[60] J. Jaromczyk and G. Toussaint. Relative neighborhood graphs and their relatives. *Proc. IEEE*, 80(9):1502–1517, 1992.

[61] L. Jia, R. Rajaraman, and T. Suel. An efficient distributed algorithm for constructing small dominating sets. In *Proc. of the 20th ACM Symposium on Principles of Distributed Computing (PODC'01)*, pages 33–42, Rhode Island, August 2001.

[62] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, pages 256–278, 1974.

[63] D. Johnson and D. Malts. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing (ed. T. Imielinski and H. Korth), chapter 5*, pages 153–181. Kluwer Academic Publishers, 1996.

[64] G. Kao, T. Fevens, and J. Opatrny. Position-based routing on 3D geometric graphs in mobile ad hoc networks. In *Proc. of the 17th Canadian Conference on Computational Geometry (CCCG'05)*, pages 88–91, Windsor, August 2005.

[65] G. Kao, T. Fevens, and J. Opatrny. 3-d localized position-based routing with nearly certain delivery in mobile ad hoc networks. In *Proc. of the Wireless Pervasive Computing, ISWPC '07*, pages 344–349, San Juan, Febuary 2007.

[66] E. Kaplan. Understanding GPS. Artech house, 1996.

[67] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless net-works. In *Proc. of the 6th ACM/IEEE Conference on Mobile Computing and Networking (Mobicom 2000)*, pages 243–254, Boston, August 2000.

[68] R. Karp. Reducibility among combinatorial problems. In *Proceedings of the Symposium on Complexity of Computer Computations*, pages 85–103, 1972.

[69] J. Keil and C. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete Computational Geometry*, 7(1):13–28, 1992.

[70] Y. Ko and N. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *ACM/Baltzer Wireless Networks (WINET)*, 6(4):307–321, 2000.

[71] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. of the 11th Canadian Conference on Computational Geometry (CCCG '99)*, pages 51–54, Vancouver, August 1999.

[72] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally. In *Proc. of the the 23rd annual symposium on Principles of distributed computing*, pages 300–309, Newfoundland, July 2004.

[73] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the locality of bounded growth. In *Proc. of the 24th annual symposium on Principles of distributed computing*, pages 60–68, Las Vegas, July 2005.

[74] F. Kuhn and R. Wattenhofer. Constant-time distributed dominating set approximation. In *Proc. of the the 22nd annual symposium on Principles of distributed computing*, pages 25–32, Boston, July 2003.

[75] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-hoc networks beyond unit disk graphs. In *Proc. of the 2003 joint workshop on the foundation of mobile computing (DIALM-POMC)*, pages 69–78, San Diego, September 2003.

[76] J. Kuruvila, A. Nayak, and I. Stojmenovic. Progress and location based localized power aware routing for ad hoc and sensor wireless networks. In *Proc. of the 3rd International Conference on AD-HOC Networks and Wireless ADHOC-NOW*, pages 294–299, Vancouver, July 2004.

[77] B. Kwak, N. Song, and L. Miller. On the scalability of ad hoc networks: a traffic analysis at the center of a network. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, Atlanta, March 2004.

[78] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proc. of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 120–130, Boston, August 2000.

[79] X. Li, G. Calinescu, and P. Wan. Distributed construction of planar spanner and routing for ad hoc wireless networks. In *Proc. of the IEEE INFOCOM (3)*, New York, June 2002.

[80] X. Li, P. Wan, and Y. Wang. Power efficient and sparse spanner for wireless ad hoc networks. In *Proc. of the IEEE International Conference on Computer*

Communications and Networks (ICCCN01), pages 564–567, Arizona, October 2001.

[81] X. Li, Y. Wang, P. Wan, W. Song, and O. Frieder. Localized low-weight graph and its applications in wireless ad hoc networks. In *Proc. of the Twenty-Third IEEE INFOCOM*, pages 431–441, Hong Kong, March 2004.

[82] Xiang-Yang Li. Topology control in wireless ad hoc networks. *Book Chapter of Ad Hoc Networking, IEEE Press, edited by Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic.*, pages 175–203, 2003.

[83] B. Liang and Z. Haas. Virtual backbone generation and maintenance in ad hoc network mobility management. In *Proc. of the IEEE INFOCOM (3)*, pages 1293–1302, March 2000.

[84] S. Liu, T. Fevens, and A.E. Abdallah. Hybrid position-based routing algorithms for 3-d mobile ad hoc networks. In *submitted for publication to the The 15th International Conference on Telecommunications*, 2008.

[85] M. Mauve, J. Widmer, and H. Hartenstein. A survey of position-based routing in mobile ad-hoc networks. *IEEE Network Magazine*, 15(6):30–39, 2001.

[86] S. Murthy and J. Garcia-Luna-Aceves. A routing protocol for packet radio networks. In *Proc. of the ACM International Conference on Mobile Computing and Networking*, pages 86–95, France, November 1995.

[87] Soumendra Nanda. Spatial multipath location aided routing. In *Master's thesis, University Hanover*, New Hampshire, June 2004.

[88] R. Nelson and L. Kleinrock. The spatial capacity of a slotted aloha multihop packet radio network with capture. *IEEE Transactions on Communications*, 32(6):684–694, 1984.

[89] The National Institute of Standards and Technology. Wireless ad hoc sensor networks, URL: http://w3.antd.nist.gov/wahn_ssn.shtml. 2001.

[90] C. Perkins and P. Bhagwat. Highly dynamic destination sequenced distance-vector routing (DSDV) for mobile computers. In *Proc. of the SIGCOMM, Conference on Communications Architectures, Protocols and Applications*, pages 234–244, London, September 1994.

[91] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing System and Application(WMCSA)*, pages 90–100, San Juan, February 1999.

[92] R. Rajaraman. Topology control and routing in ad hoc networks: a survey. *ACM SIGACT News*, 33(2):60–73, 2002.

[93] R. Ramanathan and R. Rosales-Hain. Topology control of multihop wireless networks using transmit power djustment. In *Proc. of the IEEE INFOCOM*, pages 404–413, March 2000.

[94] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. *IEEE Journal Selected Areas in Communications*, 17(8):1333–1344, 1999.

[95] E. Royer and C. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(4):46–55, 1999.

[96] R. Shah and J.M. Rabaey. Energy-aware routing for low-energy ad-hoc sensor networks. In *Proc. of the Wireless Communications and Networking Conference*, pages 812–817, Orlando, March 2002.

[97] C. Shakarji. Least-squares fitting algorithms of the nist algorithm testing system. *Research of the National Institute of Standards and Technology*, 103(6):633–641, 1998.

[98] D. Simplot-Ryl, I. Stojmenovic, and J. Wu. Energy efficient backbone construction, broadcasting, and area coverage in sensor networks. In *in: Handbook of Sensor Networks: Algorithms and Architectures (I. Stojmenovic, ed.), Wiley*, pages 343–379, 2005.

[99] P. Slavik. A tight analysis of the greedy algorithm for set cover. In *Proc. of the 28th ACM Symposium on Theory of Computing (STOC)*, pages 435–441, 1996.

[100] I. Stojmenovic. Location updates for efficient routing in ad hoc networks. In *Handbook on Wireless Networks and Mobile Computing*, pages 451–471, 2002.

[101] I. Stojmenovic. Position-based routing in ad hoc networks. *IEEE Commmunications Magazine*, 40(7):128–134, 2002.

[102] I. Stojmenovic and S. Datta. Power and cost aware localized routing with guaranteed delivery in unit graph based ad hoc networks. *Wireless Communications and Mobile Computing*, 4(2):175–188, 2004.

[103] I. Stojmenovic and X. Lin. Loop-free hybrid single-path flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Trans. on Parallel and Distributed Systems*, 12(10):1023–1032, 2001.

[104] I. Stojmenovic and X. Lin. Power aware localized routing in ad hoc networks. *IEEE Trans. on Parallel and Distributed Systems*, 12(11):1122–1133, 2001.

[105] K. Supowit. The relative neighborhood graph, with an application to minimum spanning trees. *Journal of the Association of Computer Machinery*, 30(3):428–448, 1983.

[106] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Trans. on Communications*, 32(3):246–257, 1984.

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

[108] Y. Wang and X. Li. Distributed spanner with bounded degree for wireless ad hoc networks. *International Journal on Foundations of Computer Science*, 14(2):183–200, 2000.

[109] J. Wu, M. Cardei, F. Dai, and S. Yang. Extended dominating set and its application in ad hoc networks using cooperative communication. *IEEE Transactions on Parallel and Distributed System*, 17(8):851–864, 2006.

[110] J. Wu, F. Dai, M. Gao, and I. Stojmenovic. On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. *IEEE/KICS Journal of Communications and Networks*, 4(1):59–70, 2002.

[111] J. Wu and H. Li. A dominating-set-based routing scheme in ad hoc wireless networks. *Telecommunication Systems*, 18(3):13–36, 2001.

[112] K. Yamazaki and K. Sezaki. The proposal of geographical routing protocols for location-aware services. *Electronics and Communications in Japan*, 87(4):26–34, 2004.

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

[114] C. Yu, B. Lee, and H. Youn. Energy efficient routing protocols for mobile ad hoc networks. *Wireless Communications and Mobile Computing Journal*, 3(11):959–973, 2003.

[115] B. Zhang and H. Mouftah. Energy-aware on-demand routing protocols for wireless ad hoc networks. *Wireless Networks*, 12(4):481–494, 2006.

[116] Yun Zhao. Motion vector routing protocol: A position based routing protocol for mobile ad hoc networks. In *PhD. thesis, University Of Arizona*, USA, April 2005.

List of Acronyms

Mobile Ad Hoc Network	MANET
Three-Dimensional	$3D$
Two-Dimensional	$2D$
Global Position System	GPS
Unit Disk Graph	UDG
Gabriel Graph	GG
Relative Neighborhood Graph	RNG
Yao Graph	YG
Directed Yao Graph	$\overrightarrow{YG}$
Euclidean Minimum Spanning Tree	EMST
Half Space Proximal Graph	HSB
Directed Half Space Proximal Graph	$\overrightarrow{HSB}$
Displaced Apex Adaptive Yao Graphs	DAAY
Directed Displaced Apex Adaptive Yao Graphs	$\overrightarrow{DAAY}$
Connected Dominating Set	CDS

Independent Dominating Set	IDS
Maximum Independent Dominating Set	MIDS
Distributed Database Coverage Heuristic	DDCH
Local Randomize Greedy algorithm	LRG
Local Independent Dominating Sets	LIDS
Greedy Routing	GR
Compass Routing	CM
Ellipsoid Routing	ELP
Most Forwarding Routing	MFR
Distance Sequence Distance Vector	DSDV
Wireless Routing Protocol	WRP
Cluster-head Gateway Switch Routing protocol	CGSR
Dynamic Source Routing	DSR
Ad hoc On Demand Distance Vector Routing	AODV
Zone Routing Protocol	ZRP
Greedy Perimeter Stateless Routing	GPSR
Above/Below Routing	AB
Coordinate Face Routing	Cface
Distance routing effect algorithm for mobility	DREAM
geocasting based Location-Aided Routing	LAR
Power-Aware Greedy	PAG
Power-Aware Greedy-Cost	PAGC