Minimum Energy Path Planning for Ad Hoc Networks

by

Danjie Chen

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2005

Author .....

Department of Electrical Engineering and Computer Science

June 13, 2005

Certified by .....

Philip J. Lin

Principal Communications Engineer, C.S. Draper Laboratory

Thesis Supervisor

Certified by ...

Eytan Modiano

Associate Professor, Department of Aeronautics and Astronautics

Thesis Supervisor

Accepted by .

Arthur C. Smith

Chairman, Department Committee on Graduate Students

Minimum Energy Path Planning for Ad Hoc Networks

by

Danjie Chen

Submitted to the Department of Electrical Engineering and Computer Science
on June 13, 2005, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

We introduce the problem of finding a path for a mobile node traveling from a source to a destination while communicating with at least one node from a set of stationary nodes in such a way that minimizes the transmission energy used in communication. We characterize this problem and introduce two algorithms. The first is a recursive algorithm useful for problems with one communication node. We show the limitations of this algorithm and how it can find suboptimal paths. The second algorithm, the discretized graph algorithm, can be applied to problems with more communication nodes. We find parameters that allow energy efficient paths to be found in suitable time. We demonstrate the applicability of the minimum energy path planning problem and how the discretized graph algorithm can be used in a more general context through an example.

Thesis Supervisor: Philip J. Lin
Title: Principal Communications Engineer, C.S. Draper Laboratory

Thesis Supervisor: Eytan Modiano
Title: Associate Professor, Department of Aeronautics and Astronautics

Acknowledgments

I would like to thank Philip Lin at Draper for wholeheartedly being my advisor from day one. His tireless guidance, expertise, and enthusiasm have inspired me to strive toward high standards. I also thank other members of the Distributed Information Networks group whose feedback were instrumental in guiding this work.

I thank Professor Eytan Modiano, who conceived the minimum energy path planning problem and provided a key insight to the algorithm.

I owe many thanks to Michael Bolin, who generously shared his office space and his vast knowledge, and who provided motivation and company on many long nights this spring. I also thank Clifford Choute and Carri Chan for the countless exchanges about communications and algorithms and for being my sounding boards.

Finally, I thank my friends and family who indirectly support my work by being my foundation.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Internal Company Sponsored Research Project: Enabling Edge Communication Networks.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We examine the problem of finding a path for a mobile node traveling from a source to a destination while communicating with at least one node from a set of stationary nodes in such a way that minimizes the transmission energy used in communication.

Energy efficiency is very important in wireless ad-hoc networks, where nodes have limited battery life and communication costs are a major source of energy depletion. In wireless communications, the power required to communicate with a node is a function of the distance to that node. In many applications, it is desirable for a mobile node to be connected to the network at all times. Such scenarios arise in military context where the mobile node may be deployed in hostile territory and losing communication with the network may mean the demise of that node.

One such application of this problem occurs if a soldier must travel to a destination through enemy territory. Other, more heavily armed, members of his unit are entrenched throughout the terrain and all soldiers control devices that enable communication between members of the unit. The mobile soldier must stay in constant communication with members of his unit as he moves in hostile territory so that they can exchange information and he is not isolated. Because of his communication device's limited battery life, he must traverse a path that minimizes the total energy required for communication in traveling to his destination.

Other applications of this problem arise in robotics and geographic information

systems (GIS) path planning.

In this thesis, we introduce, examine, and thoroughly understand the problem of finding a minimum communication energy path. We develop an approach and investigate the tradeoff between minimizing energy and algorithm complexity. Finally, we demonstrate how this tool can be used to solve a more general problem.

1.1 Related Work

The shortest path problem is a well-studied area with a variety of applications. Dijkstra's algorithm [4] computes the shortest path from any source to all other nodes in a given graph with non-negative edge weights. In shortest path problems, the total cost of a path is the sum of the given weights of the traversed edges. The minimum energy path planning problem differs from a shortest path problem in that the total cost is determined by the relationship of the current position on the path to a set of fixed locations. However, we can adapt the problem into a shortest path problem as we show in Chapter 4. The problem also differs from the shortest path problem in that the minimum energy path planning problem has an infinite number of possible paths from the source to the destination whereas in the shortest path problem, the fixed edges between vertices constrain the problem to have a finite number of paths assuming there are no cycles.

A related problem where possible paths are not discrete is the weighted region problem (WRP) [8], where a shortest path through regions of associated weights is determined. The WRP is motivated by the problem of navigating an autonomous vehicle through various terrains such as grass, sand, and water. The vehicle must find the best path through the terrain because different terrains have different costs associated with traversing them. Researchers have discretized this problem in [6], [9] by breaking the terrain into a grid where costs can be assigned to edges connecting two adjoining squares in the grid. Dijkstra's algorithm can then be used to find the minimum cost path.

The minimum energy path planning problem is similar to the WRP except there is no "region" where the cost is constant. Instead, the cost varies continuously as a function of the position of the mobile node. However, we are inspired by the discretized graph approach in [6], [9] and we will describe in Chapter 4 how we adapt such algorithms as part of the approach to solve this problem.

1.2 Organization

The rest of the thesis is organized as follows: In Chapter 2, we formally define the problem, describe the assumptions we make, and discuss the problem's characteristics.

In Chapters 3 and 4, we introduce two different approaches to solving the problem. First, we analyze the problem using a recursive algorithm, where we recursively divide the problem space in half and calculate points in each step to make up a path. This initial approach practically limits the problem to only one communication node but allows us to gain some insight into the problem. We show results for different types of one communication node problems and describe how the recursive approach finds suboptimal paths for certain problems.

We introduce a more general approach in Chapter 4, a discretized graph algorithm, where we divide the space into a grid, let the grid intersections be vertices, and connect the neighboring vertices with edges to form a connected graph. Then, we calculate the edge weights to be the energy costs of traveling between vertices and find an energy efficient path by searching for the shortest path on the graph. This approach allows for many communication nodes and the energy of the path found decreases as we use a finer grid and increase the graph connectivity. We compare the path found by the discretized graph approach to that of the recursive approach and show how the discretized graph algorithm is able to find paths that the recursive approach cannot.

In Chapter 5, we investigate how grid granularity and graph connectivity for the discretized graph algorithm affect the approximate path it finds. We determine the parameters for which the discretized graph algorithm finds good approximate minimum

energy paths in reasonable running time.

Using the parameters we find in Chapter 5, we demonstrate in Chapter 6 how the discretized graph algorithm can be used to solve an example problem where we determine the minimum number of communication nodes to deploy in a square area in order to minimize the communication energies of mobile nodes traveling from one side of the field to the other.

Finally, we conclude in Chapter 7 by discussing the contributions of this thesis and future work for the minimum energy path planning problem.

Chapter 2

Problem Framework

In this chapter, we formally define the problem and describe assumptions we make. We also characterize the nature of the problem and discuss its properties in depth.

2.1 Definition

Consider a set N of stationary communication nodes where $n_i \in N$ represents a communication node. The minimum energy path planning problem is the problem of finding the minimum energy path for a mobile node m from a source s to a destination d in two-dimensional space while communicating with at least one node $n_i \in N$ at all times.

The minimum power required for two nodes i and j to communicate with each other at a given rate is d_{ij}^{α} where d_{ij} is the distance between node i and node j and α depends on the multi-path environment (typically, $2 \leq \alpha \leq 4$). We only take into consideration the transmission power required to stay connected to another node and ignore all other energy such as the energy required to physically traverse a path and the overhead necessary to transfer communication between different stationary nodes, although these other sources of energy can be incorporated into the algorithm we introduce in Chapter 4.

Hence, by "minimum energy," we mean the minimum communication energy re-

quired and by "minimum energy path," we refer to the path that minimizes the total communication energy required over the duration of the time that m travels from s to d. The mobile node may communicate with many nodes throughout its path, but at any given time, m communicates with one node, called n^*. The optimal path P is therefore one that minimizes,

$$E = \int_P d^{\alpha}_{mn^*}(t)dt \tag{2.1}$$

where $d^{\alpha}_{mn^*}(t)$ is the Euclidean distance between m and n^* at time t.

2.2 Assumptions

We assume m travels at a constant velocity v, the positions of n_i are known a priori for all i, and that each node is a point in two-dimensional space. The entire problem area is open to travel and there are no obstacles or forbidden regions, although these can be introduced by easily modifying the approach we introduce in Chapter 4. We assume that α remains the same throughout the duration of m's travel from s to d. As m moves, it switches communicating with one node n_i to another node n_j because it becomes more efficient to communicate with n_j. We refer to this node that m communicates with at a given time, n^*, so $n^* = n_i$ when m is communicating with n_i and $n^* = n_j$ when n_j becomes closer and m switches communication. The mobile node m also adjusts its transmit power to the minimum power required to stay connected with n^* to conserve its energy. The mobile node m seamlessly transfers communicating with n_i to communicating with n_j as it moves and adjusts its transmission power accordingly. Finally, there is no limit on m's transmission range; in other words, there is always a node n_i that is within m's transmission range.

Figure 2-1 shows m at a given time along a sample path from s to d with multiple communication nodes scattered throughout the problem space. At the time shown, m communicates with n_6 because it is the closest node.

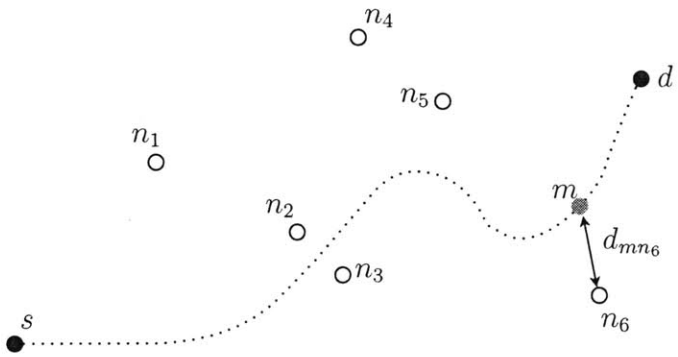


Figure 2-1: Sample path from s to d with multiple communication nodes. At the time shown, $n^* = n_6$ because n_6 is the closest node to m.

2.3 Properties and Characterization

Any problem with an arbitrary number of nodes and its minimum energy path can be scaled to fit in any area. The analysis in subsequent chapters is based on examining problems of specific sizes, but the results are general and can be scaled to fit other problems.

This minimum energy path planning problem is symmetric in that the minimum energy path from s to d is equivalent to the minimum energy path from d to s. The communication energy required at a given time depends only on the position of m, and thus the total communication energy over a path depends on the aggregate positions of the m, regardless of the direction that m moves in. Hence, s and d are interchangeable, and any path that we refer to in this thesis is reversible.

To better understand the nature of the problem, we first examine a one node path planning problem where $|N| = 1$. Thus, the total energy of a path is defined by the distance of m to a single communication node as m travels from source to destination. In this case, n^* does not changes as m moves, and the minimum energy path P is one that minimizes Equation (2.1).

There are two factors that contribute to the communication energy spent over a given path: total path length and proximity of the path to n^*. We can reduce the

energy of the path by: 1) minimizing the total path length, 2) moving m closer to n^*. These two objectives conflict. Moving closer to the communication node increases the path length. On the other hand, decreasing the path length by traveling more directly from the source to the destination does not bring m closer to the communication node. Figure 2-2 shows a one node problem with three sample paths that illustrates these two effects. On the one extreme, path A goes directly from s to d without regard to n^*. It minimizes the time spent traveling, but does not get close to the stationary node to reduce its communication energy. At the other extreme is path C that goes directly to n^* en route to d, a path that is long but gets the closest to n^*. The compromise is path B, which curves towards n^*, balancing the energy savings of traveling closer to n^* with the time spent traveling.

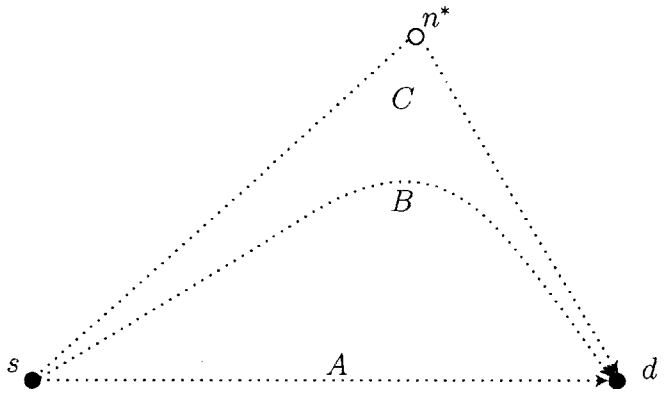


Figure 2-2: Three sample paths from s to d with one communication node

If $d = n^*$, these two objectives do not conflict, but it is possible to both get close to n^* and travel on a short path.

Claim 1 *The most energy efficient path between an arbitrary point p and n^* is a straight path between p and n^*.*

Proof The straight $p - n^*$ path achieves both objectives that minimize total path energy: it is the shortest path between p and n^* and it travels as close to n^* as possible at all times. As shown in Figure 2-3, all other possible paths are longer and do not

come closer than the straight path to n^* at a given time along the path. Thus, the straight $p - n^*$ path is optimal and all other paths are suboptimal.

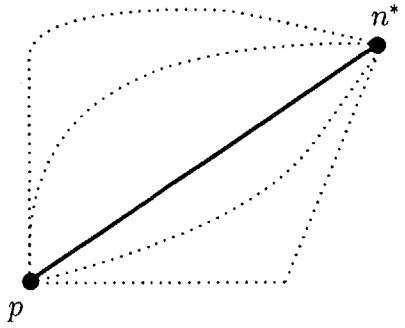


Figure 2-3: The most energy efficient path between an arbitrary point p and n^* is a straight path

Consequently, we have the following corollary.

Corollary 1 *Any path from s to d is suboptimal if n^* is along the path and the path is not made of two straight segments from s to n^* and from n^* to d.*

Proof If n^* is along the path from s to d, then the path can be broken into two segments, $s - n^*$ and $n^* - d$. According to Claim 1, the most energy efficient paths between s and n^* and n^* and d are straight paths. If neither $s - n^*$ or $n^* - d$ segment is a straight path, then part of the entire $s - d$ path is suboptimal and the entire path is suboptimal.

2.3.1 Problem Space

Because m always communicates with the closest communication node n^*, we can think of the problem as a decomposition of the problem space into regions where m communicates with the node in the region it is located in. This partitioning is a Voronoi diagram of the communication nodes N. A Voronoi diagram of N, $Vor(N)$, decomposes the space into $|N|$ regions, one for each $n_i \in N$, such that a point q lies in

the region corresponding to n_i iff $d_{qn_i} < d_{qn_j}$ for each $n_j \in N$ and $j \neq i$ [3]. Figure 2-4 shows the Voronoi diagram of N for a sample problem where $|N| = 6$. The Voronoi region containing any point p is $V(p)$, and the associated region of each stationary node n_i is $V(n_i)$. Because m communicates with the closest node n^* at a given time, n^* is the node associated with the Voronoi region that m is located in. Formally, if $V(m) = V(n_i)$, then $n^* = n_i$ and m communicates with n_i.

If the problem only has one communication node, the Voronoi diagram is simply the entire problem space since m communicates with the same node at all times.

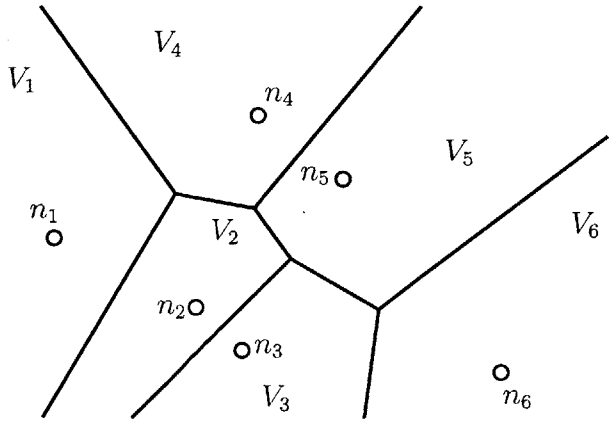


Figure 2-4: Voronoi diagram of N.

Chapter 3

Recursive Approach

In this chapter we describe a recursive approach, an initial step to gain insight into the minimum energy path planning problem. This approach finds a point along an energy efficient path and uses this point to divide the problem space in half. The algorithm recursively divides the problem space and finds points along a path based on previously determined points. The resulting path is defined by the points joined by straight segments.

The rest of this chapter describes the recursive algorithm in detail. In Section 3.3 we discuss weaknesses of this algorithm: how it can find suboptimal paths and how it is difficult to find energy efficient paths when there are multiple communication nodes. For the rest of this chapter, we analyze one node problems where $|N| = 1$ and $\alpha = 2$. Section 3.4 summarizes the key insights we gain by analyzing the problem using the recursive approach.

3.1 Algorithm

Without loss of generality, the problem can be fitted onto a Cartesian plane with the source at the origin and the destination on the x-axis such that $s = (0, 0)$ and $d = (d_x, 0)$. As shown in Figure 3-1a, the algorithm first considers two straight segments, from s to some point $c = (c_x, c_y)$ on the perpendicular bisector of $s - t$, and from c to

d. Equation (2.1) then becomes,

$$E(\theta) = \int_{t_0}^{t_m} (n_x^* - vt\cos(\theta))^2 + (n_y^* - vt\sin(\theta))^2 dt + \int_{t_m}^{t_f} (n_x^* - vt\cos(\theta))^2 + (n_y^* - vt\sin(\theta))^2 dt \quad (3.1)$$

where $t_0 = 0$, $t_m = \frac{c_x}{v\cos(\theta)}$, $t_f = \frac{d_x}{v\cos(\theta)}$, and (n_x^*, n_y^*) represents the coordinates of the stationary communication node n^*. The angle θ, as shown in Figure 3-1a is the angle with respect to the $s-d$ line at which the mobile node m must travel to reach c. The times t_0, t_m, and t_f represent when m, traveling at a constant velocity v, is at s, c and d, respectively.

We minimize Equation (3.1) with respect to θ and find the corresponding $c = (\frac{d_x}{2}, \frac{d_x}{2}\tan\theta)$ that minimizes the energy cost associated with traveling in straight segments from s to c and from c to d.

Having determined c, we then divide the problem into two different one node problems with $d_{left} = c$ in the left subproblem and $s_{right} = c$ in the right subproblem. Recursively applying Equation (3.1) to find points and joining them gives an approximate minimum energy path from source to destination. Figure 3-1 shows how the algorithm finds points at each recursion level up to a depth of three.

3.2 One Communication Node Problem

3.2.1 Equidistant Communication Node

We first use the recursive algorithm to analyze one communication node problems where $|N| = 1$ and n^* is equidistant from s and d (i.e. $n_x^* = \frac{d_x}{2}$). Because n^* lies on the $s-d$ perpendicular bisector, the path from s to c and from c to d is symmetric. For the first step in the recursion, Equation (3.1) reduces to

$$E(\theta) = 2\int_{t_0}^{t_f} (n_x^* - vt\cos(\theta))^2 + (n_y^* - vt\sin(\theta))^2 dt \quad (3.2)$$

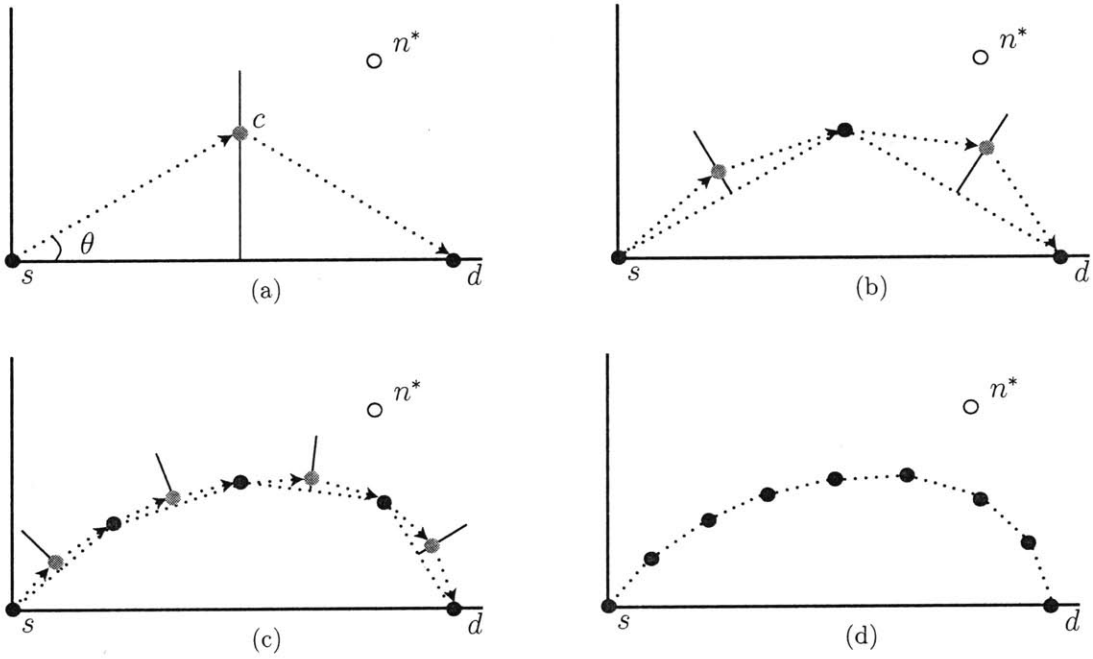


Figure 3-1: Recursive algorithm for the one communication node problem. (a) shows how the algorithm finds c on the $s - d$ perpendicular bisector. Two recursions are shown in (b) and (c) where the black points represent points previously found and gray points represent points found during that recursion. (d) shows the approximate path after three levels of recursion.

where $t_f = \frac{c_x}{v\cos(\theta)}$ and $c_x = \frac{d_x}{2}$. Solving for $\frac{dE(\theta)}{d(\theta)} = 0$, we find that $E(\theta)$ is minimized when,

$$\theta = \begin{cases} \tan^{-1}(\frac{n_y^*}{n_x^*}), & \frac{n_y^*}{n_x^*} \leq 2.1 \\ \frac{1}{2}\sin^{-1}(\frac{2n_x^*}{n_y^*}), & \frac{n_y^*}{n_x^*} > 2.1 \end{cases} \tag{3.3}$$

Therefore,

$$c_y = \begin{cases} n_y^*, & \frac{n_y^*}{n_x^*} \leq 2.1 \\ \frac{n_y^* - \sqrt{{n_y^*}^2 - (2n_x^*)^2}}{2}, & \frac{n_y^*}{n_x^*} > 2.1 \end{cases} \tag{3.4}$$

The first solution represents the path that travels through n^* from s to d while the second solution represents the path that goes more directly from s to d as $\frac{n_y^*}{n_x^*}$ increases (note that $\frac{1}{2}\sin^{-1}(\frac{2n_x^*}{n_y^*})$ is undefined for $\frac{n_y^*}{n_x^*} < 2$). Figure 3-2 shows the optimal position of c during the first recursion as a function of n^*'s distance from the $s-d$ line. There is a clear disconnect between the two solutions. For $\frac{n_y^*}{n_x^*} \leq 2.1$ the optimal point c found in the first recursion is n^* because the objective of traveling close to n^* dominates energy considerations. For $\frac{n_y^*}{n_x^*} > 2.1$, the objective of minimizing path length dominates energy considerations and c tends towards $s-d$ and away from n^* as n^* become far from s and d.

Figure 3-3 shows two examples of the path found when $\frac{n_y^*}{n_x^*} \leq 2.1$ where the c found during the first recursion is the same as n^*. According to Claim 1, the minimum energy paths from s to n^* and from n^* to d are straight paths. Indeed, the recursive algorithm finds such straight paths as shown in the figure.

Figure 3-4 shows two examples of the curved paths found for when $\frac{n_y^*}{n_x^*} > 2.1$. During the first recursion in each of the examples, $c_y < n_y^*$. Furthermore, c_y decreases as n_y increases.

We find that the minimum energy path found by this algorithm depends on how far n^* deviates from straight $s-d$ line. As discussed in Section 2.3, the two elements of a path that determine its total energy is the total length and proximity to the communication node. If n^* is close to s and d, as in Figure 3-3, then m can travel close

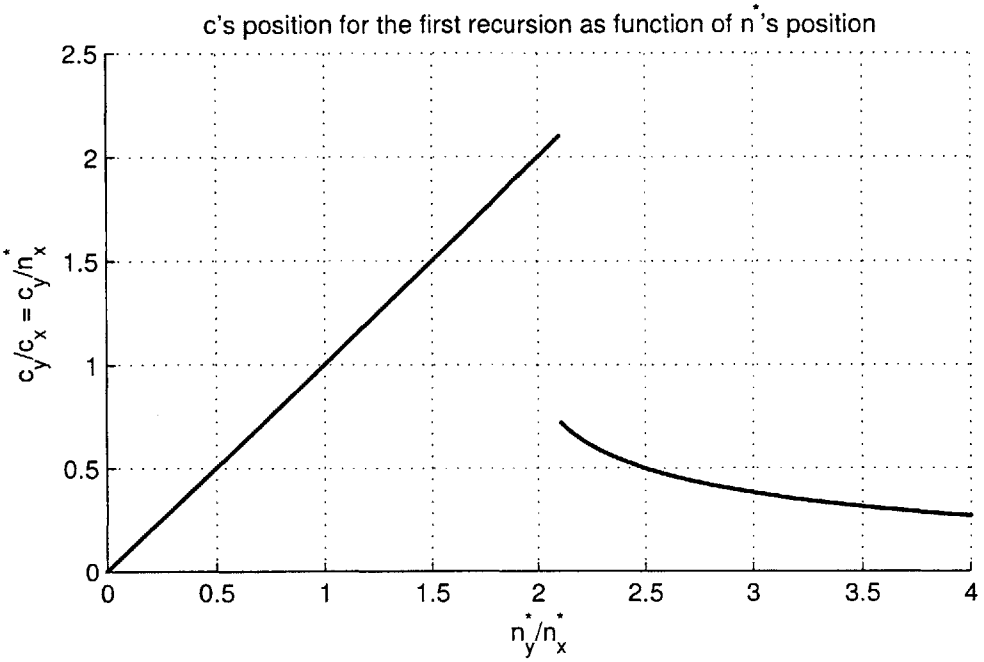


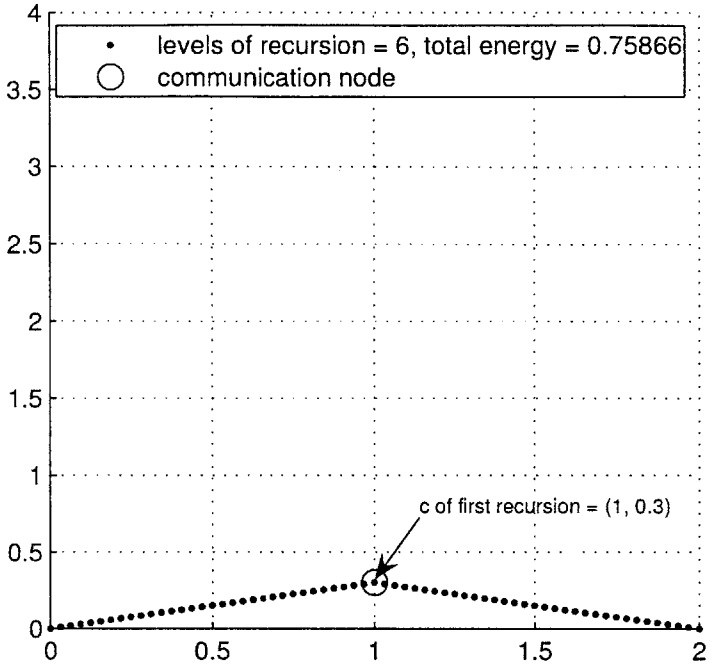
Figure 3-2: Position of c for the first recursion as function of position of n^*.

to n^* without drastically increasing its path length. Thus it is more energy efficient to travel directly from s to d through n^*. On the other hand, if n^* is far from s and d, as in Figure 3-4, m will spend significantly more time traveling if it travels close to n^*. In this case, a more direct path from s to d is more energy efficient.

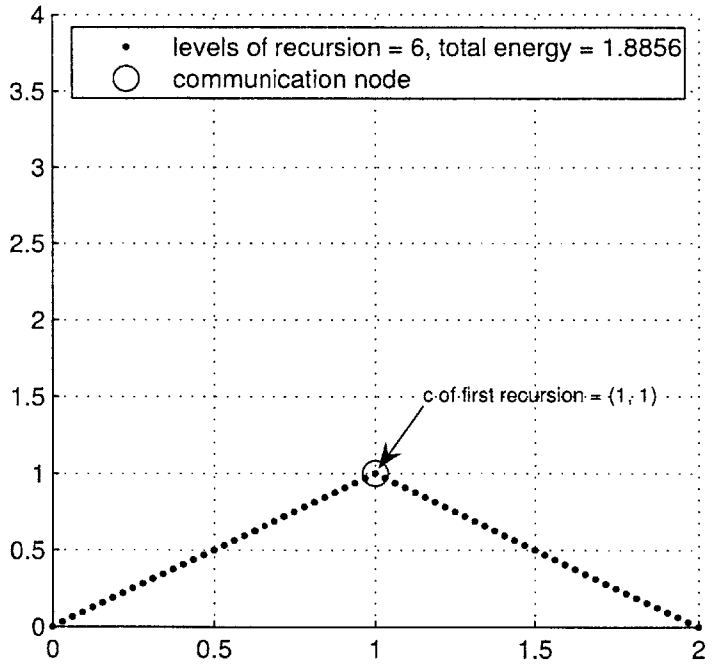
3.2.2 Recursion Depth

If the approximate path found by this algorithm is a curve, increasing the recursion depth allows the path to better approximate the curved path because the approximate path is defined by more points. Thus, the path energy decreases if we increase the recursion depth. However, the algorithm's runtime is $O(2^r)$, where r is the recursion depth, and computing a path defined by many points may be very time consuming. In this section we examine how the total energy of the path decreases as the recursion depth increases, and determine the appropriate depth that yields a good approximate energy efficient path in acceptable computation time.

Figure 3-5 shows how the energy of the path found by the recursive algorithm

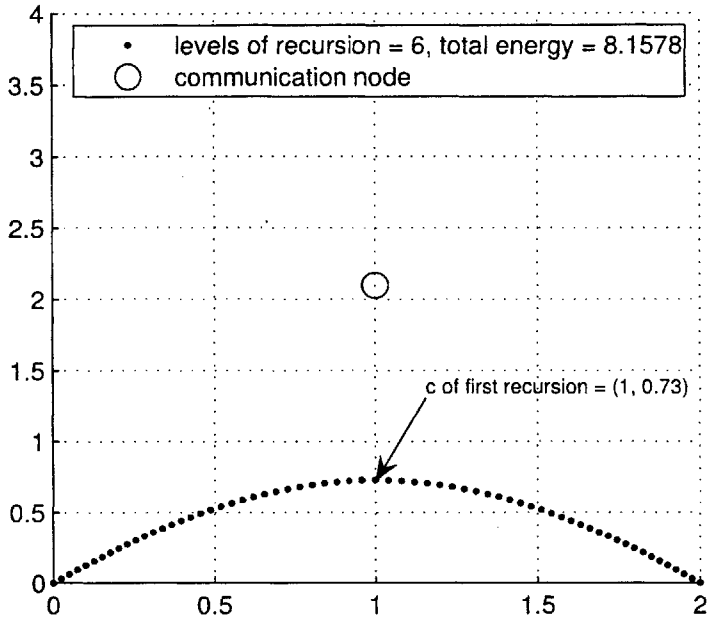


(a)

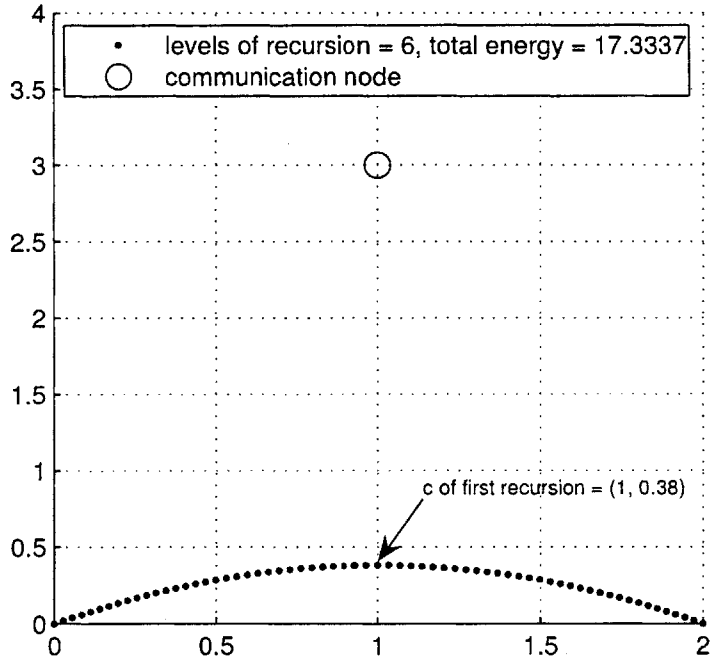


(b)

Figure 3-3: Paths found by recursive algorithm for equidistant node problems where $\frac{n_y^*}{n_x^*} \leq 2.1$.



(a)



(b)

Figure 3-4: Paths found by recursive algorithm for equidistant node problems where $\frac{n_y^*}{n_x^*} > 2.1$.

decreases as recursion depth increases in an example equidistant node problem where $\frac{n_y^*}{n_x^*} = 2.4$. There is an initial 2% decrease in energy as the algorithm depth increases from one to two. The decrease in energy as the depth further increases becomes negligible and the advantage of defining the path by more points significantly diminishes as the total energy of the path converges.

Figure 3-6 shows the total energy of the path found by the algorithm as a function of recursion depth for various positions of the equidistant communication node. As before in Figure 3-5, the total energy of the path initially decreases as the depth increases from one to two but does not significantly decrease as recursion depth increases further. Moreover, as the communication node's distance from s and d increases, the change in energy over different recursion depths also decreases. Because n^* is far from the mobile node and m requires so much energy to communicate with n^*, increasing the number of points that defines the path does not significantly change the the total path energy. Note that for $\frac{n_y^*}{n_x^*} \leq 2.1$, there is no effect of recursion depth because the path consists of two straight segments and the algorithm finds the path on the first recursion.

For the equidistant node problem, it is therefore not worth the exponential computation time to find paths defined by points found by the algorithm beyond a certain depth because further refinements do not significantly reduce the total communication energy. For the examples given in this chapter, we use a recursion depth of six to obtain a clearly defined path in acceptable computation time.

3.2.3 General Communication Node

We now remove the constraint of the equidistant communication node and examine paths found for problems where n^* is unconfined with $\alpha = 2$ as before. We use Equation (3.1) to recursively find points along an energy efficient path, and show two sample paths in Figure 3-7. As described in Sections 2.3 and 3.2.1, the paths this algorithm finds depend on the total path length and the proximity to n^*. Figure 3-7a shows a path that tends toward n^* when the communication node is close to s and d in order to reduce the communication energy. If n^* is farther, as in Figure 3-7b,

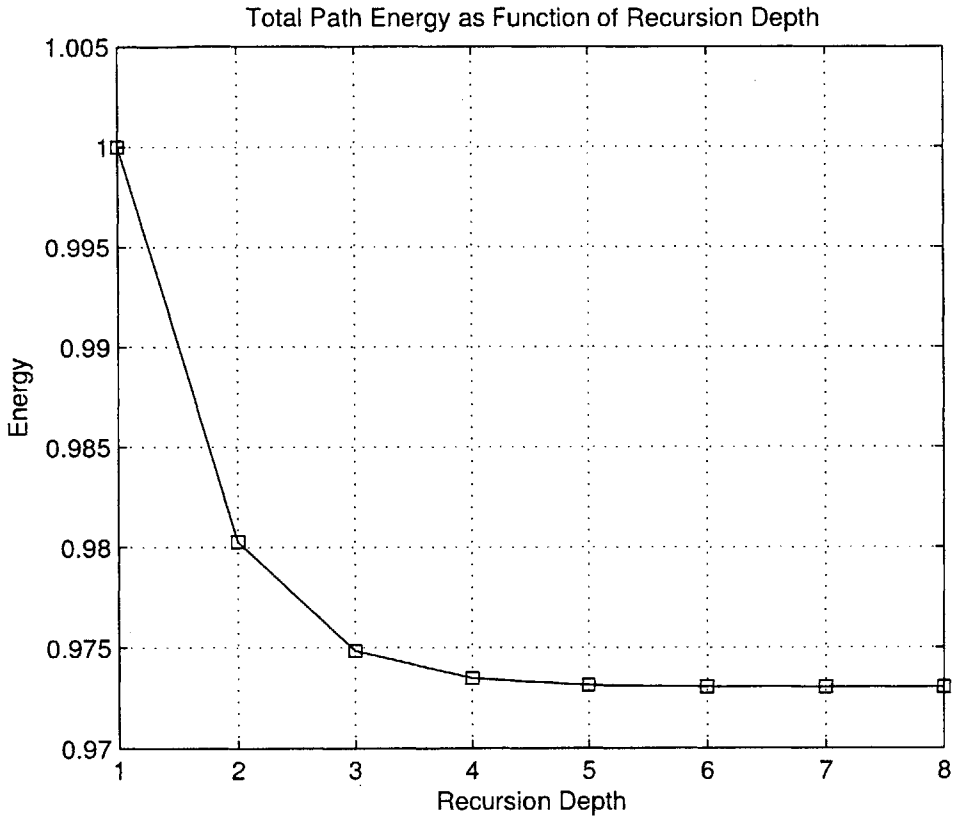


Figure 3-5: Energy as function of recursion depth for $s = (0,0)$, $d = (2,0)$, and $n^* = (1, 2.4)$. Energy is normalized by the energy of the path found by the recursive algorithm of depth 1.

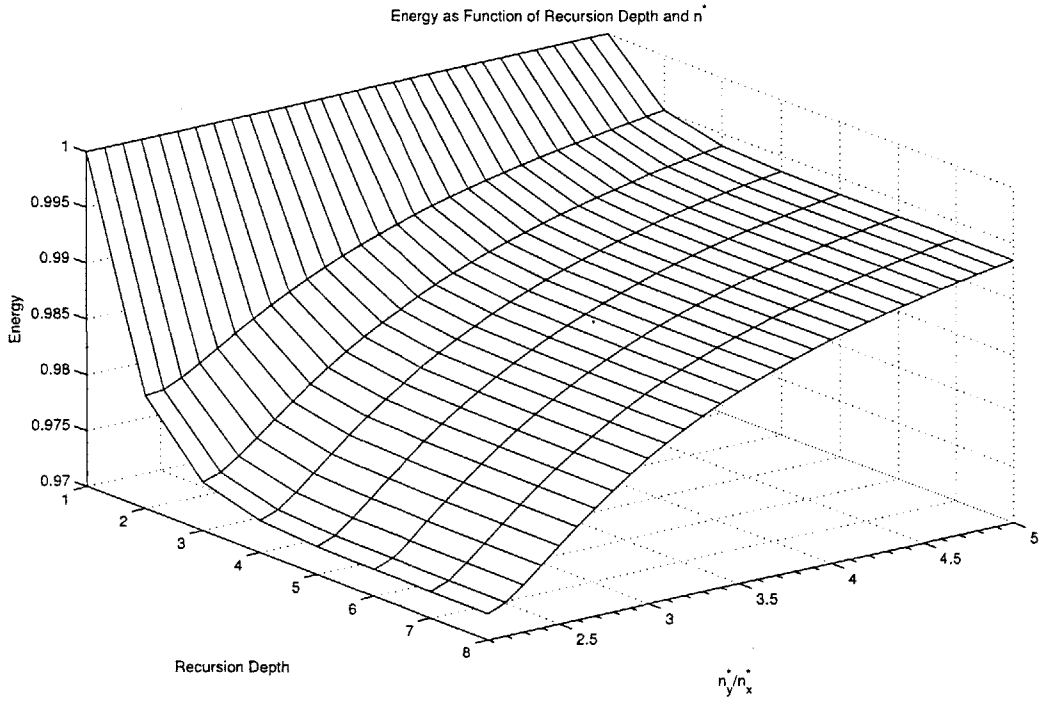


Figure 3-6: Energy as function of recursion depth and position of n^* for the problem where $s = (0, 0)$ and $d = (2, 0)$. For each n^* position, energy is normalized by the energy of the path found by the recursive algorithm at depth 1.

the resulting energy efficient path goes more directly from source to destination to minimize the total distance traveled.

3.3 Shortcomings

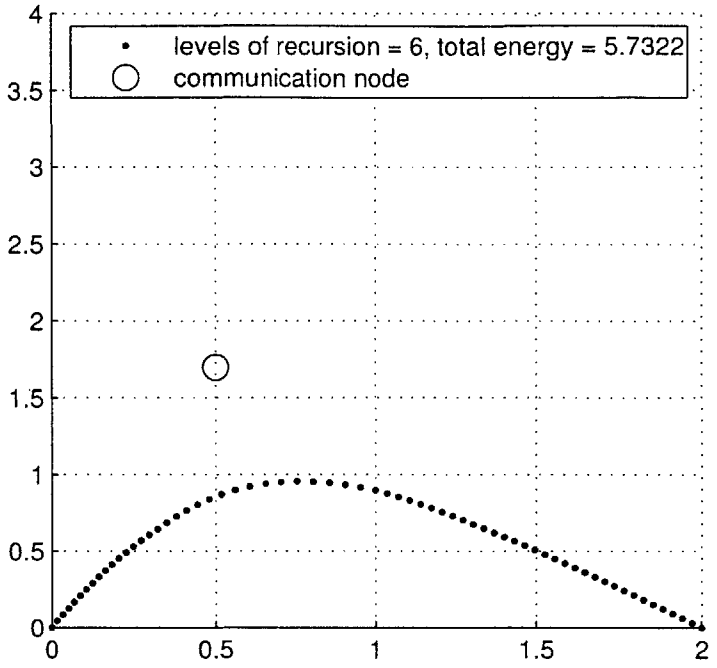
We use the recursive algorithm as an initial analytic tool to gain understanding of the problem. In this section, we discuss two main shortcomings of this algorithm, how it is difficult to use this algorithm to solve problems with multiple communication nodes and how this recursive approach finds suboptimal paths.

3.3.1 Multiple Communication Nodes

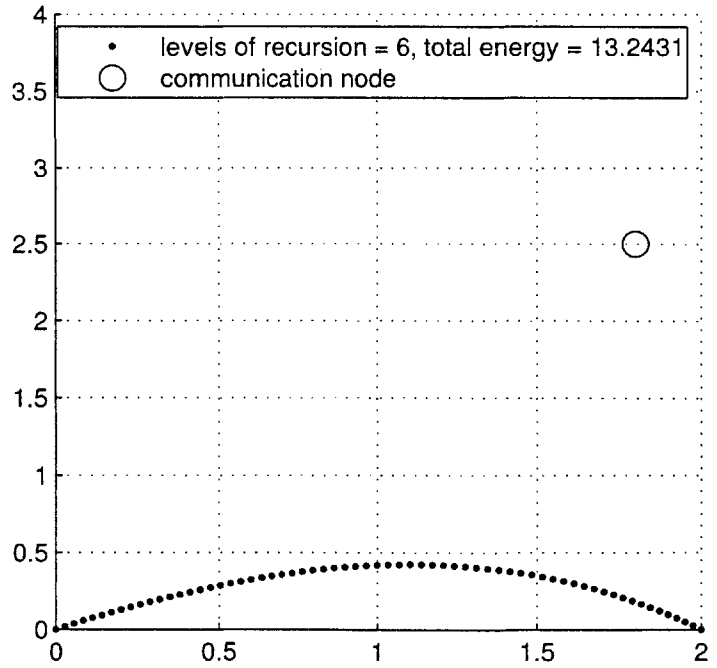
The recursive algorithm is not computationally efficient for problems with multiple communication nodes. If $|N| > 1$, the algorithm must consider all nodes that m communicates with when examining the energies of the straight segments during each recursion. Figure 3-8 shows a problem with multiple nodes and the Voronoi regions associated with each node. In the first recursive step shown, the algorithm must find the optimal point c on the $s-d$ perpendicular bisector. The figure shows one such $s-c-d$ path, made of the $s-c$ segment and the $c-d$ segment, and the Voronoi regions it crosses through. The $s-c-d$ path traverses through five different Voronoi regions, V_1, V_2, V_4, V_5 and V_6, and the total energy of the $s-c-d$ path is the sum of the energy used in communicating with each of the associated nodes, n_1, n_2, n_4, n_5, and n_6, for a portion of the path. The energy of traveling from s to d through c that the algorithm must minimize is thus,

$$E(\theta) = \sum_{V} \int_{t_{V_i}} (n_x^* - vt\cos(\theta))^2 + (n_y^* - vt\sin(\theta))^2 dt \tag{3.5}$$

where t_{V_i} is the time spent in each Voronoi region V_i that m must travel through from s to d. The number of Voronoi regions $|V|$ in the energy calculations for a given $s-c-d$ path is $O(|N|)$. To find the optimal point c during each recursion, the algorithm must



(a)



(b)

Figure 3-7: Paths found by recursive algorithm for general one communication node problems.

consider all possible $s-c-d$ paths that cross through different sets of Voronoi regions. There are $O(2^{|N|})$ $s-c-d$ paths that travel through different sets of Voronoi regions for the various potential c's so calculating the c that minimizes the energy of traveling from s to d for *each recursion* can take $O(|N|2^{|N|})$. Finding a path using the recursive algorithm of depth r therefore takes $O(|N|2^{|N|+r})$ time. Clearly, this approach can be very time consuming if there are many communication nodes and is not practical for solving multiple node problems.

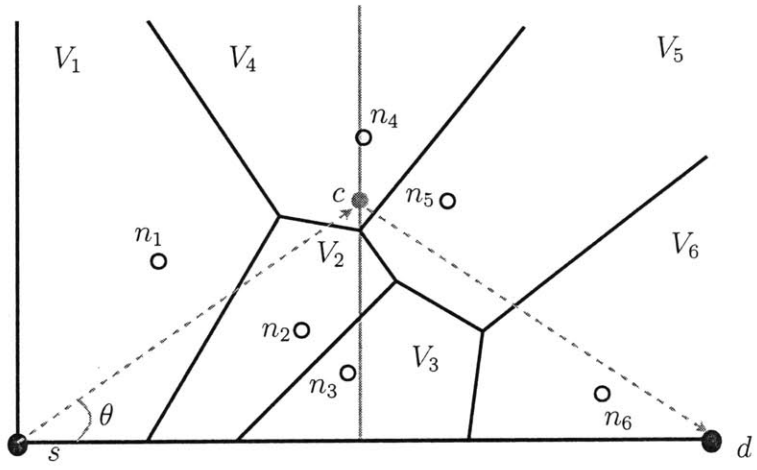


Figure 3-8: Calculating c for the first recursion in a multiple node problem. Each region V_i represents the Voronoi region associated with node n_i. A sample $s-c-d$ path traverses through five different Voronoi regions.

3.3.2 Suboptimal Paths

The recursive algorithm may find paths that are clearly not optimal. At each recursion level, it finds the local optimal solution by assuming the path from one point to another is straight. For certain one communication node problems, the approximate path found using the recursive approach is close to optimal. The recursive algorithm finds good paths when the communication node is equidistant or far from s and d. However, in other cases, the recursive approach finds paths that are clearly suboptimal.

This is best illustrated by an example, seen in Figure 3-9. As we argue in Corollary 1, the path seen in the figure is clearly suboptimal because the path goes through n^* but is not made of straight paths $s - n^*$ and $n^* - d$. The suboptimal path found by the algorithm is both longer than the direct $s - n^* - d$ path and does not get closer to the communication node at any point along the path than the direct $s - n^* - d$ path.

The algorithm chooses the suboptimal path because during the first recursion it finds the point c by only considering the straight path segments from s to c and from c to d. In the example in Figure 3-9, the straight $s - c$ segment is far from the actual optimal path, which is from s to n^* to some point on the bisector. Finding c during the first recursion to be (1, 0.6) anchors that point to be part of the path. Hence, only considering the straight segments to and from the bisector fixes the path to suboptimal points which the algorithm must use in further calculations.

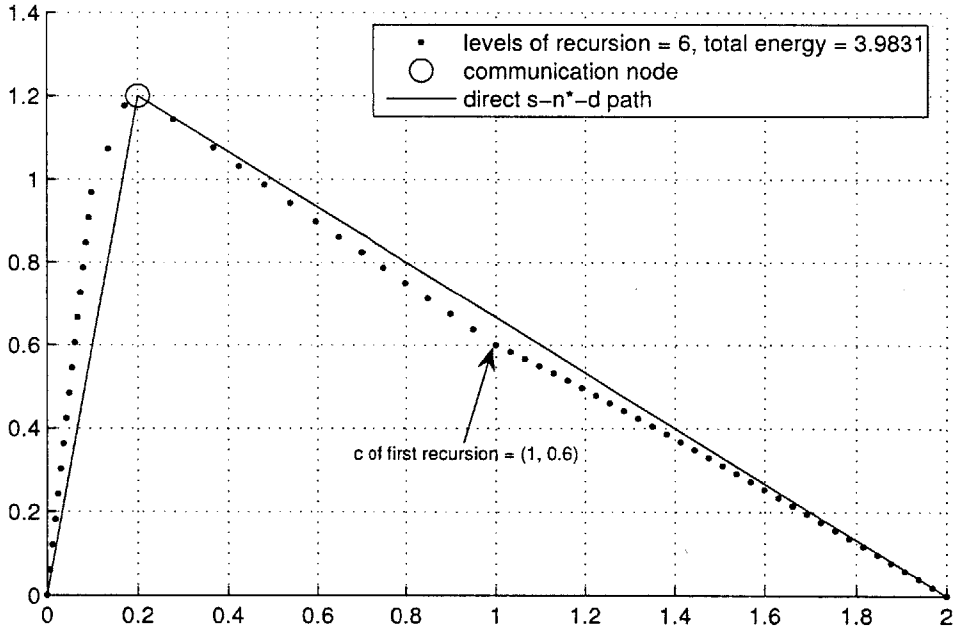


Figure 3-9: Example of suboptimal path found by recursive algorithm. The points represent the path find by the algorithm, where the solid line represents the straight path from source to destination through the communication node.

3.4 Insights

This initial recursive approach is an important step in solving the minimum energy path planning problem as we have been able to use the recursive algorithm to gain key insights.

By analyzing the equidistant node problem, we determined that the algorithm was able to find energy efficient paths by running the algorithm to a recursion depth of six. We concretely demonstrated the effect of the two opposing objectives of minimizing total path energy. If the communication node is close enough to the source and destination, it is more energy efficient for the mobile node to travel directly to the node en route to the destination. On the other hand, if the communication node is far, minimizing the path length is more important and the minimum energy path goes more directly from source to destination. We also illustrated how the algorithm finds paths for problems where the communication node is not constrained to be equidistant from s and d.

We have also understood the limitations of this algorithm. Specifically, we recognized that a divide-and-conquer approach does not yield a global optimal solution to this problem and that for the general problem consisting of multiple communication nodes, using the algorithm to recursively divide the problem space makes it difficult to find a path.

We now use these insights in a different approach where we avoid the shortcomings of the recursive approach.

Chapter 4

Discretized Graph Approach

In this chapter we describe a second approximation algorithm that is more general than the recursive approach. As in Chapter 3, we analyze one node problems where $|N| = 1$ and $\alpha = 2$ in this chapter. However, we also describe and demonstrate how this algorithm can be used to solve multiple node problems where $|N| > 1$.

We discretize the problem by finding a regular set of vertices in the problem space and connecting these vertices to form edges. We limit the path from source to destination to be made up of these edges, whose costs are the communication energies used to travel along the edges. The approximate minimum energy path is thus made up of a combination of edges from source to destination with the least cost.

Similar to the approaches used in [6], [9], we lay a square grid on top of the problem area. As illustrated in Figure 4-1, the grid points form a set of vertices V and edges connect vertices to form the set E. Each vertex can be connected to its neighboring vertices to form the graph $G = (V, E)$. A graph where each vertex is connected to c other vertices is a c-connected graph. Figure 4-1 shows a 4-connected graph where each vertex is connected to 4 neighbors while Figure 4-2 shows an 8-connected graph where each vertex has 8 neighbors.

The cost of each edge e_{ij} is E_{ij}, the energy required to travel from vertex i to vertex j. This energy can be calculated as,

$$E_{ij} = \int_{t_i}^{t_j} d_{mn^*}^{\alpha}(t)dt, \tag{4.1}$$

where t_i and t_j are the times the mobile node m is at vertex i and vertex j, respectively, and n^* is the node the m communicates with as it travels from i to j. If m is at a given vertex i, then it can travel to any of i's neighbors, $j \in neighbors(v)$, using communication energy E_{ij}. Because the problem is symmetric, the energy required to travel from i to j is equal to the energy required to travel from j to i and $E_{ij} = E_{ji}$.

We find an approximate minimum energy path by performing a shortest path search from source s to destination d on the undirected graph G. Increasing the grid granularity and graph connectivity increases the accuracy to which the approximate path fits the optimal path, but also increases the computation time required to find the path. Chapter 5 analyzes these parameters' effects on the minimum energy path found using the discretized graph approach.

In we described in the problem framework and assumptions defined in Chapter 2, we only consider the communication energy required to maintain connectivity to the network and assume that the entire problem area is open to travel. We can easily extend this discretized algorithm to ease both of these restrictions, as we discuss in Chapter 7.

We fit the grid onto the problem such that lower left vertex of the grid is located at, $(\min(s_x, d_x, n_x), \min(s_y, d_y, n_y)), \forall n \in N$.

If s or d does not lie on a vertex, the algorithm finds the closest vertex v_c, fixes the segment $s - v_c$ or $d - v_c$ be part of the minimum energy path found, and chooses the new s or d to be v_c in order to perform the shortest path search. This method introduces some error, but the added energy of fixing the path in such a way decreases as we impose a finer grid granularity.

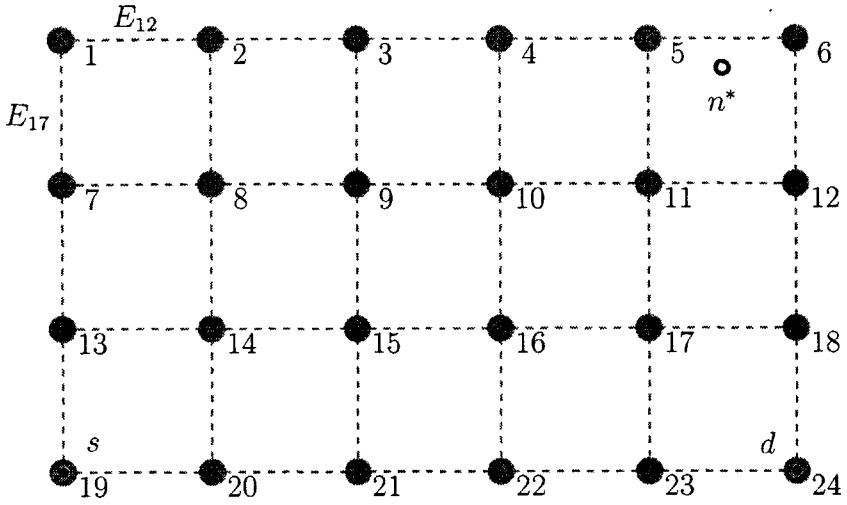


Figure 4-1: Imposing a grid on the problem area to produce a 4-connected graph

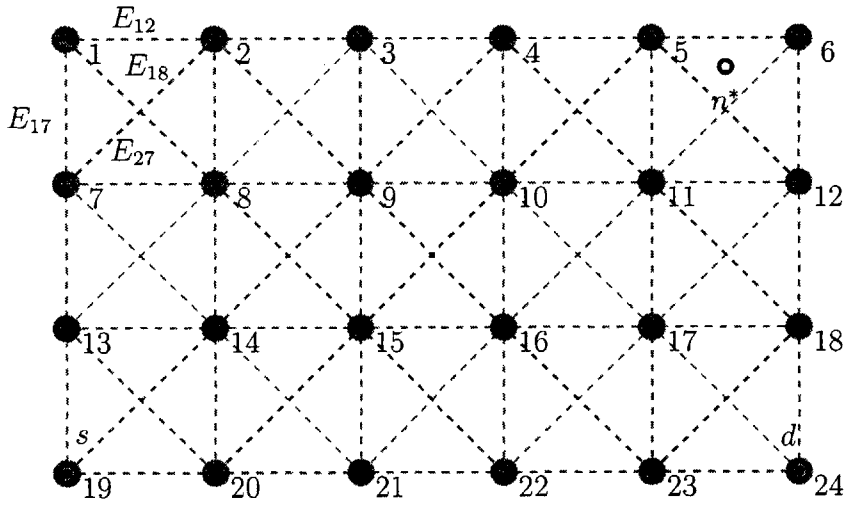


Figure 4-2: Imposing a grid on the problem area to produce an 8-connected graph

4.1 Finding the Closest Node

If there are multiple communication nodes, m communicates with the closest node n^*. For each edge, we must find n^* in Equation (4.1) in order to calculate its weight. As discussed in Section 2.3, we can decompose the problem space into the Voronoi diagram of N whereby m communicates with the node n_i associated with $V(m)$, the Voronoi region that contains m. Figure 4-3 shows the Voronoi diagram of a multiple node problem and a grid the algorithm imposes on the problem area.

For a given edge e_{ij}, it is easy to determine the correct n^* in calculating E_{ij} if both i and j lie in the same Voronoi region because Voronoi regions are convex. However, if e_{ij} crosses Voronoi regions, then we approximate the energy used in traversing the edge by determining n^* to be the node associated with the Voronoi region of the midpoint of $i-j$. Because each Voronoi edge represents all points that are equidistant from the two closest nodes in $V(i)$ and $V(j)$, all points along an edge e_{ij} that cross a Voronoi edge are approximately equidistant from the two communication nodes. The error of calculating such edge weights is small if the grid granularity is fine enough such that the distance between communication nodes is much greater than the distance between adjacent vertices in G.

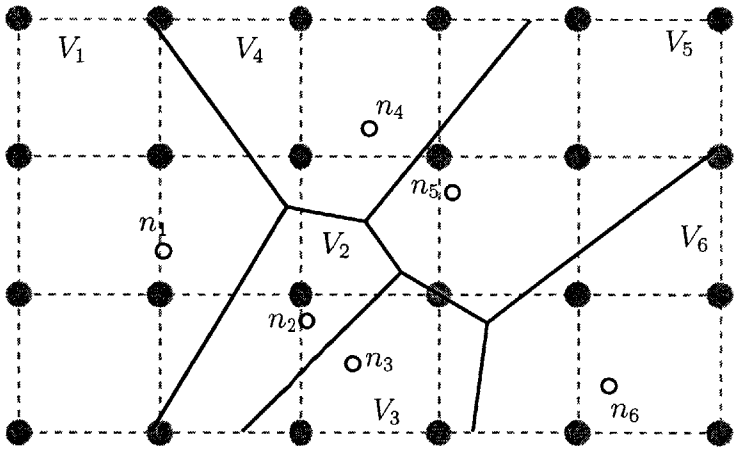


Figure 4-3: Calculating edge weights according to the Voronoi diagram of N.

4.2 Grid Granularity

We formalize the notion of a grid granularity for the problem to represent the granularity of vertices and edges with respect to the problem area. In problem granularity g, grid size and the length of the straight $s - d$ segment are related,

$$g = \frac{d_{sd}}{gridsize} \tag{4.2}$$

where *gridsize* is the length of a side of a square formed by the grid. We discuss how g affects the minimum energy path in Chapter 5. In the subsequent sections, we examine one communication node problems where $d_{sd} = 2$ and choose $gridsize = 0.01$ and $gridsize = 0.001$ for a problem granularity of $g = 200$ and $g = 2000$, respectively. In Section 4.5, we consider a multiple communication node problem using $d_{sd} = 1$ and $gridsize = 0.01$ for a $g = 100$.

4.3 Shortest Path Algorithm

Given a graph $G = (V, E)$, we perform a shortest path search from s to d using an implementation of Dijkstra's algorithm.

Shortest path algorithms have the optimality substructure property, such that the shortest path between two vertices in G contains other shortest paths within it [2]. Formally, if $p_{1k} =< v_1, v_2, ..., v_k >$ is the shortest path from vertex v_1 to vertex v_k, then $p_{ij} =< v_i, v_{i+1}, ..., v_j >$ is the shortest path from v_i to v_j, where $1 \leq i \leq j \leq k$.

Our approach uses SPLIB's [5] implementation of Dijkstra's algorithm using a k-ary heap that runs in $O(E \log V)$ time for $k = 3$ [1].

We choose this implementation of a shortest path algorithm because it is a popular and widely-known version. However, the majority of the computation time of the overall algorithm is in calculating the weights of E and thus implementing the fastest shortest path search is not a priority. In practice, the implementation we use performs very well for the graphs in this problem: finding the shortest path of 48,010,000 edges

between 6,005,000 nodes takes less than four minutes. [1]

4.4 Graph Connectivity

The number of edges at any vertex determines the degree of freedom m has in path selection. Increasing the connectivity of G increases the number of possible paths from s to d, but also increases the time needed to compute all the edge weights of E.

In the subsequent sections we discuss how the minimum energy path found using this approach differs as graph connectivity c increases. We first discuss how the algorithm performs when each vertex is connected to four and eight neighbors. We then generalize the graph we construct to provide more directional freedom for the path by allowing a vertex to be connected to more neighbors in its immediate neighborhood. If the algorithm only considers four or eight neighbors for each vertex, G has connectivity $c = 4$ and $c = 8$, respectively, and the angles of the edges are evenly spaced. For $c > 8$, the edges angles at each vertex are no longer evenly distributed, and increasing connectivity does not allow uniform directional movement for m, as we discuss in depth in Section 4.4.3.

4.4.1 4-Connectivity

We first consider a graph G where each vertex is connected to its four immediate neighbors, as shown in Figure 4-1. With $c = 4$, each edge is at an angle $\tau = 2\pi j/c$ where the angle $\tau = 0$ is in the direction of the positive x axis and $j = 0, ..., c-1$. For a 4-connected graph, the angles of the edges connected to a vertex are regularly spaced as shown in Figure 4-4.

We apply Dijkstra's algorithm to G to find the minimum energy path. Figure 4-5 shows the two different paths found for the problem where $s = (0,0)$, $d = (2,0)$, and $n^* = (1,1)$,. Figure 4-5a shows the path found for $g = 200$ while Figure 4-

[1] We ran our simulations on a Dell OptiPlex GX270 workstation with a Pentium 4, 3.2GHz processor and 1GB of RAM running Windows XP.

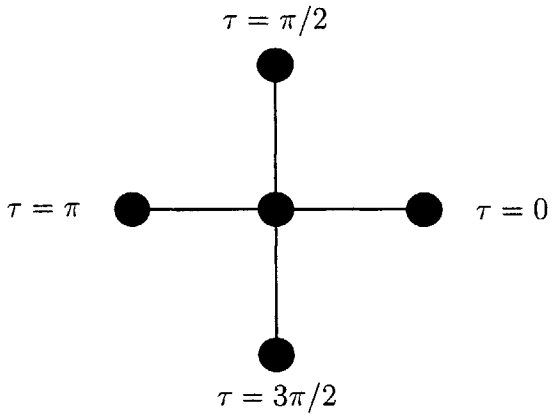
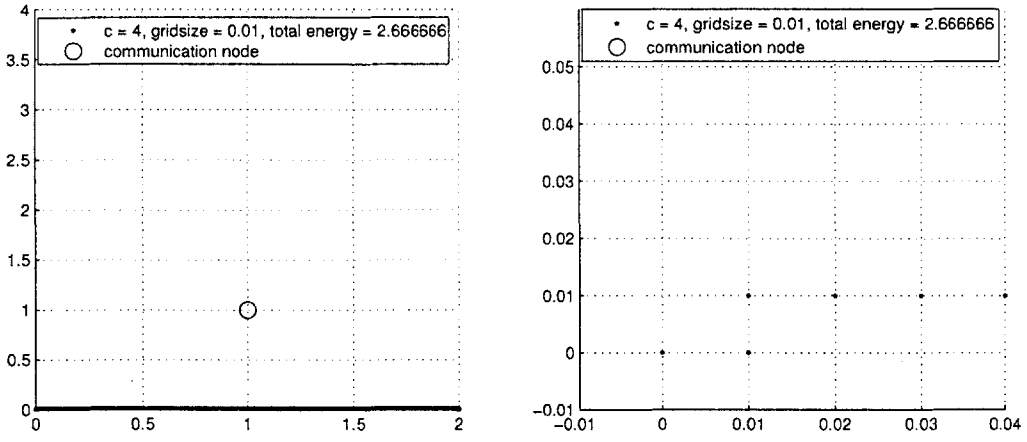
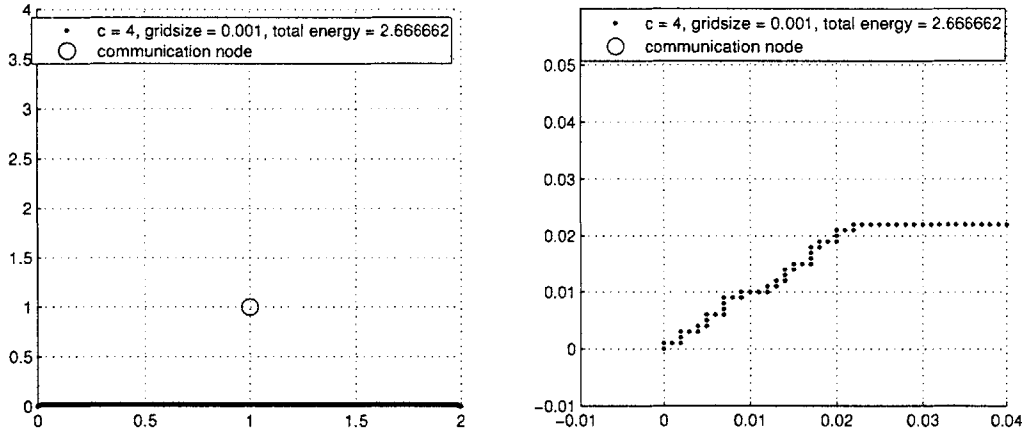


Figure 4-4: A vertex and its neighbors in a 4-connected graph

5b shows the path that has a lower energy for a higher granularity $g = 2000$. The path found for $g = 2000$ initially consists of a series of 90-degree turns towards n^* before staying straight towards d. The path found using problem granularity $g = 200$ only tends toward n^* slightly in two 90-degree turns. The energy costs of both paths are essentially the same since neither path strays far from the straight $s - d$ path. Although the energies of the two paths are very close, Figure 4-5 demonstrates that increasing the granularity can change the minimum energy path found. We examine the effect that problem granularity has on the path found and the associated energy in Chapter 5. The paths shown in Figure 4-5 found using the discretized 4-connected graph algorithm are suboptimal because they have higher total energy than the path found for the same problem using the recursive algorithm which was a direct $s - n^* - d$ path, shown in Figure 3-3b. A 4-connected graph does not naturally allow the direct $s - n^* - d$ path of energy 1.8856 shown in Figure 3-3b because such a path moves at $\tau = \pi/4$ and $\tau = 7\pi/4$ and a 4-connected graph constrains the path to vertical or horizontal movement. Although a 4-connected graph can approximate $\tau = \pi/4$ and $\tau = 7\pi/4$, the additional distance of the approximation introduces and error known as "digitization bias" that we explain in depth in Section 4.4.2.



(a)



(b)

Figure 4-5: Paths found using the discretized 4-connected graph algorithm for (a) $g = 200$ and (b) $g = 2000$. The graphs of the paths on the right magnify parts of the paths of the graph on the left.

4.4.2 8-Connectivity

To allow for diagonal movement between vertices, we consider an 8-connected graph for the discretized graph approach. Similar to the 4-connected graph, we connect each vertex to its eight immediate neighbors. The angles of the resulting edges are uniformly distributed. For $c = 8$, each edge is at an angle $\tau = 2\pi j/c$ where $j = 1, ..., c-1$, as shown in Figure 4-6. Whereas a 4-connected graph restricts the path to 90-degree turns, using an 8-connected graph allows the path to travel in eight directions in multiples of 45-degree angles.

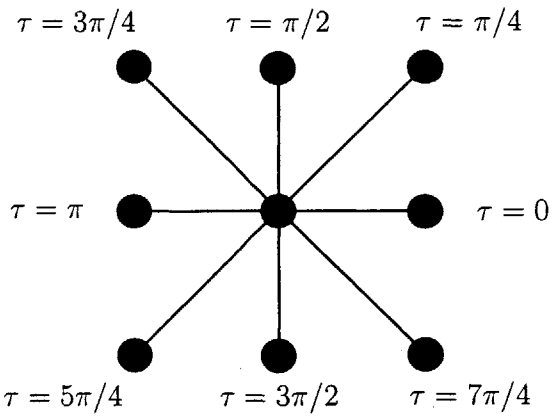


Figure 4-6: A vertex and its neighbors in an 8-connected graph

We again examine the problem where $s = (0, 0)$, $d = (2, 0)$, and $n^* = (1, 1)$. The paths previously found using the recursive algorithm and the discretized 4-connected graph algorithm are shown in Figures 3-3b and 4-5, respectively. The optimal direct $s-n^*-d$ path has movement in angles $\tau = \pi/4$ or $\tau = 7\pi/4$. Whereas this movement is not feasible for a 4-connected graph, it is feasible within an 8-connected graph, as Figure 4-7 demonstrates. While the recursive and discretized 8-connected grid algorithms both find $s - n^* - d$ to be the minimum energy path for $n^* = (1, 1)$, the discretized 4-connected grid algorithm finds a path of higher energy because it is restricted to move in angles of 90-degrees multiples.

For the problem where $n^* = (1, 0.3)$, the recursive approach finds a more energy

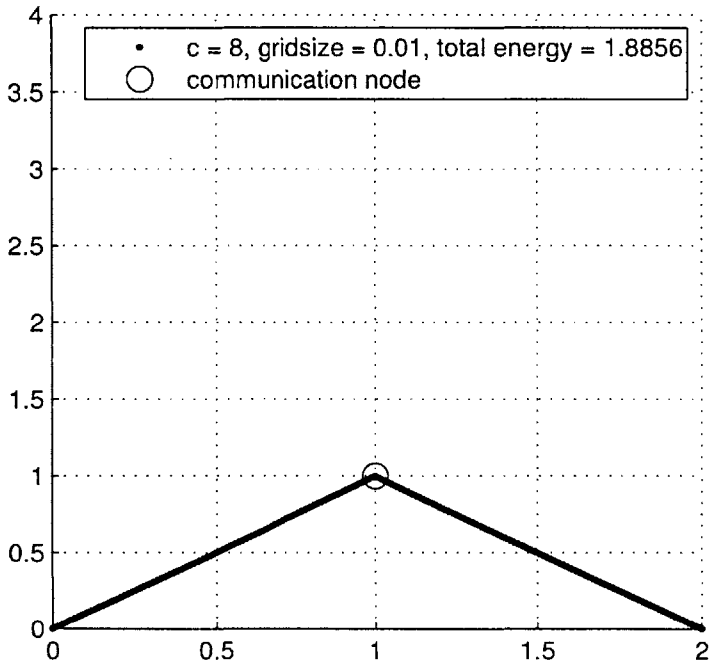


Figure 4-7: Path found using discretized 8-connected graph algorithm where $s = (0, 0)$, $d = (2, 0)$, and $n^* = (1, 1)$.

efficient path than the path found by the discretized 8-connected grid algorithm, as seen in Figure 3-3a for the recursive algorithm and Figure 4-8 for the discretized algorithm. The path shown in Figure 4-8 is clearly suboptimal because it does not travel from s to n^* and from n^* to d in straight segments. As Corollary 1 argues, any path that goes through n^* but is not made of two straight segments is suboptimal. Similar to the case with the 4-connected graph, the direct $s - n^* - d$ path travels at angles that the 8-connected grid does not naturally allow.

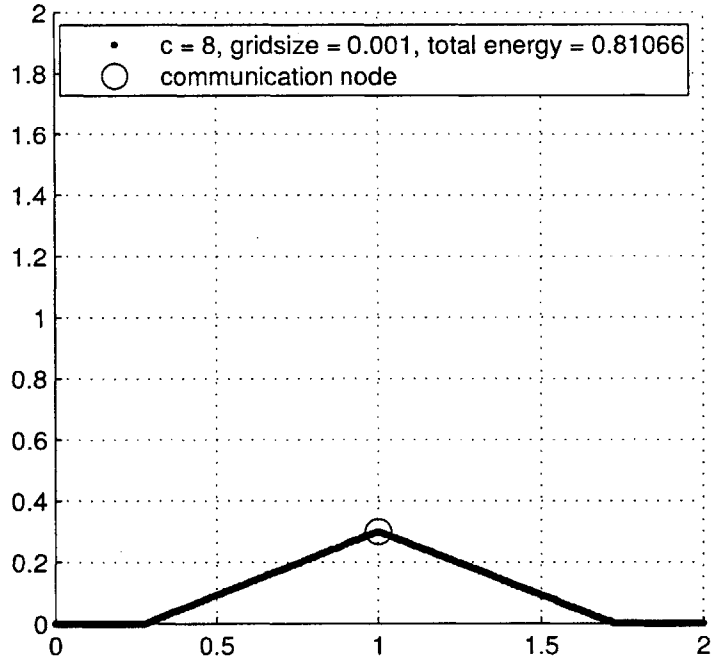
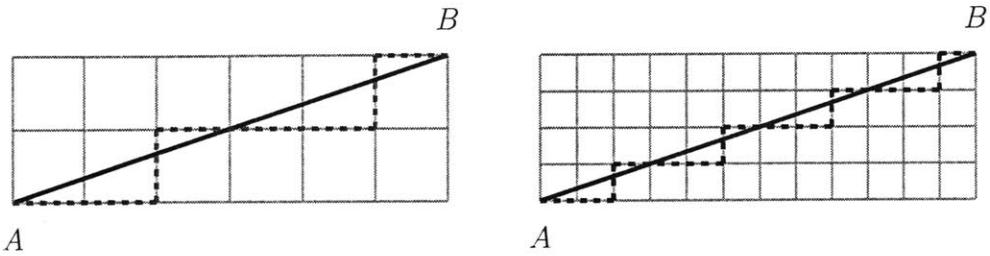


Figure 4-8: Suboptimal path found by the discretized 8-connected graph algorithm where $s = (0, 0)$, $d = (2, 0)$ and $n^* = (1, 0.3)$.

On initial examination, coarse grid granularity may be the cause of the discretized graph approach finding suboptimal paths, and that using grids of finer granularity would allow the algorithm to yield paths of lower energy that can approximate paths of unnatural angles.

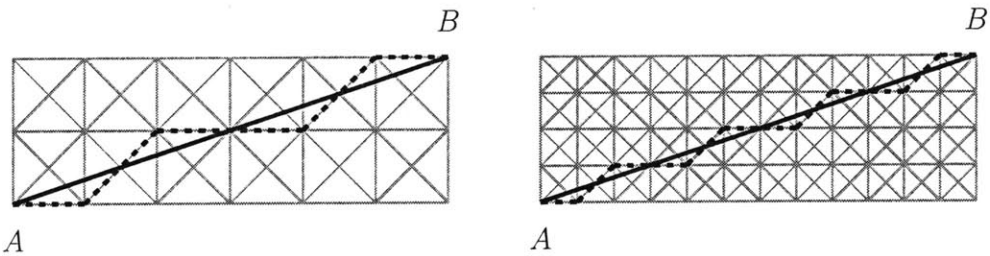
However, upon closer inspection, the discretization of the problem using a grid and discrete angles introduces a "digitization bias" [7] that cannot be fixed by increasing

grid granularity. Consider Figure 4-9 where a diagonal straight path from A to B, shown as a solid path, is approximated by edges from a 4-connected graph, shown as dashed paths. Although the approximate path in Figure 4-9b has finer granularity than the approximate path in Figure 4-9a, the two paths have the same length. Figure 4-10 shows that this is also true for paths made of edges from 8-connected graphs. Regardless of granularity, each of the finite number of paths have the same length. The difference between the length of the approximate path and the true path is called metrication error.



(a) (b)

Figure 4-9: Approximating a straight line using edges from a 4-connected graph. The solid path from A to B is approximated by edges from a 4-connected graph represented by the dashed path. (a) has coarser granularity than (b).



(a) (b)

Figure 4-10: Approximating a straight line using edges from an 8-connected graph. The solid path from A to B is approximated by edges from an 8-connected graph represented by the dashed path. (a) has coarser granularity than (b).

We show through an example how the digitization bias can cause the algorithm to find suboptimal paths. Consider the problem of finding the most energy efficient path from source to destination using the discretized 8-connected graph algorithm as

shown in Figure 4-11, where $d = n^*$. The path shown in Figure 4-11a is the optimal path, a path that cannot be composed of edges from an 8-connected graph because of the graph's discrete directional freedom. Figures 4-11b and 4-11c show two possible paths, I and II, made from edges of an 8-connected graph that are being considered by the algorithm as minimum energy paths. These two paths have the same length regardless of granularity. Path I better visually approximates the optimal path, but the discretization bias causes the algorithm to choose Path II as the approximate path because it actually has lower total energy, a phenomenon we now describe.

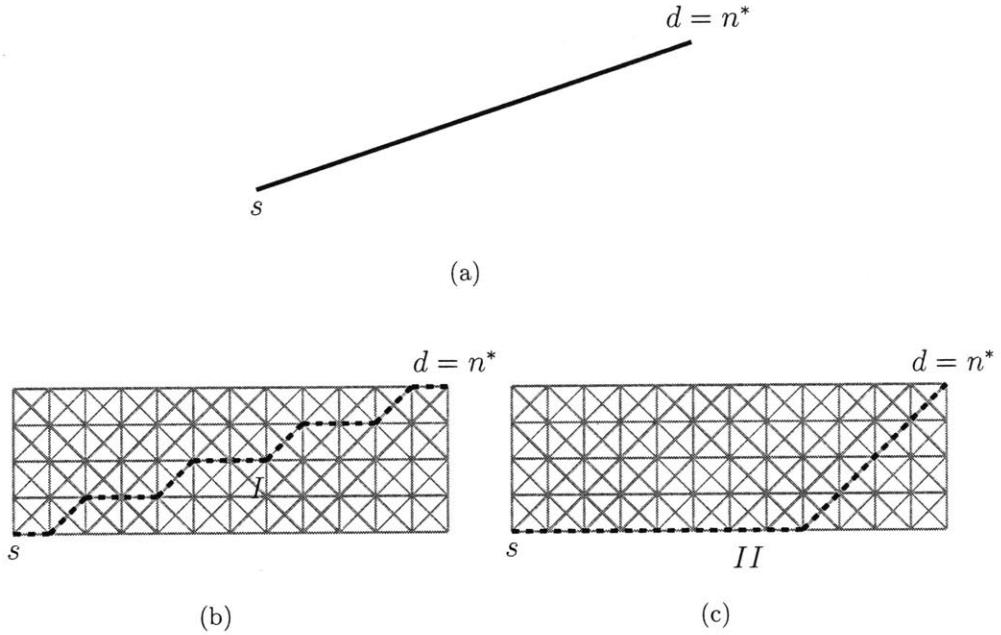
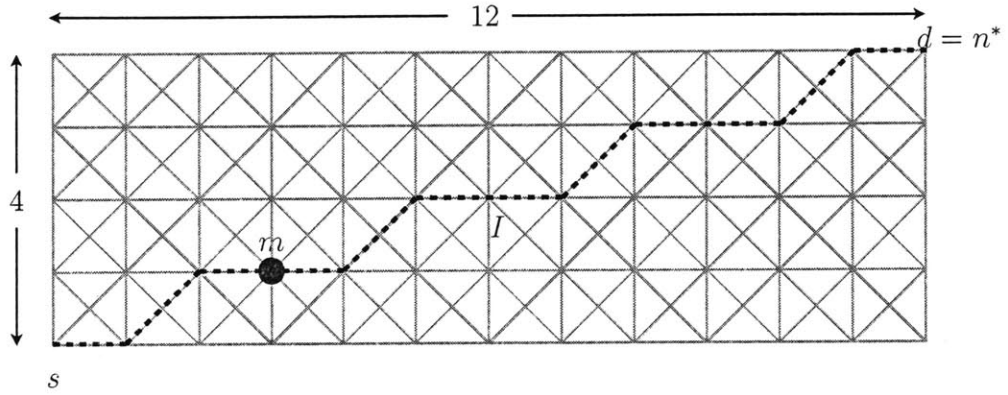


Figure 4-11: Digitization bias forces the discretized graph algorithm to find suboptimal paths. (a) shows the optimal path from s to d. (b) and (c) show two sample paths that approximate the optimal path.

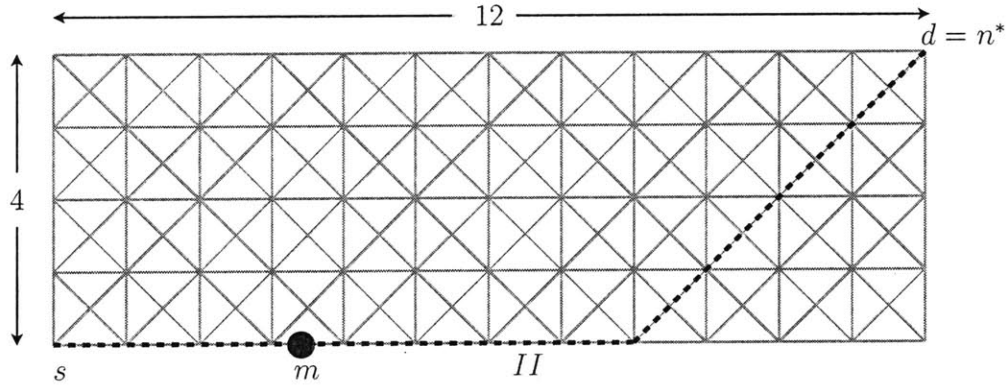
Recall the two conflicting objectives to reducing path energy: decrease the path length and move closer to n^*. The optimal path is the shortest and gets closest to n^* at any given time. However, because the optimal path $s - n^*$ path does not lie at a natural angle for an 8-connected graph, the algorithm must choose among longer paths such as Path I and Path II. Both paths have the same length, but Path II moves closer to n^* earlier, as Figure 4-12 shows for m moving at constant velocity. At the time shown in the figure, m is closer to n^* if it travels along Path II than if it travels

on Path I.

The discretized algorithm therefore chooses Path II over Path I because Path II has lower total energy than Path I even though Path I better approximates the optimal path visually. Thus, the digitization bias forces the algorithm to choose a poor path to approximate the optimal path. Because the metrication error is constant regardless of granularity, this problem cannot be solved by increasing granularity.



(a) at $t = 2 + \sqrt{2}$, $d_{mn^*} = 9.487$



(b) at $t = 2 + \sqrt{2}$, $d_{mn^*} = 9.472$

Figure 4-12: Comparison of two paths' proximity to n^*. At time $t = 2 + \sqrt{2}$, m moving at constant velocity $v = 1$ is closer to n^* if it travels along Path II than if it travels along Path I.

4.4.3 c-Connectivity

To counter the effect of digitization bias and to find a better approximate path, we allow more angular variations along the path. To provide more degrees of freedom for movement at each vertex, we expand the neighbors of a vertex to more than its eight immediate neighbors. We connect each vertex to l levels of nodes in its neighborhood, where the vertices in different levels form concentric squares around the vertex in question. In the 8-connected graph discussed in Section 4.4.2, a vertex's 8 neighbors are all neighbors of the vertex at level $l = 1$. Figure 4-13a shows how a vertex is connected to two levels of neighbors, its 8 neighbors at $l = 1$ and 16 neighbors at $l = 2$ while Figure 4-13b shows the vertex connected to all nodes at $l = 1, 2, 3$.

Each level l consists of $8l$ vertices. When we refer to a graph that is level l-connected, we mean that each vertex is connected to all vertices at levels $\leq l$. Thus a level l-connected graph has connectivity,

$$c = \sum_{k=1}^{l} 8k = 4l(l+1) \tag{4.3}$$

However, not all nodes at a level add new angles to the directional freedom at a vertex. For example, connecting to neighbors at $\tau = j\pi/4$ for $j = 1, ..., 8$ in all levels $l > 1$ does not add new angles. The dashed edges in Figure 4-13 are the edges in level 2 and level 3 that add new angles to a vertex's freedom of movement. For a given level l, the number of edges that add to the degree of freedom is (see Appendix A),

$$\text{edges} = \begin{cases} 4l, & l \text{ even} \\ 8(l-1), & l \text{ odd} \end{cases} \tag{4.4}$$

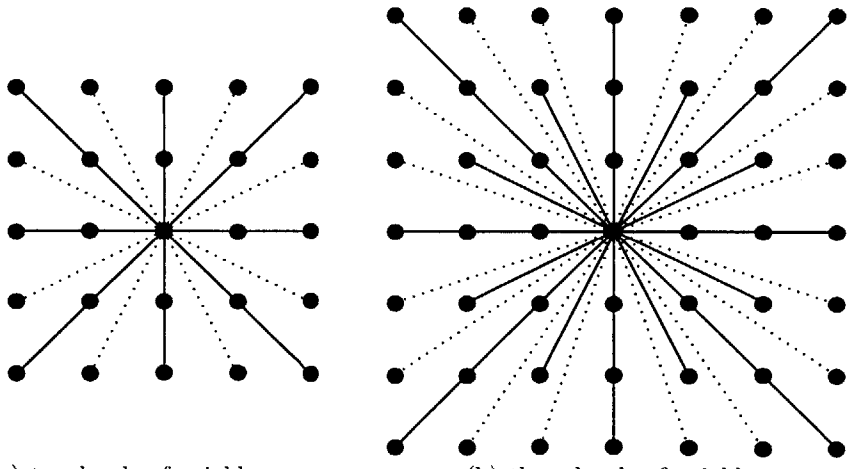
Therefore, for a level l-connected graph where $l > 1$ is odd, the total number of unique angles for a vertex is,

$$8 + \sum_{k=1}^{\frac{l-1}{2}} (4(2k) + 8(2k)) = 3l^2 + 5 \tag{4.5}$$

and for a level l-connected graph where $l > 1$ is even, the total number of unique angles for a vertex is,

$$8 + \sum_{k=1}^{\frac{l}{2}} (4(2k) + 8(2k)) - 8l = 3l^2 - 2l + 8 \tag{4.6}$$

Even though the number of angles increases with l, the angles of the edges they form are not uniform, as is the case for 4- and 8-connected graphs. Figure 4-13b shows the edges that connect a vertex to its third level of neighbors. The edges are more densely concentrated in eight wedge-shaped regions, as shown in Figure 4-14. Connecting a vertex to many levels of neighbors does not uniformly distribute the edges' angles, but only increases the density of edges within the eight wedges. Hence, arbitrarily increasing graph connectivity does not necessarily allow the algorithm to find paths of decreasing energy. We investigate graph connectivity and the diminishing returns of increasing l in Chapter 5.



(a) two levels of neighbors (b) three levels of neighbors

Figure 4-13: Level l-connected graphs. The dashed lines represent edges in the (a) $l = 2$ and (b) $l = 3$ that contribute new angles to the vertex's degrees of freedom.

We analyze the one communication node problem we examined in Figures 3-3a and 4-15 where $n^* = (1, 0.3)$ using the discretized c-connected grid algorithm. We use problem granularity $g = 200$ and vary graph connectivity. Figure 4-15 shows the path

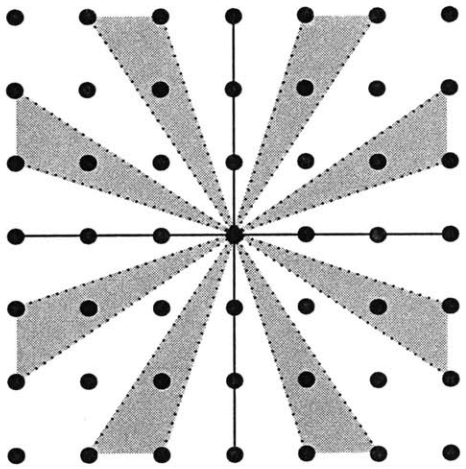
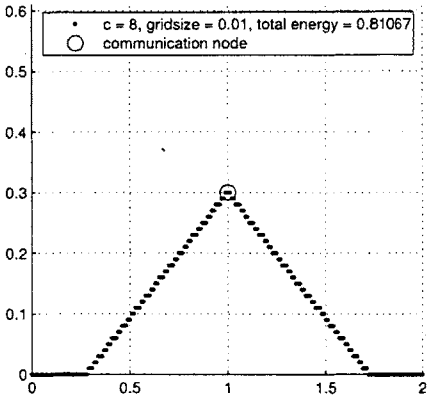


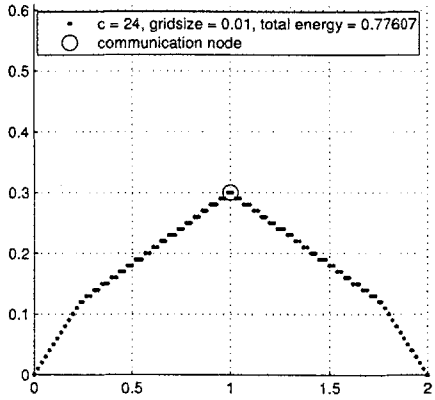
Figure 4-14: Level l-connected graphs do not uniformly distribute the edges' angles for $l > 1$. The edges are distributed more densely in the eight wedge-shaped shaded regions shown for this example where $l = 3$.

found as the connectivity of the graph increases. Figure 4-15a shows the path found for only one level of connectivity, the same as the path found using the 8-connected graph algorithm seen in Figure 4-7. As l increases and vertices have more neighbors, the path found resembles the direct $s - n^* - d$ path more and the total energy decreases. Figure 4-15d and Figure 4-15e show the same path because vertices in a level 4-connected graph have enough degrees of freedom to find the optimal path so that increasing the connectivity of the graph does not find a more energy efficient path for this problem. We discuss in detail how graph connectivity affects the path found in Chapter 5.

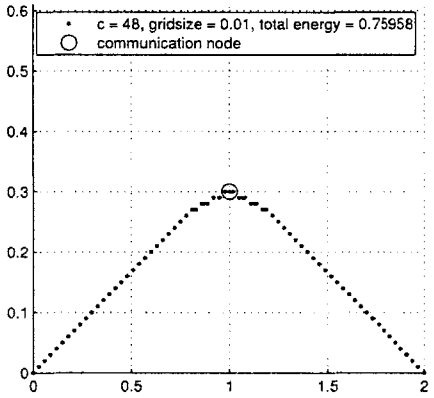
In Section 3.3.2, we discussed how the recursive algorithm finds the local optimal solution at each recursion level which collectively do not yield a good approximation to the global optimal solution. This discretized algorithm yields a better approximation of the optimal path (assuming adequate angular freedom) because the shortest path search is global. For example, we apply the discretized graph algorithm to the problem we introduced in Section 3.3.2 and shown in Figure 3-9. We compare the solutions of the two algorithms. The discretized graph algorithm finds a path of lower energy, shown in Figure 4-16.



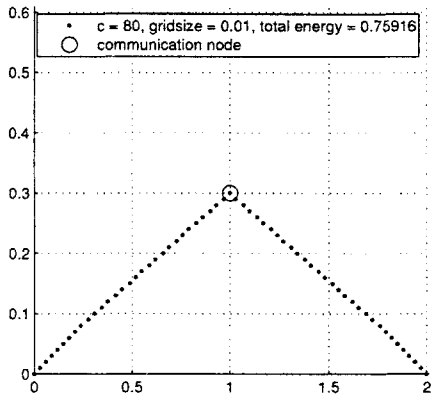
(a) $l = 1$



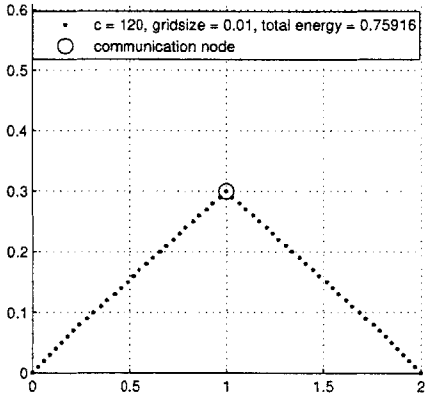
(b) $l = 2$



(c) $l = 3$



(d) $l = 4$



(e) $l = 5$

Figure 4-15: Paths found for the problem where $n^* = (1, 0.3)$. As the connectivity of the graph increases from level $l = 1$ to $l = 5$, the algorithm find paths of decreasing energy.

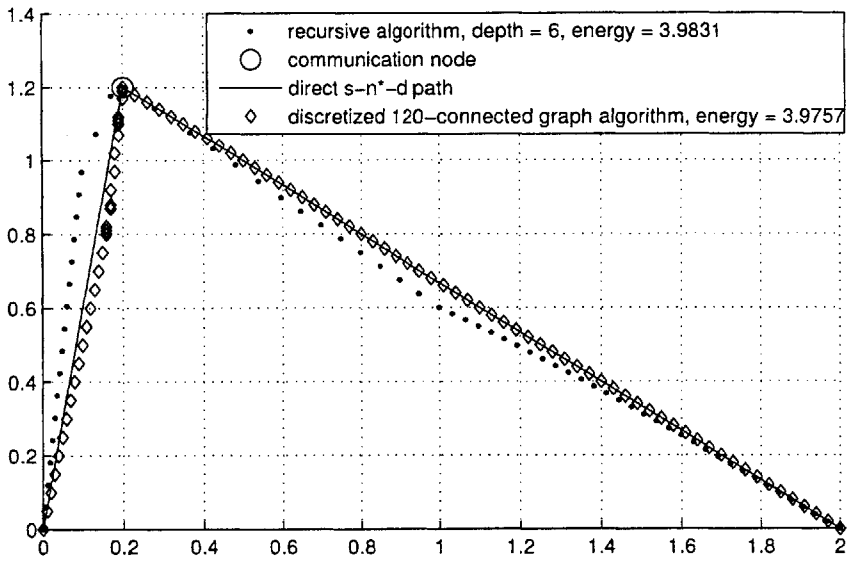


Figure 4-16: Comparison of direct $s - n^* - d$ path and paths found by recursive and discretized graph algorithms. For the problem where $s = (0, 0)$, $d = (2, 0)$, and $n^* = (0.2, 1.2)$, the discretized 120-connected graph algorithm using $g = 100$ finds a more energy efficient path than the path found by the recursive algorithm.

4.5 Multiple Communication Nodes

The discretized graph algorithm is more general than the recursive algorithm because it easily allows for multiple communication nodes in the problem. As discussed in Section 4.1, we calculate each edge weight with respect to the communication node associated with the Voronoi region the edge lies in. The minimum energy path may not be unique, as the two paths in the example in Figure 4-17 shows. In the case of multiple optimal paths with the same energy, the algorithm returns one of the paths, where the chosen path depends on the implementation of the shortest path algorithm.

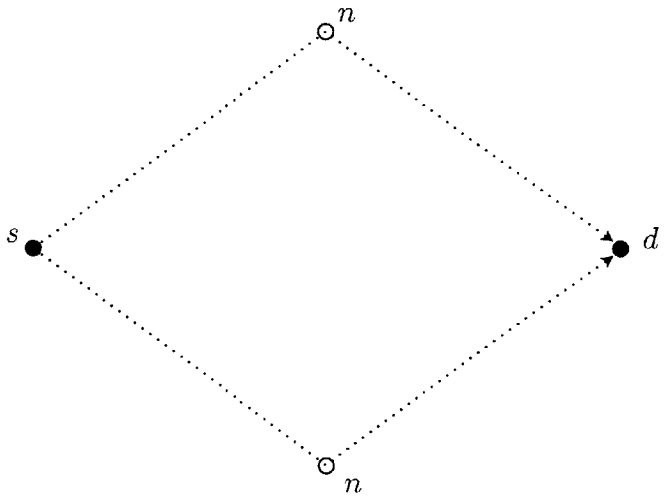


Figure 4-17: Two minimum energy paths with the same energy.

Figure 4-18 shows the path found from s to d given five randomly placed communication nodes using a level 4-connected graph. The path shown in the figure goes through one communication node en route to d from s but the communication energy required in directly traveling to any of the other nodes is too high. Hence, the path curves towards the communication node at (0.93, 1.57) in order to get close to that node but curves quickly toward d to decrease the path length.

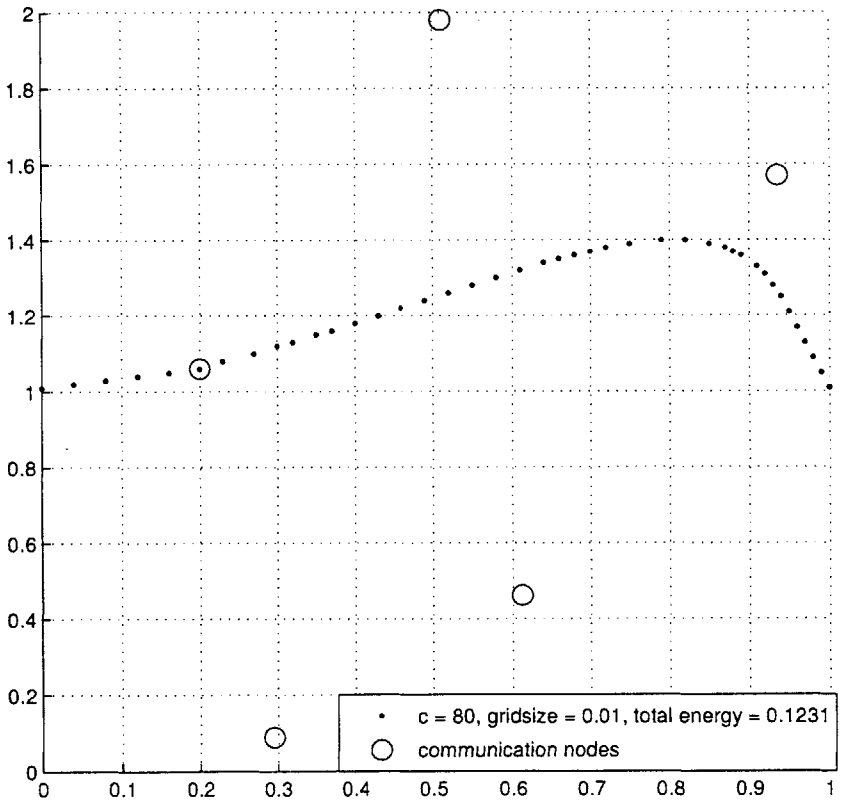


Figure 4-18: The path found for a problem of finding a minimum energy path from $s = (0, 1)$ to $d = (1, 1)$ with five randomly placed communication nodes using problem granularity $g = 100$ and an 80-connected graph.

Chapter 5

Discretized Graph Algorithm Parameters

In this chapter, we investigate how the parameters for the discretized graph algorithm affect the approximate minimum energy path the algorithm finds. We vary the problem granularity g and graph connectivity c and determine good values of g and c for the algorithm that balance having adequate running time with finding energy efficient paths.

In Section 4.4.1, we showed how the discretized 4-connected graph algorithm using different problem granularities can find different paths (see Figure 4-5). In this chapter, we examine the specific one communication node problem where $s = (0, 0)$, $d = (2, 0)$ and $n^* = (1, 3)$.

First, we explore how problem granularity affects the path. Because $d_{sd} = 2$, we vary the *gridsize* and use problem granularities as seen in Table 5.1.

As Figure 5-1 shows, two paths found by the algorithm using an 8-connected graph ($l = 1$) and different problem granularities have the same shape and similar energy costs despite their large difference in problem granularity.

Figure 5-2 shows how the path energy decreases as problem granularity g increases for six different levels of graph connectivity; for each, the path energy decreases less than 1% as g increases. For all levels of connectivity, we find that a problem granularity

gridsize	g
0.2	10
0.1	20
0.05	40
0.025	80
0.0125	160
0.00625	320
0.003125	640

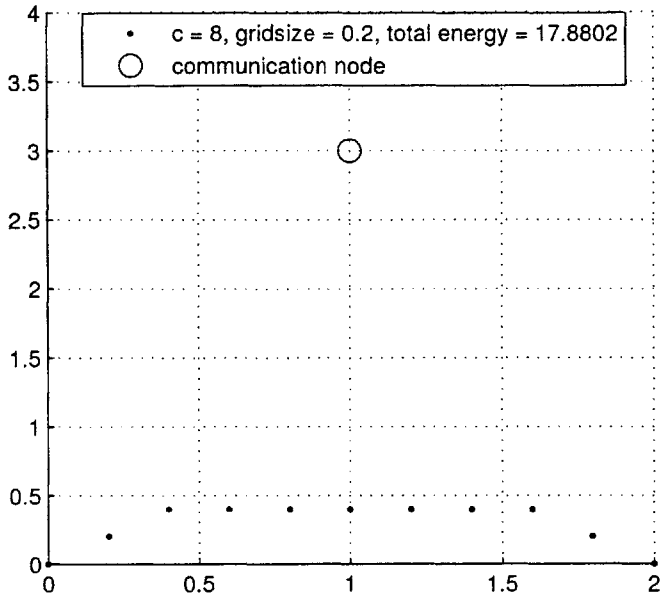
Table 5.1: Problem granularities used to examine how parameters for the discretized graph algorithm affect path energy

of $g = 80$ suffices and that using grids of finer granularity does not allow the algorithm to find paths of lower energy. Overall, problem granularity does not significantly affect the path found: paths of different granularities have the same trajectory, but are defined by more points as problem granularity increases. Beyond the initial slight decrease in energy as g increases from 10 to 20, defining the path by more points does not significantly affect the energy of the path found.

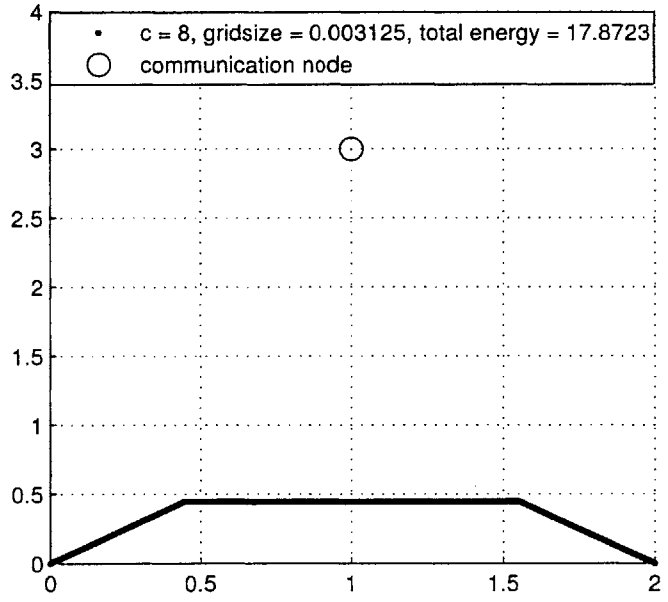
We now investigate how graph connectivity affects the path. Section 4.4 detailed how we connect each vertex in the graph to c-neighbors and Figure 4-15 shows how increasing graph connectivity can alter the path and decrease the energy. However, as we discussed in Section 4.4.3, increasing the connectivity beyond a certain level does not uniformly increase the distribution of the edges' angles and may have diminishing returns in terms of lowering energy.

We study how graph connectivity affects the approximate path by examining the problem where $s = (0, 0)$, $d = (2, 0)$ and $n^* = (1, 3)$, as before. We apply graphs of various connectivity to the problem for the granularities listed in Table 5.1. In contrast to varying granularity, increasing the connectivity can change the shape of the path the algorithm finds, as Figure 5-3 shows for $g = 80$. The path found using an 8-connected graph, shown in Figure 5-3a, is made up of straight segments; as graph connectivity increases in Figures 5-3b-f, the path better approximates a curve.

Figure 5-4 shows how the path energy decreases as graph connectivity increases for the seven problem granularities in Table 5.1. For all values of g, path energy decreases

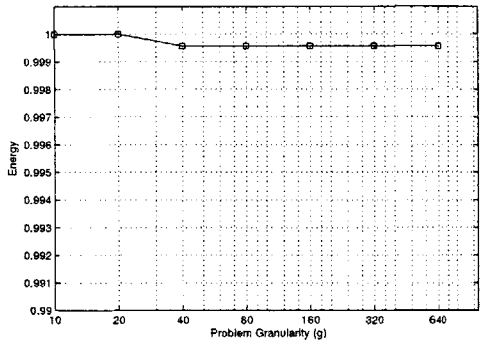


(a) $g = 10$

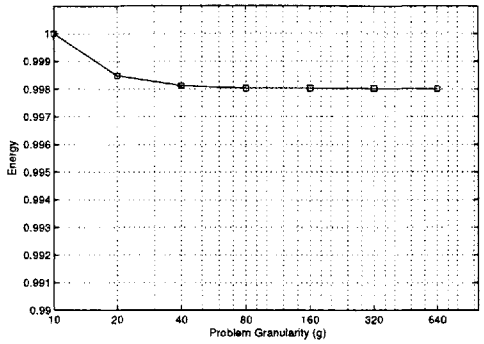


(b) $g = 640$

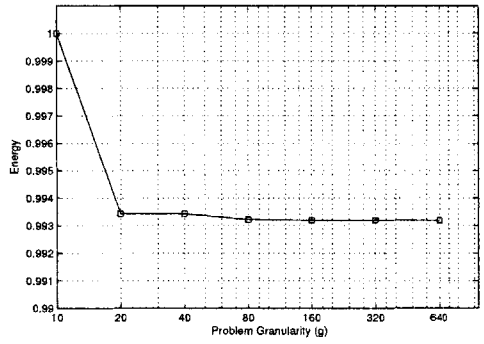
Figure 5-1: Paths found by discretized graph algorithm using different problem granularities and an 8-connected graph.



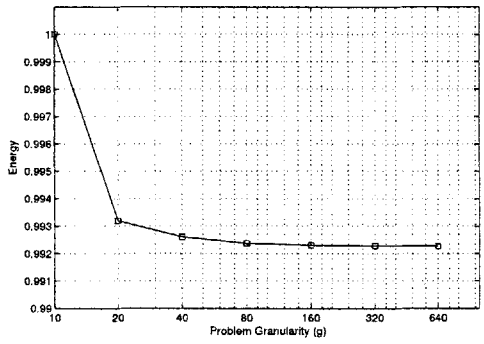
(a) $l = 1$



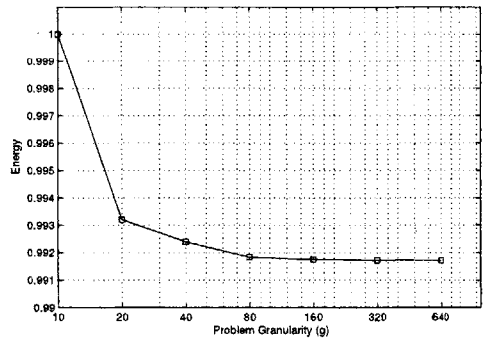
(b) $l = 2$



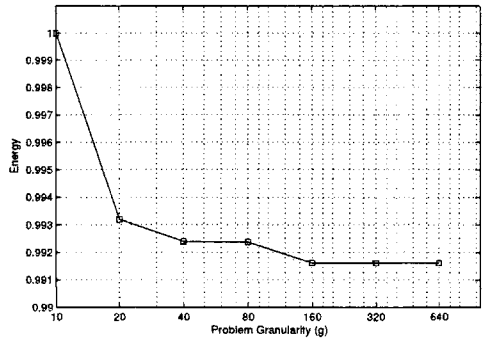
(c) $l = 3$



(d) $l = 4$

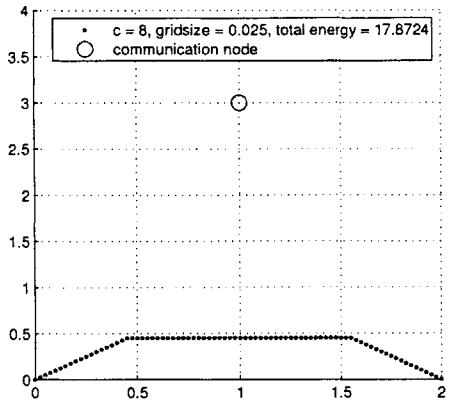


(e) $l = 5$

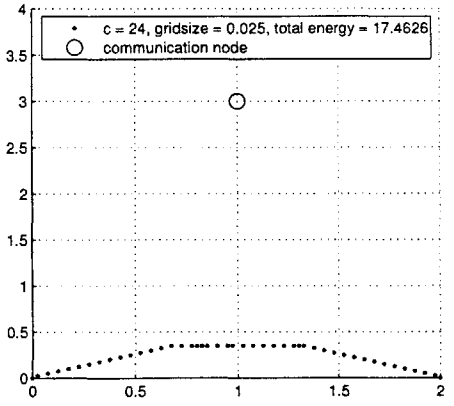


(f) $l = 6$

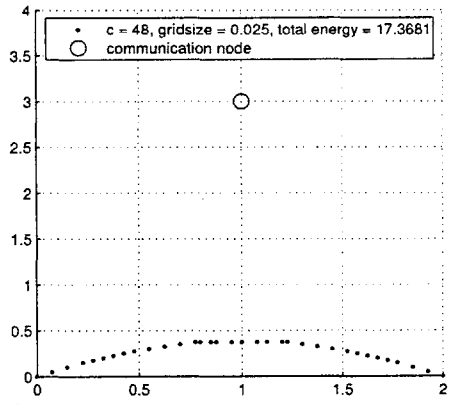
Figure 5-2: Energy of paths found by the discretized graph algorithm decreases as problem granularity g increases. For each figure of level l-connectivity, energies are normalized by the path energy found using $g = 10$.



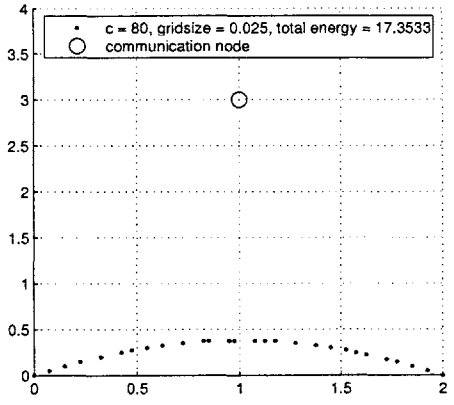
(a) $l = 1$



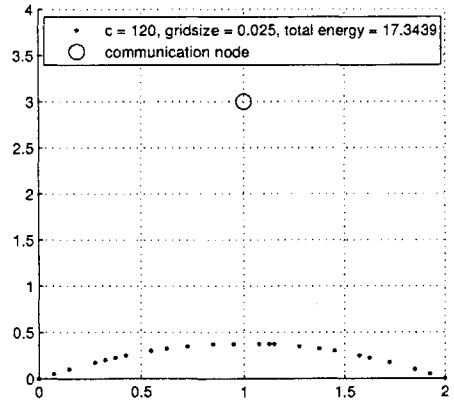
(b) $l = 2$



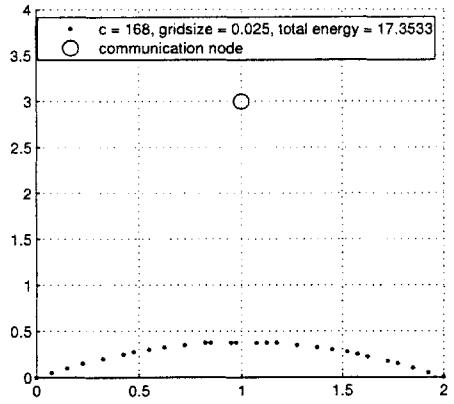
(c) $l = 3$



(d) $l = 4$



(e) $l = 5$



(f) $l = 6$

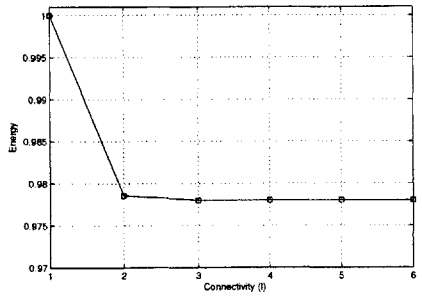
Figure 5-3: Paths found by discretized graph algorithm using different levels of graph connectivity and problem granularity $g = 80$.

approximately 3% as the level of connectivity increases. There is an initial energy decrease of less than 2.5% as l increases from one to two (and c increases from 8 to 24), but further increases in connectivity do not significantly reduce the energy.

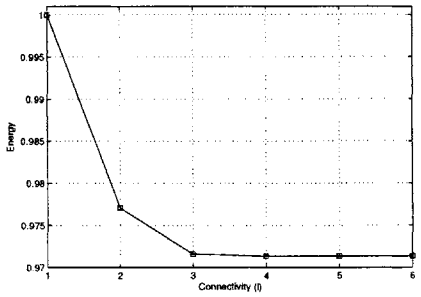
For all problem granularities, but specifically for $g = 80$ that we previously found to be adequate, a level 4-, or an 80-connected graph suffices to find an good approximate minimum energy path and connecting a vertex to more than 80 neighbors does not significantly alter the resulting path. Overall, graph connectivity affects the trajectory of the path found by the discretized graph algorithm, and using a more connected graph reduces path energy.

The effect of connectivity on path energy is slightly greater than the effect of problem granularity on path energy, as comparison of Figure 5-2 and Figure 5-4 shows. Increasing granularity decreases energy by about 1% whereas increasing connectivity decreases energy by approximately 3%. Therefore, to decrease path energy, it is more important to increase c than to increase g. Figure 5-5 shows how path energy decreases as a function of both problem granularity and graph connectivity.

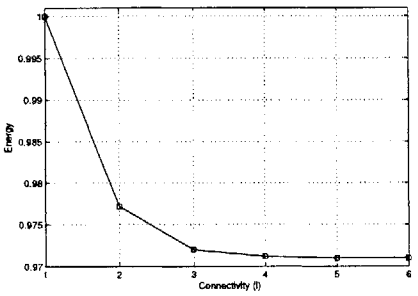
We find that using a problem granularity of $g = 80$ and level 4-, 80-connected graph allows the discretized graph algorithm to find good approximate minimum energy paths.



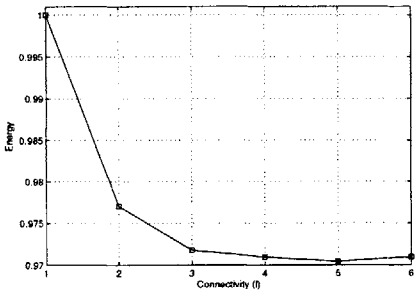
(a) $g = 10$



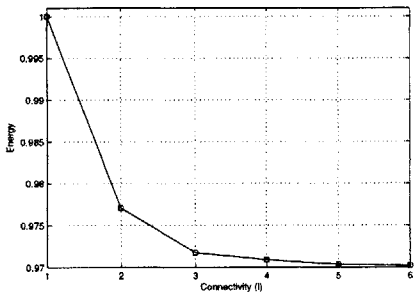
(b) $g = 20$



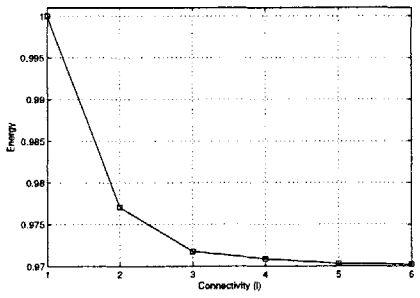
(c) $g = 40$



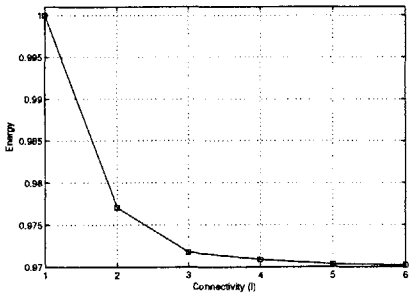
(d) $g = 80$



(e) $g = 160$



(f) $g = 320$



(g) $g = 640$

Figure 5-4: Path energies found by discretized graph algorithm using different levels l of graph connectivity. For each figure of granularity g, the energies are normalized by the path energy found using $l = 1$ connectivity

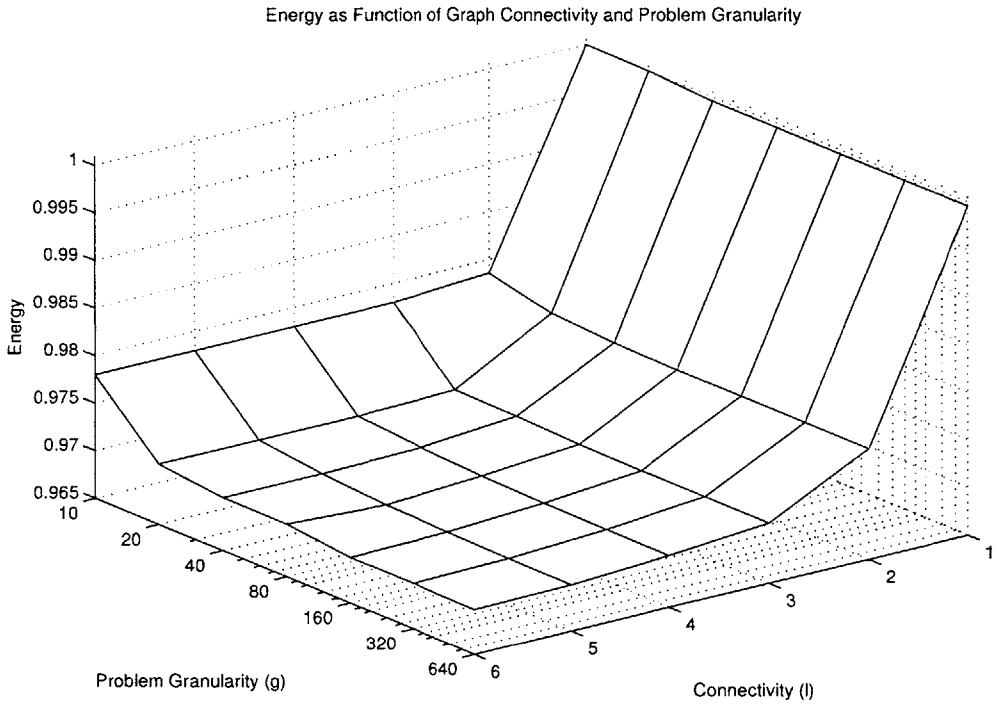


Figure 5-5: Path energy as function of level l connectivity and problem granularity g. Energies are normalized by the energy of the path found using $g = 10$ and $l = 1$.

Chapter 6

Sample Problem

We introduced the discretized graph algorithm in Chapter 4 and determined the parameters that find good approximate minimum energy paths in Chapter 5. In this chapter, we use the discretized graph algorithm and the previously determined values of problem granularity g and graph connectivity c to solve a sample problem.

We consider the problem of determining an adequate number of communication nodes to deploy in the field that minimizes the energy of all paths from one side of the field to the other. This problem can occur if we want to facilitate mobile nodes traveling across enemy territory while minimizing their communication energies. If we deploy communication nodes in enemy territory by scattering them from the air, we know the general area that they fall into, but we cannot predict the exact locations of the nodes.

For a given $s - d$ pair, deploying the communication nodes in a collinear configuration along the $s - d$ line will result in the path of least energy from s to d. Because we cannot control the exact locations of the communication nodes and want the flexibility to facilitate energy efficient paths from different sources to different destinations, we hope to deploy enough nodes such that the average $s - d$ path will be energy efficient. Since communications nodes are expensive, we want to deploy the least number of nodes while minimizing the energies of all paths from one side of the field to the other.

Using the discretized graph algorithm, we investigate this problem by scaling the

enemy territory to a 1×1 square and fitting the square onto a Cartesian plane with the lower left corner at $(0, 0)$. A mobile node m must cross from one side of the square to another while minimizing its communication energy, as defined in Equation (2.1), where $\alpha = 2$.

We randomly choose ten sets of communication nodes of ten nodes each, where the node positions are uniformly distributed over the 1×1 square. For each set, we randomly choose ten $s-d$ pairs such that $s = (0, s_y)$ and $d = (1, d_y)$, where s_y and d_y are uniformly distributed between 0 and 1. In Chapter 5, we found that using a problem granularity $g = 80$ and connectivity $c = 80$ as parameters to the discretized graph algorithm allows us to find good approximate minimum energy paths. For this problem $\min(d_{sd}) = 1$ and $\max(d_{sd}) = \sqrt{2}$, so we choose $gridsize = .01$ for $\min(g) = 100$ and $\max(g) = 141$. We construct a level 4-, or 80-connected graphs to analyze our problem.

For each set of communication nodes, we randomly choose one of the ten nodes to place first in the square. We then find the minimum energy path for each of the ten $s - d$ pairs. We keep previously added nodes and introduce more communication nodes from the set into the square to analyze how the paths change. Figure 6-1 shows how the path from a randomly chosen s to a randomly chosen d changes as communication nodes are randomly placed in the field. As nodes are added, the path changes and the energy decreases because the additional nodes allow for more energy efficient paths.

Figure 6-2 shows how the average energy of the $s - d$ paths found by the discretized graph algorithm decreases as communication nodes are deployed onto the field, where each data point represents the average energy of 100 paths, ten paths for each of the ten sets of communication nodes. Initially when there are few nodes to communicate with, the introduction of new nodes facilitates significant decreases in path energies. When there are few nodes in the field, adding new nodes changes the path and decreases the path energy for many of the $s - d$ pairs. For example, the addition of the second node reduces the average energy by 34%. If there are many deployed communication nodes, introducing new nodes changes the minimum energy paths for fewer $s - d$ pairs since a mobile node is more likely to communicate with existing nodes than newly added

nodes. Hence, adding communication nodes to an existing deployment of many nodes has a less significant effect on the average energy. Indeed, as seen in Figure 6-2, the addition of the fifth node to the set of four existing communication nodes decreases average energy by an additional 5% in comparison to the average path energy where there is only one communication node.

Therefore, it may not be worth the cost of deploying nodes in order to achieve slight energy savings. Depending on the objectives of the deployment and the tradeoff between the cost of each communication node and the energy constraints of the mobile nodes, we can determine the adequate number of communication nodes for energy efficient travel between two sides of the field.

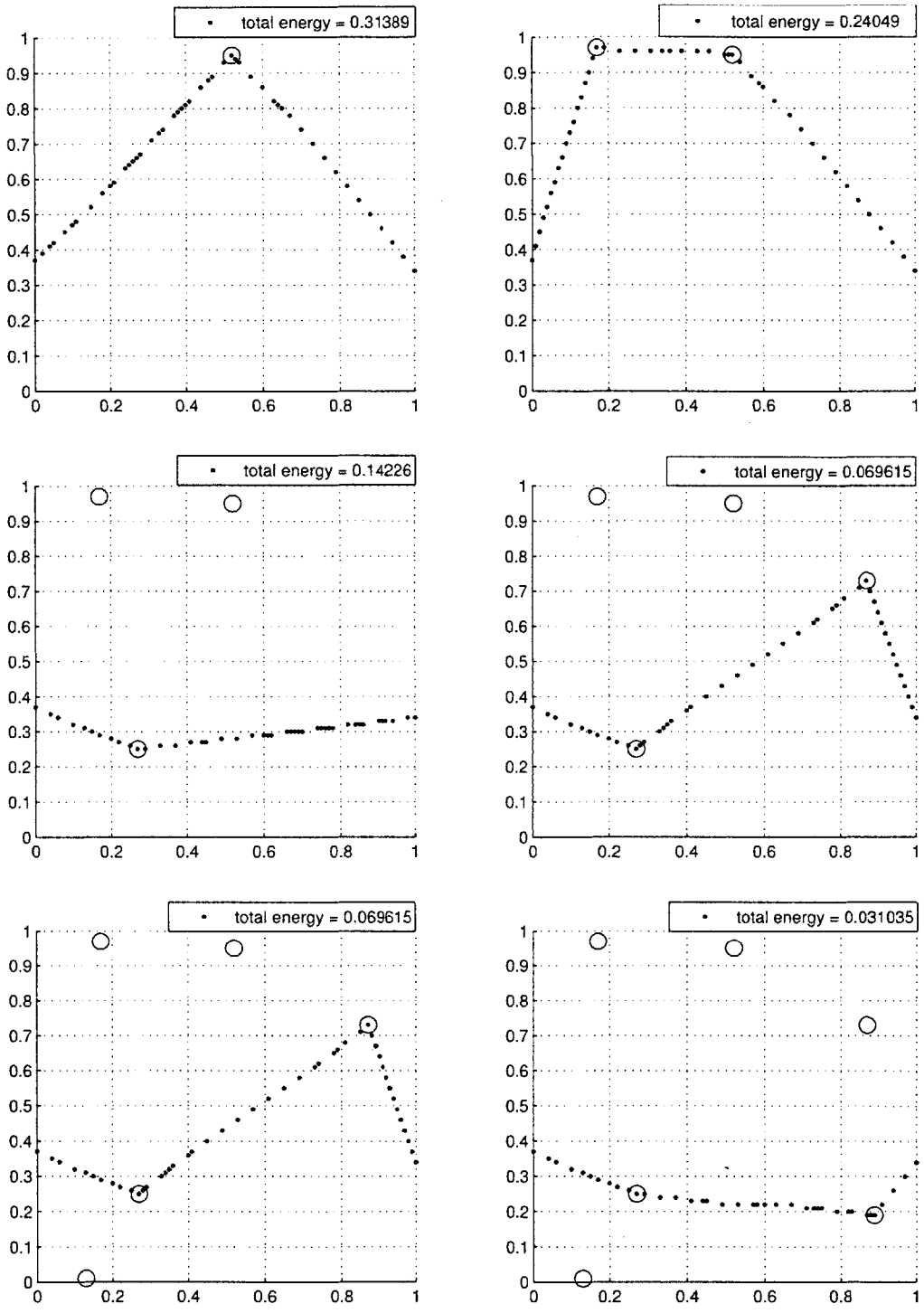


Figure 6-1: Energy of paths decrease as communication nodes are added to the problem. Minimum energy paths from $s = (0, 0.37)$ to $d = (1, 0.34)$ as communication nodes are randomly placed in the field.

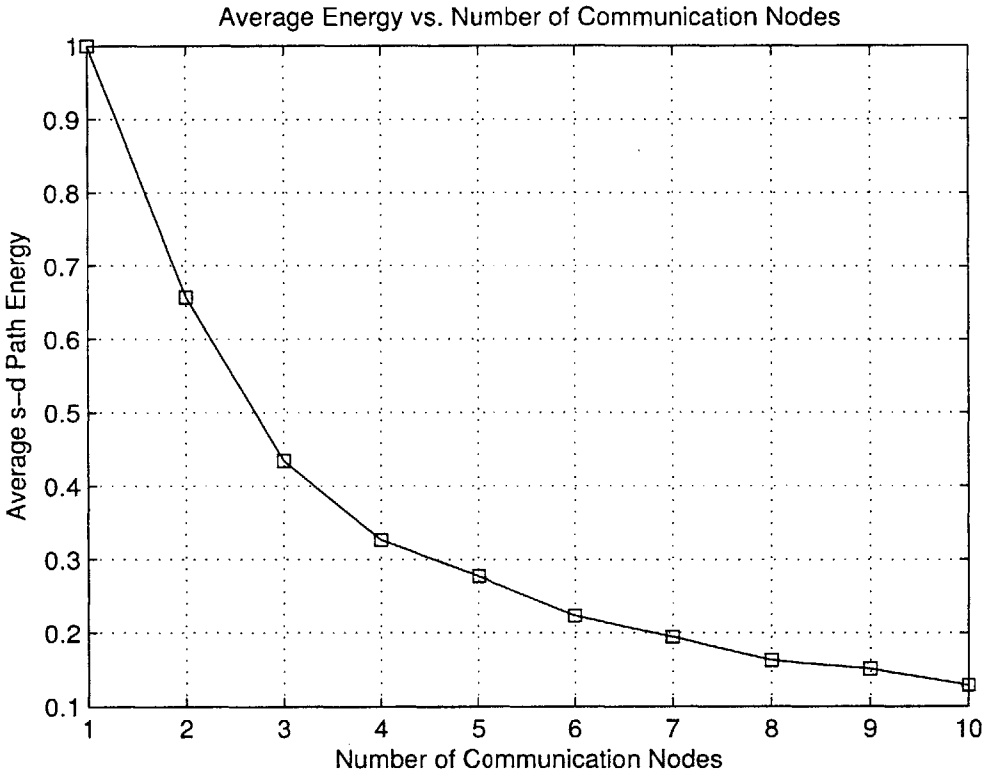


Figure 6-2: Average energy of minimum energy path decreases as the number of communication nodes increases. Each data point represents the average energy of 100 paths, ten random $s - d$ pairs for each of ten sets of randomly placed communication nodes. Path energies are normalized by the average energy of paths when there is one communication node.

Chapter 7

Conclusions

7.1 Contributions

In this thesis, we introduced the minimum energy path planning problem for ad hoc networks. In this problem, we find a path that requires the least communication energy for a mobile node from one point to another throughout which it stays connected to a set of communication nodes.

We discussed the problem's properties and analyzed the two factors involved in minimizing energy: traveling a short path and moving close to the communication nodes it communicates with. We demonstrated how these two objectives affect the minimum energy paths found using a recursive approach for problems with one equidistant communication node. If the communication node is close to the source and the destination, the mobile node travels straight to the communication node in the optimal path. On the other hand, if the communication node is far from the source and the destination, the mobile node travels more directly from the source to the destination and does not travel as close to the communication node.

Although we used the recursive algorithm to gain key insights to the minimum energy path planning problem where there is one communication node, we showed the limitations of the algorithm in finding good approximate paths and in finding paths for problems with multiple communication nodes.

We presented a discretized graph algorithm that finds approximate minimum energy paths for problems of one or multiple communication nodes. We determined good values for the algorithm's parameters, problem granularity and graph connectivity, that allow the algorithm to find good approximate paths while having an acceptable running time. We found that using a level 4-, 80-connected graph and problem granularity of $g = 80$ is adequate for the algorithm to find good approximate paths.

Finally, we demonstrated the applicability of the discretized graph algorithm by using it to solve a sample problem. In this problem, we determined an adequate number of communication nodes to randomly deploy over a field to facilitate energy efficient travel across the field.

7.2 Future Work

In this thesis, we focused on problems where $\alpha = 2$ in the communication power. However, α can be greater than two, and future work can investigate how different values of α affect the path found.

In our problem framework, we did not factor in the energy used to physically travel from source to destination in finding the minimum energy path. Realistically, this energy should be taken into consideration and can influence the path taken. We can extend the discretized graph algorithm to include this energy by adding an additional cost to each edge in the graph that is proportional to the length of the edge. This additional energy would place more emphasis on the total path length factor influencing the total path energy because the added energy is a function of only path length and not proximity to communication nodes. In a similar fashion, we can incorporate the Weighted Region Problem by taking into account different energy required to travel in different types of terrain. Future work can investigate how the minimum energy path changes if we consider the movement energy.

The discretized graph algorithm can also be extended to facilitate other constraints in the problem. For example, if a mobile node only has a certain amount of time to

travel to the destination, we can alter the shortest path search to constrain the total path length.

Another constraint may arise if there are physical obstacles in the terrain that a mobile node cannot travel through, such as mountains, buildings, or bodies of water. Revisiting the example of the soldier traveling through hostile territory, he may wish to avoid certain areas of the region because there are more enemy combatants concentrated in those areas. In such situations, the algorithm can be altered by removing the edges corresponding to the forbidden regions from the graph.

Appendix A

Directional Freedom in c-Connected Graphs

In the discretized c-connected graph algorithm, each level l consists of $8l$ neighbors for a given vertex. However, not all vertices in a level add new angles to the directional freedom at a vertex.

Connecting to neighbors at $\tau = j\pi/4$ for $j = 1, ..., 8$ in any level $l > 1$ does not add new angles. For neighbors at "even" levels, where l is even and $l > 1$, only half of the vertices contribute new angles. Figure A-1 illustrates how for every other vertex in an even level (shown as white vertices), there is a vertex at a lower level with the same angle. Therefore, out of $8l$ neighbors in an even level l, there are $4l$ vertices that add new angles of direction freedom.

Vertices at "odd" levels contribute more angular freedom, as the black vertices in Figure A-1 show for $l = 3$. All vertices except those at $\tau = j\pi/4$ for $j = 1, ..., 8$ add new angles. Therefore, there are $8(l-1)$ neighbors in an odd level l that add directional freedom.

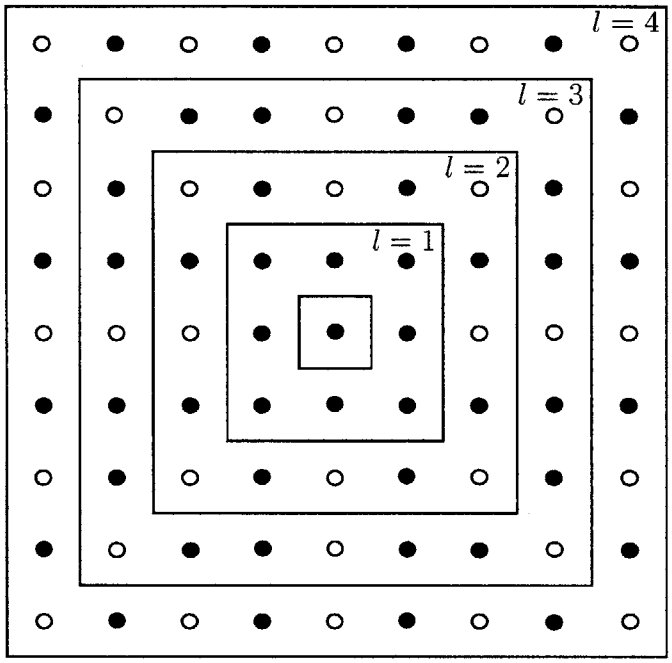


Figure A-1: Number of vertices in a given level that add new angles of directional freedom. Black vertices add new angles of freedom while white vertices do not.

Appendix B

Implementation Details

We implemented the recursive algorithm using Matlab 7. We implemented the discretized graph algorithm in Matlab 7 and used the GNU C compiler to compile the shortest path algorithm code from SPLIB. The discretized graph algorithm is as follows:

DISCRETIZED-GRAPH-ALGORITHM$(s, d, N, gridsize, l)$

$y_{max} = \max(s_y, d_y, n_y), \forall n \in N$

$y_{min} = \min(s_y, d_y, n_y), \forall n \in N$

$x_{max} = \max(s_x, d_x, n_x), \forall n \in N$

$x_{min} = \min(s_x, d_x, n_x), \forall n \in N$

$rows = (y_{max} - y_{min})/gridsize$

$cols = (x_{max} - x_{min})/gridsize$

$V \leftarrow rows \times cols$

foreach $i \in V$

    $e_{ij} \leftarrow$ edges to neighbors j in levels $\leq l$

$T \leftarrow Vor(N)$

foreach $e_{ij} \in E$

    $n^* \leftarrow n$ in VoronoiRegion$(T, \text{midpoint}(i, j))$

    $E_{ij} = \text{ENERGY}(i, j, n^*)$

$G = (V, E)$

minimum energy path $\leftarrow$ DIJKSTRA(s, d, G)

Bibliography

[1] Boris V. Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest paths algorithms: theory and experimental evaluation. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 516–525, Arlington, Virginia, January 1994.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Intoduction to Algorithms.* MIT Press, Cambridge, Massachusetts, second edition, September 2001.

[3] Mark de Berg, Marc van Kreveld, Mark Overmars, and Ottfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Spring-Verlag, Berlin, Heidelberg, Germany, second edition, 2000.

[4] Edsger Wybe Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] http://www.avglab.com/andrew/soft.html.

[6] D.M. Keirsey and Joseph S.B. Mitchell. Planning strategic paths through variable terrain data. In *In Proc. SPIE Applications of Artificial Intelligence*, 1984.

[7] Joseph S.B. Mitchell and David M. Keirsey. Planning strategic paths through variable terrain data. In *Proc. of SPIE*, 1984.

[8] Joseph S.B. Mitchell and Christos H. Papadimitriou. The weighted region problem. In *Proceedings of the 3rd Annual ACM Symposium on Computational Geometry*, pages 30–38, Waterloo, Ont., Canada, June 1987.

[9] Francis K. H. Quek, R.F. Franklin, and F. Pont. A decision system for autonomous robot navigation over rough terrain. In *Proc. SPIE Applications of Artificial Intelligence*, Boston, 1985.

[10] A.M. Vossepoel and A.W.M Smeulders. Vector code probability and metrication error in the representation of straight lines of finite length. *Computer Graphics and Image Processing*, 20(4):347–64, 1982.