INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

Compass Routing on Geometric Graphs

by

Harvinder Singh

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirement for the degree of
Master of Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa
Ottawa, Ontario, Canada

The Master of Computer Science program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute for Computer Science.

Abstract

In this thesis, we introduce a novel routing algorithm which we call "compass routing" to find paths between pairs of points in planar geometric graphs.

Our main goal was that of developing, whenever possible, routing algorithms that, using only "local information", the position of our destination and a finite amount of extra memory, find *a path* from a starting position to our destination.

We developed "compass routing" based routing algorithms for trees, Delaunay triangulations and orthogonal convexly embedded geometric graphs. Several related results on various types of geometric graphs were also studied.

Acknowledgements

I would like to thank my thesis supervisor Dr. Jorge Urrutia and my thesis co-supervisor Dr. Jurek Czyzowicz for their advice, patience and confidence in my work, as well as for the many hours they spent helping me. Their guidance and counselling have been of great importance in this stage of my academic career. I would like to also thank Dr. Evangelos Kranakis from Carleton University for his contribution to the result presented in Chapter 7 of this thesis.

I thank Margaret Urrutia for reading my thesis.

A very special thanks to Gordon Beatty for his advice, for reading this thesis and his extremely valuable help throughout my university years.

I would like to thank Ralph Boland and Felipe Contreras for very helpful meetings regarding the thesis work.

I would like to thank my many other colleagues both from Carleton University and University of Ottawa for their suggestions.

Lastly, I really appreciate the financial assistance from my supervisor Dr. Jorge Urrutia during my stay in Mexico City, Mexico.

Contents

List of Figures

Chapter 1

Introduction

The shortest path problem is one of the most fundamental problems in computer science. Briefly, given a graph G in which there is a cost (weight) associated with each edge, the goal is to compute the least expensive path between pairs of vertices of G. A shortest path algorithm is studied in Chapter 2.

In this thesis, we are interested in developing, whenever possible, "local routing algorithms" for graphs. A local routing algorithm is defined to be an algorithm that finds a path $v_0 = u, v_1, v_2,...,v_m = v$ between two vertices u (source) and v (destination) of a graph G such that vertex v_{i+1} is chosen among the neighbours of v_i using only the identity of v and the information stored at v_i concerning its neighbours; that is, the full information about G at any time during execution is not available to us.

The idea of using local routing algorithms arises from the following problem:

Suppose that we are given a road map (see Figure 1.1) of a town. Furthermore, assume that we are at point s of the map and we wish to travel to point d in the map. We start by choosing the road which points the closest to the direction determined by dotted line segment joining s to

d, that is, we pick the road sa to arrive at intersection point a. Similarly, we next choose the road ab, followed by bc, and finally cd to arrive at our destination d.

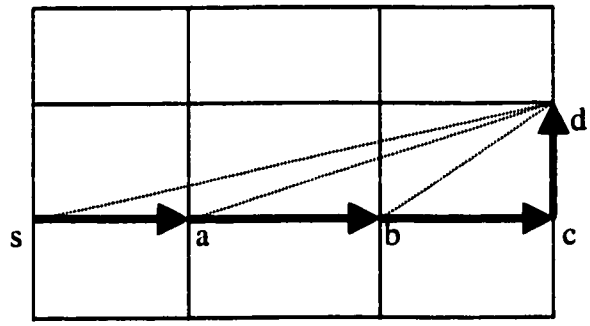


Figure 1.1: A road map

In the context of this thesis, we will think of a road map as a geometric graph and the way we select roads upon reaching intersection points as compass routing.

Let us now look at the notion of compass routing. First we will consider compass routing from a historical perspective. Using a compass, we can direct ships, airplanes and road vehicles. Generally speaking, a map and a compass are the two main items required both for land navigation and marine navigation. Land navigation is considered formally in [Kals83]. In land navigation, a compass and a map are of extreme importance when we wish to explore undeveloped country which is rarely travelled. For complete details regarding navigation with map and compass in wilderness, refer to [Blan92], [Gear80], [Flem82] and [Kjel76].

In this thesis, we study routing problems in planar geometric graphs. A planar geometric graph is a graph embedded on the plane such that its vertices are represented by points and its edges by non-intersecting straight line segments.

Figure 1.2: Road map of Mexico City

The above mentioned procedure motivates the following algorithm which we call *compass routing:*

Suppose we wish to travel from vertex s (source) to vertex d (destination) in a geometric graph G. We start to walk along the edge of graph G incident to vertex s which has the closest slope to that of the line segment joining vertices s and d. This process is repeated at each vertex we arrive at, until we reach the destination vertex d.

Using this procedure, we compute two paths (not necessarily the shortest) from Viveros to Terminal Aerea and Tlatilco to Tacuba on the road map of Mexico City shown in Figure 1.2. The resulting paths are shown in red and green respectively.

We present some definitions in Chapter 3 which will be used throughout this thesis.

This motivates the study of the following problem: Let G be a planar graph. Can we find a planar geometric embedding of G such that compass routing always produces the shortest path between any pair of vertices of G? As we will see in Chapter 4, any tree can be geometrically embedded such that compass routing always finds the shortest path between any pair of vertices; however the same is not true for both planar and outerplanar graphs.

In Chapter 5, we study orthogonal geometric embeddings. An orthogonal geometric embedding is defined to be an embedding in which every edge is either horizontal or vertical (also called rectilinear geometric embedding). We will look at orthogonal tree embeddings and balanced trees (a tree in which every node except the leaves has degree 4).

In Chapter 6, we study a slight variation of the original problem; namely, when can we guarantee *a path* (not necessarily the shortest) between any pair of vertices using compass routing. Here, we look at two

cases; convexly embedded geometric graphs (a graph in which every face except the exterior one is a convex polygon) and Delaunay triangulations. As we will see, we use local geometric information to compute a path between any pair of vertices in both cases; that is, the algorithms presented for both cases are "local algorithms".

Finally, in Chapter 7, we present a routing algorithm which we call Algorithmic Geometric Routing which, given any convexly embedded geometric graph, and using only the positions of starting and destination points produces *a path* between them. Related work has appeared lately in the context of routing in wireless networks. In [Stoj98], the authors presented a new location based Geographic Distance Routing algorithm. They use concepts similar but different to those studied in this thesis.

Some of the results in this thesis are presented in [Kran98].

Chapter 2

Shortest Path Problem

In this chapter, we look at a classical and fundamental problem in graph theory, namely the shortest path problem. This problem has generated much interest since the conception of the idea in the 1950's and hence, a lot of research has been carried out for finding the possible solutions to it.

The shortest path problem also appears as a sub-problem of other optimization problems, particularly those concerning transportation and communication networks. This problem can be solved efficiently by numerous algorithms that have evolved during recent years. Some examples include the algorithms by Dijkstra, Ford, Floyd and Dantzig. All these algorithms have a polynomial time complexity.

2.1 Shortest Path Problem

Suppose that a motorist wishes to determine the shortest possible route from Ottawa to Vancouver. Given a road map of Canada containing the distance between each pair of adjacent intersections, how can we find this shortest path?

One possible method is to list all the routes from Ottawa to Vancouver, compute the total distance by adding up the distances on each

route, and choose the shortest. It is easy to see that there are too many possibilities, most of which are simply worth not considering. For example, the route from Ottawa to Vancouver through Halifax is obviously a poor choice.

In a shortest-path problem, we are given a directed graph G=(V,E) along with a weight function W: E ---> R which maps edges to real-valued weights. The weight of a path $p=[v_0,v_1,\ldots,v_k]$ is the sum of the weights of its constituent edges i.e.,

$$w(p) = \sum_{i=1}^{k} w(v_{i-1},v_i)$$

We can define the weight of a shortest path from vertex u to vertex v by

$$\delta(u,v) = \begin{cases} \text{MIN}\{w(p): u \rightarrow v\} & \text{if there exists a path from u to v} \\ \infty & \text{otherwise} \end{cases}$$

The problem of determining the shortest path from vertex u to vertex v can then be defined as finding any path with weight $w(p) = \delta(u,v)$.

Now recall the Ottawa-to-Vancouver example. One can model the road map as a graph in which the intersections are represented by vertices, the road segments between intersections by edges, and the road distances by edge weights. The goal is to determine the shortest path from a given intersection in Ottawa to a given intersection in Vancouver.

Other than distances, the weights of edges can also be interpreted as metrics. They are often used to represent time, cost, penalties, loss or waste, or any other quantity which accumulates linearly along a path and which one seeks to minimize.

As mentioned before, there are several different algorithms that determine the shortest path between two vertices in a weighted

(unweighted) graph. For complete details please refer to [John90], [Rose91], [Corm90] and [Gold91]. We will look at one of these algorithms, namely Dijkstra's algorithm, in detail:

2.1.1 Dijkstra's Algorithm

This algorithm solves the single-source shortest-path problem given a weighted, directed (undirected) graph G=(V,E) for the case in which all the edge weights are positive numbers. As a result, we assume that the weights are positive numbers (i.e., $w(u,v) > 0$) and that we wish to determine a shortest path from source s to destination d.

Dijkstra's algorithm involves labelling all the vertices. Let us denote the label of a vertex v by L(v). At any given moment, some of the vertices will have temporary labels and the remaining ones will have permanent labels. The vertices with temporary labels are stored in a set, say T. Initially, all the vertices except s have temporary labels. During each iteration of the algorithm, the status of one label changes from temporary to permanent. As soon as a vertex receives a permanent label it is circled. At the termination of the algorithm, the vertex d receives a permanent label. The label L(d) of vertex d is the length of a shortest path from s to d.

Input: A connected, weighted graph in which all edge weights are positive.

Output: L(d), the length of a shortest path from vertex s to vertex d.

Step 1: [Initialization] Set $L(s):=0$. For all vertices x adjacent to s, set $L(x):=w(s,x)$. For all vertices x not adjacent to s, set $L(x):=\infty$. Let $S:=\{s\}$ and $T:=V(G)-\{s\}$.

Step 2: [Done?] If $d \notin T$, then stop. (L(d) gives us the length of a

shortest path from s to d).

Step 3: [select next vertex] Choose $v \in T$ such that L(v) is the smallest and then set $T:=T-\{v\}$ and set $S:=S+\{v\}$.

Step 4: [Revise labels] For every vertex $x \in T$ which is adjacent to v, set

$L(x):=MIN\{L(x), L(v)+w(v,x)\}$

Now go to step 2.

Now we give the proof of correctness of Dijkstra's algorithm as follows:

We will use induction on the number of iterations of the loop (which consists of steps 2-4) to show that before step 2 is executed:-

(i) The label on a circled vertex v gives the length of a shortest path from s to v.

(ii) There is a shortest path from s to a circled vertex v consisting only of circled vertices.

(iii) The label on an uncircled vertex v gives the length of a shortest path of the form $(s, v_1, ..., v_m)$, where $v=v_m$ and $s, v_1, ..., v_{m-1}$ are circled. If there is no path of the form $(s, v_1, ..., v_m)$ where $v=v_m$ and $s, v_1, ..., v_{m-1}$ are circled then the label on v is ∞.

Note that this algorithm terminates as soon as vertex d is circled, therefore it follows from (i) that L(d) gives us the length of a shortest path from s to d as required.

Initially, the vertex s has the label 0, for any vertex x which is adjacent to s has the label w(s,x) and all other vertices have label ∞. At this point (i), (ii) and (iii) hold. Hence, the basis step has been verified.

Now assume that (i), (ii) and (iii) hold the ith time that we arrive at step 2. Consider an additional iteration of the algorithm. First, at step 3, the uncircled vertex v having the minimum label is circled. We first show that (i) holds for vertex v, i.e., the label L(v) now gives the length of a shortest path from s to v. Suppose that there is a path

(iv) $(s, v_1, ..., v_m)$,

where $v=v_m$ and whose length is less than L(v). According to (iii), we must have some vertex v_j (where $1 \leq j \leq m$) uncircled. Let k be the least index with v_k uncircled. Since every edge has positive weight, the length L' of the path

(v) $(s, v_1, ..., v_k)$

is less than the length of path (iv). In (v), v_k is the only uncircled vertex therefore the label on v_k is at most L' using (iii). Since $L' < L(v)$ therefore we have contradicted the choice of v. Hence, (i) holds for v. A similar argument shows that (ii) holds for v.

Finally, we prove that after the labels of the uncircled vertices adjacent to v are updated in step 4, (iii) holds. Suppose that we have an uncircled vertex x which is adjacent to v. There exist two types of path of the form

(vi) $(s, v_1, ..., v_m)$, where $x=v_m$ and only $s, v_1, ..., v_{m-1}$ are circled.

namely those in which $v_{m-1}= v$ and those in which $v_{m-1}\neq v$.

First we consider the case when $v_{m-1}\neq v$. Recall that according to (ii) there exists a shortest path from s to v_{m-1} consisting of only circled vertices, none of which is v. The length of this path is equal to the length of the subpath $(s,v_1,...,v_{m-1})$ in (vi). Hence, the length of a shortest path in (vi) with $v_{m-1}\neq v$ is L(x).

When $v_{m-1}=v$ then a shortest path of the form (vi) is obtained by taking a shortest path from s to v with length L(v), then appending the edge (v,x). The length of this path is L(v)+w(v,x). Thus the length of a shortest path of the form (vi) is

MIN{L(x), L(v)+w(v,x)}.

Therefore, after updating the labels of the uncircled vertices adjacent to v in step 4, (iii) holds. We now have verified that (i), (ii) and (iii) hold prior to the (i+1)st execution of step 2. The inductive step is complete. Hence, the algorithm is correct.

Theorem 1 *Given a simple, connected, weighted graph G on v vertices and letting f(n) denote the number of times Dijkstra's Algorithm examines an edge of G in the worst case, then* $f(n)=O(n^2)$.

Proof: Note that Dijkstra's Algorithm examines edges at step 4. Since the graph G has n vertices, the maximum number of edges examined at step 4 is n−1. Note that step 4 is inside a loop consisting of steps 2, 3 and 4. This loop is executed as long as $d\in T$. Since in each iteration of this loop, one element is removed from T, and initially T has n−1 elements, this loop is executed at most n−1 times. Therefore, the number of times Dijkstra's Algorithm examines an edge of graph G is at most $(n-1)^2$. We

have

$$f(n) \leq (n-1)^2 \leq n^2 = O(n^2)$$

as wanted. QED

Chapter 3

Definitions

The purpose of this chapter is to formally explain the notions of geometric graph and compass routing along with some other definitions.

Let us begin by looking at two types of connected and weighted graphs:

(i) the edges of the graph have arbitrary weights (see Figure 3.1).

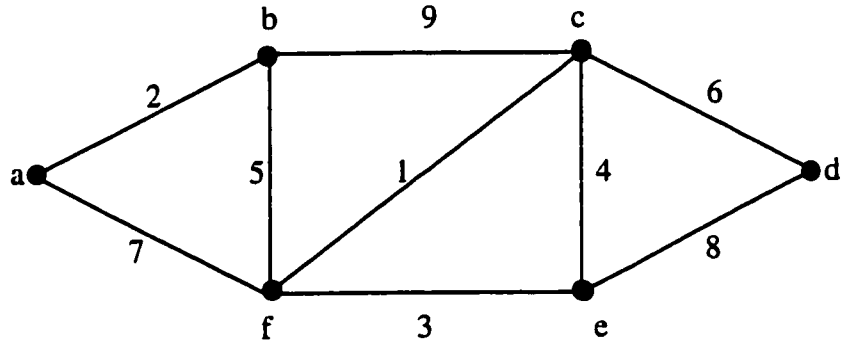


Figure 3.1: Edges have arbitrary weights

(ii) all the edges of the graph have the same weight 1 (see Figure 3.2).

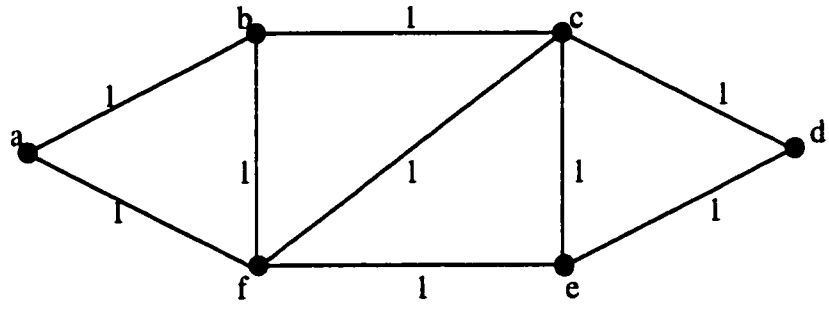


Figure 3.2: All the edges have the same weight 1

In this thesis, we will work with graphs of type (ii), i.e., the graphs in which all the edges have weight 1.

Let us now look at the concept of geometric graph.

Definition 1 *A graph is called planar if it can be drawn on the plane without any edge crossings.*

Now we define a geometric graph as follows:

Definition 2 *A geometric graph is a graph drawn on the plane such that each vertex corresponds to a point and each edge is a closed straight line segment connecting two vertices but not passing through a third vertex.*

The C(n,2) segments determined by n points in the plane, no three of which are collinear, form a complete geometric graph with n vertices (see [Pach95]).

In this thesis, we assume that the edges do not intersect one another, i.e., we will be working with planar geometric graphs throughout our work.

An example of a geometric graph is given in Figure 3.3. Given the set of vertices, their coordinates and the adjacency matrix, we draw a geometric graph as shown.

<a,b,c,d,e,f,g,h>

a(0,8), b(4,7), c(3,4), d(4,1), e(0,0), f(-4,1), g(-3,4), h(-4,7)

	a	b	c	d	e	f	g	h
a	0	1	0	0	0	0	0	1
b	1	0	1	0	0	0	0	1
c	0	1	0	1	0	0	1	0
d	0	0	1	0	1	1	0	0
e	0	0	0	1	0	1	0	0
f	0	0	0	1	1	0	1	0
g	0	0	1	0	0	1	0	1
h	1	1	0	0	0	0	1	0

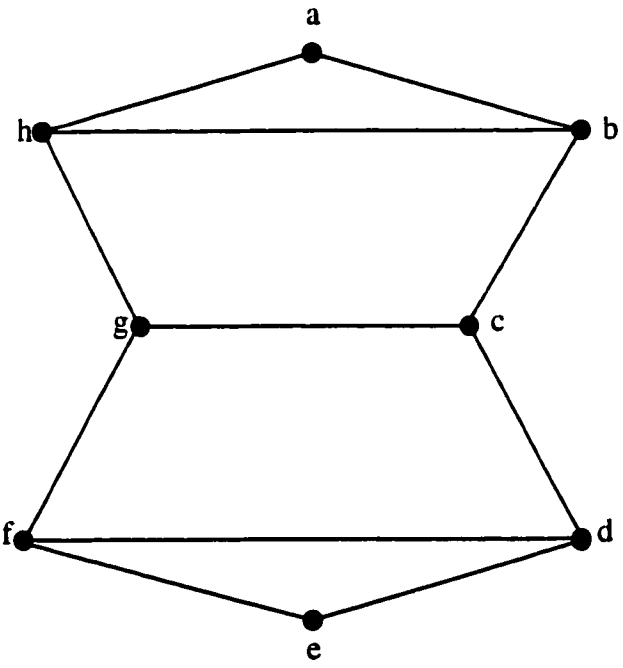


Figure 3.3: A Geometric Graph

In the past there have been numerous studies concerning planar graph embeddings with straight line segments (see Figure 3.4). For

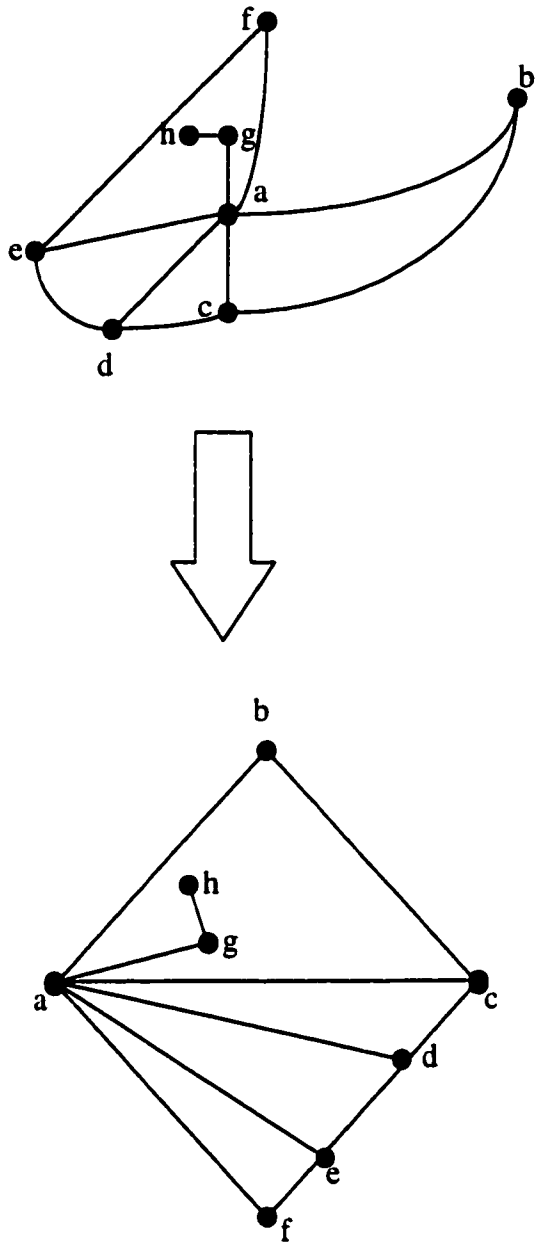


Figure 3.4: Planar Graph embedding

instance, recall that the theorem of I. Fary [Fary48] shows that every plane graph has an embedding (or drawing) in which the edges are straight line segments and the vertices are points in the plane. Starting with the algorithm of Tutte in 1963, there have been many algorithms

presented for straight-line embedding of plane graphs, see [Tutt63], [Chib82] and [Read87]. In [DeFray90], the authors prove that every plane graph having n vertices has a straight-line embedding on the (2n–4) by (n–2) grid and an O(n*log n) algorithm is presented to embed the graph.

In [Cast96], Castañeda and Urrutia looked at straight line embeddings of planar graphs on point sets. Given a finite set P_n of n points in the plane in general position, we say that P_n supports an n-vertex planar graph G if there exists an embedding of G on the plane such that the vertices of G are mapped to the elements of P_n and its edges to non-intersecting open straight line segments joining pairs of elements of P_n which correspond to the pairs of adjacent vertices in G. Any such embedding is known as a straight-line embedding of G on P_n. If any point set P_n supports G then graph G is said to be a universal graph. The authors in [Cast96] proved that the set of universal graphs is exactly the set of outerplanar graphs. Furthermore, an $O(n^2)$ algorithm is presented which produces planar embeddings of outerplanar graphs on point sets. This result was improved in [Bose97] in which the author presented an O(n*log n) algorithm. Another study which has been motivated by geometric embedding problem is by Frances Yao [Yao91]. Given a set of points S in the plane, define $G_k = (S,E_k)$ to be the k-nearest-neighbors graph of S where $(u,v) \in E_k$ if u is one of the k nearest neighbors of v. The author in [Yao91] has presented some results regarding the graph properties of G_k. In particular, the bounds on the diameter of the graph G_k have been derived.

Suppose that we have to traverse a path in a geometric graph G starting at its vertex s and ending at vertex d. We say that we apply *compass routing* if we start at the edge incident to s having the smallest angle with the segment joining s and d. This principle is applied at every vertex we arrive at until vertex d is reached.

We express this formally as follows:

Definition 3 *We say that the path $s=v_0,v_1,...,v_n=d$ of a geometric graph G was obtained using compass routing if for all $i=1,2,...,n-1$ the edge $v_{i-1}v_i$ has the smallest angle with segment $v_{i-1}v_n$ among all the edges incident to v_{i-1}.*

In this thesis, we are interested in the geometric graphs in which applying compass routing would result in a shortest path between s and d.

Definition 4 *We say that a geometric graph G supports compass routing if for every pair of vertices s and d of G, compass routing (starting at s) produces a shortest path between s and d.*

As mentioned before, the term "shortest path" is expressed according to the link distance function, i.e., the distance refers to the number of edges in a path.

It is interesting to note that given a geometric graph, compass routing may fail for either of two different reasons:

(i) We never reach our destination d. This case is illustrated in Figure 3.5. Observe that in this embedding of the pentagon, compass routing fails to produce a path from s to d since $\alpha < \beta$ and hence we get into an infinite loop between the vertices s and b.

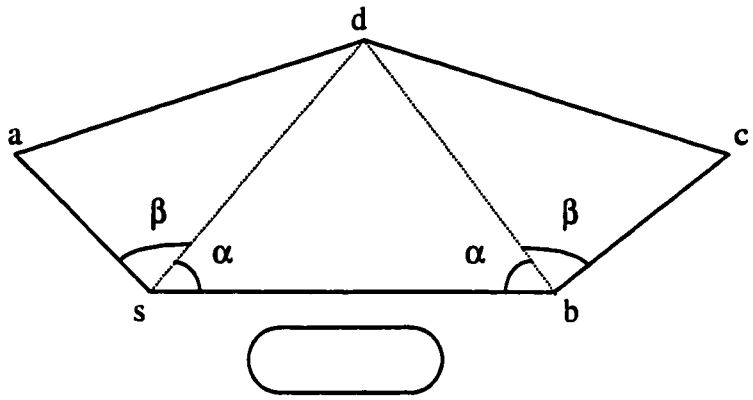


Figure 3.5: The destination d is never reached.

(ii) Compass routing produces a valid path between the vertices s and d but it is not the shortest path. This case is illustrated in Figure 3.6. Observe that in this embedding of the pentagon, compass routing does not produce the shortest path from s to d.

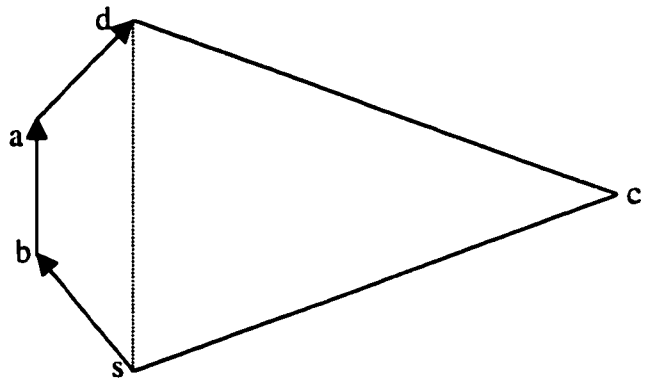


Figure 3.6: The path between s and d is not the shortest path.

Observe that for the geometric graph in Figure 3.5, compass routing works for any pair of vertices except (s,d) and (b,d). It is worth noting that while compass routing does not work for pair (s,d), it does work for pair (d,s).

We shall now introduce the concept of single source compass routing (SS-CR).

Definition 5 *Given a geometric graph G and a vertex $s \in G$ we say that G supports single source compass routing from s if for any vertex $d \neq s$, compass routing finds a shortest path between s and d.*

Observe that the geometric graph shown in Figure 3.5 supports single source compass routing from d but not from s.

Note that the following embedding of the pentagon is optimal since compass routing produces the shortest path between any pair of vertices, unlike the embeddings in Figures 3.5 and 3.6.

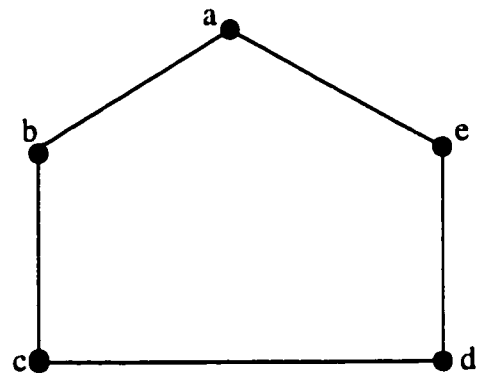


Figure 3.7: An optimal embedding of the pentagon

Now we look at single source compass routing for any given class of graphs.

Definition 6 *We say that a given class of graphs supports single source compass routing if, for any graph G of the class and any vertex $s \in G$,*

there exists a geometrical embedding of G that supports single source compass routing from s.

Observe that, trivially, K_n, the complete graph on n vertices, supports compass routing.

It is also easy to prove the following:

Theorem 2 *The class of polygons supports compass routing.*

Proof: Each graph of this class can be represented as a regular polygon in the plane (see Figure 3.8).

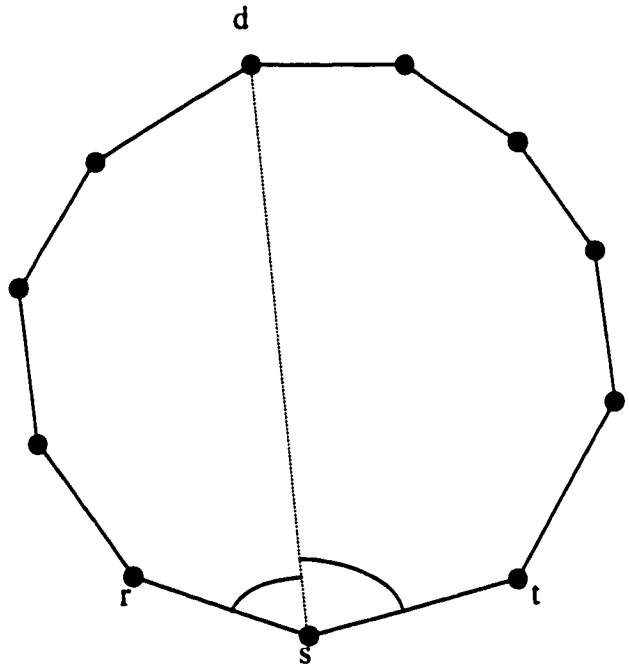


Figure 3.8: A convex polygon with 11 edges

Observe that for any two vertices s and d of this polygon and any two neighbours r and t of the vertex s, $\angle rsd < \angle tsd$ if and only if distance(r,d) < distance(t,d). QED

Finally, we define the concepts of SS-CR embedding and CR-embedding.

Definition 7 *Given a vertex s of a graph G, a SS-CR embedding of graph G is defined to be a geometric embedding which supports single source compass routing from s.*

Definition 8 *A CR-embedding of graph G is defined to be a geometric embedding which supports compass routing.*

Chapter 4

Geometric Embeddings

In this chapter, we study the geometric embedddings of planar graphs, trees and outerplanar graphs.

4.1 Planar Geometric Embeddings

In this section, we study the following question:

Does there exist a CR-embedding for every planar graph? We prove:

Theorem 3 *There does not exist a CR-embedding for every planar graph.*

The above theorem is proved in Section 4.3, in which we show that not every outerplanar graph has a CR-embedding. Since outerplanar graphs are a subset of planar graphs, it follows that the above theorem holds. On the other hand we prove in the next section that there does exist a CR-embedding for every tree.

4.2 Tree Geometric Embeddings

In this section, we study the following:

Theorem 4 *Every tree has a CR-embedding.*

Proof: We will prove the result for balanced trees with respect to r. For non-balanced trees, we can add dummy vertices to balance them such that the degree of every non-leaf vertex is greater than or equal to 3. We then find a CR-embedding of the resulting tree and then delete the dummy vertices at the end.

Let $T_{k,L}$ be a rooted tree with root r such that each vertex except its leaves has k children and the diameter of $T_{k,L}$ is 2*L, i.e., the distance from any leaf of $T_{k,L}$ to the root is L.

We now prove by induction on L that $T_{k,L}$ has a CR-embedding.

We will show that there exists a geometric embedding of $T_{k,L}$ ($k \geq 3$) which supports compass routing such that there exists a family of concentric circles $C_1,...,C_L$ centered at r with diameters $d_1 < d_2 < ... < d_L$ such that all the vertices of $T_{k,L}$ which are at distance j from r lie on the circle C_j where $1 \leq j \leq L$.

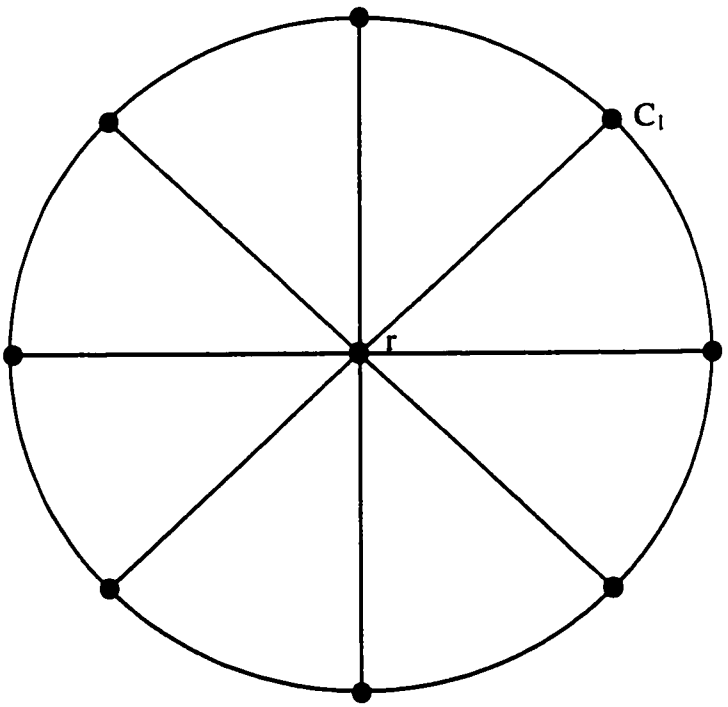


Figure 4.1: Layer L = 1

Let L = 1. For our first iteration, we place r at the center of a circle C_1 and its children at equidistant points on C_1 as shown in Figure 4.1. Now suppose that we have an embedding of $T_{k,L}$ on $C_1,...,C_L$ which supports compass routing. Next we show how to extend it to an embedding of $T_{k,L+1}$ which supports compass routing.

First we determine the placement of a circle C_{L+1} as follows:

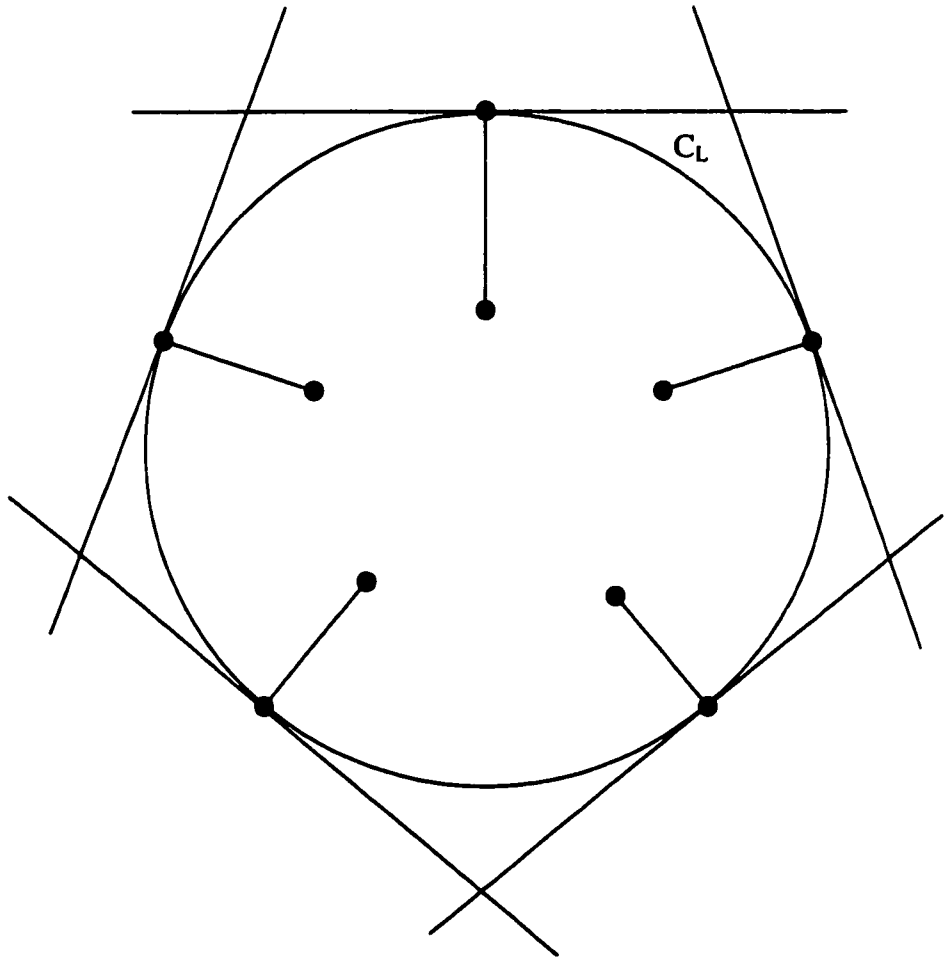


Figure 4.2: The circle C_L and the tangent points

For every $v_i \in C_L$, we draw a tangent line to circle C_L at v_i as shown in Figure 4.2. Notice that these tangent lines form a convex polygon enclosing C_L. Here, we assume that there are always enough vertices on

C_L so that the tangent lines form a convex bounded polygon. Now we place circle C_{L+1} around C_L such that all the vertices of the convex polygon are left outside the circle C_{L+1} as shown in Figure 4.3.

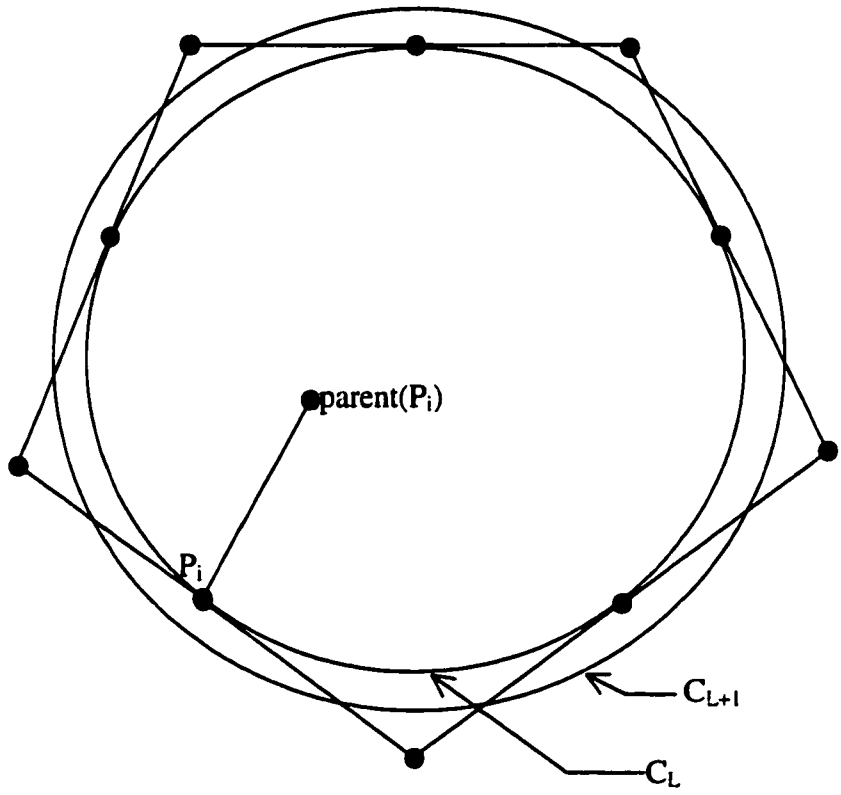
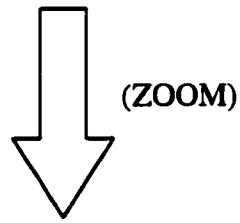


Figure 4.3



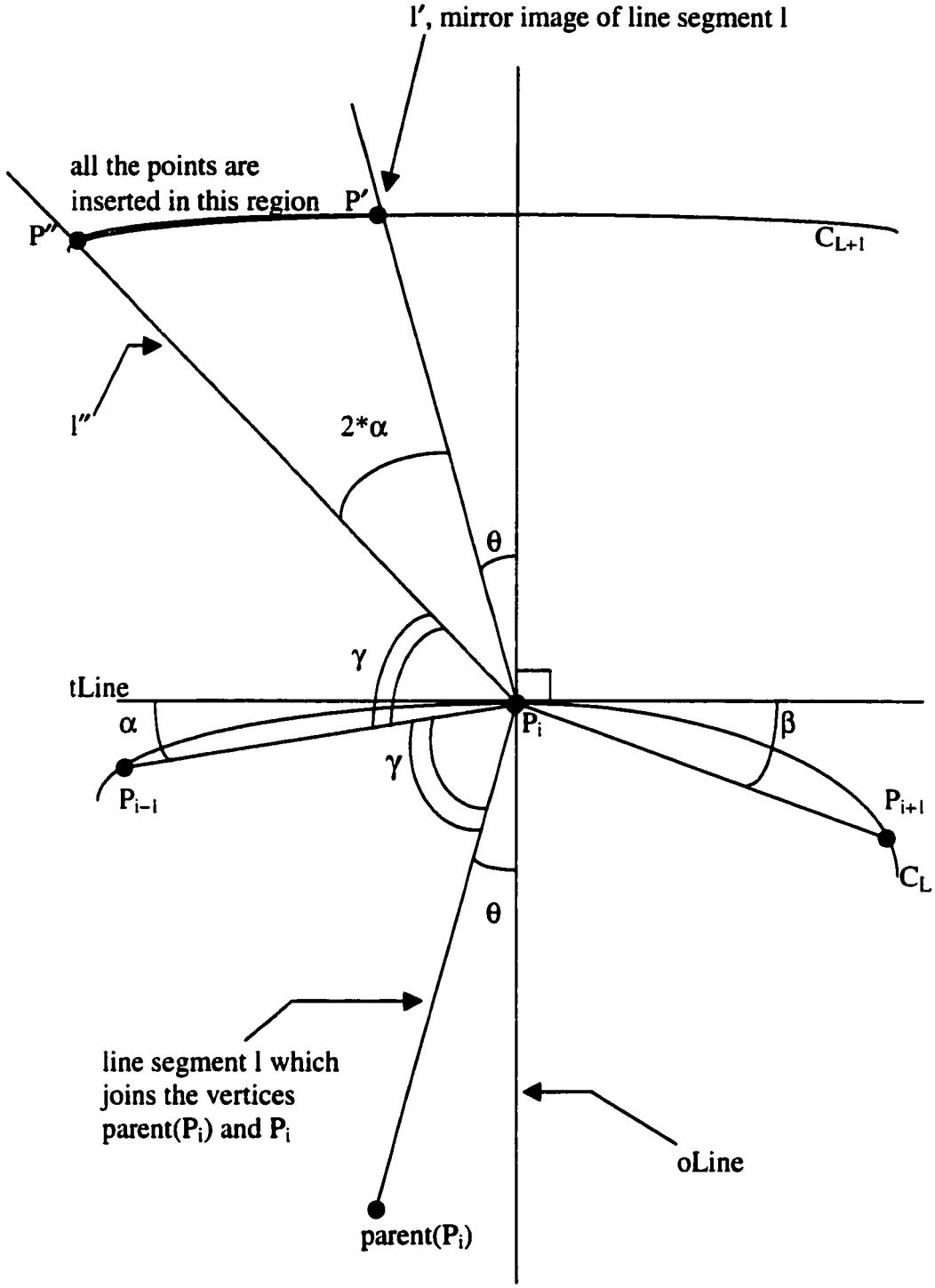


Figure 4.4: Proving the inductive step

Let P_i be any point on the circle C_L representing a vertex of $T_{k,L}$ at a distance L from r (see Figure 4.4). We would like to find the region on the circle C_{L+1} in which to place the children of point P_i such that the

resulting graph supports compass routing. We first draw a tangent line tLine to the circle C_L at point P_i and then draw another line segment oLine orthogonal to tLine at P_i. Without loss of generality, we assume that tLine is parallel to the x-axis and oLine is parallel to the y-axis. Let the points P_{i-1}, P_i and P_{i+1} be three consecutive points on circle C_L. Let α be the angle between tLine and the line segment joining the points P_{i-1} and P_i and let β be the angle between tLine and the line segment joining the points P_i and P_{i+1}. Let l be the line segment joining the points parent(P_i) and P_i. We now take the mirror image l′ of l with respect to tLine as shown in the figure above. Let γ be the angle between the line segment joining the points P_{i-1} and P_i and l.

Now we draw a straight line l″ from P_i such that the angle between l″ and the line segment joining P_{i-1} and P_i is γ. Notice that the size of the angle between l″ and l′ is $2*\alpha$. Let P′ be the point where the line segment l′ intersects the circle C_{L+1} and let P″ be the point where the line segment l″ intersects the circle C_{L+1}. We place all the children of point P_i on the arc joining the points P′ and P″.

Let u and v be any two vertices of $T_{k,L}$. Let $\delta_{u,v}$ be the smallest angle that an edge incident to u forms with the dotted line segment

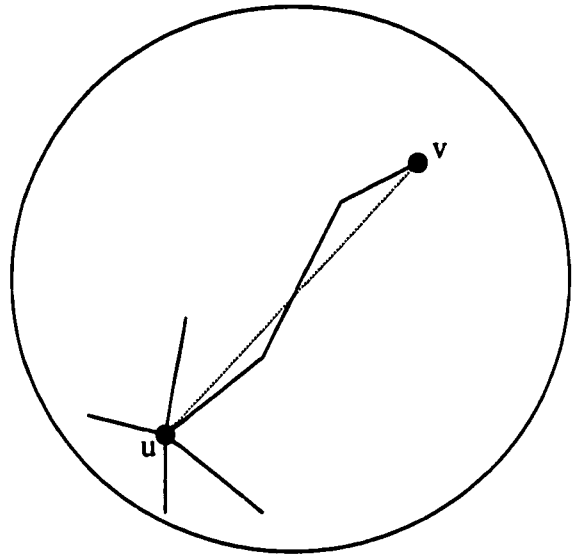


Figure 4.5

joining vertices u and v (see Figure 4.5). Let $\beta_{u,v}$ be the second smallest angle that an edge incident to u forms with the dotted line segment joining vertices u and v. Now we define $\varepsilon_{u,v} = \beta_{u,v} - \delta_{u,v}$. Next we define

$$\varepsilon = \text{MIN } \{\varepsilon_{u,v} \text{ where } u,v \in V(T_{k,j}\}.$$

We now make a final modification to the choice of $C_{k,L+1}$ as follows:

We shrink the radius of circle C_{L+1} so that the angle formed between any vertex P at P of the embedding of $T_{k,L}$ and any two children of any point P_i on C_L is strictly less than ε where $P \neq P_i$.

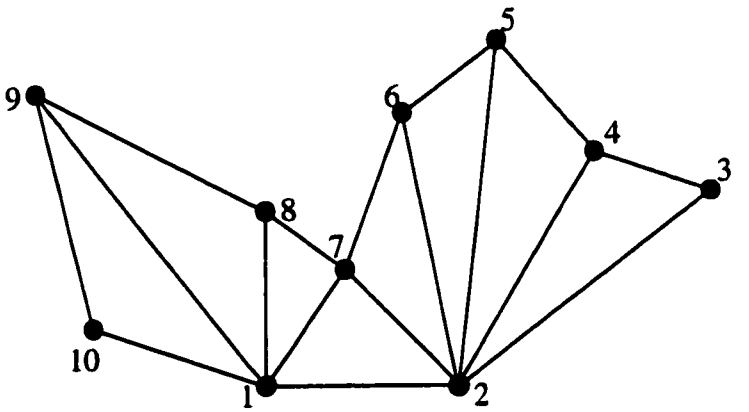
Now we show that with this final choice of C_{L+1} we obtain an embedding of $T_{k,L+1}$ which supports compass routing. Again letting u and v be any two vertices of $T_{k,L+1}$, the following two cases arise:

(i) both u and v are at distance at most L from r. It is easy to see that if neither of u and v are at distance L from r then the path produced by compass routing from u to v will be unaffected. Suppose then that at least one of them, say u, lies on C_L. By the choice of the position of the children of u, we can verify that when travelling from u to v, compass routing will not choose any of the edges connecting u to one of its children.

(ii) at least one of u or v lies on C_{L+1}. Suppose that both of them lie on C_{L+1} and that we want to go from u to v. In the first step, compass routing chooses the edge joining u to its parent p(u). We now observe that by the choice of ε compass routing will choose edges as if we were travelling from p(u) to the parent of v, namely p(v). Once we arrive at p(v) we will take the edge connecting it to v. A similar analysis can be done for the case when only one of u or v lies on C_{L+1}. QED

4.3 Outerplanar Geometric Embeddings

An outerplanar graph is a graph which can be embedded on the plane in such a way that its vertices are the vertices of a convex polygon and its edges are represented by a set of non-intersecting straight line segments. An example of an outerplanar graph appears in Figure 4.6.



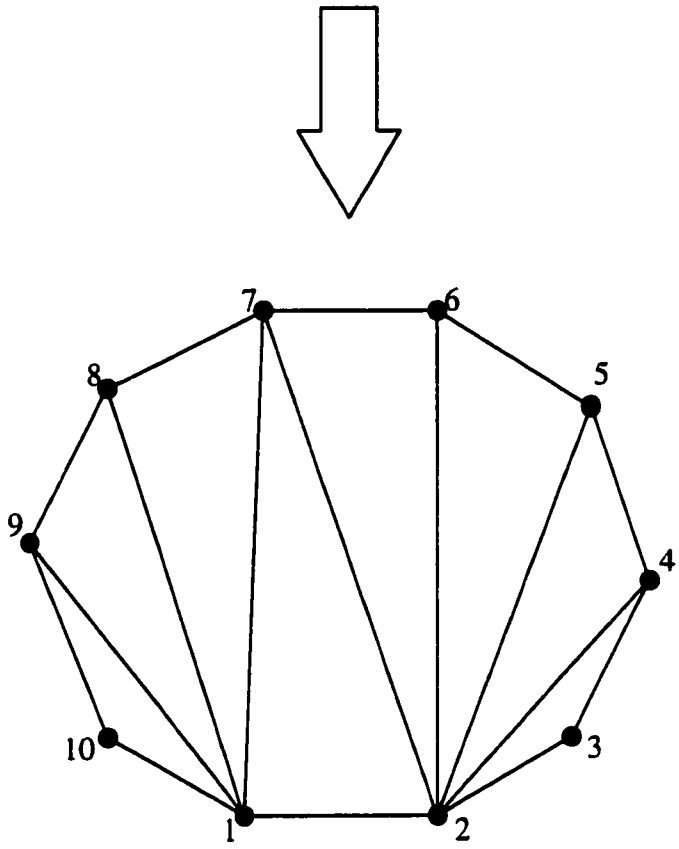


Figure 4.6: An Outerplanar Graph

Now we ask the following question:

Does there exist a CR-embedding for every outerplanar graph?

Theorem 5 *Not every outerplanar graph has a CR-embedding.*

Proof: We start by showing that convex embeddings of outerplanar graphs do not support compass routing. Consider an embedding of a fan F_n in which all its vertices form the vertices of a convex polygon, where a fan F_n is a maximal outerplanar graph with a vertex v, called its apex, adjacent to all the other vertices of F_n (see Figure 4.7). We will assume

that the vertices of F_n are labelled $v_0,v_1,...,v_n,v$ such that v_i is adjacent to v_{i+1} where $0 \leq i \leq n-1$ and v is adjacent to v_i where $0 \leq i \leq n$.

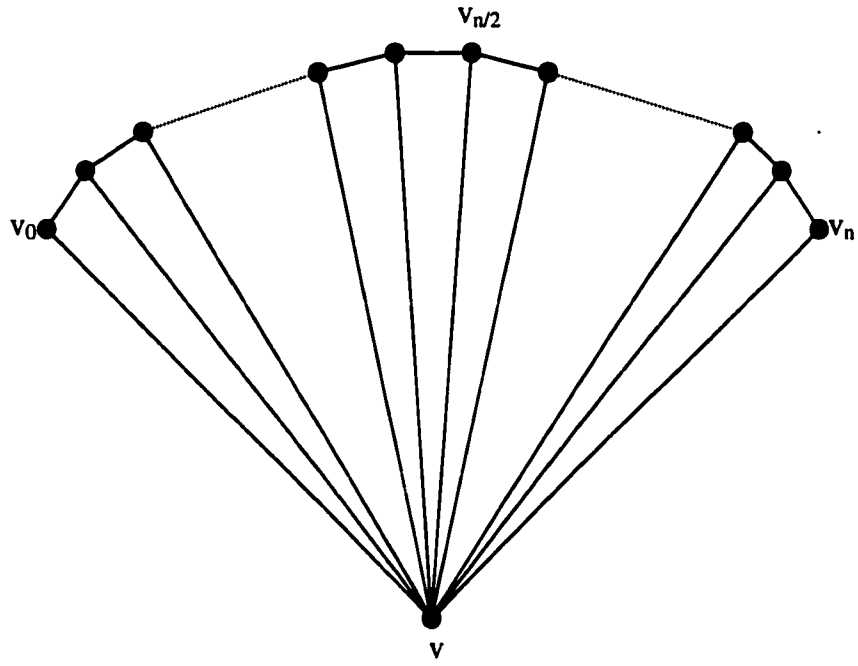


Figure 4.7: A convex fan

We need the following result before completing our proof.

Lemma 1 For every $\varepsilon > 0$ there is an n_0 such that any convex polygon with $n \geq n_0$ vertices contains eight consecutive vertices such that their interior angles are greater than or equal to π-ε.

Proof: Let $\varepsilon > 0$ be a constant. Observe that any convex polygon contains at most $m_\varepsilon = \lceil 2*\pi / \varepsilon \rceil$ vertices with interior angles smaller than or equal to π-ε.

Let $n_0 = 10*m_\varepsilon$. It now follows that we can select eight consecutive vertices such that their interior angles are greater than or equal to π-ε.

Lemma 2 *If n is very large then no convex embedding of F_n supports compass routing.*

Proof: Let $\varepsilon > 0$ and $n > n_0$. Suppose that we have a convex embedding of F_n, that is, an embedding of F_n such that all the vertices of F_n are the vertices of a convex polygon which supports compass routing. By Lemma 1, we pick eight consecutive vertices $v_i,...,v_{i+7}$ from n+1 vertices such that each interior angle is greater than or equal to $\pi-\varepsilon$ where ε is very small. In other words, the eight consecutive vertices which we select are almost on a straight line (see Figure 4.8).

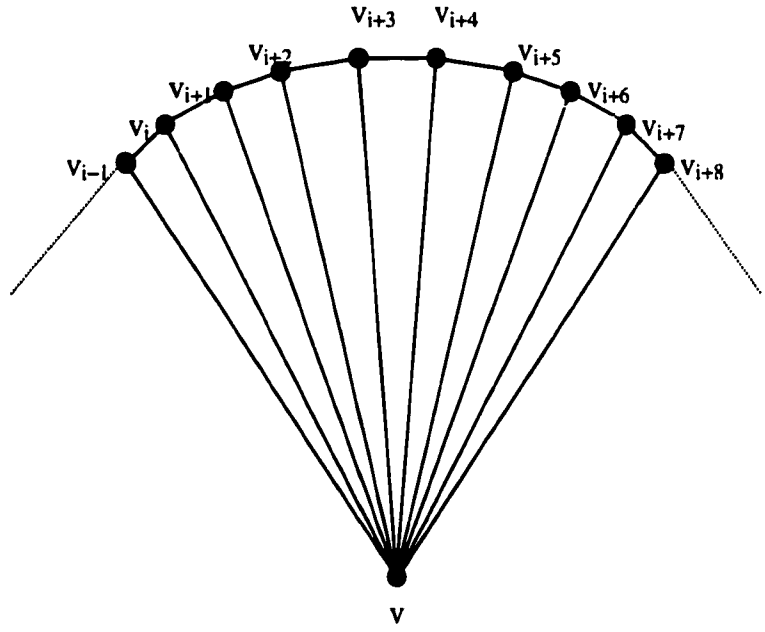


Figure 4.8: Choosing eight consecutive vertices

Observe now that $\angle v_{i+2}v_{i+3}v_i$ has size at most 2ε. Now suppose that we want to go from v_{i+3} to v_i. Notice that the shortest path from v_{i+3} to v_i is of length 2 via v, and thus $\angle v_iv_{i+3}v_i$ is smaller than $\angle v_{i+2}v_{i+3}v_i$ which is smaller than 2ε (see Figure 4.9), that is, compass routing chooses the edge $v_{i+3}v$ followed by the edge vv_i to go from v_{i+3} to v_i.

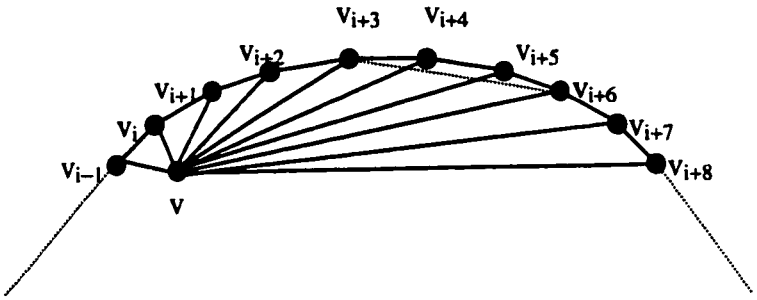


Figure 4.9

Now suppose that we want to go from v_{i+3} to v_{i+6}. It is now easy to see from Figure 4.9 that $\angle v v_{i+3} v_{i+6}$ is almost π and thus compass routing chooses the edge $v_{i+3}v_{i+4}$. However, the shortest path from v_{i+3} to v_{i+6} is of length 2 via v. Hence, we have a contradiction.

Next we show that the non-convex embedding of F_n does not support compass routing. We consider only the case in which the angle $v_0 v v_n$ is less than π with a large number of vertices (see Figure 4.10).

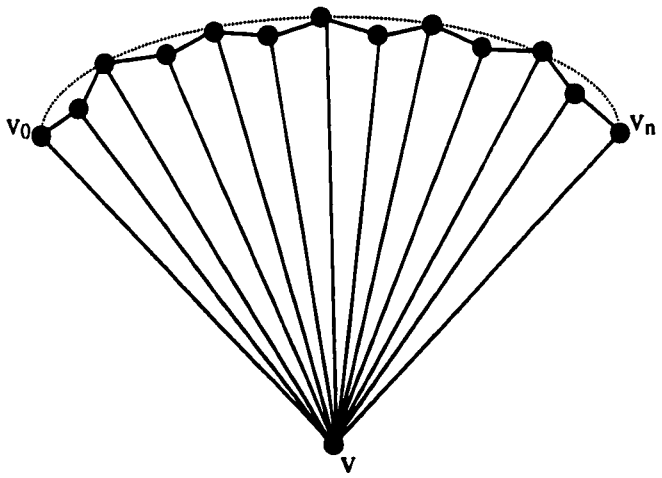


Figure 4.10: A non-convex fan with very large n

Let *Conv*($v,v_0,...,v_n$) be the convex hull of $\{v,v_0,...,v_n\}$.

Lemma 3 *At least half of vertices $\{v_0,...,v_n\}$ must be convex hull vertices of Conv($v,v_0,...,v_n$).*

Proof: Let P_m be the convex hull of the embedding of F_n and let v_i and v_{i+j} be the two consecutive vertices on P_m as shown in Figure 4.11.

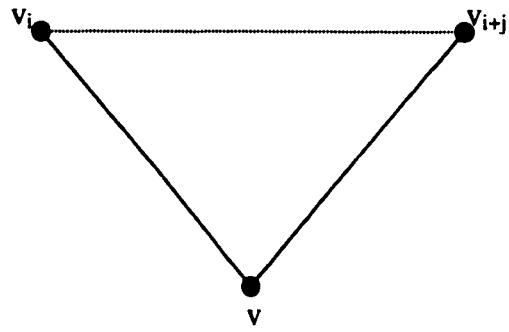


Figure 4.11: v_i and v_{i+j} are consecutive convex hull vertices

We want to show that $j \leq 2$, that is, if v_i is a convex hull vertex then either v_{i+1} or v_{i+2} must be a convex hull vertex as well. Suppose that this is not the case, that is, $j > 2$.

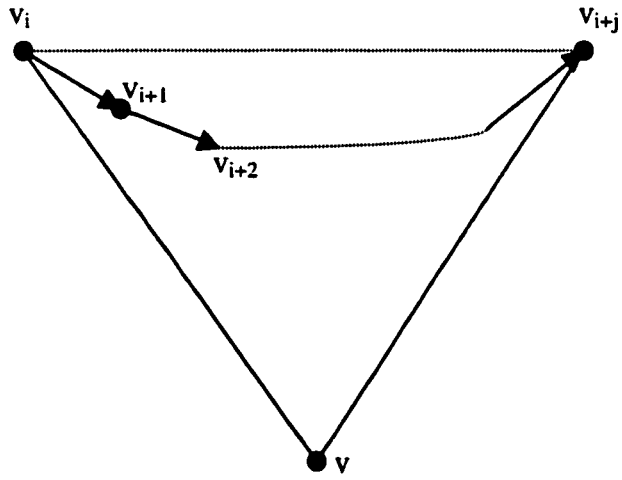


Figure 4.12: v_{i+1} and v_{i+2} are not convex hull vertices

Now suppose that we want to go from v_i to v_{i+j} using compass routing (see Figure 4.12). Observe that using compass routing we will first go to vertex v_{i+1}. Notice that vertex v_{i+j} is not adjacent to vertex v_{i+1} since $j > 2$. Hence, compass routing will produce a path with length greater than or equal to 3 from v_i to v_{i+j}. However, the shortest path from v_i to v_{i+j} is of length 2 via v. Thus at least half of vertices $\{v_0,...,v_n\}$ are convex hull vertices (see Figure 4.13).

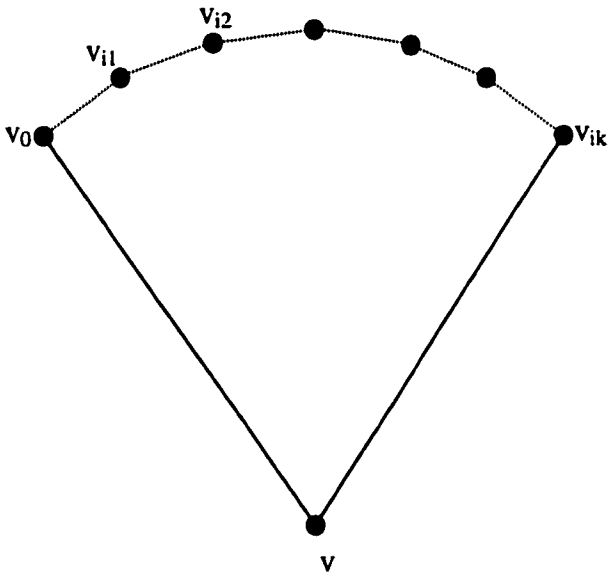


Figure 4.13

Note that in the above figure, $|i_t - i_{t+1}| \leq 2$ where $0 \leq t \leq k-1$.

This completes our proof of the lemma.

Using the same technique described in Lemma 1, we now pick twelve convex hull vertices which are almost on a straight line. From these vertices we choose the middle eight vertices. In these middle eight vertices there exists a non-convex hull vertex, say v_y, and we select three vertices on each side of v_y as shown in Figure 4.14.

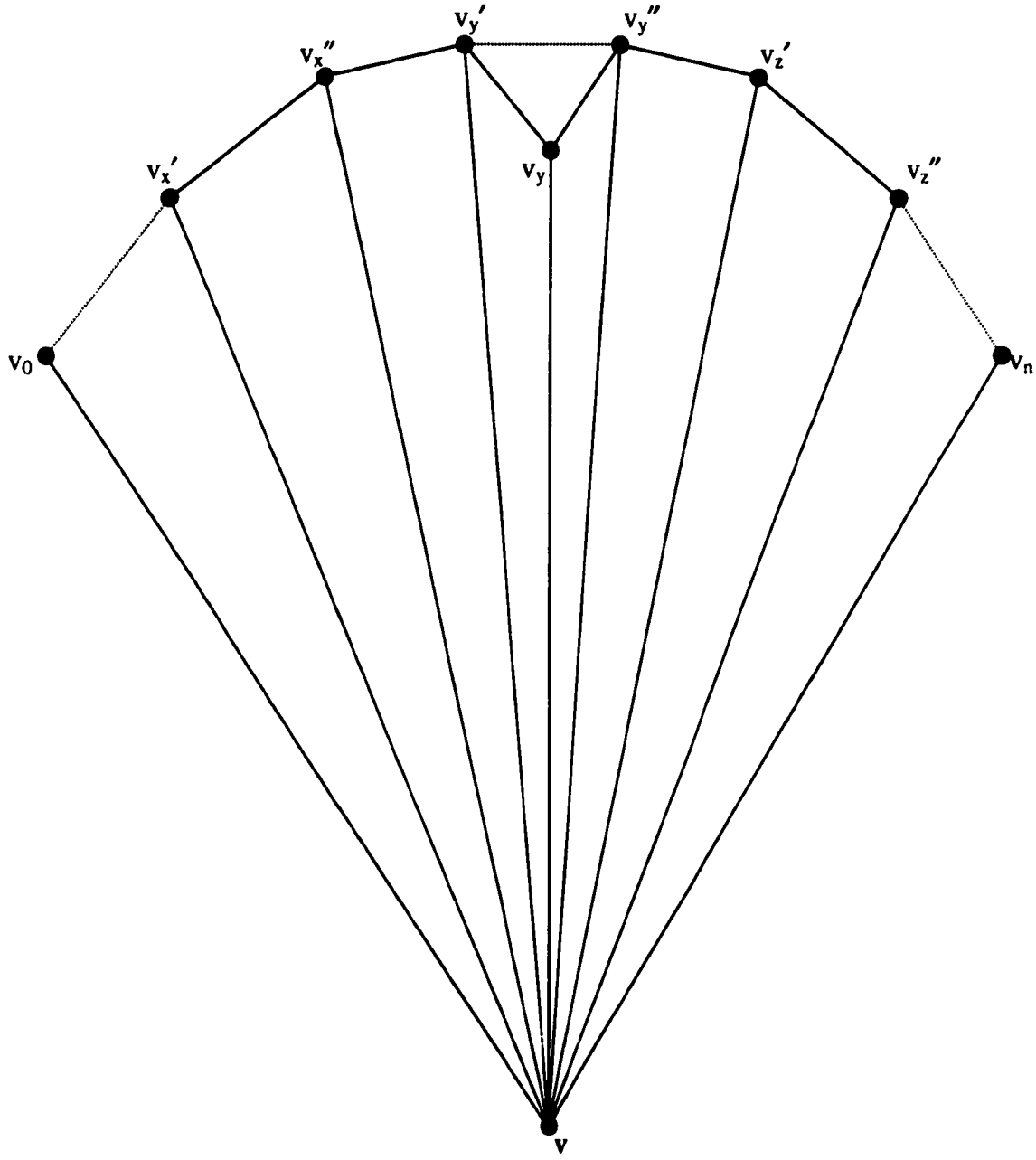


Figure 4.14

Notice that there are two cases concerning the location of vertex v_y namely (i) v_y is outside the triangle $v_y'v_y''x$ and (ii) v_y is inside the triangle $v_y'v_y''x$ where the point x is the intersection point of the dotted line segments joining v_y' to v_z'' and v_y'' to v_x'.

Let us now consider Case (i) in which the vertex v_y is outside the triangle $v_y'v_y''x$ (see Figure 4.15).

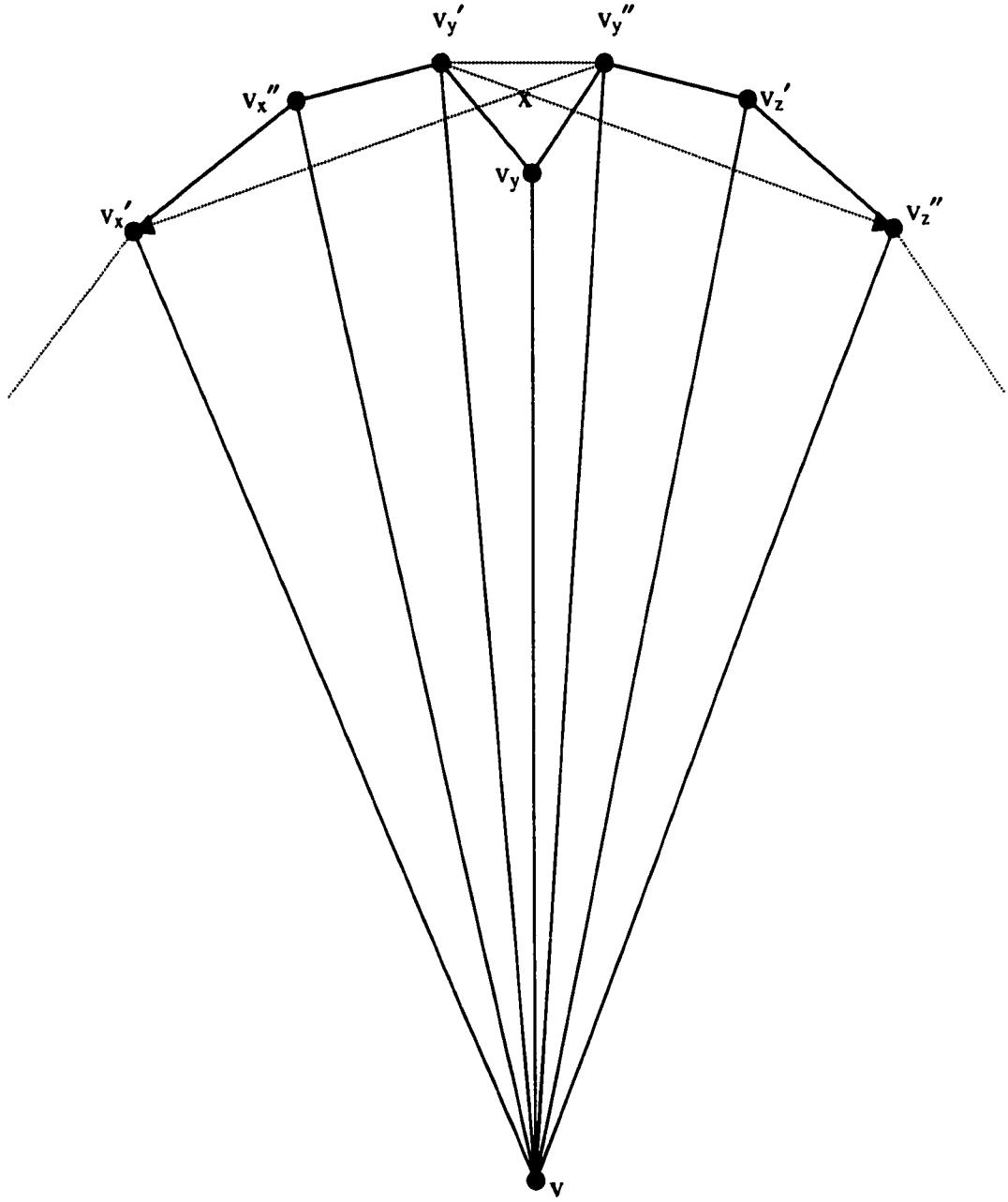


Figure 4.15 Case (i)

Suppose that we want to go from v_y'' to v_x' using compass routing. Notice that we choose the edge $v_y''v_y$ since it forms the smallest angle of all the edges incident to v_y'' with the dotted line segment joining the

vertices v_y'' and v_x'. Therefore, we arrive at vertex v_y. Note that v_x' is not adjacent to v_y. Hence, this is not the shortest path, since the shortest path from v_y'' to v_x' is 2 via v. As a result, we have a contradiction.

Let us now consider Case (ii) in which the vertex v_y is inside the triangle $v_y'v_y''x$ (see Figure 4.16).

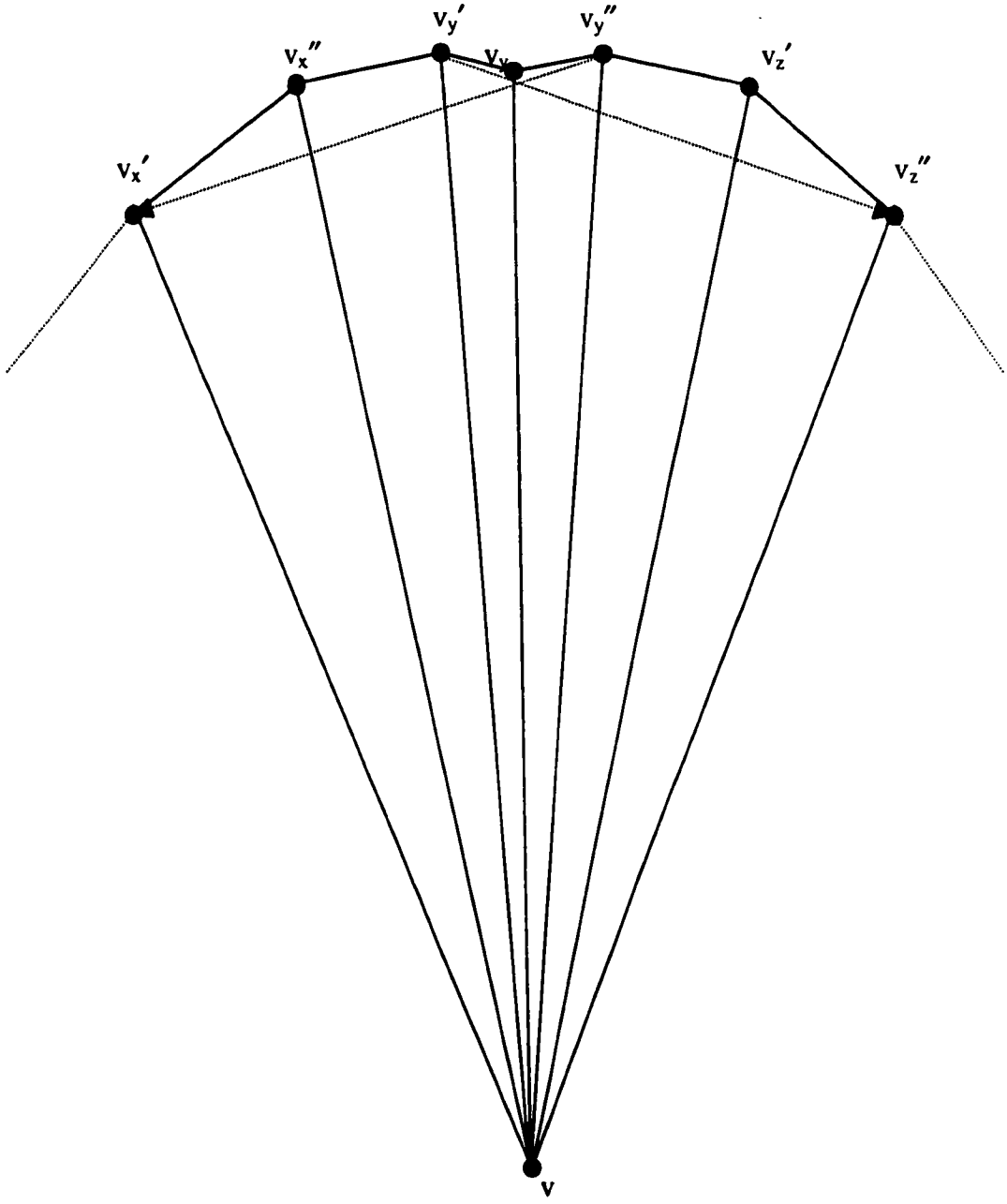


Figure 4.16 Case (ii)

Suppose that we want to go from v_y'' to v_x' using compass routing. Notice that the shortest path from v_y'' to v_x' is of length 2 via v. The vertex v must be placed such that compass routing will pick vertex v to go from v_y'' to v_x', that is, we choose the edge $v_y''v$ followed by the edge vv_x'.

Now suppose that we want to go from v_y' to v_z''. It is now easy to see that $\angle vv_y'v_z''$ is almost π and thus we choose the edge $v_y'v_y$. However, the shortest path from v_y' to v_z'' is of length 2 via v. Hence, we have a contradiction. QED

Chapter 5

The Orthogonal Case

In this chapter, we study the problem of finding orthogonal embeddings of geometric graphs such that several instances of compass routing would work.

Recall that an orthogonal geometric graph is a graph such that all edges are either parallel to the x-axis or the y-axis. Therefore it follows that the maximum degree of any vertex in such graphs is at most 4.

5.1 Orthogonal Tree Embeddings

We start by proving that not all trees in which every vertex has degree at most 4 have a CR-embedding.

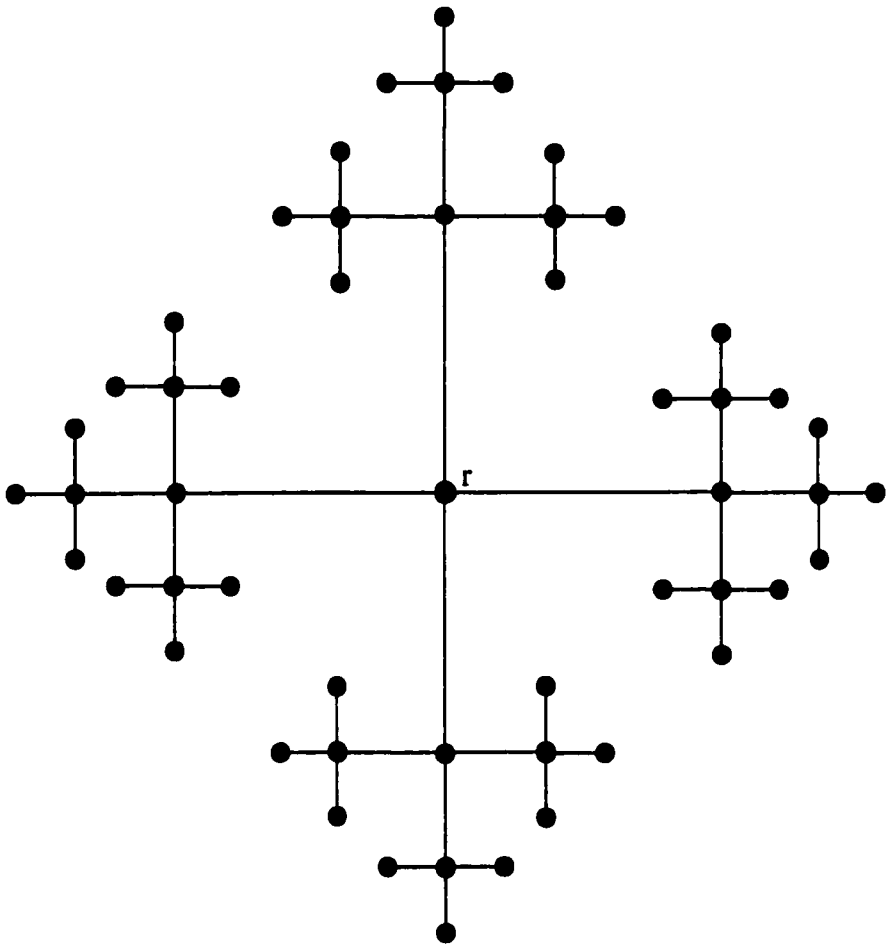


Figure 5.1: An Orthogonal Embedding

Let T be the tree shown in Figure 5.1 and r be the vertex at the center of T.

Given any orthogonal embedding of T and a vertex v of T with degree 4, let N(v), S(v), E(v) and W(v) be its neighbours where N(v) represents the northern neighbour, S(v) represents the southern neighbour, E(v) represents the eastern neightbour and W(v) represents the western neighbour as shown in Figure 5.2.

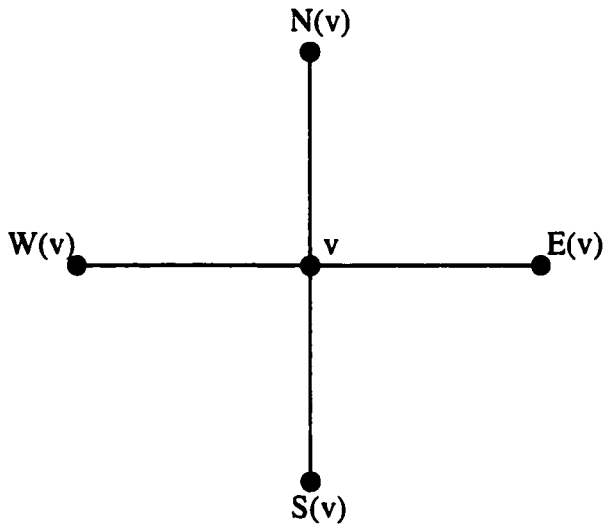


Figure 5.2: Vertex v and its neighbours

Notice that since any non-leaf vertex v of T has degree 4, therefore N(v), S(v), E(v) and W(v) are well defined for such vertices. Furthermore, any vertex in T which is at distance less than or equal to 2 from root r has degree 4 and all the leaves of T are at distance 3 from r.

We now show that there does not exist an orthogonal CR-embedding of T. We prove this by contradiction as follows:

Suppose that there exists an orthogonal CR-embedding of T. Using this orthogonal embedding of T, we let $p_4 = W(r)$, $p_2 = S(p_4)$, $p_1 = W(p_2)$, $p_3 = E(p_2)$ and $s = S(p_2)$ (see Figure 5.3).

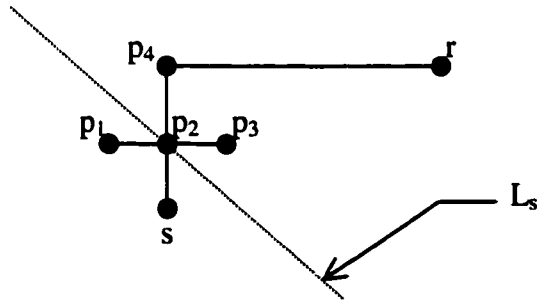


Figure 5.3

Similarly let $p_8 = S(r)$, $p_6 = W(p_8)$, $p_7 = N(p_6)$, $d = W(p_6)$ and $p_5 = S(p_6)$ (see Figure 5.4).

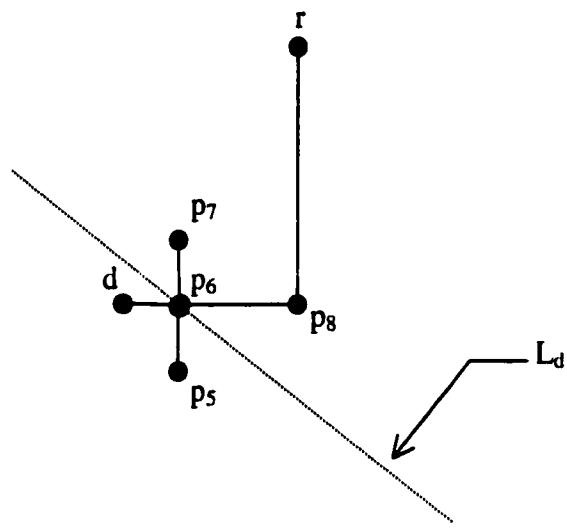


Figure 5.4

Let the coordinates of p_4 be $(-x_1,0)$. Let the coordinates of one of the children of p_4, namely p_2, be $(-x_1,-y_1)$. The vertex p_2 has three children, p_1, s and p_3. Let the coordinates of p_1 be $(-x_1-x_2,-y_1)$, the coordinates of p_3 be $(-x_1+x_3,-y_1)$ and the coordinates of s be $(-x_1,-y_1-y_2)$. Next let the coordinates of p_8 be $(0,-y_3)$. Now let the coordinates of one of the children of p_8, namely p_6, be $(-x_4,-y_3)$. The vertex p_6 has three children, p_7, d and p_5. Let the coordinates of p_7 be $(-x_4,-y_3+y_4)$, the coordinates of d be $(-x_4-x_5,-y_3)$ and the coordinates of p_5 be $(-x_4,-y_3-y_5)$.

Now suppose we want to go from vertex s to vertex d in the orthogonal embedding of T. We must first go to vertex p_2. Now notice that d must be above the line L_s with slope -1 going through p_2, otherwise using compass routing we would fail to reach p_4 in our next step, which is a contradiction since the shortest path from s to d must go through p_4 (see Figure 5.3).

Similarly, suppose we want to go from vertex d to vertex s in the orthogonal embedding. We must first go to vertex p_6. Now notice that s must be above the line L_d with slope -1 going through p_6, otherwise using compass routing we would fail to reach p_8 in our next step which is a contradiction since the shortest path from d to s must go through p_6 (see Figure 5.4).

However, it is not possible to have an orthogonal embedding which satisfies the above two cases. Hence, this shows that there does not exist an orthogonal CR-embedding for every tree in which each vertex has degree at most 4.

Now we ask the following question?

Does there exist an orthogonal SS-CR embedding for any given tree in which each vertex has degree at most 4?

We will now prove that this is indeed the case.

Theorem 6 *There exists an orthogonal SS-CR embedding for any given tree T in which the degree of every vertex is less than or equal to 4.*

Proof: Let the root r be any vertex of T. We will prove the result for balanced trees with respect to r. For non-balanced trees, we can add dummy vertices to make them balanced trees and then these vertices will be deleted at the end.

Let T be a balanced tree of depth k with root r. We label each vertex of T. The label of the vertex depends on the distance of that vertex from the root of T. All the vertices having depth 1 are labelled 1, all the vertices having depth 2 are labelled 2, all the vertices having depth 3 are labelled 3 and so on until every vertex of T has received a label. Note that the root of T has depth 0 and hence it is labelled with a label 0.

Next we orthogonally embed T such that the resulting orthogonal embedding supports SS-CR from r. We use the origin as the root of T. All the vertices with depth 1 are placed at distance 2^{k-1} from the root, all the vertices with depth 2 are placed at distance 2^{k-2} from their parents, all the vertices with depth 3 are placed at distance 2^{k-3} from their parents and so on. Finally, all the vertices with depth k are placed at distance 2^0 from their parents (see Figure 5.5).

Now we prove that the resulting orthogonal embedding does indeed support SS-CR.

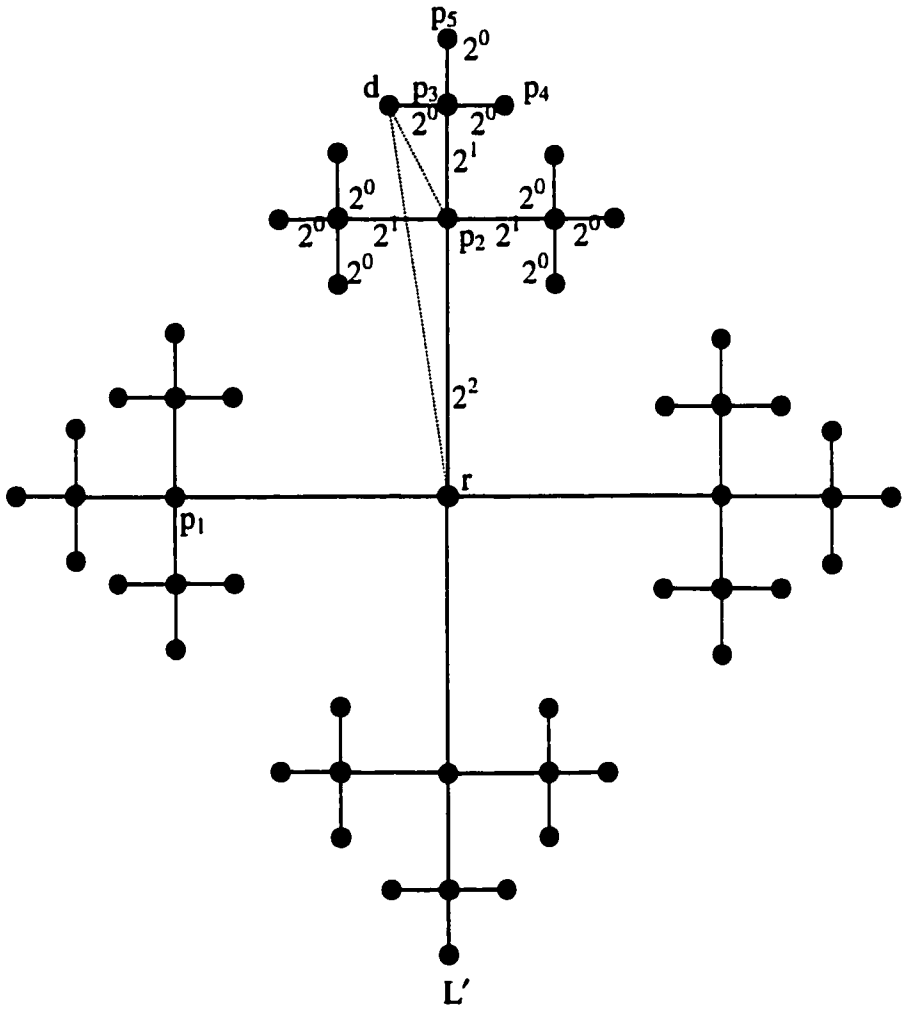


Figure 5.5: T with depth k=3

Suppose that we want to go from source vertex r to any vertex d using compass routing (see Figure 5.5). Without loss of generality, let us assume that d is on the branch of T generated by $N(r) = p_2$. We join vertex r to vertex d by a dotted line segment. Now notice that the edge rp_2 forms the smallest angle of all the edges incident to vertex r with the dotted line segment rd. Therefore, we take this edge and arrive at vertex p_2. Our result now follows by an inductive argument. QED

5.2 Balanced Trees

In this section, we will study orthogonal embeddings of balanced trees in which the vertices of our trees are restricted to have integer coordinates.

We now concentrate on the following problem:

Find the smallest $m \times m$ grid in which any balanced tree of depth k can be orthogonally embedded such that the embedding supports SS-CR.

Theorem 7 *The smallest $m \times m$ grid for which there exists a SS-CR embedding of T of depth k is the $(2^{k+1} - 2) \times (2^{k+1} - 2)$ grid.*

Proof: We first show that the embedding of the balanced tree T presented in the previous section fits in the $m \times m$ grid where $m = 2^{k+1} - 2$.

From Figure 5.5, we can notice that the distance from the source vertex r to the unique leaf L on the y-axis and above r is

$$2^0 + 2^1 + 2^2 + \ldots + 2^{k-1}$$

which equals

$$\sum_{n=0}^{k-1} 2^n = 2^k - 1$$

Similarly, the distance from r to the leaf L′ of T on the y-axis and below r is $2^k - 1$. Hence, the distance from L′ to L is

$$2(2^k - 1) = 2^{k+1} - 2.$$

Finally, we show that T cannot be embedded on any smaller grid, that is, our embedding of T is optimal. To show this, we first prove that the width T(k) of the optimal m × m grid in which T can be embedded satisfies the following recurrence equation:

$$T(k) = 2(1 + T(k-1)), \quad k \geq 1, \quad T(0) = 0.$$

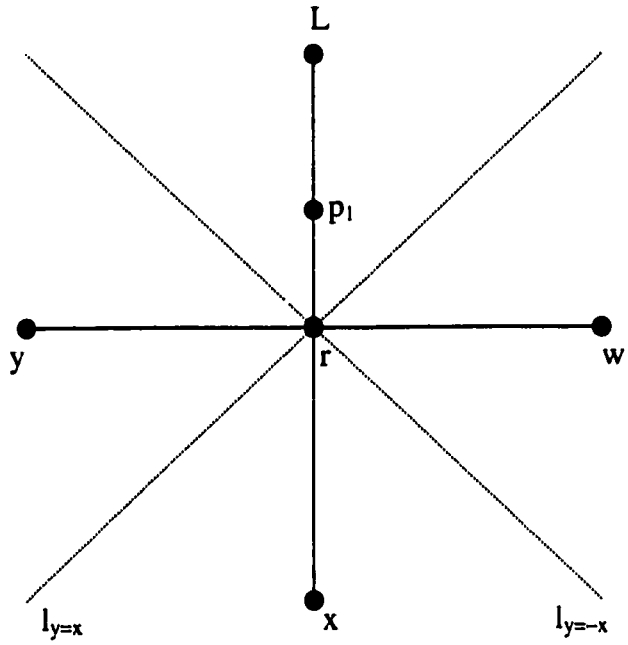


Figure 5.6

To prove this, let us consider an optimal embedding of T in which root r lies at the origin. Let $l_{y=x}$ and $l_{y=-x}$ be the lines going through r. Now notice that in our embedding of T, no edges of the subtree of T generated by N(r) = p_1 intersect either of these lines going through r, otherwise compass routing would fail to produce shortest path between r and any vertex of this subtree (see Figure 5.6).

Let $l'_{y=x}$ and $l'_{y=-x}$ be the lines going through p_1 (see Figure 5.7). It is easy to see now that $l'_{y=x}$ and $l'_{y=-x}$ are not intersected by any of the edges of the subtree of T generated by p_1. Next we note that the

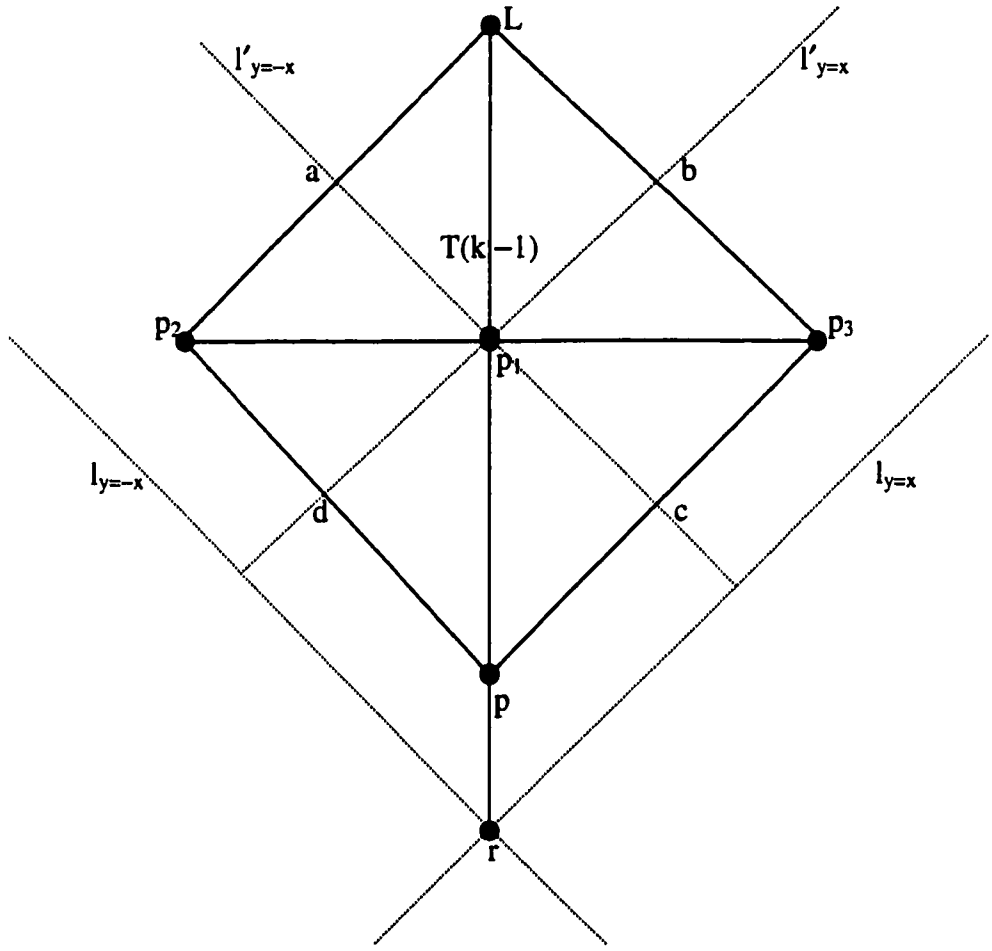


Figure 5.7: The area T(k−1) × T(k−1)

lengths of the edges used in each of the rectangles ap_1dp_2, bp_1aL and cp_1bp_3 by the subtrees of p_1 generated by $W(p_1)$, $N(p_1)$ and $E(p_1)$ respectively are optimal since otherwise we would be able to reduce these lengths further and get a better embedding of T which would be a contradiction. Notice that the rectangle dp_1cp contains no vertex of T

and hence, by copying any one of the subtrees generated by $W(p_1)$, $N(p_1)$ or $E(p_1)$ into it, we get an embedding of a tree with depth k−1 which can be embedded on a grid of optimal size with width T(k−1). Note that the distance $p_1p = p_1L$. If this embedding is not optimal then we can easily get a better embedding of T which is a contradiction. Notice that the closest point to r on the y-axis where p could lie is at (0,1), that is, the distance rp equals 1.

Hence, the distance from r to L is 1 + T(k−1) and therefore the distance from L′ to L is 2(1 + T(k−1)). Therefore we have the recurrence equation

$$T(k) = 2(1 + T(k-1))$$

By expanding the above recurrence, we have

$$T(k) = 2^1 + 2^2 + 2^3 + \ldots + 2^k$$

which can be rewritten as

$$T(k) = 2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^k - 2^0$$

Therefore,

$$T(k) = \sum_{n=0}^{k} 2^n - 2^0 = 2^{k+1} - 1 - 1 = 2^{k+1} - 2$$

Notice that this value of T(k) equals twice the distance from r to L computed earlier. QED

Chapter 6

Approximations and Variations

We now turn our attention to the following problem:

Given some class of geometric graphs, is there a routing scheme based on compass routing which will at least guarantee that for any starting position s and final destination d, we always find *a path* from s to d?

It is important to recall here that we are interested in routing algorithms which use only a finite amount of memory and "local information", that is, if we are located at a vertex v of a geometric graph, the only information available to us is our destination d, the location of v and the location of neighbours of v. In addition we may assume that we have some constant amount of extra information available to us. Furthermore, any previously visited vertex cannot be "marked" as this would lead to the accumulation of information stored at vertices.

6.1 Delaunay Triangulations

Let P_n be a set of points and D be the Delaunay triangulation of it. (The Delaunay triangulation D of a point set P_n is defined as the triangulation in which every circle goes through the vertices of a triangle

and it does not contain any vertices of P_n in its interior [Prep85]). We now prove the following result:

Theorem 8 *Let D be the Delaunay triangulation of a point set P_n. Then compass routing always finds a path between any pair of vertices of D.*

Proof: Suppose that we want to go from vertex s to vertex d in D. We will show that starting at s, compass routing will choose an edge, say sv, such that the euclidean distance from v to d is strictly smaller than the euclidean distance from s to d. This will prove our result.

Suppose that we join s to d by a dotted line segment (see Figure 6.1). Let the triangle sxy be the triangle incident with s intersected by line segment sd. Let C be the circle determined by the vertices s, x and y.

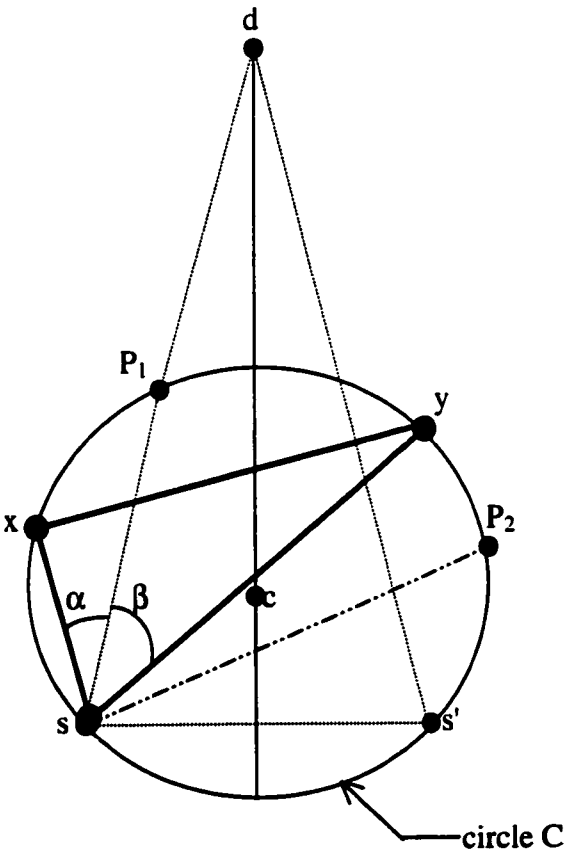


Figure 6.1

Now let s' be the mirror image of s through the line segment going through d and c, the center of C. We join s' and d with a dotted line segment. Let the point where the line segment sd intersects the circle C be P_1. Furthermore, let the arc distance between vertex s and point P_1 be I and choose point P_2 such that the arc distance between vertex s and point P_2 is 2*I. Let $\alpha = \angle xsd$ and $\beta = \angle ysd$.

Now there are two cases to be considered regarding the edge to be chosen from s using compass routing, namely

Case (i) $\alpha < \beta$

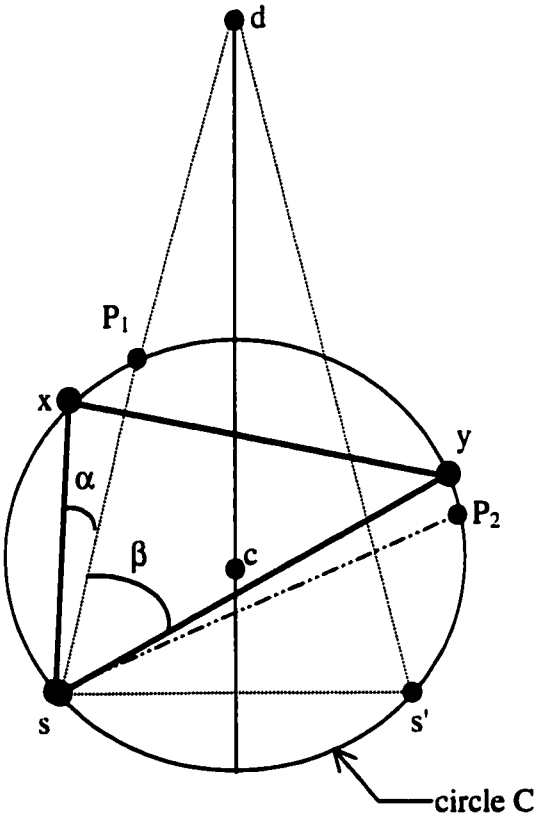


Figure 6.2: Here $\alpha < \beta$

In this case, it follows easily that x is closer to d than s (see Figure 6.2). Hence, we move closer to d upon choosing edge sx.

Case (ii) $\alpha > \beta$

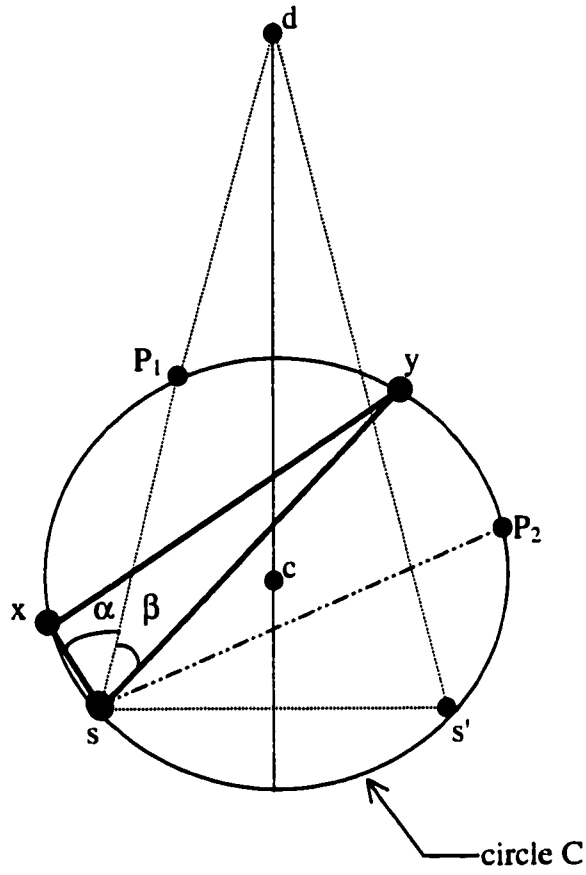


Figure 6.3: Here $\alpha > \beta$

Observe that the length of the arc joining point P_1 to vertex x is greater than the length of the arc joining point P_1 to vertex y (see Figure 6.3). Therefore, it follows that vertex y lies on the arc joining point P_1 to point P_2 and therefore it is closer to d than s. Hence, we choose the edge sy and move closer to d.

This completes our proof. QED

It is interesting to note that in Chapter 4 we proved that the fan triangulation cannot be embedded in such a way that compass routing produces a shortest path between any pair of vertices. Notice, however, that the fan can be embedded in such a way that it is a Delaunay triangulation of a point set P_n.

Suppose that we are given a circle C' with center c' and points $p_0,\ldots,p_n$ on its upper half such that $\angle p_0c'p_n < \pi$. Observe that the Delaunay triangulation of the point set $\{c',p_0,\ldots,p_n\}$ is a convex fan F_n with apex at c' as shown in Figure 6.4.

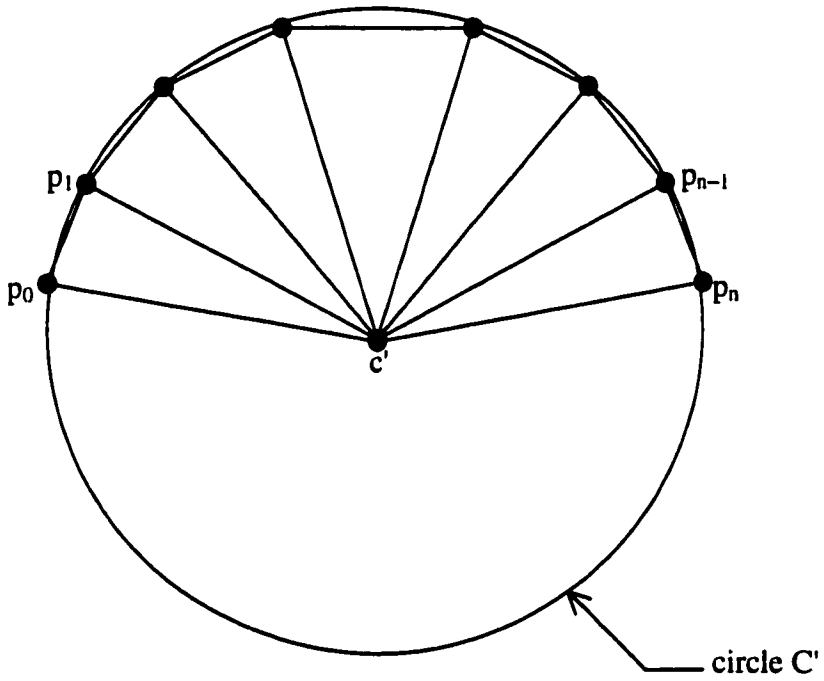


Figure 6.4

It now follows from our previous result that using this embedding of the fan F_n we can always find *a path* between any pair of its vertices.

6.2 Convexly Embedded Geometric Graphs

A geometric graph is called convexly embedded if all its faces except the exterior one are convex polygons. We also require the exterior face to be the complement of a convex polygon. An example of such a graph appears in Figure 6.5.

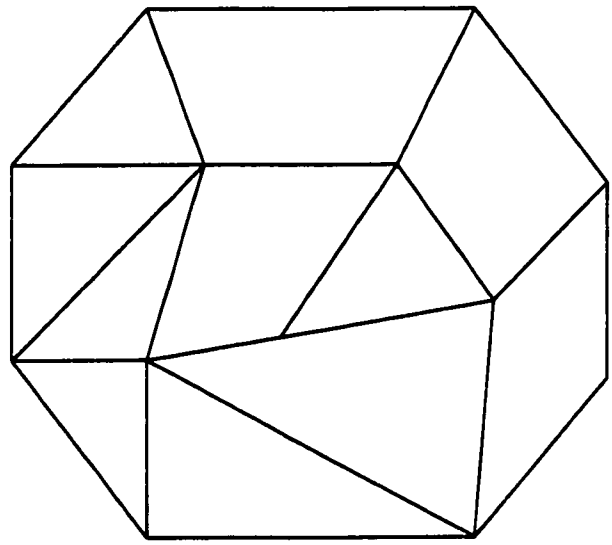


Figure 6.5: An n-gon with convex faces

We now study the following question:

Is it true that compass routing would always find *a path* between any pair of vertices s and d of a convexly embedded geometric graph?

Clearly the answer to this question is "no", as the graph shown in Figure 3.5 is convexly embedded. Suppose now that we add the following rule to compass routing:

Rule 1: If edge uv is chosen last to go from u to v then we cannot use it immediately to go back to u from v.

We now show that even in the presence of this restriction, compass routing fails to find *a path* between some vertices of a convexly embedded geometric graph.

The following example is interesting, for as we will now see, using modified compass routing we get caught in a cycle which contains our intended destination in its interior.

Let G be the geometric graph shown in Figure 6.6. Suppose we

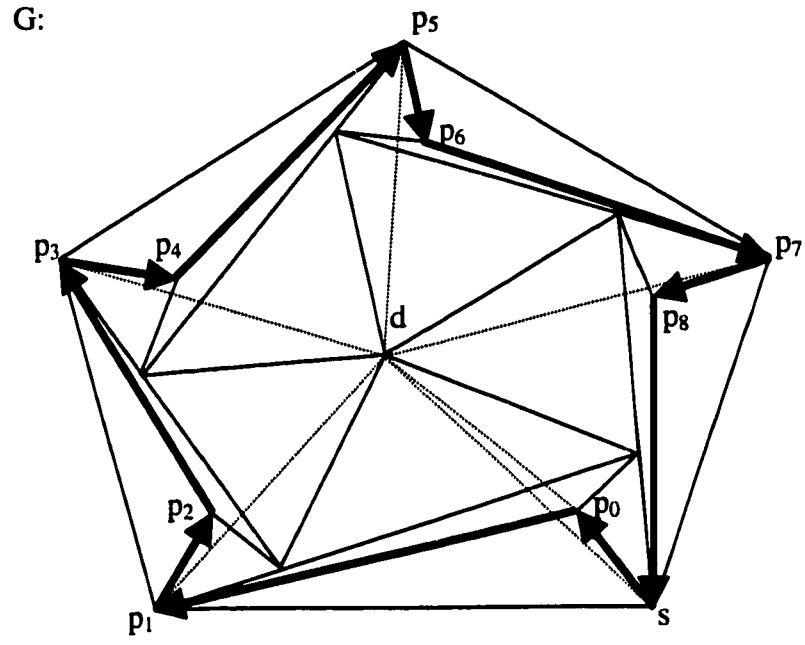


Figure 6.6: A counterexample

want to go from vertex s to vertex d in graph G using compass routing. We first arrive at p_0 since the edge sp_0 forms the smallest angle of all the edges incident to s with the dotted line segment joining vertices s and d. Now we note that the edge p_0p_1 forms the smallest angle of all the edges incident to p_0 with the dotted line segment joining vertices p_0 and d, therefore we choose the edge p_0p_1 and arrive at p_1. By symmetry, the same argument holds for all the vertices we reach. Note that we never

reach our destination d using compass routing; instead we arrive back at s, that is, we keep going in a cycle s-p_0-p_1-p_2-p_3-p_4-p_5-p_6-p_7-p_8-s shown in the figure above in bold.

We now have the following open problem:

Problem: Let G be a planar graph which can be convexly embedded. Is there a convex embedding of G such that compass routing (including Rule 1) always finds *a path* between any pair of its vertices?

6.3 Orthogonal Convexly Embedded Geometric Graphs

We now restrict our attention to orthogonal convexly embedded geometric graphs, that is, the graphs in which all the faces are rectangles. An example of such a graph appears in Figure 6.7.

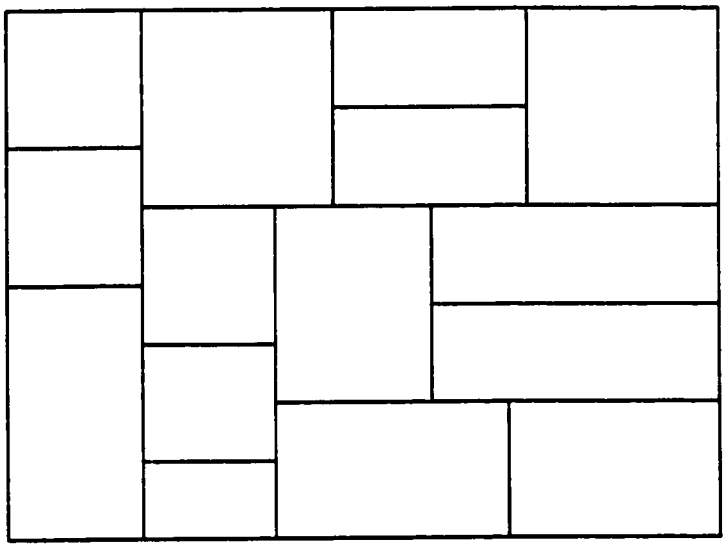


Figure 6.7: Orthogonal convexly embedded geometric graph

We start by asking the following question:

Does modified compass routing (compass routing + Rule 1) guarantee *a path* between any starting vertex s and destination d?

Consider the orthogonal graph shown in Figure 6.8. Suppose

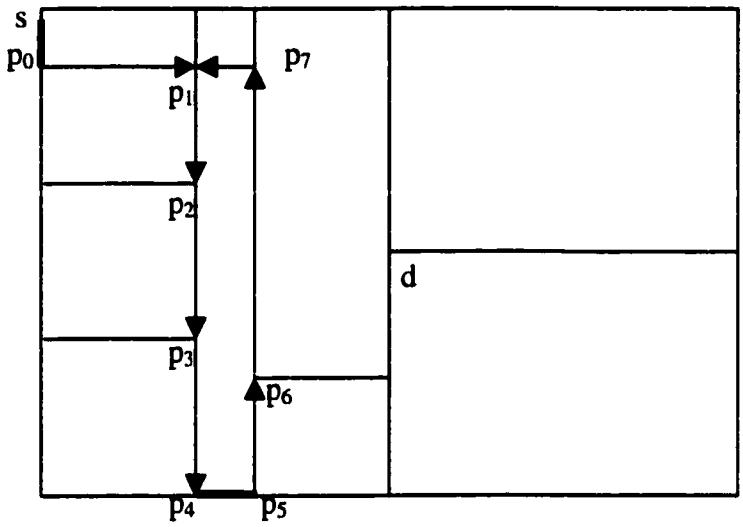


Figure 6.8: Cycle not containing d

we want to go from s to d. It is easy to verify that using compass routing we will first go to p_0, followed by p_1 and then we will go around the cycle p_1-p_2-p_3-p_4-p_5-p_6-p_7-p_1 infinitely.

Since orthogonal graphs are of special interest in many real life applications, it is particularly important to see if we can add an extra (local) rule such that new modified compass routing will find the desired path. With this in mind, we add the following Rule 2 to modified compass routing.

Without loss of generality, assume that destination d is at the origin. We let Q_1 represent the first quadrant, Q_2 represent the second quadrant, Q_3 represent the third quadrant and Q_4 represent the fourth quadrant. Now we define our rule as follows:

Rule 2: Suppose that we are at point u in Q_i and we enter into Q_{i+1} by choosing edge uv (see Figure 6.9). The restriction we impose is that we cannot make right turns upon entering Q_{i+1}, that is, we cannot choose (if available) the edge connecting v to a vertex w above it. Similarly, if we cross over into Q_{i-1} from Q_i then we cannot make any left turns upon entering Q_{i-1}.

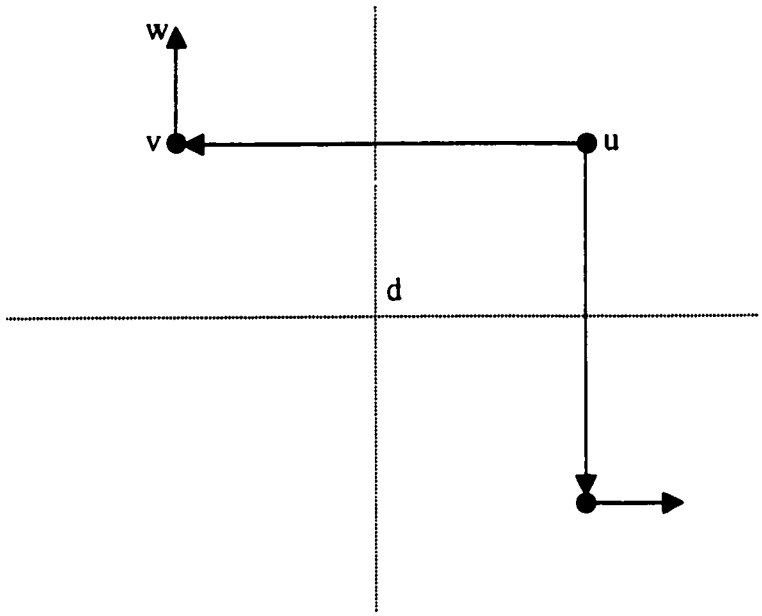


Figure 6.9: Forbidden turns

Now we are ready to prove:

Theorem 9 *Given any orthogonal convexly embedded geometric graph G, the new modified compass routing always produces a path (not necessarily the shortest) between any pair of vertices of G.*

Proof: Suppose that we want to go from vertex s to vertex d using new modified compass routing. To prove this theorem it is enough to show that we do not get into a closed cycle while trying to reach d.

Now assume that we do get into an anti-clockwise cycle C while trying to reach destination d where $C=\{p_1,p_2,...,p_m\}$. We enclose cycle C into a smallest possible rectangle, say R (see Figure 6.10).

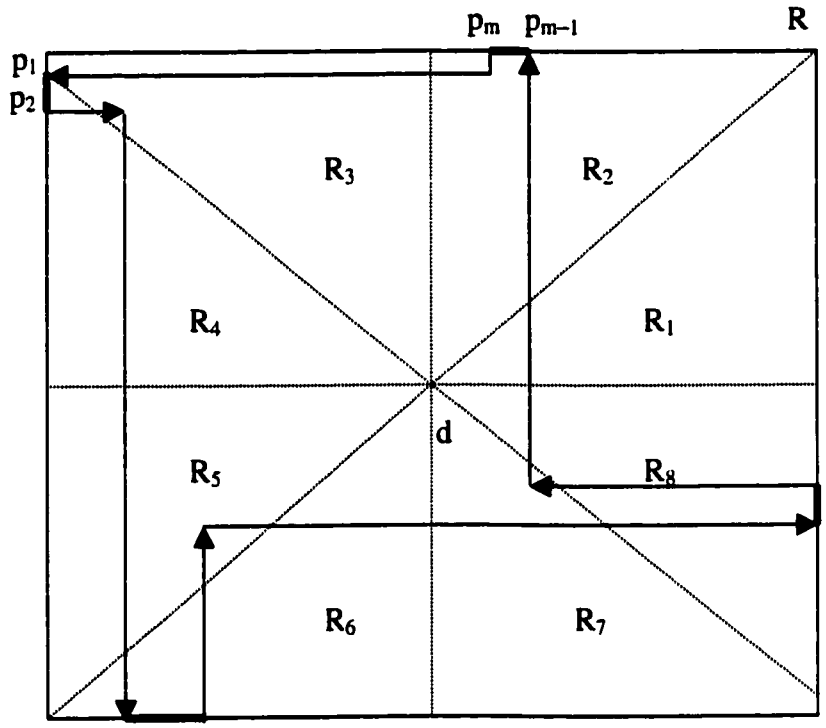


Figure 6.10: R containing cycle C

Next we divide R into eight regions R_1, R_2, R_3, R_4, R_5, R_6, R_7 and R_8 using the line segments x-axis, y-axis, y=x and y=–x passing through d as shown in the figure above.

Let v be any vertex of G. Using the same notation as in Chapter 5, let N(v), S(v), E(v) and W(v) be the neighbours of v if they exist. Now let $e_N(v)$, $e_S(v)$, $e_E(v)$ and $e_W(v)$ be the edges connecting v to N(v), S(v), E(v) and W(v) respectively.

We first make the following observation.

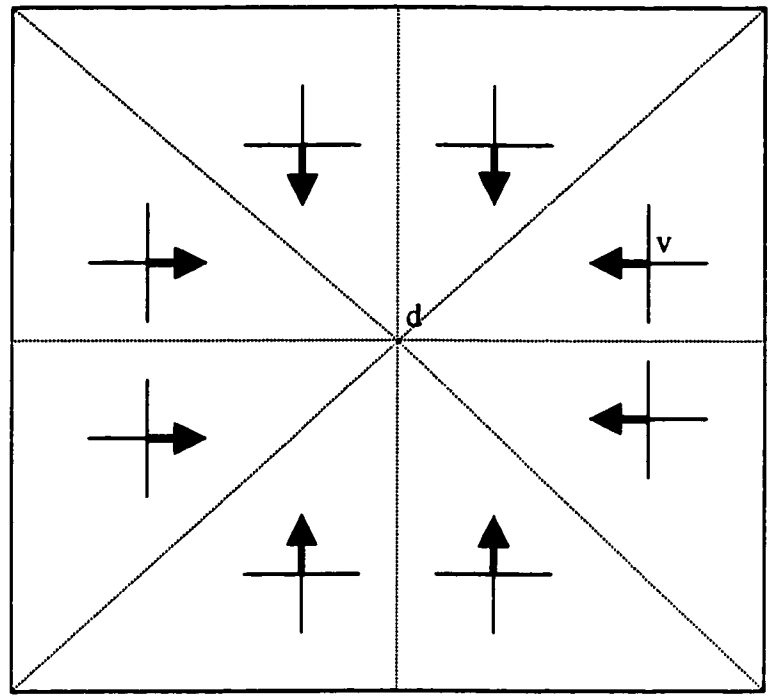


Figure 6.11: The edge selection

Observation 1: If a vertex $v \in R_1$ and W(v) exist as seen in Figure 6.11, then new modified compass routing will choose $e_W(v)$ if available, otherwise it will choose $e_S(v)$ (which in the second case must exist in order to guarantee that we have an orthogonal convexly embedded geometric graph).

By symmetry, the same argument holds true for all other regions.

Next we make our second observation as follows:

Observation 2: If a vertex $v \in R_1$ then $e_N(v)$ and $e_E(v)$ cannot be consecutive edges in C. Suppose that we take the edge $e_N(v)$ to arrive at vertex v (see Figure 6.12). Observe now that if either of the edges $e_W(v)$ or $e_S(v)$ exist, then new modified compass routing will choose one of them instead of $e_E(v)$. Notice that one of the edges $e_W(v)$ or $e_S(v)$ must be present, otherwise we have a non-convex face in the original graph

which is a contradiction since we are working with graphs in which all the interior faces are convex. A similar argument reveals that if we arrive at v by choosing the edge $e_E(v)$ then modified compass routing will not choose the edge $e_N(v)$ next.

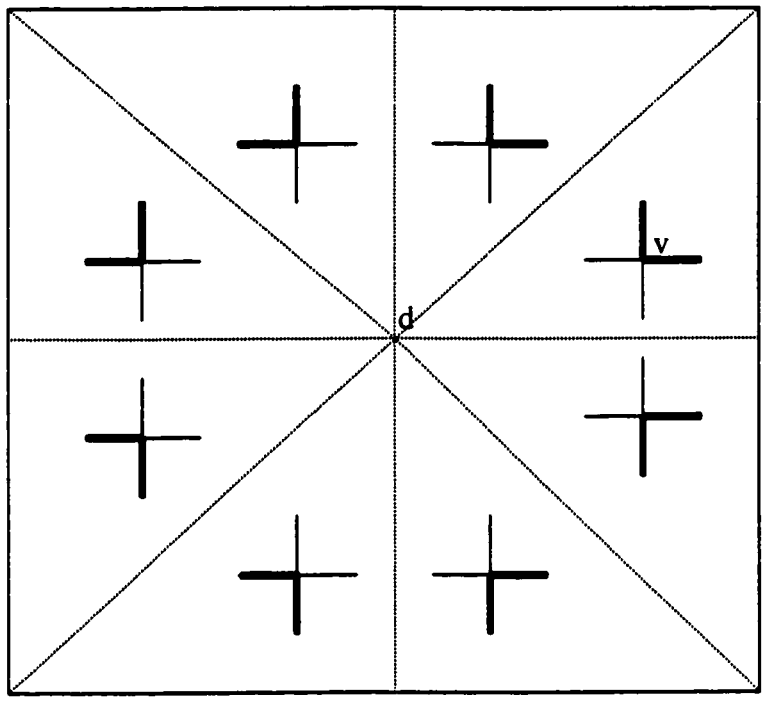


Figure 6.12: Invalid edge selections

By symmetry, the same argument holds true for all other regions shown in the figure above.

Now we are ready to prove our theorem as follows:

We observe that since R is the smallest rectangle containing C, each side of R contains at least one edge of C. Let p_ip_{i+1} be the

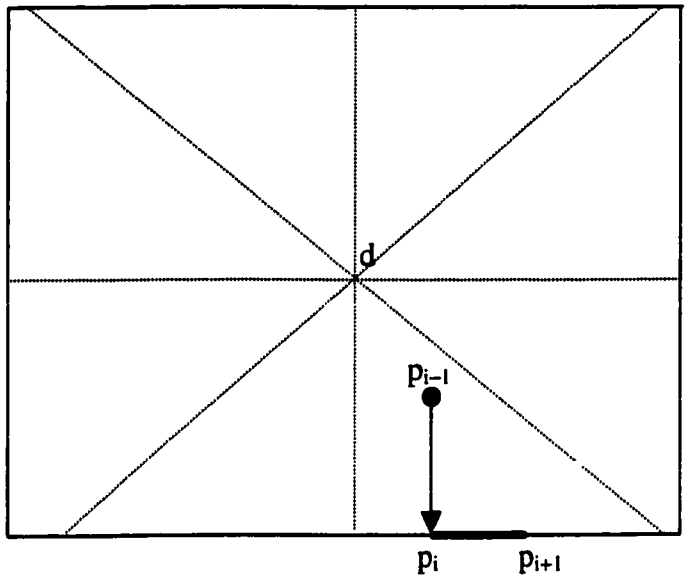


Figure 6.13: p_ip_{i+1} lies in Q_4

leftmost such edge on the bottom edge of R as shown in Figure 6.13. We now show that p_ip_{i+1} cannot be contained in Q_4. Observe now that p_{i-1} must be directly above p_i.

We now prove that p_{i-1} is in Q_1. Suppose that p_{i-1} is not in Q_1, but lies in Q_4 as shown in the figure above. If the edge $p_{i-2}p_{i-1}$ of C is horizontal then in our next step we would either continue in the horizontal direction or turn upwards at p_{i-1}, thus failing to choose the edge $p_{i-1}p_i$ (see Figures 6.14 (a) and 6.14 (b)).

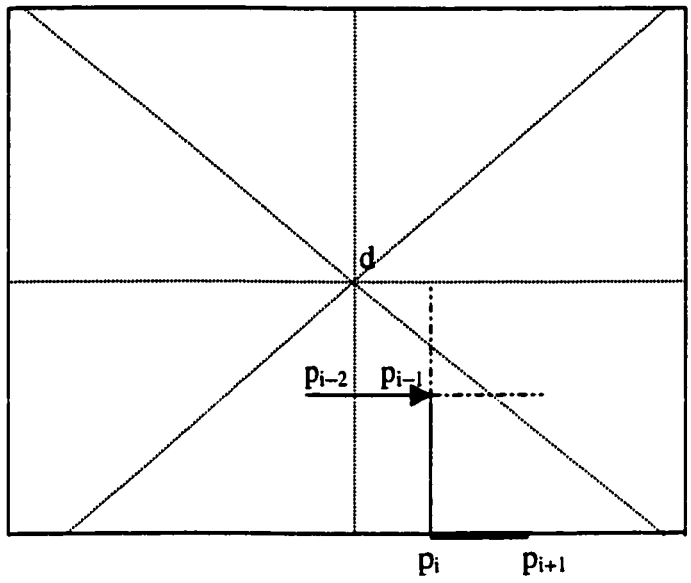


Figure 6.14 (a)

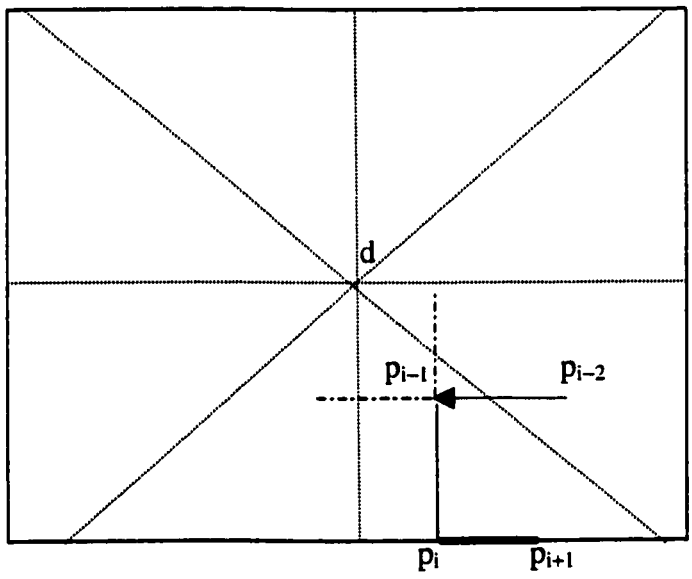


Figure 6.14 (b)

A similar argument applies if the edge $p_{i-2}p_{i-1}$ of C is vertical. It now follows that p_{i-1} must be in Q_1.

Next we notice that due to Rule 2 we cannot make a right turn upon reaching p_i, that is, we cannot choose the edge $p_{i-1}p_i$, which is a contradiction. It follows now that the edge $p_{i-1}p_i$ must be in Q_3. By symmetry, the same argument holds true for all other sides of R.

Suppose that the edges $p_ip_{i+1},\ldots,p_{i+k-1}p_{i+k}$ of C lie on the bottom edge of R (see Figure 6.15). Let $p_{i-1}p_i$ be the edge which precedes p_ip_{i+1} in C.

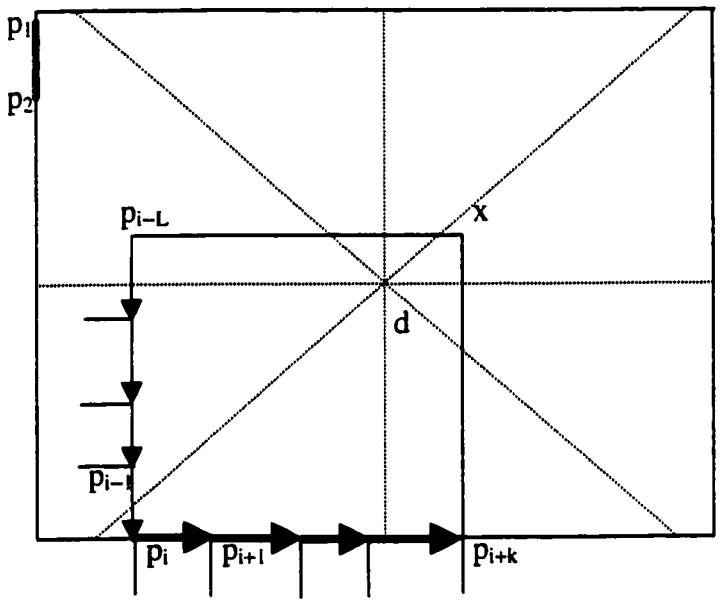


Figure 6.15: R containing cycle C

If p_{i-1} is in the same quadrant as p_i then it must have a western edge $e_W(p_{i-1})$ and a northern edge $e_N(p_{i-1})$ present. Notice that p_{i-1} cannot have an eastern edge $e_E(p_{i-1})$ present since modified compass routing would select it instead of $p_{i-1}p_i$. Similarly we can show that p_{i-2} must lie directly above p_{i-1} and so on until we arrive at a vertex p_{i-L} which lies in Q_2. It now follows that our path must have crossed into Q_3 from Q_2.

We now observe that p_{i+k} does not belong to Q_4 since in such a case none of the vertices $p_{i+1},\ldots,p_{i+k-1}$ has a northern edge $e_N(v)$ where $v \in p_{i+1},\ldots,p_{i+k-1}$. This along with the fact that none of the vertices $p_{i-1},\ldots,p_{i-L+1}$ has an eastern edge $e_E(v)$ where $v \in p_{i-1},\ldots,p_{i-L+1}$ implies that there exists a rectangular face, namely $xp_{i+k}p_ip_{i-L}$ in our graph, which encloses d. Now it follows that our destination vertex d is disconnected since the rectangle $xp_{i+k}p_ip_{i-L}$ contains nothing but vertex d.

Hence, the vertex p_{i+k} is in Q_3 as shown in Figure 6.16. Let q_1 be

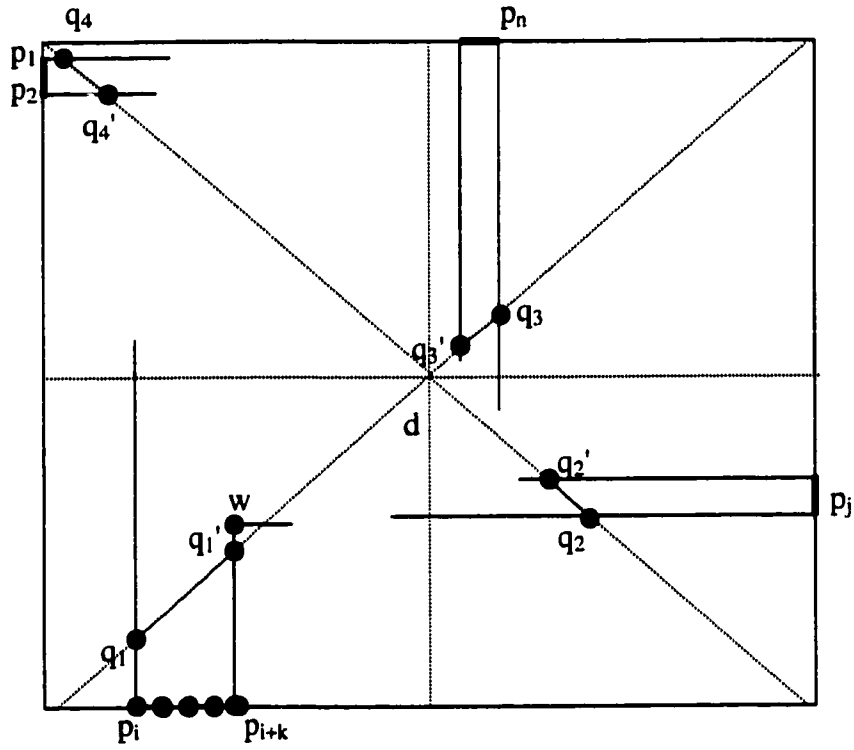


Figure 6.16: R containing cycle C

the point at which the line y=x intersects the vertical line through p_i and let d_1 be the distance between q_1 and d. Notice that q_1 is not necessarily above p_i; it could be below p_i.

Similarly we define q_2 to be the point of intersection between the line y=–x and the horizontal line through the lowest vertex of C on the right side of R and let d_2 be the distance between q_2 and d. In a similar way, we define q_3, d_3 and q_4, d_4 as shown in the figure above.

We now show that $d_1 > d_2$. Let q_1' be the point at which the line y=x intersects the vertical line through p_{i+k} and let d_1' be the distance between q_1' and d. Notice that $d_1 > d_1'$. Observe first that the first right turn at vertex w that C makes must occur in R_5. Due to Observation 2 and Rule 2, the y coordinates of the vertices in C between w and p_j cannot decrease and thus q_2 lies on or above the horizontal line through q_1'. It now follows that $d_1 > d_1' \geq d_2$, that is, $d_1 > d_2$.

By symmetry, $d_2 > d_3$, $d_3 > d_4$ and $d_4 > d_1$. Hence, we have a contradiction.

This completes our proof. QED

What are the advantages and disadvantages of this variation?

The main advantage of this variation is that it guarantees *a path* between any pair of vertices and hence it differs from other typical algorithms which either use routing tables or generate spanning trees.

The disadvantage of this variation is that it does not always determine the shortest path in terms of euclidean distance or link distance.

Chapter 7

Algorithmic Geometric Routing

To finish this thesis, we present a routing algorithm which on any convexly embedded geometric graph and using only the positions of our starting and destination points finds *a path* between them.

Suppose then that we have two vertices s and d of a convexly embedded geometric graph G. Our algorithm, which we call Algorithmic Geometric Routing, proceeds as follows:

a) Let L(s,d) be the line segment joining s to d and let f_0 be the face of G containing s on its boundary that intersects L(s,d).
b) Choose any edge su of f_0 incident to s and continue travelling along the edges of f_0 until we reach d or find an edge, say u_0v_0, of f_0 that is intersected by L(s,d). Let f_1 be the second face of G containing u_0v_0 on its boundary. At this point, we proceed as follows:

Let i=1 and suppose that we have just traversed the edge $u_{i-1}v_{i-1}$ which intersects L(s,d) from u_{i-1} to v_{i-1}. Continue traversing the edges of f_i in the direction u_{i-1} to v_{i-1} until we reach d or an edge u_iv_i on the boundary of f_i that intersects L(s,d). Increase i by 1 and repeat this procedure until we reach d (see Figure 7.1).

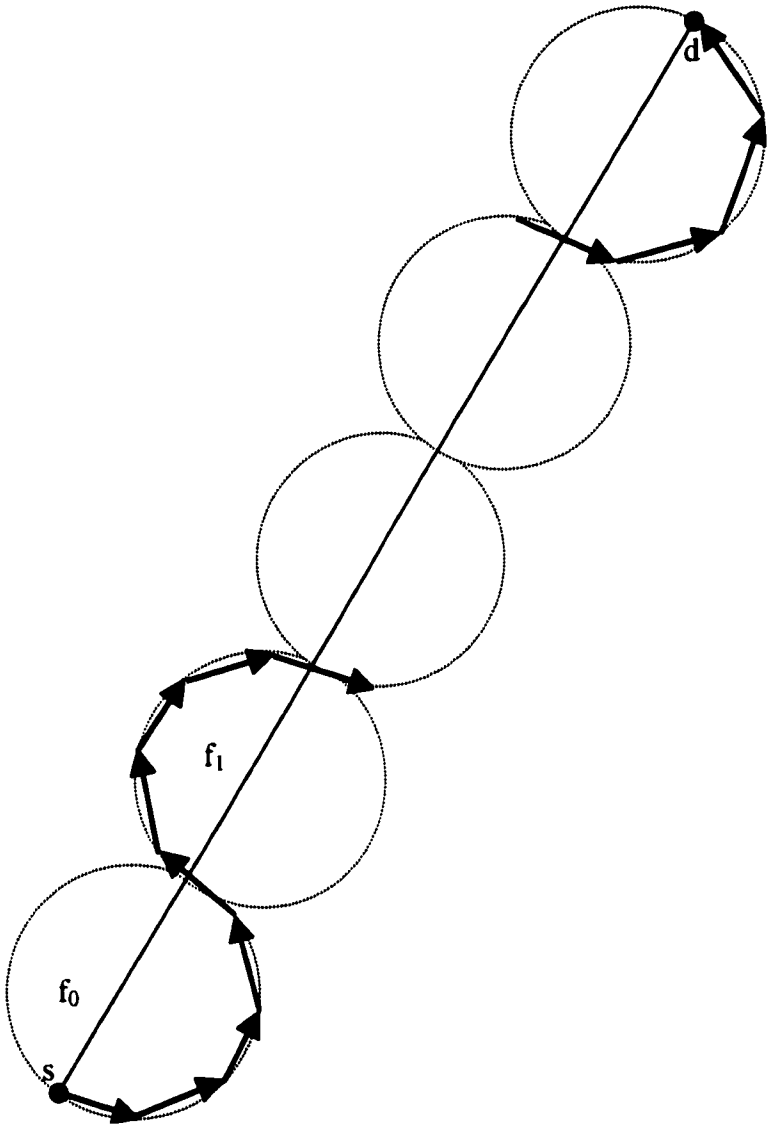


Figure 7.1

Notice that if L(s,d) intersects m edges of G then our algorithm will visit m+1 faces of G. It follows trivially that our algorithm always terminates.

Thus we have proved:

Theorem 10 *There is a routing algorithm on convexly embedded geometric graphs that using only local information at the vertices of G and the positions of the initial and destination points s and d finds a path connecting them.*

It is worth mentioning that our algorithm relies heavily on the fact that we know G is a convexly embedded planar geometric graph. We now show how to modify our algorithm so that it will also work for arbitrary geometric graphs.

Observe first that the vertices and edges of any geometric graph G induce a partitioning of the plane into a set of connected regions with disjoint interiors, not necessarily convex, called faces of G. The boundary B_i of each of these faces is a closed polygonal in which some edge of G is allowed to appear twice. For example, in the graph shown in Figure 7.2, the polygonal bounding the external face is $\{v_1, v_2, v_3, v_4, v_5, v_6, v_5, v_7, v_8, v_9, v_{10}, v_4, v_3, v_{11}, v_1\}$, notice that the edge v_3v_4 is traversed twice, once from v_3 to v_4, and again from v_4 to v_3. Suppose now that we want to go from a vertex s to a vertex d of G. As before, we calculate the line segment joining s to d, and determine the face $F = F_0$ incident to s intersected by sd. We now traverse the polygonal determined by F_0, checking if the last line traversed intersects sd. If it does, we calculate the distance from intersection point to s. Upon returning to s, (unless we reach d in which case we stop) all we need to recall is the point p_0 at which the polygonal bounding F_0 intersects sd which maximizes its distance to s. We then travel the boundary of F_0 until p_0 is reached at which point we update F to be the second face whose face contains p_0. It is now straightforward to see that we eventually reach d. Furthermore, we notice that each edge e of G is

traversed at most twice regardless of whether e belongs to the polygonals determined by one or two faces of G.

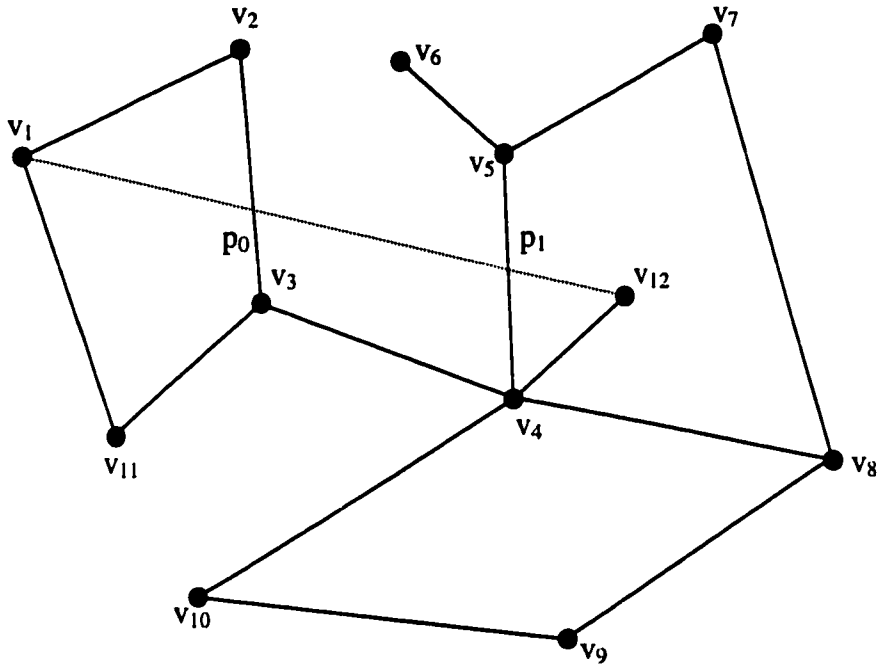


Figure 7.2: Travelling from v_1 to v_{12}

For the graph shown in Figure 7.2, we first traverse the edges of the face bounded by the polygonal $v_1v_2v_3v_{11}v_1$, we then traverse the edges of the external face and finally the polygonal $v_4v_5v_7v_8v_4v_{12}$ at which point we stop. In summary, we have:

Theorem 11 *There exists a local information routing algorithm on geometric graphs which guarantees that destination is reached. Moreover, our algorithm is such that a linear number of edges are traversed.*

Notice that our algorithm does not have any "markers" on the vertices we visit on our way from s to d and that the only information

used during the execution of our algorithm was which pairs of edges incident to a vertex v belong to the boundary of a face of G.

It is easy to see that if this information is not available at the vertices of G then none of the algorithms presented here would work.

Bibliography

[Blan92] Blandford, Percy W., *Maps and Compasses (2nd edition)*, TAB Books, PA, 1992.

[Bose97] Bose, P., "On embedding Outerplanar Graphs on a point set", in Graph Drawing, LNCS 1353, 25-36, 1997.

[Cast96] Castañeda, N. and Urrutia, J., "Straight Line Embeddings of Planar Graphs on Point Sets", Proceedings 8th Canadian Conference on Computational Geometry, August 1996, Ottawa, 312-318.

[Chib82] Chiba, N., Yamanouchi, T. and Nishizeki, T., "Linear algorithms for convex drawings of planar graphs", *Progress in Graph Theory*, 1982, 153-173.

[Corm90] Cormen, T. H., Leiserson, C. E. and Rivert, R. L., *Introduction to Algorithms*, The MIT Press, McGraw-Hill Book Company, 1990.

[DeFray90] De Fraysseix, H., Pach, J. and Pollack, R., "How to draw a planar graph on a grid", *Combinatoric*, Vol. 10, 1990, 41-51.

[Fary48] Fary, I., "On straight line representation of planar graphs", *Acta. Sci. Math. (Szeged)*, Vol. 11, 1948, 229-233.

[Fial78] Fiala, F., *Vehicle Routing Problem,* School of Computer Science, Carleton University, Ottawa, 1978.

[Flem82] Fleming, June, *Staying Found: The Complete Map and Compass Handbook,* VINTAGE Books, New York, 1982.

[Forf90] Forfar Chaundy, Duncan John, *Rural School Bus Routing and Scheduling,* School of Computer Science, Carleton University, Ottawa, 1990.

[Gear80] Geary, Don, *Step in the Right Direction:* A basic Map and Compass book, STACKPOLE Books, PA, 1980.

[Gold91] Goldfarb, D., Hao, J. and Kai, S., "Shortest path algorithms using dynamic breadth-first search", *Networks*, Vol. 21, 1991, 29-50.

[Hoff85] Hoffman, A. J. and Wolfe, P., The travelling salesman Problem. *A guided tour of Combinatorial Optimization*, John Wiley & Sons Ltd., 1985, 1-15.

[John90] Johnsonbaugh, Richard, *Discrete Mathematics,* (2^{nd} edition), Macmillan Publishing Company, New York, 1990.

[Kals83] Kals, W. S., *Land Navigation Handbook: The Sierra Club Guide to Map and Compass,* Sierra Club Books, San Francisco, 1983.

[Kjel76] Kjellstrom, Bjorn, *Map & Compass: The Complete "Orienteering" Handbook*, Charles Scribner's Sons, New York, 1976.

[Kran98] Kranakis, E., Singh, H. and Urrutia, J., "Compass Routing on Geometric Graphs", manuscript, SITE, University of Ottawa, Ottawa, 1998.

[Pach95] Pach, J. and Agarwal, P. K., *Combinatorial Geometry*, John Wiley, New York, 1995.

[Prep85] Preparata, Franco P. and Shamos, Michael I., *Computational Geometry (An Introduction)*, Springer Verlag, New York, 1985, 209-210.

[Read87] Read, R. C., "A new method for drawing a planar graph given the cyclic order of the edges at each vertex", *Congressus Numerantium*, Vol. 56, 1987, 31-44.

[Rose91] Rosen, Kenneth H., "Shortest Path Problems" on *Discrete Mathematics and its Applications*, (2^{nd} edition), McGraw-Hill Inc., 1991, 467-474.

[Rose77] Rosenbrantz, D. J. Stearns, R. E. and Lewis, P. M. II, "An analysis of several heuristics for the travelling salesman problem", *SIAM J. Comput*, Vol. 6, 1977, 563-581.

[Stoj98] Stojmenovic, I. and Lin, X., "Geographic distance routing in ad hoc wireless networks", manuscript, SITE, University of Ottawa, Ottawa, 1998.

[Tutt63] Tutte, W. T., "How to draw a graph", *Proc. London Math. Soc.*, Vol. 13, 1963, 743-768.

[Yao91] Yao, F., "Neighborhood Graphs and Geometric Embedding", *Lecture Notes in Computer Science*, Vol. 519, 1991, P. 201.