AN ABSTRACT OF THE THESIS OF

Youn-su Kim for the degree of Master of Science in Computer Science presented on March 17, 1983.

Title: Geometric Computer Graphics Package with Applications to Steiner's Problem

Abstract approved: Redacted for Privacy

Alan Coppola

An interactive computer graphics package was developed for the purpose of solving certain geometric problems in the Euclidean plane. The Geometric Graphics Package provides basic application-independent functions for creating arbitrary views of two-dimensional objects and for supporting interaction between the application program and its user. The application program contains a number of useful functions for our purpose.

Our specific application of the Geometric Graphics Package was to develop a heuristic algorithm for the Euclidean Steiner Minimal Tree, known as Steiner's Problem. The problem has been proved to be at least as difficult as any of the NP-complete problems. We present a heuristic algorithm using fixed tension physical relations for the problem. The Geometric Graphics Package is to be used as a tool and to guide us in improving the heuristic.

Geometric Computer Graphics Package with Applications to Steiner's Problem

by

Youn-su Kim

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed March 17, 1983

Commencement June 1983

Approved:

Redacted for Privacy

Professor of Computer Science in charge of major

Redacted for Privacy

Head of department of Computer Science

Redacted for Privacy

Dean of Graduate School

Date thesis is presented March 17, 1983

Typed by Young-sook Sitt for Youn-su Kim

TABLE OF CONTENTS

LIST OF FIGURES

GEOMETRIC COMPUTER GRAPHICS PACKAGE WITH APPLICATIONS TO STEINER'S PROBLEM

INTRODUCTION

Seeing is believing. This might have been one of the major motivations for the development of computer graphics, which is a rapidly growing field. At the present time, interactive computer graphics is an extremely effective medium for communication between man and computer. It has become a major facilitator of man/machine interaction.

Our interest here is to see how interactive computer graphics fulfils its promise in a particular application. It can enhance our understanding of complex phenomena and also provide us with pictorial communication to design or improve methods to solve difficult problems.

In this thesis, an interactive computer graphics package was developed for the purpose of solving certain geometric problems in the Euclidean plane. These are of practical importance because in many applications the real physical objects are arranged in the Euclidean plane. Our Geometric Graphics Package serves the purpose of modeling the Euclidean plane, and it contains many useful functions. Two of these functions find two important geometric structures. These structures can be used to solve various geometric problems and a number of practical problems. One such geometric structure is the Euclidean Minimal Spanning Tree(MST), an interconnecting tree of minimum total Euclidean length whose vertices are the given points in the plane. The MST is a common component in applications involving networks, and has been used as a tool in clustering [Zahn (71)], in pattern recognition [Osteen (74)], and in minimizing wire length in computer circuitry [Loberman (57)]. Another powerful geometric structure is the Voronoi diagram in the Euclidean plane, a graph containing the proximity information defined by the given point set in the plane. The Voronoi diagram can be used to solve a number of geometric problems, such as finding a nearest neighbor of each given points, the smallest circle enclosing the set, and the largest circle containing no points of the set [Shamos (78)]. Also the Voronoi diagram gives rise to other useful geometric structures, such as (1) a triangulation with property that the circumcircle of every triangle is empty, called the Delaunay triangulation (DT), (2) the MST, (3) the relative neighborhood graph, a proximity graph like DT and MST [Supowit (81)], and (4) the Gabriel graph, the least square adjacency graph with Euclidean distance [Matula (80)]. Numerous applications of the solutions to these geometric problems and of geometric structures can be found in [Shamos (78)], [Urquhart (82)], [Toussaint (80)], [Matula (80)], [Smith (81)], and [Dasarathy (80)].

Besides providing a useful geometric graphics package for many practical applications, our specific goal was to develop an alogorithm for finding the Euclidean Steiner Minimal Tree (SMT) using the Geometric Graphics Package as a tool. The problem of finding the SMT is known as Steiner's Problem. The SMT is a spanning tree with the shortest possible total length whose vertices contain the given points. The SMT differs from the Euclidean Minimal Spanning Tree(see the definition above) in that the SMT may contain other points besides the given fixed points as its vertices in order to reduce the total length. An extra vertex which is added to a tree to reduce its length is called a Steiner point. The simplest form of Steiner's Problem was first posed by Fermat early in the 17th centry. It is as follows: "Given three points in the plane, find a fourth point such that the sum of its distances to the three given points is a minimum[Kuhn (75)]" Steiner's problem with three points is exactly solved (The answer is shown in Chapter 2). However, Steiner's Problem with any number of given points, a generalization of Format's Problem has been proved to be at least as difficult as any of the NP-complete problems [Garey (77a)]. In Chapter 3 we present an iterative heuristic using fixed tension physical relations for the problem. The Geometric Graphics Package is to be used to visualize the behavior of the heuristic, to understand the natural phenomena simulated by our model, and finally to guide us to improve the heuristic.

Chapter 1 introduces the Geometric Computer Graphics Package. Its 7 major functions are described. Also we study the problem of constructing the Voronoi diagram and the Euclidean Minimal Spanning Tree, which are two important functions in the package. Efficient algorithms due to [Shamos (78)]are presented along with their applications. We give an example of a sequence of function invocations to illustrate the role of each function. Finally, the data structures used in the package are explained.

In Chapter 2 we review Steiner's Problem. Known facts and various properties of the optimal solution for the problem are listed along with their consequences and implications.

In Chapter 3 we present the heuristic algorithm for Steiner's Problem. Experimental results for the heuristic on certain point sets are included. This demonstrates how the graphics package can be used as a tool.

CHAPTER 1

GEOMETRIC COMPUTER GRAPHICS PACKAGE

In this Chapter, we introduce the Geometric Computer Graphics Package developed in this thesis. Its major functions are described. In section 2, we study two important functions, the Voronoi diagram and the Euclidean Minimal Spanning Tree. In section 3, we show an example of a sequence of function invocations. The data structures for the graphics package are explained in the last section.

The Geometric Graphics Package follows the principal concepts of interactive graphics programming using the Core System, as described in [Foley(82)] and [Bergeron(78)]. The Core System is a proposed standard graphics package developed by the ACM/SIGGRAPH Graphics Standards Planning Committee. It is designed as a general purpose subroutine package that provides an interface between an application program and graphics hardware. The major goal of the Core System is to have the interface be independent of the specific hardware available so that the application program is portable [Bergeron(78)].

The Gemetric Graphics Package provides the basic application-independent facilities for creating arbitrary views of two-dimensional objects and for supporting interaction between an application program and its user. The principal application of the graphics package is to help in the solutions of various geometric problems in the Euclidean plane (see the appendix for specific implementation of the package).

1.1 Main Functions in the Geometric Graphics Package

In this graphics package, the main objects that are created and represented pictorially are points and graphs in the Euclidean plane. The interactive application program contains 7 major functions which can be applied to a set of points or a graph. These functions are:

1. ADD

   This function adds a points to the set of points.

2. DELETE

   This function deletes a point from the set of points.

3. GEOMETRY

   This function has subfunctions which are used to display geometric structures which arise from the given point set (details in the next section).

   a. VORONOI_D: Display the Voronoi diagram, a graph in which every edge defines a nearest neighbor of each given point.

   b. DUAL : Display the Delaunay triangulation, the planar straight line dual graph of the Voronoi diagram.

   c. MST : Display the Euclidean Minimal Spanning Tree, an interconnecting tree of minimum total length whose vertices are the given points.

4. SKETCH_MST

   This function displays the Euclidean Minimal Spanning Tree, as in 3.c.

5. CHANGE_VIEW

   This function is to used to show the graph on the screen with different views. It consists of 6 subfunctions:

a. ZOOM : Set a new window and display the graph with the new window.

1) NEW_BOX : This allows the redefinition of a window.

2) BOX_done: Display the graph with the new window.

b. TRANSLATE: Translate the graph.

c. ROTATE : Rotate the graph.

d. OVERVIEW : Display the original graph.

e. EARTHVIEW: Display the previous graph with the current window (This is only to be used after OVERVIEW).

f. GEOMETRY : Same functions as above in (3). The graph in the EARTHVIEW is used as input.

Note that each function can be applied to point sets or graphs. An example of a sequence of operations is given in section 1.3.

6. LIBRARY

This function allows the user to access and store various point sets and graphs. It consists of 3 subfunctions:

a. GET_POINTSET : Get a new point set with the name provided by the user.

b. RETRIEVE_GRAPH: Retrieve a graph which has been saved. The user provides the name of the graph.

c. SAVE_GRAPH : Save the graph on the screen with the name provided by the user.

7. EXIT

Terminate the application program.

All functions are contained in menus so that the invocation of each function is done by selecting a menu entry. The function displaying the MST was set up in two menus for convenience' sake. A grah G = (V, E) consists of a finite, nonempty set of vertices V and a set of edges E. Figure 1.1 shows a module hierachy for MENU_LAYOUT and the interconnections between the menus. The function, HEURISTIC_SMT in the Main_menu is to invoke the heuristic algorithm for Steiner's Problem (see Chapters 2 and 3).

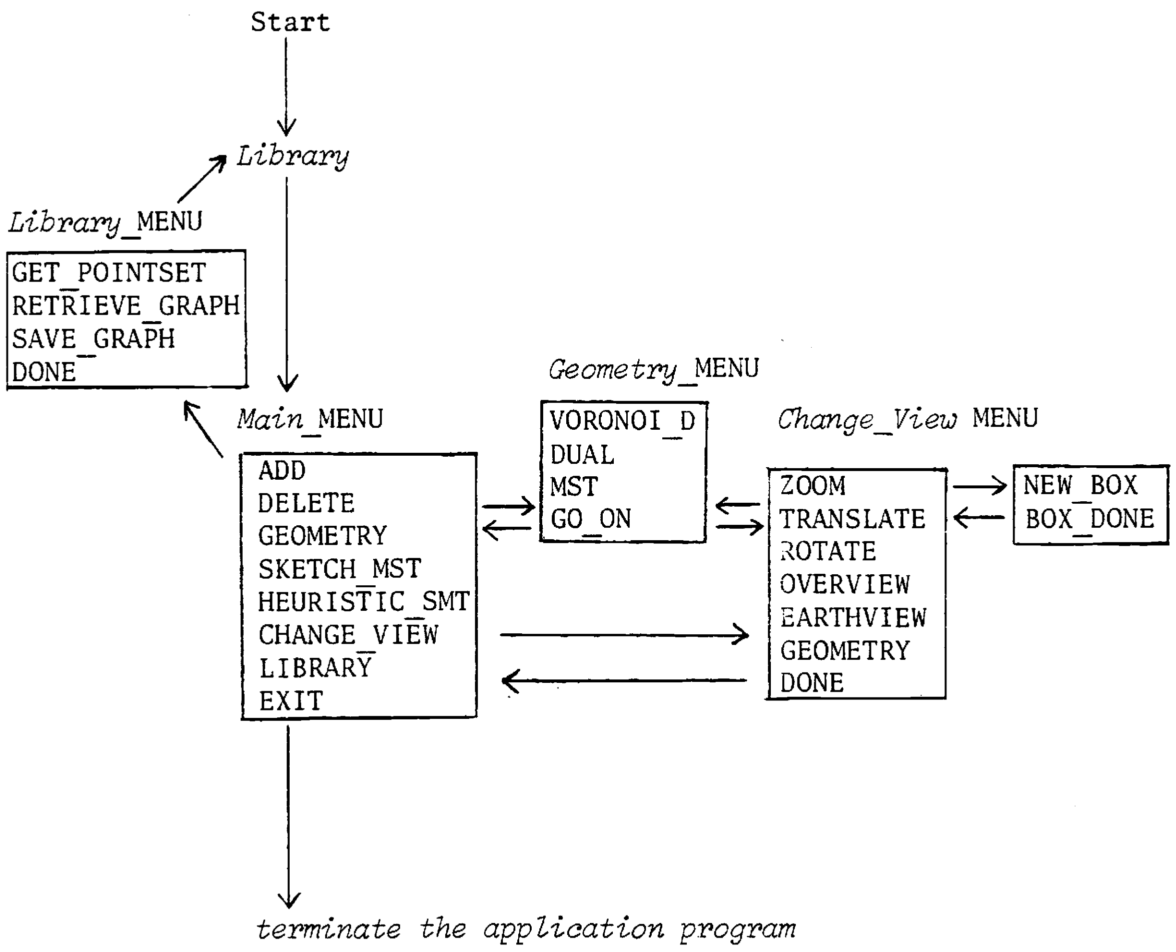


Figure 1.1 Module Hierarchy for MENU_LAYOUT

1.2 Voronoi Diagram and Euclidean Minimal Spanning Tree

In this section, we give the definitions of the Voronoi diagram and the Euclidean Minimal Spanning Tree, which are two important functions in our Geometric Graphics Package. Then we briefly introduce efficient algorithms for these functions, along with their applications. The algorithms are due to [Shamos(78)] and represent a break-through in computational geometry.

Given a finite set S of n points in the plane, for each point p_i there is a convex polygon VP(i), called the Voronoi polygon associated with the point p_i. This polygon has the property that the polygonal region encompasses the locus of points closer to p_i than any other point in S. It is formally defined as the intersection of half-planes determined by the perpendicular bisector of the straight lines joining P_i and all other points in S. If H(i,j) denotes the half-plane determined by the perpendicular bisector of points P_i and P_j, then

$$VP(i) = \bigcap_{j \neq i} H(i,j) .$$

This shows that VP(i) is a convex polygonal region having at most n-1 sides. The collection of Voronoi polygons VP(i), for each p_i in S, partitions the plane into n regions, some of which may be unbounded. This collection is referred to as the Voronoi diagram V(S) for the set S (Figure 1.2).

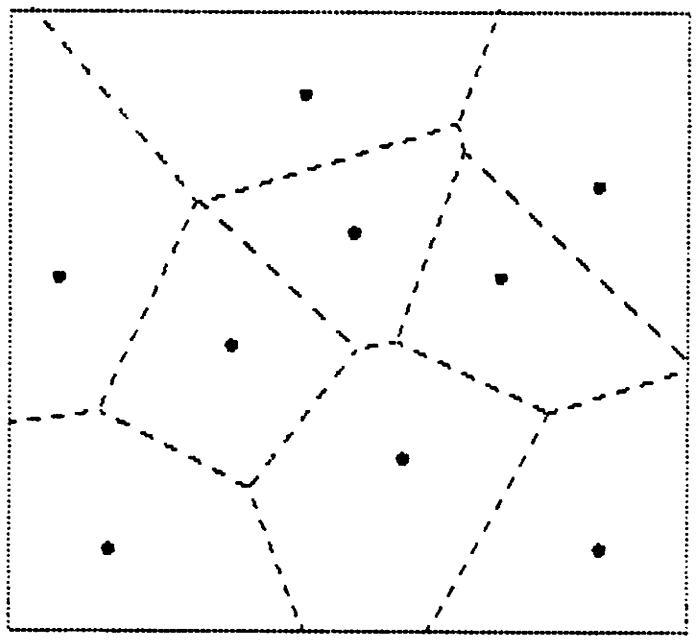


Figure 1.2 Voronoi Diagram

As we see in Figure 1.2, each of the original n points belongs to a unique Voronoi polygon, and thus any point inside the polygon VP(i) has P_i as its nearest neighbor. The Voronoi diagram having all the proximity information defined by the given set S turns out to be a single geometric structure which can be used to solve efficiently a number of seemingly diverse geometric problems, such as (1) finding a Euclidean Minimal Spanning Tree, (2) finding the convex hull of S, (3) finding the large empty circle inside the convex hull of S, (4) finding all nearest neighbors, and (5) finding a triangulation with the property that the circumcircle of every triangle is empty.

Shamos [Shamos (78)] has shown that the Voronoi diagram can be constructed in O(nlogn) time in the worst case, and obtained O(nlogn) algorithm for all these problems using the Voronoi diagram.

Even though the Voronoi diagram appears to be a complex object, a Voronoi diagram on n points has only O(n) vertices and O(n) edges joining them. Thus the construction of the Voronoi diagram can be done in linear space, and it is elegantely solved by the "Divide and Conquer" technique. The algorithm [Shamos (78)] first divides the n points into two sets L and R, each of equal size, by a vertical medium line so that every point of L lies to the left of every point in R, and every point of R lies to the right of every point in L. Find the Voronoi diagrams VD(L) and VD(R) recursively. The various structural properties [Shamos (78)] of the Voronoi diagram enables the algorithm to merge VD(L) and VD(R) in O(n) time to get the Voronoi diagram of the whole set. Thus this is an O(nlogn) algorithm.

Shamos and Hoey [Shamos (75)] have collected a group of problems in computational geometry which they refer to as closest point problems. They show how the Voronoi diagram can be used to solve all the closest point problems efficiently (some of these problems are mentioned above). Also recent applications can be found in [Matula (80)], [Supowit (81)], and [Smith (81)]. Here we illustrate how the Voronoi diagram is used to find the Euclidean Minimal Spanning Tree. The definition follows first.

Given n points in the plane, the Euclidean Minimal Spanning Tree (MST) is an interconnecting tree of minimum total Euclidean length whose vertices are the given points. The MST is the one of the classical examples in computational geometry showing that certain problems in graph theory can be solved with substantially lower time complexity when restricted to the Euclidean plane. The problem for finding the MST is usually solved by regarding the points as vertices of a complete n-graph whose edge weights are the Euclidean distances. The usual algorithm, Kruskal's algorithm [Aho (74)], finds the EMST in $O(n^2 \log n)$. Shamos showed that the MST can be constructed in $O(n \log n)$ time using the Voronoi diagram [Shamos (78)].

Given a finite set S of n points in the plane, we first construct the Voronoi diagram for S, $VD(S)$. Every edge of the Voronoi diagram is a segment of the perpendicular bisector of a pair of given points, and is thus common to exactly two polygons (see Figure 1.2). Consider the straight-line dual of $VD(S)$; that is, join p_i and p_j by a line segment if and only if $VP(i)$, the Voronoi polygon of p_i and $VP(j)$ share an edge. One of the important properties of the Voronoi diagram is that the straight-line dual of $VD(S)$ is a triangulation [Shamos (78)], called the Delaunay triangulation (Figure 1.3). The Delaunay triangulation is a planar graph on the given n points consisting entirely of triangles. Hence, it has at most $3n-6$ edges [Harary (71)]. One of the important consequences of this triangulation is that the corresponding Voronoi diagram has at most $2n-4$ vertices and $3n-6$ edges. Also, every Euclidean Minimal Spanning Tree of the given points is a subgraph of the Voronoi dual. Hence the MST must also be a Minimal Spanning Tree of the dual graph [Shamos (78)]. Since the dual graph has at most $3n-6$ edges, the MST can be constructed in $O(n \log n)$ time if we use Kruskal's algorithm on the dual graph. Since the Voronoi diagram can be constructed in $O(n \log n)$ time and the dual graph

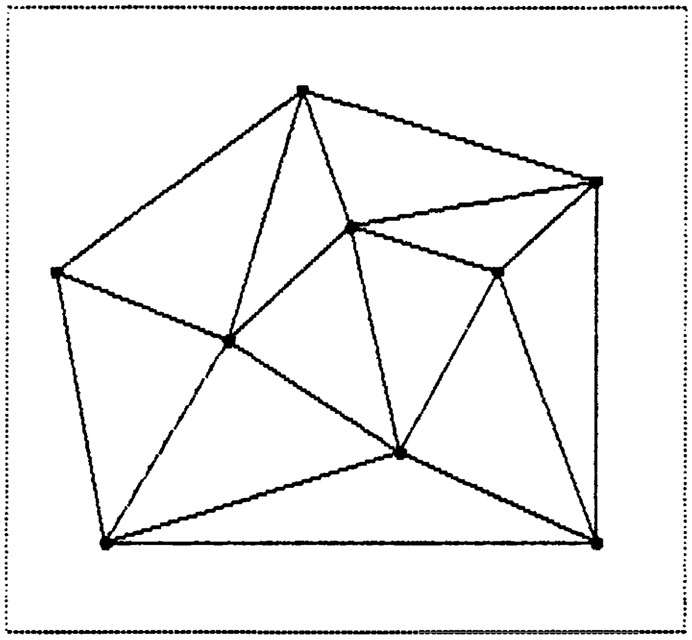


Figure 1.3 Delaunay Triangulation

can be found in O(n) time, the overall time to construct the MST is O(nlogn).

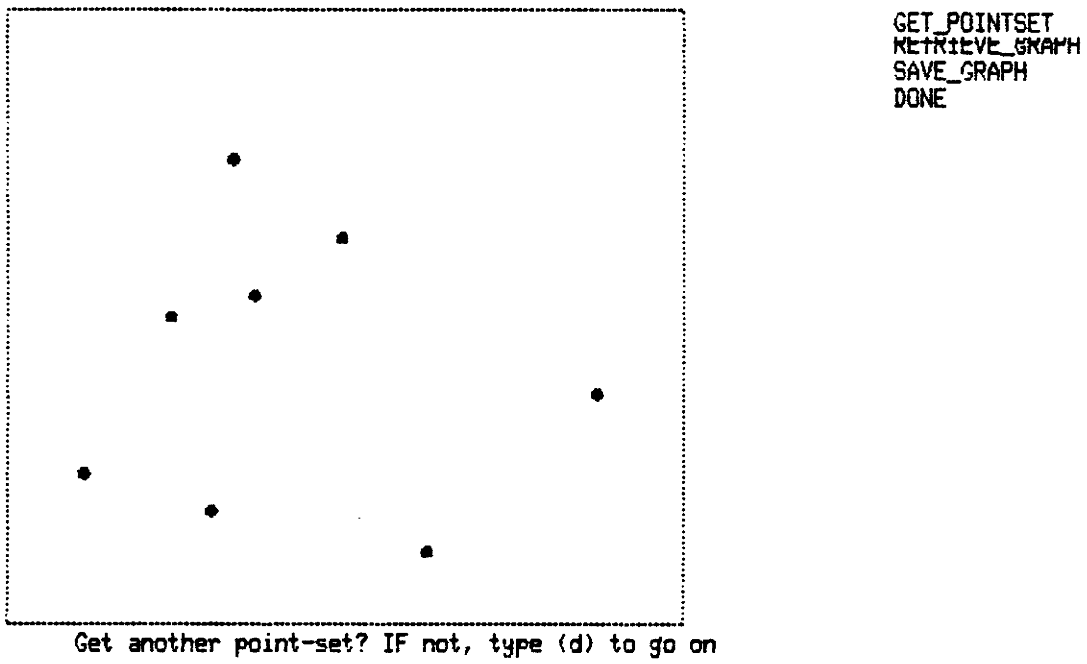
The MST problem is a common component in geometric minimization problems. Some of these problems are: (1) designing a network of minimum cost for a communication system, (2) minimizing wire length for connecting terminals [Loberman (57)], (3) calculating the tariff for private telephone line service (proportional to the length of the minimal network containing the customer's stations), (4) clustering (detecting and describing the structure of point clusters) [Zahn (71)], (5) pattern recognition [Osteen (74)], (6) obtaining approximate solutions to the Traveling Salesman problem [Shamos (78)], and (7) developing a heuristic for Steiner's Problem [Chang (72)], [Smith (81)]. We investigate (7) further in Chapter 3.

1.3 Example Of A Sequence Of Function Invocations

The Geometric Graphics Package contains a function, LIBRARY which enables us to work with various point sets and graphs. The LIBRARY command has three primitives; one for saving a graph, the second for retrieving a new point set, and the third for retrieving a graph which has been saved. This command can be easily implemented on most machines. The main purpose of this function is to build a library containing various graphs and point sets identified by their names. At the beginning of the session, the user is directed to the library to retrieve a point set or retrieve a graph (see Figure 1.1).

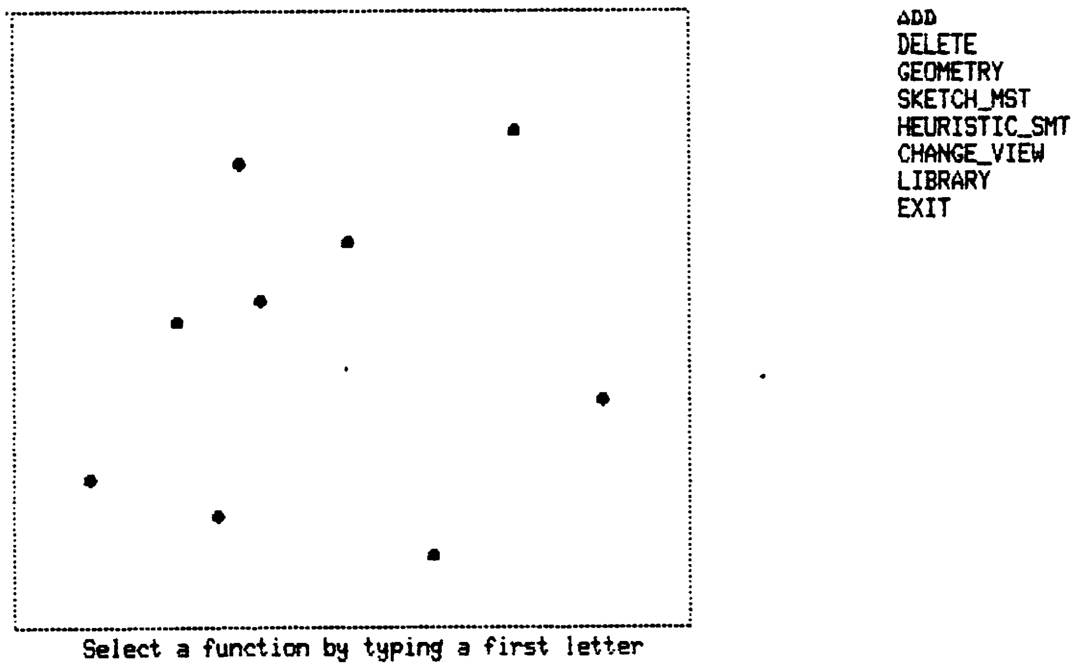
The purpose of this section is to guide the reader through a typical sequence of commands, which will illustrate the power of the Geometric Graphics Package.

Suppose we get a point set from the library (Figure 1.4a). Then the Main_menu is displayed, and the user should select a function. For example, one can add or delete a point (Figure 1.4b). In order to see the geometric structures of the given point set, select GEOMETRY; its menu will appear, and the user can select one of the sub-functions: VORONOI_D, DUAL, MST in any order (Figure 1.4c,d). After examining the geometric structures, select GO_ON, control returns to the main program, and the Main_menu is displayed. If we want to see part of the MST more closely, select SKETCH_MST (Figure 1.4e), CHANGE_VIEW, and ZOOM in that order. The user is allowed to specify a new window (Figure 1.4f) by giving the lower left and upper right corners of the box and redefine a window until the user picks BOX_DONE. The new picture is displayed (Figure 1.4g). Now if the user wants to see the original graph, select OVERVIEW (Figure 1.4h). In OVERVIEW, the current window for the 'earthview' is showed to indicated where the new window has been located relative to the original picture. At this point, we have two choices: go back to the 'earthview' by selecting EARTHVIEW or select another item (ZOOM, TRANSLATE, ROTATE etc). Suppose we pick EARTHVIEW (Figure 1.4i) and we want to rotate the graph. Select ROTATE and provides a point, say the center of the screen, and a 60 degree angle of rotation. Then the graph is rotated 60 degree about the point (Figure 1.4j). To see the Voronoi diagram and the Delaunay triangulation with the current window, select GEOMETRY followed by VORONOI_D and DUAL (Figure 1.4k). Finally, suppose we want to save the original graph for later use and try some other operations on a new point set. Go back to the original window to see the entire graph by selecting OVERVIEW (Figure 1.4ℓ), and then pick DONE. When the Main_menu is displayed select LIBRARY to save the graph and get a new point set (Figure 1.4m). Then the user will manipulate the new point set in same fashion as the old point set.



Get another point-set? IF not, type (d) to go on

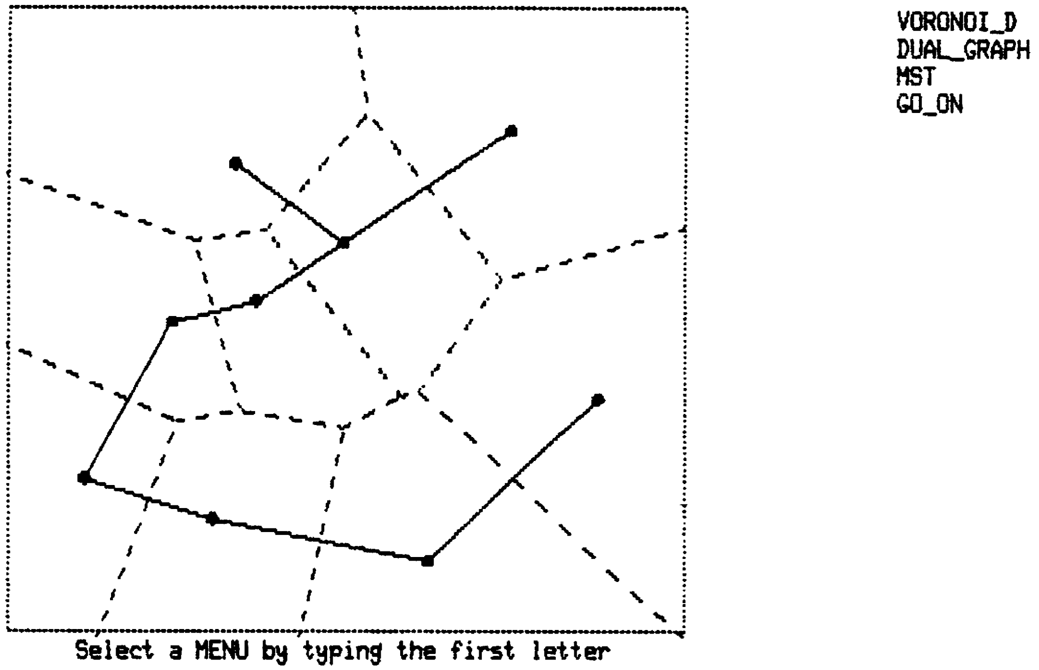
Figure 1.4a A Point-Set from the Library



Select a function by typing a first letter

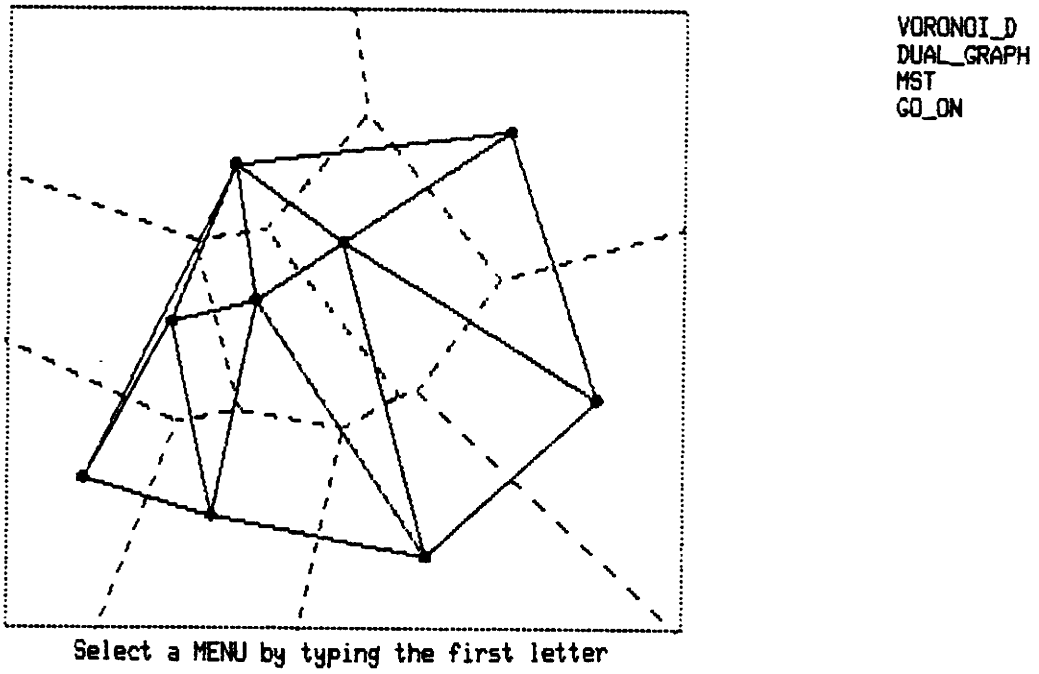
Figure 1.4b Add a point

Figure 1.4 A Sequence of Function Invocations



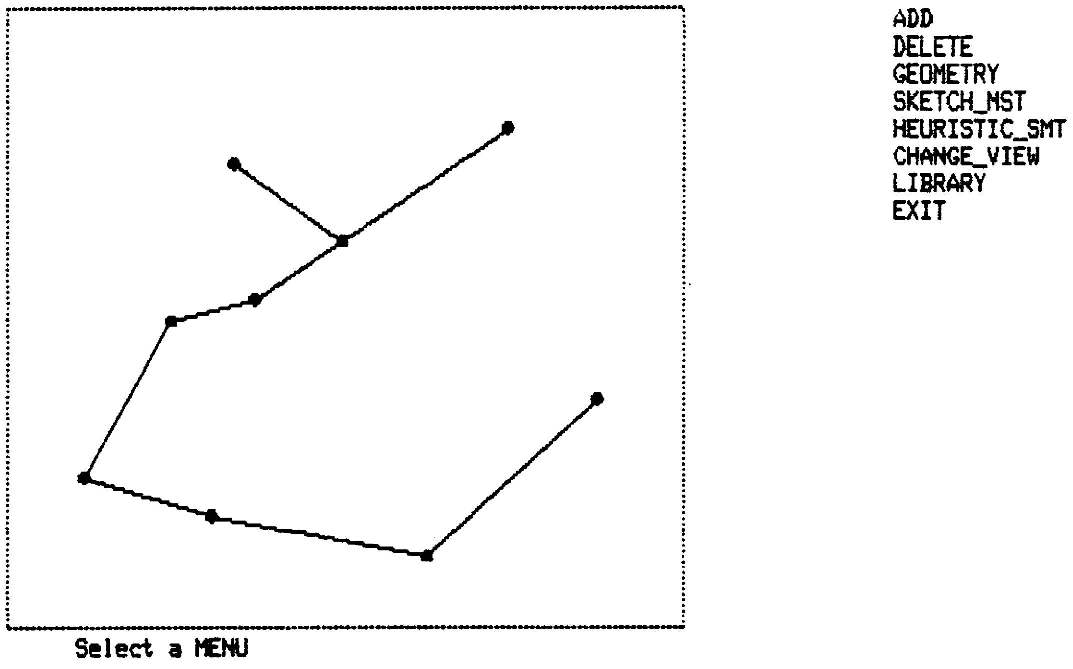
Select a MENU by typing the first letter

Figure 1.4c Voronoi diagram and MST



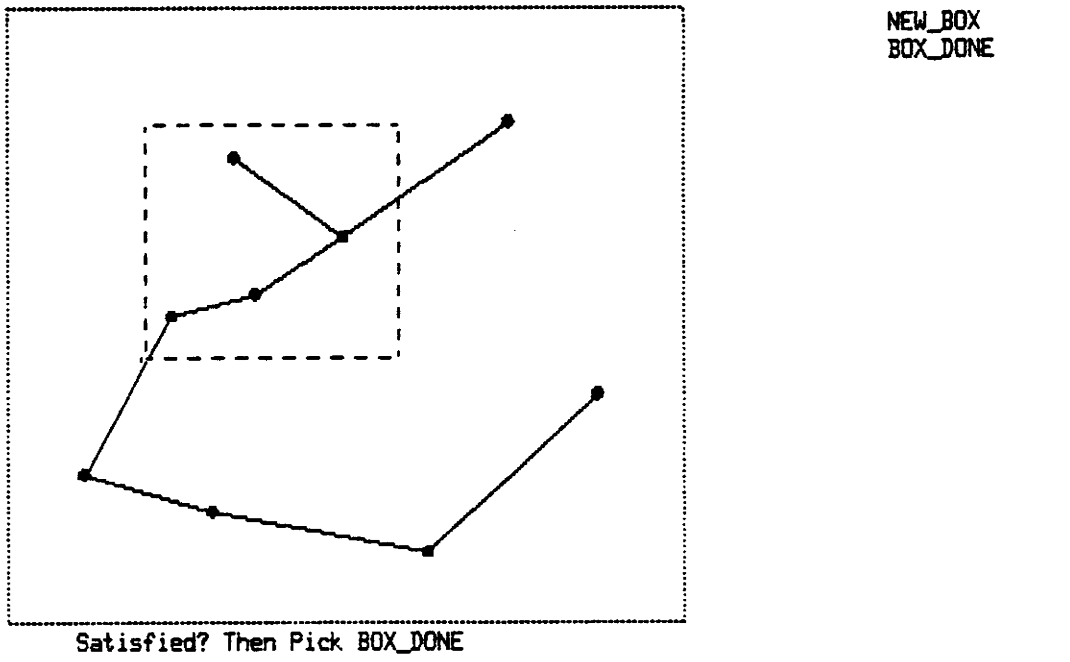
Select a MENU by typing the first letter

Figure 1.4d Voronoi diagram and Delaunay Triangulation



Select a MENU

Figure 1.4e Minimal Spanning Tree



Satisfied? Then Pick BOX_DONE

Figure 1.4f Window Box

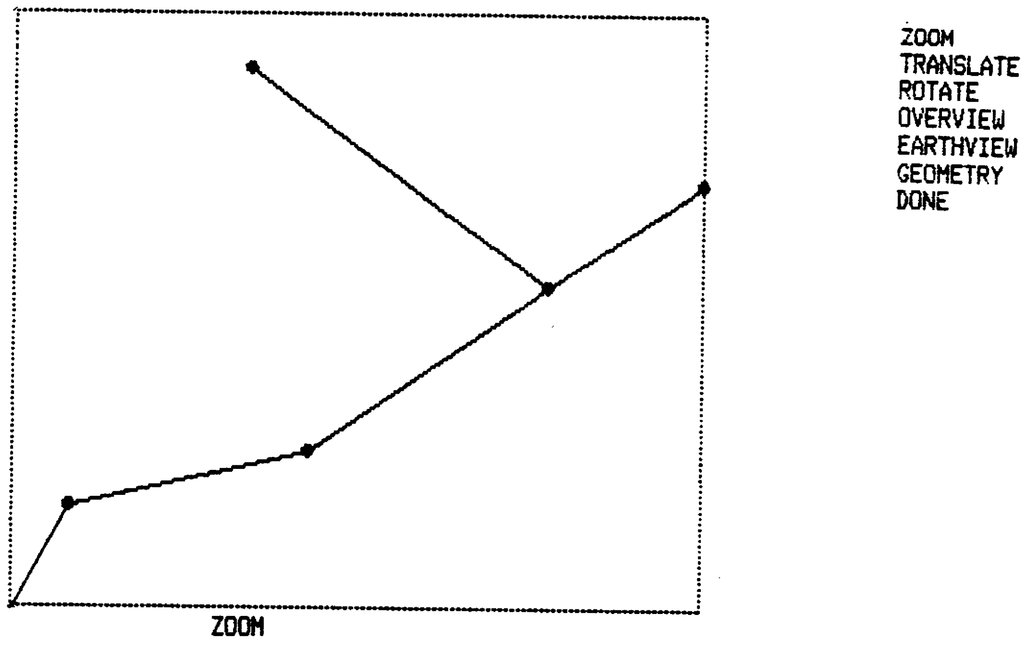


Figure 1.4g MST with the New Window

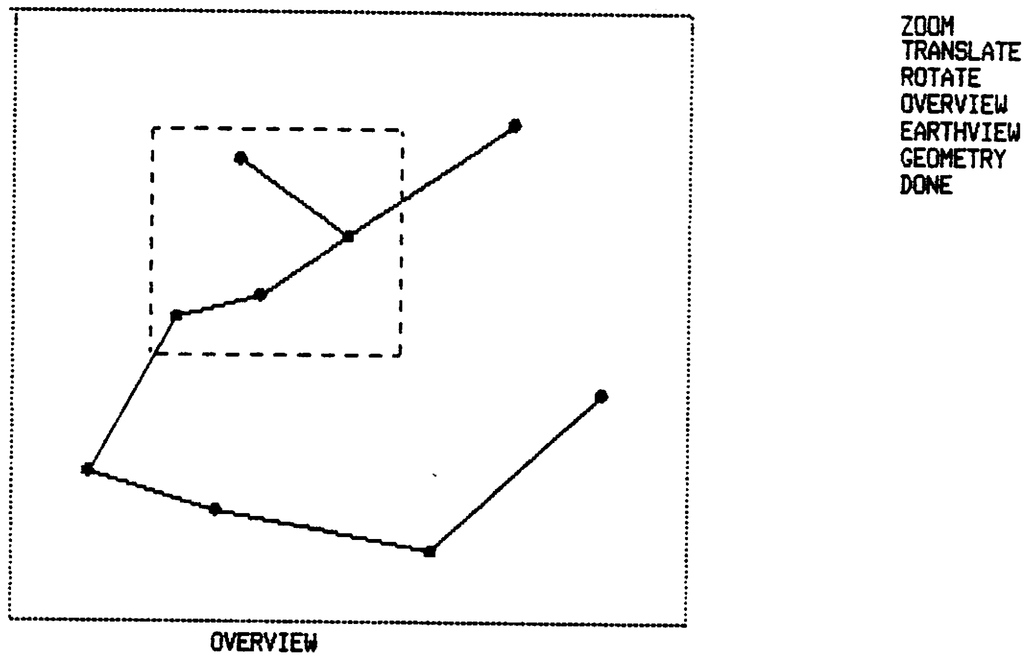


Figure 1.4h MST with the Original Window

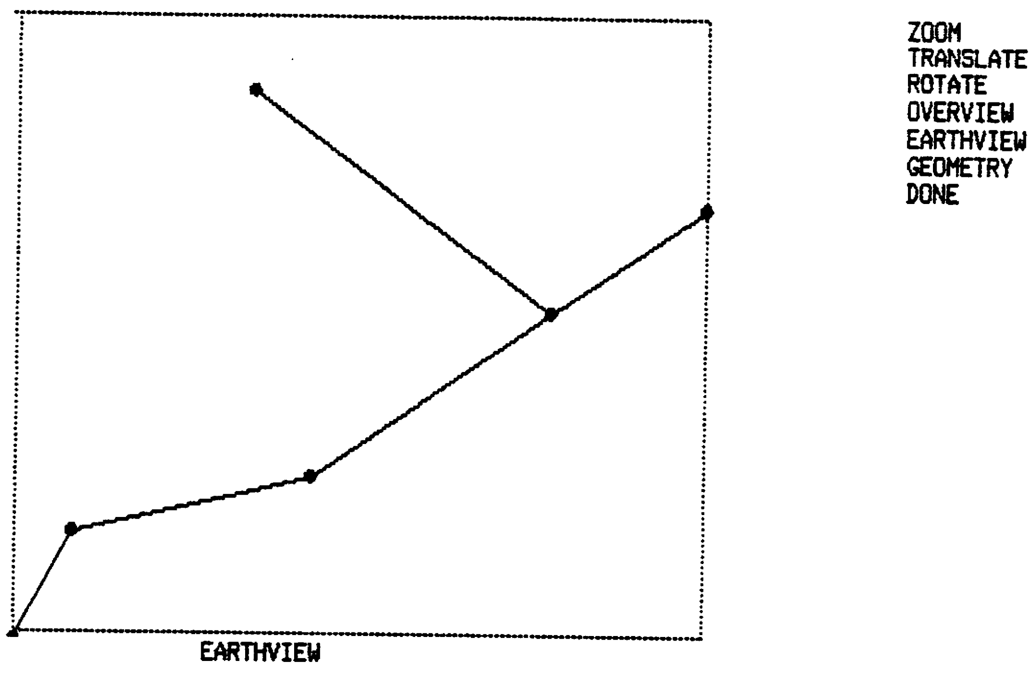


Figure 1.4i Go Back to 'Earthview'

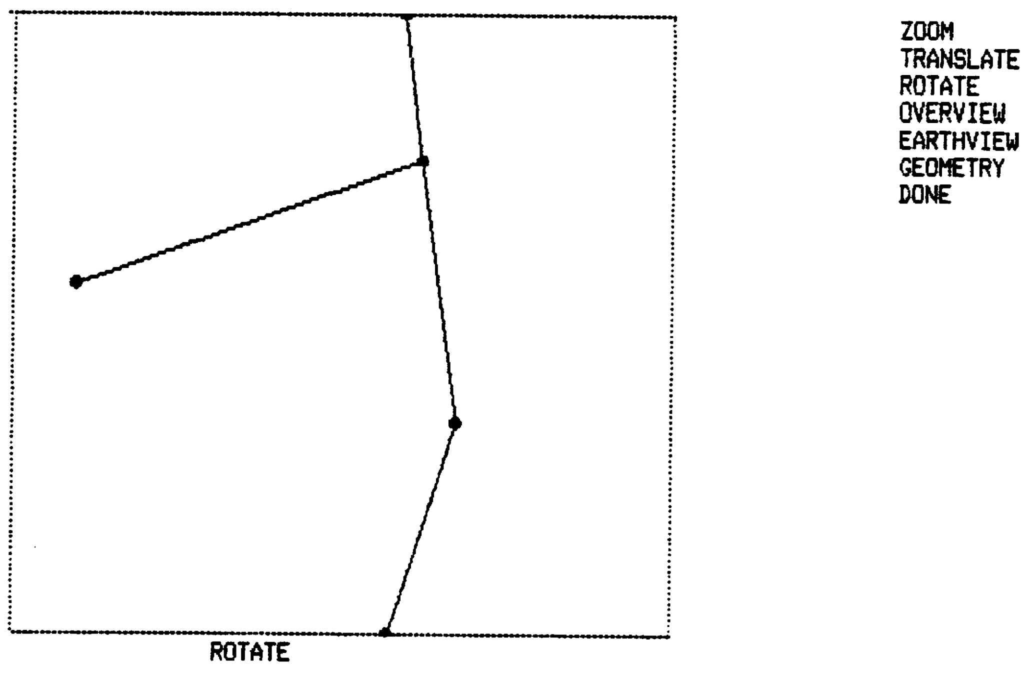


Figure 1.4j Rotate 60° about the center of the screen

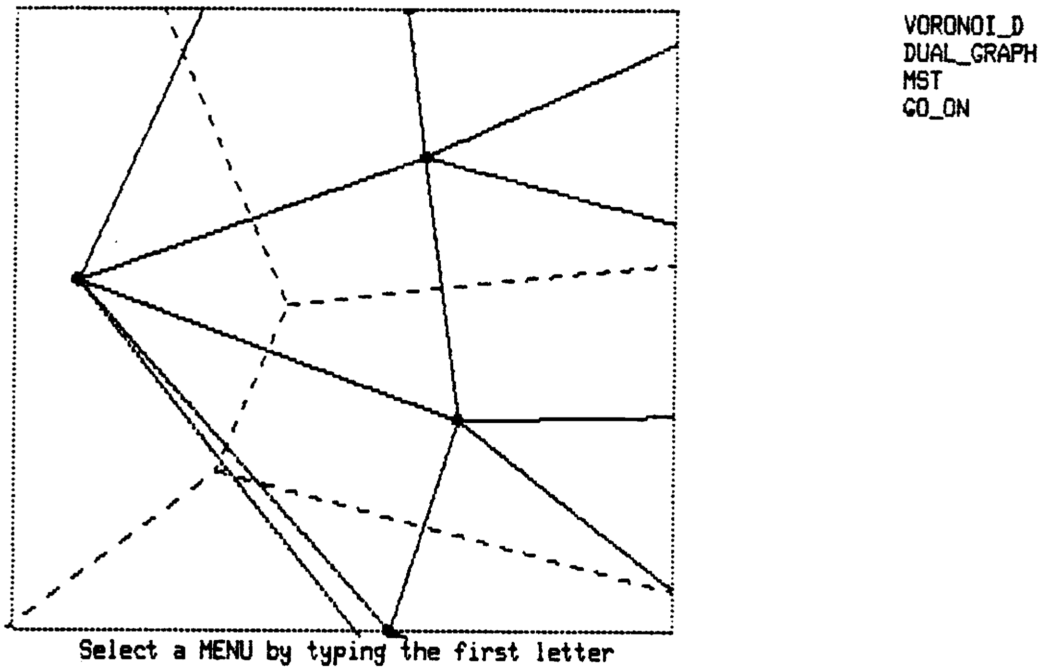


Figure 1.4k Voronoi diagram and Delaunay Triangulation

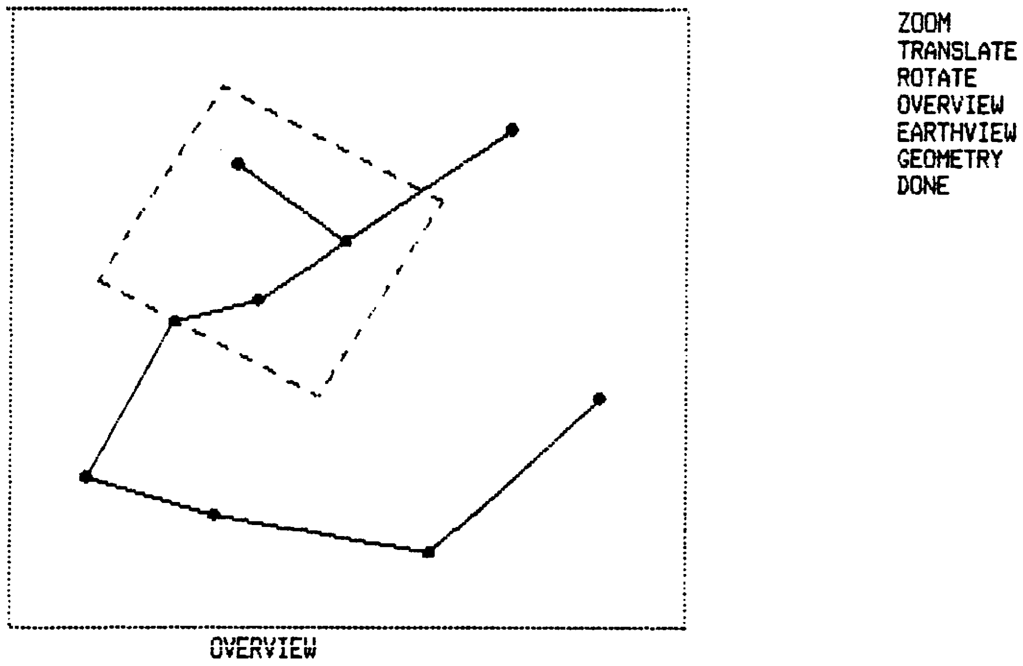
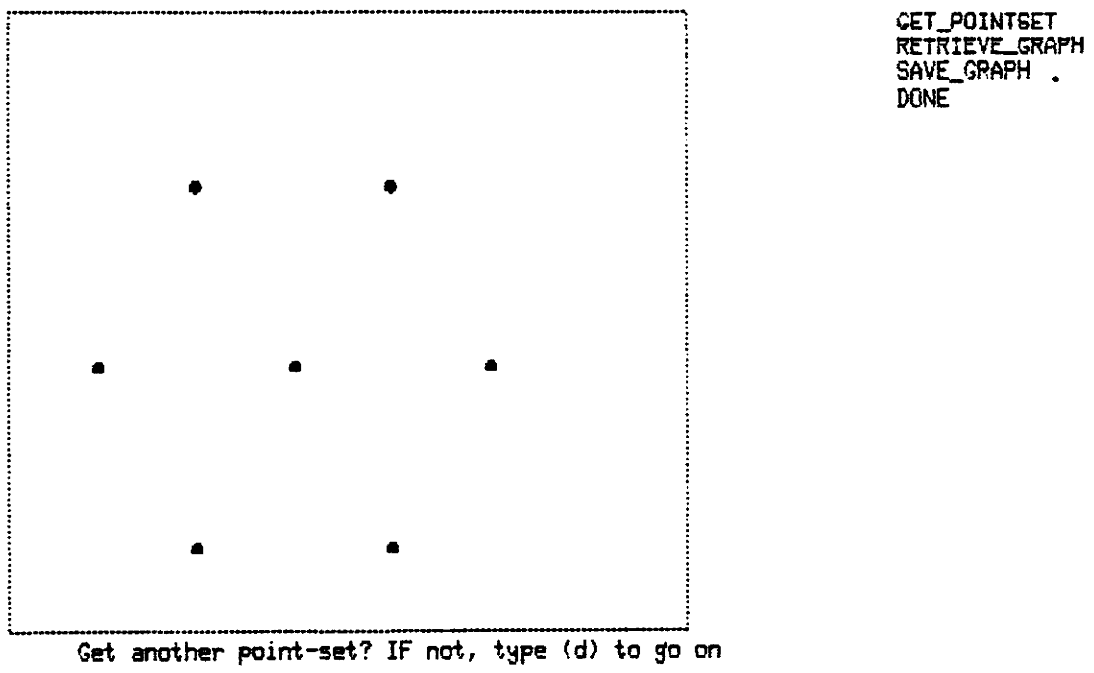


Figure 1.4ℓ Original Graph



Get another point-set? IF not, type (d) to go on

Figure 1.4m New Point-Set from the Library

1.4 Data Structures

In this **graphics package,** the main objects that are created and represented pictorially are sets of points and graphs in the Euclidean plane. The data structures of the application program is very simple.

Point sets, the lowest level sub-objects, are represented as a linked-list of structures, each structure containing the (x,y) coordinates of a point and a pointer pointing to the next point. This data structure makes it easy to alter points individually **(Figure 1.5a).** For a graph the data structure consists of an array of structures, one for each vertex in the graph and a linked list containing a list of edges for the topology of the graph (Figure 1.5b and 1.5c). Each structure in the array has the (x,y) coordinates for a vertex and an attribute of the vertex which can be used to treat points differently, for example, draw a vertex with different color or shape. The indices of the array are used as names of vertices in the graph so that the topology of a graph is represented as a set of integer pairs. This representation of a graph is easily transformed into an adjacency list representation of graph by a single scan of the list of edges.

For the purpose of clean and compact programming, a set of points is regarded as a graph with no edges. Thus procedures in CHANGE_VIEW function operate on the data structure of a single object, a graph.

The followings are the declarations of data structures in the C language:

```
(a)  Struct ℓ-point {
             float  xcord, ycord;
             struct  ℓ-point   *next;
                     }  *org_header;

(b)  Struct vertex  {
             float  xcord, ycord;
             unsigned code;
                     }   org_v [MAXV];

(c)  Struct graph   {
             int  v1, v2;
             double length;
             structure graph *next;
                     }  *org_graph
```

First (a) is for point sets, and (b) and (c) for graphs. MAXV is the maximum number of vertices in a graph, and 'length' in (c) is the Euclidean distance between vertices, v1 and v2.

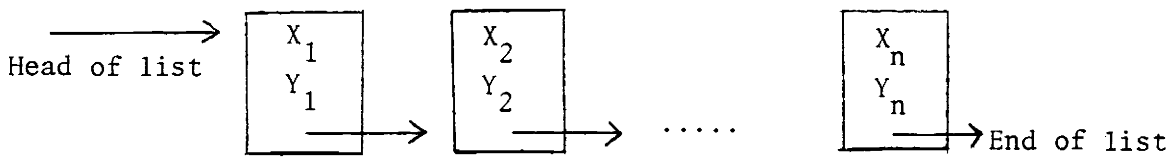


Figure 1.5a Linked list of structures for a set of points

1	2	3	.....	n
X_1 Y_1 Code1	X_2 Y_2 Code2	X_3 Y_3 Code3	.....	X_n Y_n Code n

Figure 1.5b Array of structures for the vertices

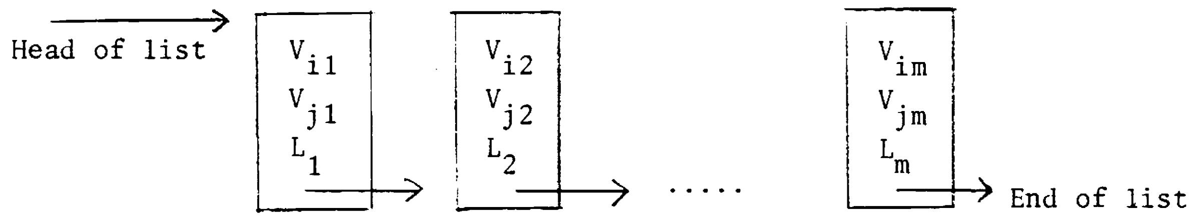


Figure 1.5c Linked list of structures for the edges

Figure 1.5 Data Structures of the Geometric Graphics Package. Figure 1.5a is for point sets, and Figure 1.5b and Figure 1.5c for graphs. In Figure 1.5c, V_{jk} is the name of a vertex which is the index of the array in Figure 1.5b.

CHAPTER 2

STEINER'S PROBLEM IN EUCLIDEAN PLANE

In previous chapter, for given n points in the plane we constructed a spanning tree having these points as its vertices and having the minimum total length (the MST) in O(nlogn) time with Shamos' algorithm. A fundamental condition of the problem finding the EMST is that edge intersections in the tree may only occur at the given n points. What happens when this basic condition is relaxed; that is, suppose new vertices may be added to the original set in order to reduce the total length of a tree connecting the given n points. Is the solution to this new problem still practical to compute? It has been shown [Garey (77a)] that the new problem becomes at least as hard as any of the NP-complete problems. In this Chaper we study this hard (computationally) problem. It is known as _Steiner's Problem_. First we give some definitions and formally define the problem. Then we study a number of basic known properties of the problem.

2.1 _Problem Definitions_

Given n points, A_1, A_2, ..., A_n in the plane, $n > 1$, the _Euclidean Steiner Minimal Tree_ (SMT) is a spanning tree with the shortest possible total length whose vertices contain the given n points. The length of the tree is the sum of the Euclidean length of its edges. The same questions we consider in this thesis can be asked for other metrics, for example the rectilinear metric [Hwang (76)] [Garey (77b)] [Hwang (79)]. Henceforth the length will always refer to the Euclidean metric. In order to achieve the minimum length, the SMT often contains other vertices besides $A_1, A_2, \ldots, A_n$. For example, if A_1, A_2, A_3 are three vertices of an equilateral triangle, then the shortest tree consists of 3 lines from A_1, A_2, A_3 to an extra vertex S inside the triangle (Figure 2.1).

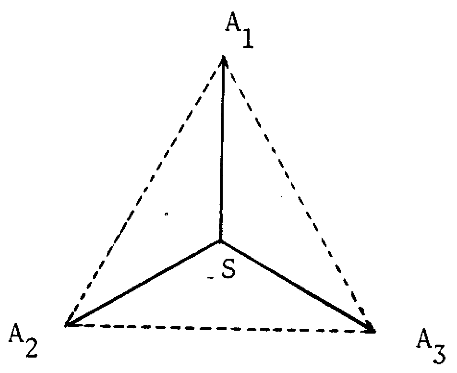


Figure 2.1 Steiner Minimal Tree with 3 points of equilateral triangle

Any extra point S which is added to a tree to reduce its length is called a <u>Steiner</u> point, after Jacob Steiner, the famous geometer at the University of Berlin in the early 19th century. By adding extra Steiner points, a tree of shorter length than the MST is almost always obtainable. The problem of finding the shortest possible tree is called <u>Steiner's Problem</u>.

The Steiner Problem is one of the oldest optimization problems. in Mathematics. It starts with Fermat's Problem (see next section) early in the 17th century. Since the Fermat Problem has been widely popularized by Courant and Robins [Courant (41)] under the name "Steiner Problem", the problem has appeared in numerous places due to its potential applications to practical situations. The SMT arises frequently in problems concerning network design [Werner (69)], optimal location of facilities [Soukup (75)], and in locating processors with minimum line cost in distributed computer system [Chang (76)].

1.2 Basic Properties

When any number of extra Steiner points may be added at will to the given set of points, the problem of finding a shortest tree(SMT) appears an infinite problem. But in 1961, Melzak [Melzak (61)] gave an algorithm for finding the SMT using a number of basic properties of the SMT. Though his algorithm is effective, it is extremely inefficient. We outline his algorithm in this section.

By the topology we shall mean an adjacency matrix, or any equivalent description specifying the connections between points in the given set of points and the Steiner points. The difficulty of the Steiner Problem is in finding the optimal Steiner tree topology. There are too many topologies for even small n that one has to consider in order to find the optimal solution even though many known properties of the SMT rule out a large number of possibilities. Before we study some key properties of the SMT, we give the solution for the simplest case of the Steiner Problem and show the construction of the SMT for that case.

The simplest case of the Steiner Problem is posed as Fermat's Problem:

> Given a triangle T with vertices A, B, C in the plane, find a fourth point S which minimizes the sum of the Euclidean distances
>
> $$|SA| + |SB| + |SC| .$$
>
> The 4th points may coincide with one of A, B, C.

This problem is always exactly solvable. If an angle of T is greater than or equal to 120° at a vertex, then S is that vertex. Otherwise S lies inside of T, and S is the point at which the sides of T subtend an angle of 120° (The proof can be found in [Gilbert (68)], [Coxeter (61)], or try it yourself). Thus, if given a triangle with no angle greater than or equal to 120°, then the unique Steiner point S exists, and S can be constructed as follows.

Let ABC' denote the equilateral triangle exterior to given triangle ABC. Draw a circle circumscribing ABC'. The point S is the intersection of the circle and the line segment CC' (Figure 2.2a). [Coxeter (61)] gives an alternate construction for S. Draw any two of three equilateral triangles ABC', ACB', and BCA' outwards on the sides of ABC. Then S is the common intersection point of any two of the three lines AA', BB', and CC' (Figure 2.2b).

From the proof of Fermat's Problem [Coxeter (61)] it follows directly that the length of the Steiner Minimal Tree on three points is $|CC'|$ (or $|BB'|$ or $|AA'|$ in the latter case) if S exists. (Note that this fact together with above construction is used to find

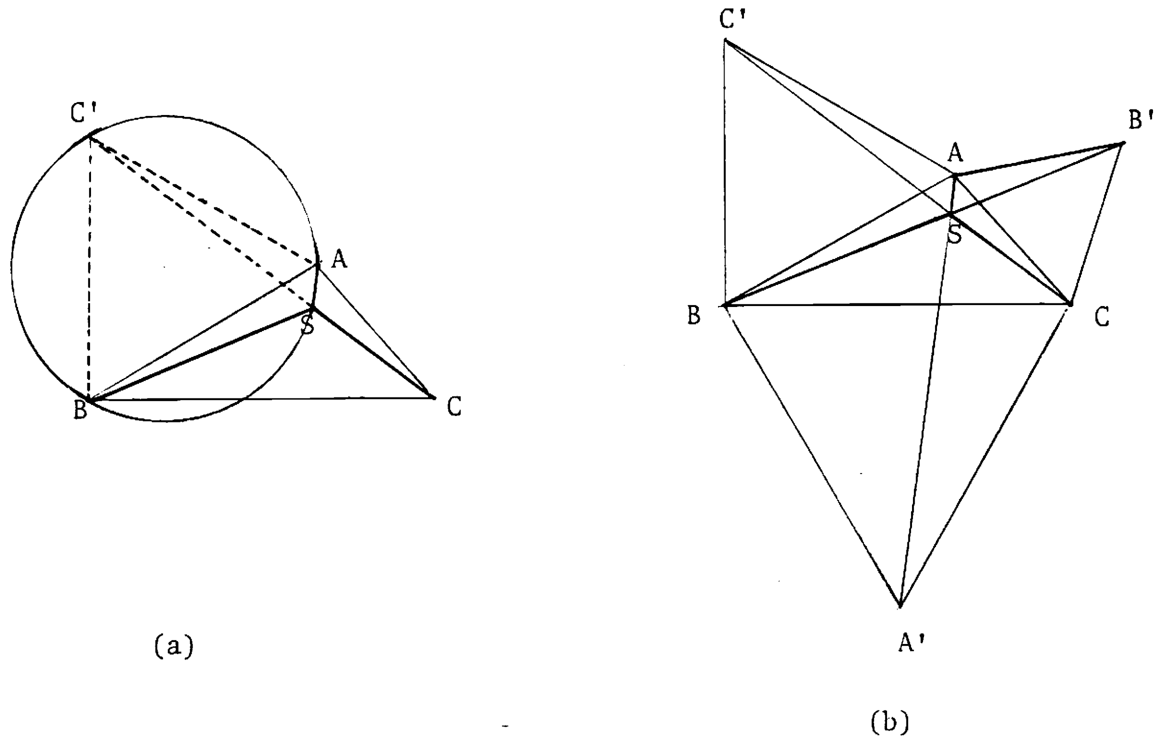


Figure 2.2 The Construction of the Stainer point S for n=3.

the Steiner Minimal Tree with a given topology in general case, $n \geq 3$. Imagine for a moment we have more than 3 given points. From Figure 2.2a, $|CC'|$ allows the replacement of two given fixed points, A and B by a new point C'. Then we have one less given point and one less Steiner point. This simpler problem has the same length as the old one!)

Let an Euclidean Steiner Minimal Tree have $A_1, A_2, \ldots, A_n$, $n \geq 3$, given points and $S_1, S_2, \ldots, S_m$, Steiner points. We study 7 key properties of a SMT.

(Property 1) In the SMT, two lines (edges) which intersect, whether at a fixed point or a Steiner point, form an angle which is always greater or equal to 120°.

The proof can be found in [Gilbert (68)], [Werner (69)]. From this property the followings are self-evident:

(1) Each original point A_i, $1 \leq i \leq n$, has at most degree 3.

(2) Each Steiner point S_j, $1 \leq j \leq m$, has exactly degree 3. If a Steiner point has two incident lines, then they can be replaced with a straight line by eliminating the Steiner point.

(3) Each Steiner point S_j, $1 \leq j \leq n$ is the Steiner point of the triangle formed by the points which are directly adjacent to S_j in the SMT. Thus if we know that 3 points should be connected to a Steiner point, we can find this Steiner point by the above construction.

(Property 2) The number of Steiner points in the SMT is at most n-2; that is, $0 \leq m \leq n-2$.

Proof: Let E be the number of edges in the SMT. Then from Property 1 $2E = 3m + n_1 + 2n_2 + 3n_3$, where n_k is the number of given points with degree k. Since a tree of x vertices has x-1 edges, $E = (n_1+n_2+n_3+m)-1$. Thus,

$$m = n - 2 - n_2 - 2n_3.$$

In particular,

$$m \leq n - 2,$$

with equality holding if and only if each A_i is of degree 1. Q.E.D.

This is the key property which is used to show the SMT can be constructed in a finite number of steps.

(Property 3) the SMT is planar; that is, its edges do not cross.

A tree satisfying Properties 1, 2, and 3 is called the Steiner tree.

Before preceeding to Property 4, we need some definitions. The convex hull of a set is the intersection of all convex set containing it. A line L is a line of support of a set S if it meets the boundary of S and S lies entirely on one side of L [Shamos (78)].

(Property 4) In the SMT , every Steiner point lies in the convex hull of the given fixed points, A_1, A_2, ..., A_n. That is, no supporting line L of the convex hull separates certain Steiner points, S_1, S_2, ... from the given points A_1, A_2,...,A_n.

The proof is in [Gilbert (68)]. Because of this property, we do not consider any possible Steiner points outside of the convex hull of given n points. For example, for the case $n = 3$ the Steiner point lies inside of the triangle formed by three given points - if it exists.

(Property 5) For every n, a SMT can be constructed in a finite number of steps.

This says that the Steiner problem is at least a finite problem. This was first proved by [Melzak (61)]. His algorithm basically tries (enumerates) all the possible topologies for given n fixed points and m Steiner points, $0 \leq m \leq n-2$. We now outline the proof of Melzak.

Given n points, A_1, A_2,...,A_n, $n \geq 3$ together with m Steiner points, S_1, S_2,..., S_m and a topology, the (unique) relatively minimal tree (minimal tree relative to a given topology) is found by induction on m, the number of Steiner points. The construction of a tree with $m = 0$ is trivial. Just construct an MST. Consider the case $m \geq 1$.

The first step is to find a Steiner point S which the topology connects to two vertices A_i, A_j. Such an S exists for if no such S exists, then every Steiner point has at least 2 adjacent Steiner points for deg(S)=3 and thus $n < m$. But we know $m \geq n-2$. Thus at least one Steiner point S must connect to two given points. As in the case n = 3, find the substitution point C' by drawing the quilateral triangle with A_iA_j. We must try 2 choices for C'. Using Property 4 or else we may not consider the other possibility for C'. C' is added as a new fixed point in place of A_i and A_j, and we remember the connection of A_i, A_j with S. The remaining topology is same as before and we have n-1 fixed points and m-1 Steiner points. The tree for the smaller problem is found by the inductive procedure. Then the location of the point C is known if it were a Steiner point. Now using the same construction as n = 3, the S is located and we simply replace the line CC' by A_iS, A_jS and CS to get the desired tree (also see the note in the case when n = 3).

When the relatively minimal tree is found, we check whether the tree is a Steiner tree, a tree satisfying Properties 1,2, and 3. The reason for this is that a SMT is a Steiner tree of minmum total length. To find a SMT by exhausting all possible relatively minimal trees one needs a way of listing the topologies. Gilbert and Pollark [Gilbert (68)] gave a way to enumerate the topologies and derived the formula for the number of different topologies for Steiner trees with n given points and m Steiner points. With their conventions used in counting topologies:

$$\frac{2^{-m} \binom{n}{m+2} (n+m-2)!}{m!}$$

Some representative values for this formula are shown in Table 1. This completes the outline of the proof of Property 5.

Table 1

Number of Possible Topologies for Steiner Trees with n Given Vertices and m Steiner points

m \ n	3	4	5	6	7
0	3	12	60	360	2520
1	1	12	120	1200	12600
2		3	75	1350	22050
3			15	630	17640
4				105	6615
5					945
Total	4	27	270	3645	62370

Source: Gilbert and Pollak (1968)

Although a number of geometric properties [Gilbert (68)] of the SMT could be used to rule out a large number of topologies, there are still too many cases. We are told that with present technology the problem with more than about 15 points cannot be solved [Boyce (75)]. For the moment, there appears to be little hope for finding the optimal solution for the problem. Indeed the following property destroys any hope for finding an efficient algorithm for the Steiner Problem.

(Property 6) The Steiner Problem in the Euclidean plane is inherently at least as difficult as any of the NP-complete problems.

The proof is in [Garey (77a)]. For technical reasons, which involve the possibility of Steiner points of irrational coordinates and because of the irrationality of the Euclidean metric, they discretize every point appearing in the problem to integer coordinates. Then they use the discretized Euclidean metric and show that the Discrete SMT problem is NP-complete, by a transformation from X3C (Exact cover by 3 sets), a known NP-complete problem.

Thus finding the optimal solution for the Steiner Problem is computationally hopeless unless P = NP and the emphasis on heuristics and special case algorithms for the problem is well justified, In next Chapter we present an interative heuristic using fixed tension physical relations for the Steiner Problem. Other heuristics can be found in [Chang (72)],[Smith (81)].

(Property 7) Conjecture: The lower bound on the length of the SMT was conjectured as $\sqrt{3}/2$ (.8660254...) times the length of the MST by Gilbert and Pollark[Gilbert (68)].

Since the length of the MST (an upper bound) can be easily computed, if we know the exact lower bound on the ratio of the SMT/MST, then it would be very useful in determining the accuracy of an approximate solution. The conjecture has been shown to be true for n = 3 [Gilbert (68)] and n=4 [Pollark (78)], [Du (82)]. In fact, equality is achieved. The known lower bound on the ratio of the

SMT/MST has been improved from $1/\sqrt{3}$ (.5771...) [Graham (76)] to $1/3(2\sqrt{3} + 2 - \sqrt{7+2\sqrt{3}})$ (.74309...) [Chung (78)].

CHAPTER 3

A HEURISTIC FOR STEINER'S PROBLEM

In this Chapter, we present an iterative heuristic using fixed tension physical model for Steiner's Problem. We have already mentioned on Chapter 2 that the difficulty with Steiner's Problem is in finding an optimal Steiner tree topology. One approach to find such a topology is motivated by a physical interpretation of the Steiner Minimal Tree. If a tree is interpreted as a mechanical system in which the potential energy is a sum of distances between adjacent vertices, then the mechanical system described by the Steiner Minimal Tree is in stable equilibrium. Consider the following mechanical model: given fixed points are represented by fixed pegs, and Steiner points are represented by movable pegs on a board. A string is looped about each peg according to the desired topology. By pulling the string with a fixed tension, the movable pegs approach an equilibrium positions which minimizes the length of strings. This model assumes that the topology is given. Thus the equilibrium position will be a relatively minimal tree (A detailed description of this model can be found in [Miehle (58)]). The idea behind this model is that the resultant of the forces in the strings acts on each point to drive the system to equilibrium. Our heuristic is based on the same idea and uses the the Euclidean Minimal Spanning Tree as a guide for obtaining an approximate solution for Steiner's Problem. It should be mentioned that the heuristic algorithm is in an experimental stage. The Graphics Package developed in this thesis is to be used as a tool to help in the further development of the heuristic algorithm described in this chapter. We will illustrate the behavior of the heuristic on certain point sets after we formally describe the algorithm in Section 1.

3.1 The Heuristic Algorithm

Let A be the set of given points, $P_1, P_2, \ldots, P_n$, $N \geq 3$. First we construct an Euclidean Minimal Spanning Tree for A. From the MST of A, we obtain a set S_1 of Steiner points in the following manner. For each vertex P_i, if P_i is of degree greater than 1, then a new Steiner point is introduced in the direction of the resultant force on the point P_i, where each component force on P_i is given by applying a fixed unit tension on the edges eminating from P_i. Let R_i denote the resultant vector, then

$$R_i = \sum_{k=1}^{d} \frac{p_{j_k} - p_i}{\|p_{j_k} - p_i\|} , \qquad (1)$$

where p_{j_k} is the adjacent vertex of P_i, $k = 1, 2, \ldots, d$, and $d = \deg(P_i)$. Each vertex P_i is considered to be two component vectors and $\| P_i - P_j \|$ is the Euclidean norm. With R_i we define a new Steiner point s_i by

$$s_i = p_i + \lambda R_i , \qquad (2)$$

where λ is the step size, a constant. It is not necessary that the step size be a constant, but we assume so for the purpose of simplicity.

We now have a set S_1 of Steiner points. Let A_1 be a new set of points containing the given fixed points in A and the Steiner points in S_1. We construct a MST of A_1. From the MST of A_1, we obtain a new set of Steiner points in somewhat the same manner as above. To those vertices in the MST of A_1 which are the given fixed points in A, we apply the same process as before. But now each point will have an adjacent Steiner point. With a vertex which is a Steiner point, we do one of several things depending on the degree of the Steiner point. Let s denote a Steiner point in the MST of A_1. If s is of degree 3, then we simply move the Steiner point to a new location obtained by the formulas (1) and (2) with s. If s is of degree greater than or equal to 4, we introduce a new Steiner point by applying the formulas (1) and (2) with s, and keep the old Steiner point s as it is. If s is of degree 2; that is, it has two incident lines, then they are replaced with a straight line by eliminating the Steiner point. We do this process for each Steiner point in the MST of A_1. At the end, we combine the fixed points in A and a new set S_2 of Steiner points. Then we construct a MST with A_2 and the process continues as in previous step.

Because the algorithm is iterative in nature, the process can be terminated at any stage. Currently, we use two stopping criteria. First, we stop when the maximum change of magnitude from an old Steiner point to a new Steiner point becomes small. Second, we stop if the number of iterations exceeds a given bound. Before we discuss more about the heuristic, we formally state the algorithm.

ITERATIVE HEURISTIC ALGORITHM

```
[Step 1] Construct a MST with A', A' = {A U S}, where
         A = $\{p_1, p_2, \ldots, p_n\}$, the set of given points, $n \geq 3$, and
         S = $\{s_1, s_2, \ldots, s_m\}$, the set of Steiner points, for some m.
         Initially, m = 0.

[Step 2] With the MST of A' do the following.
         Set S' = ∅ .
         For each vertex $v_i$ in the MST of A', i = 1,2,..., |A'|.
             If $v_i \in A$ and $\deg(v_i) > 1$ Then
                 Introduce a new Steiner point, using the formulas
                 (1) and (2). Add the Steiner point s to S'.
             If $v_i \in S$ Then
                 (Case 1) $\deg(v_i) = 3$
                          Apply (1) and (2) to $v_i$ to get a new
                          Steiner point s. Add s to S'.
                 (Case 2) $\deg(v_i) > 4$
                          Apply (1) and (2) to $v_i$ to get a new
                          Steiner point s. Add $v_i$ and s to S'.

[Step 3] Check the termination conditions. Halt if so indicated by
         the stopping criteria.

[Step 4] Set S to S', and go to Step 1.
```

The basic idea of this heuristic algorithm is that with the appropriate step size λ in formula (2), say small enough or big enough, the algorithm will halt at a desired solution. By introducing appropriate Steiner points in each stage of the algorithm, we use the Euclidean Minimal Spanning Tree on the given points and new Steiner points to prescribe an overall topology. If, for each iteration, the topology never changes, the algorithm reduces to the peg and string model. At this stage we do not have a good way to compute the step size for a given point set. Thus we initially set the step size to a constant, but the algorithm is set up in such way that we are able to change the step size at any iteration step. Experimental studies indicate that a good choice of the step size is needed not only for fast convergence, but also in changing the overall topology of the newly computed point sets. We conjecture that the step size should vary in each iteration and at each point. Right now the algorithm only converges successfully on certain point sets, and it fails to converge on others. In next section, we will illustrate some of these cases.

By simply visualizing the behavior of the heuristic, using the Graphics Package developed in this thesis, we were able to get desired solutions to certain point sets and also draw many inferences needed for refining the heuristic in future studies. Also the further studies should indicate how the known properties of the MST (in previous Chapter or elsewhere) can be fully exploited. Using the geometric structures, the Voronoi diagram and the Delaunay triangulation, available in our Graphics Package should help in improving any heuristic for Steiner's Problem.

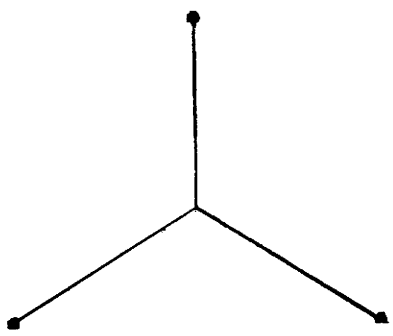
3.2 Experimental Results

In order to use the Graphics Package, the Heuristic algorithm was implemented as a function in the Main-menu. It was set up in such a way that the MST for given fixed points and a new set of Steiner points at each iteration is displayed on the screen.

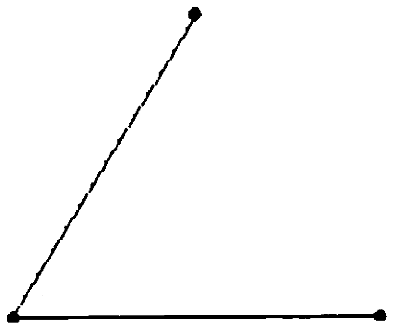
We know that a MST has at most n-2 Steiner points where n is the number of given points. A Steiner tree with n-2 Steiner points is called the full Steiner tree. In the full Steiner tree, each of given points has one incident edge and it leads to a Steiner point. We mainly illustrate the experiments with certain point sets of which the optimal solutions are the full Steiner trees. We conjectured that since our heuristic introduces a new Steiner point when a given point in the MST is of degree greater than 1, it should perform relatively well on those point sets. It turned out that it is true for small n, n = 3,4,5, but it is not so when n gets large. One advantage with those point sets was that we were able to compare the experimental results and the exact answers.

Gilbert and Pollark [Gilbert (68)] showed full Steiner tree with the number of Steiner points up to 7. We illustrate the experiment on six of those trees with S = 1,2,3,4,5. The set of fixed points for each tree is the input point set. Figure 3.1 shows three trees for each point set: first the full Steiner tree, second the MST of given fixed points, which the heuristic algorithm constructed at the beginning, and third the tree to which the heuristic algorithm converged with the specified step sizes. For the first three cases when s = 1,2,3 (Figure 1.3 a,b,c), the heuristic algorithm converged to the optimal solution. For the case of s = 4 (Figure 1.3d), we do get the full Steiner tree topology, but the heuristic did not converge to the desired tree. We have another full Steiner tree with s = 4 (Figure 1.3e). This time the heuristic reduced to the peg and string model represented by the MST of given fixed points. For the case of s = 5 (Figure 1.3f), the heuristic again fails to converge to the full Steiner tree. Figure 4.1 also shows the total lengths of the MST and the resulting tree and the ratio of two trees.

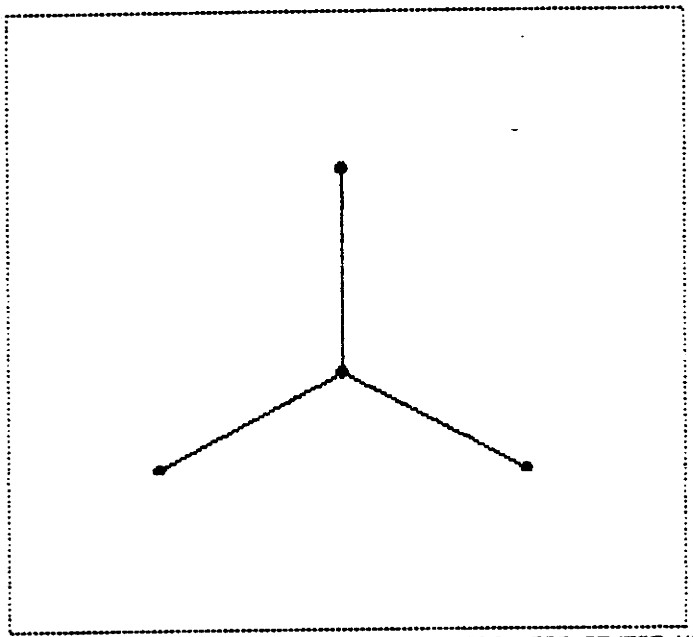
At present, the heuristic algorithm is in experimental stage. We did not exploit fully the geometric properties of the optimal solution for the Steiner Problem. Many inferences drawn from the experiments must be analyzed to find their places. Having the Geometric Graphics Package containing many useful functions at hand, the refined version of our somewhat crude heuristic algorithm should come in the near future.



Full Steiner tree with s=1



MST

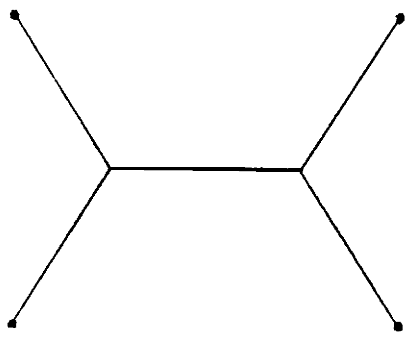


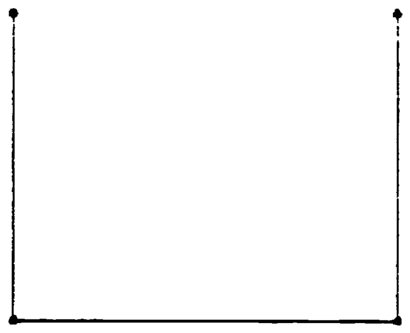
Step size: 0.6

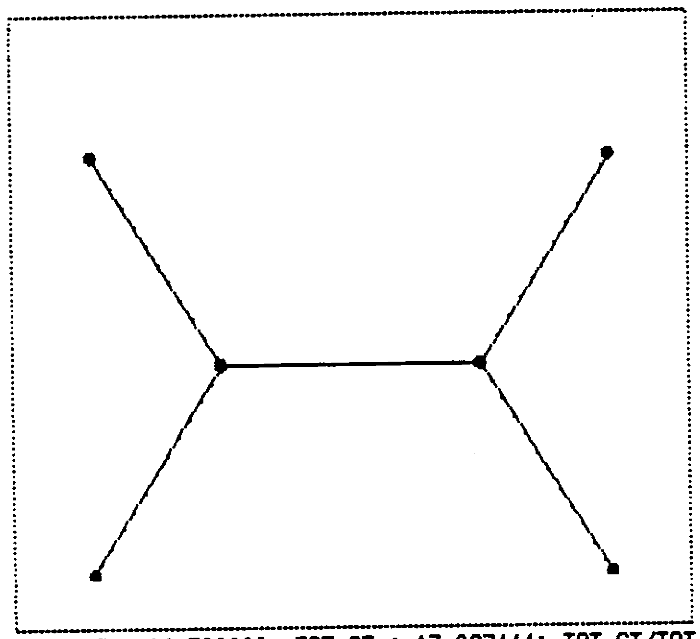
```
TOT_MST : 7.610861; TOT_ST : 6.622595; TOT_ST/TOT_MST :  0.8701506
```

Figure 3.1a

Figure 3.1 Experiments on 6 full Steiner trees with s Steiner points. First tree is the full Steiner tree with s Steiner points. Second tree is the MST of given fixed points. Third tree is the resulting tree from the heuristic algorithm.

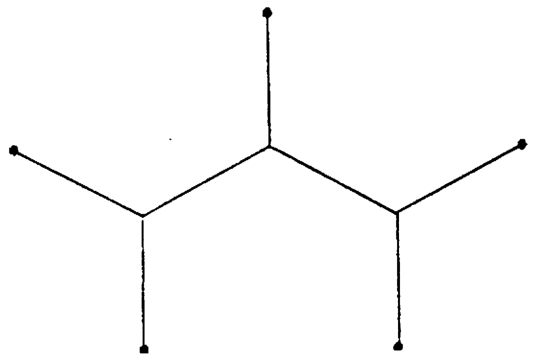

Full Steiner tree with s=2


MST

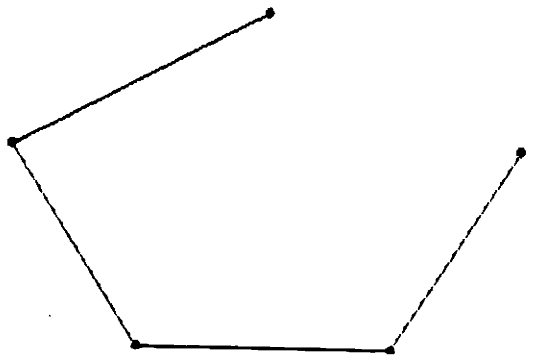

Step size: 0.5

```
TOT_MST : 14.500000; TOT_ST : 13.267444; TOT_ST/TOT_MST :  0.9149961
```

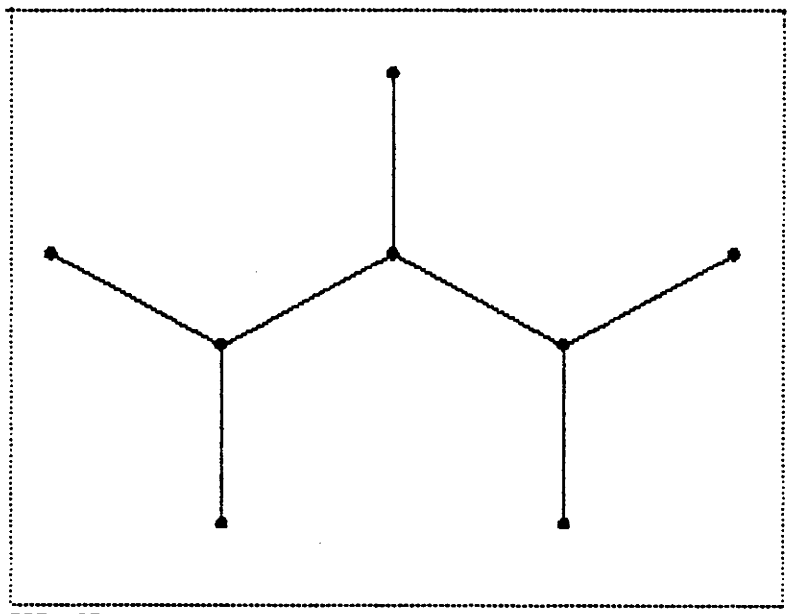
Figure 3.1b



Full Steiner tree with s=3



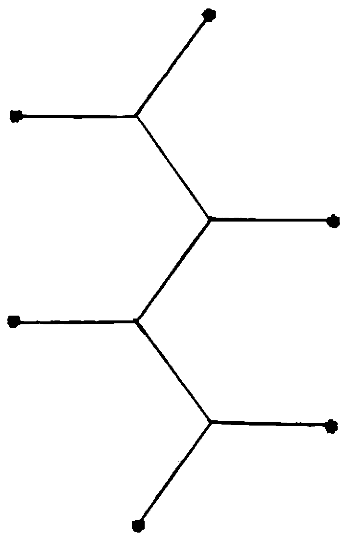
MST



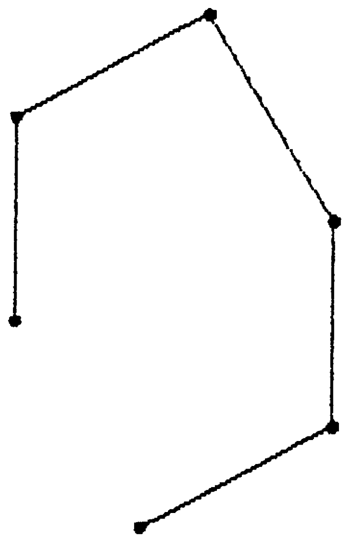
TOT_MST : 14.477351; TOT_ST : 14.062178; TOT_ST/TOT_MST : 0.9713226

Step size: 2.2,0.3
1.0 in that
order

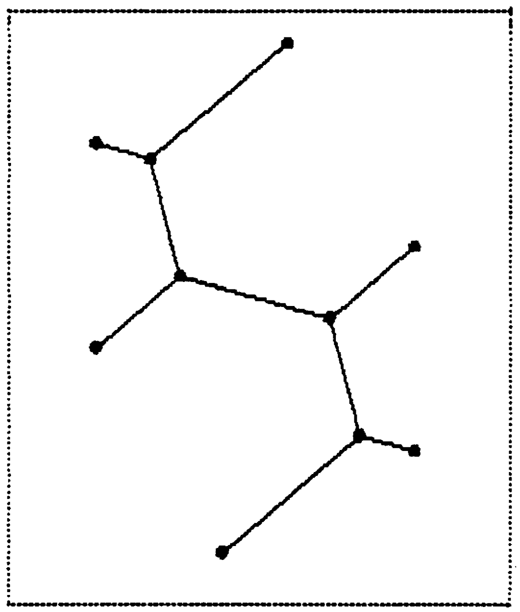
Figure 3.1c



Full Steiner tree with s=4



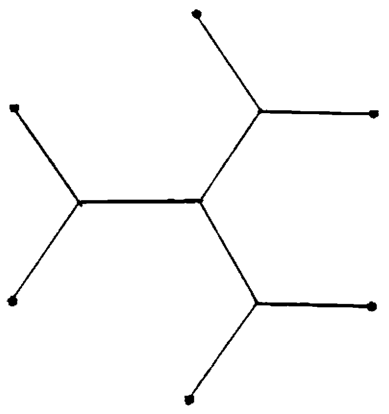
MST



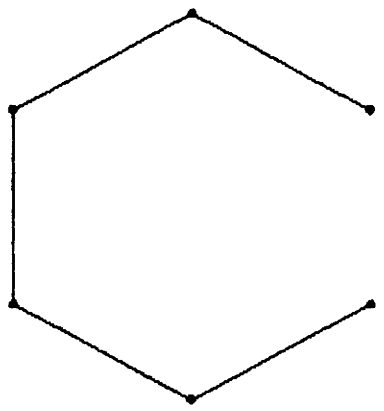
Step size: 0.8, 0.4 in that order

TOT_MST : 17.977351; TOT_ST : 17.865610; TOT_ST/TOT_MST : 0.9937844

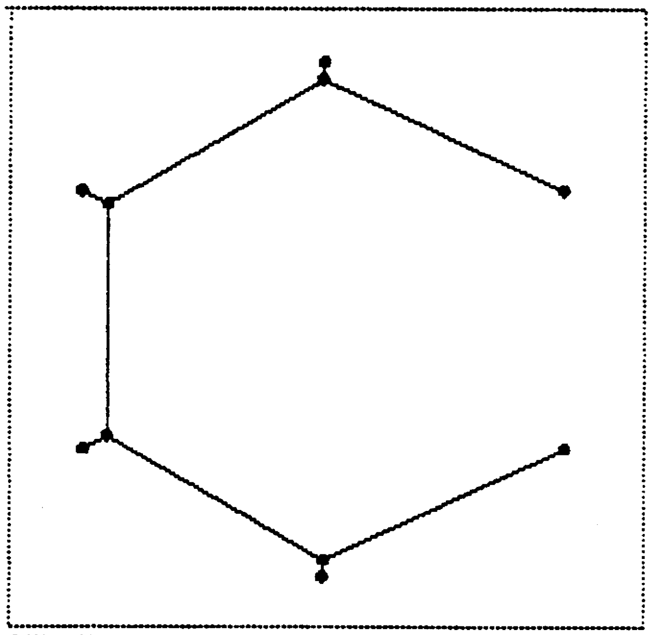
Figure 3.1d



Full Steiner tree with s=4



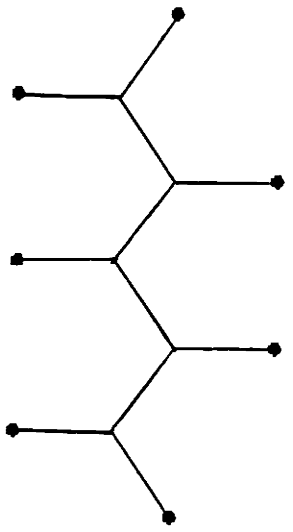
MST

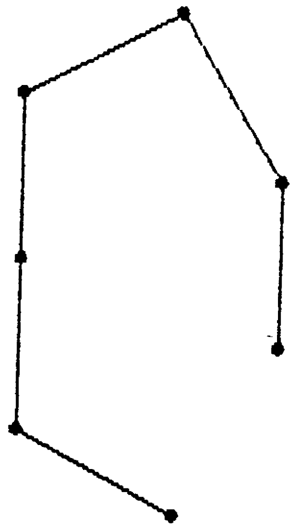


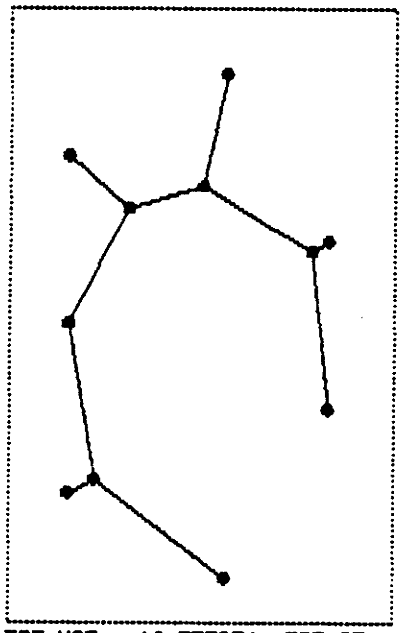
Step size: 0.5

```
TOT_MST : 17.392444; TOT_ST : 17.407738; TOT_ST/TOT_MST :  1.0008794
```

Figure 3.1e


Full Steiner tree with s=5


MST



Step size: 1.5,0.6,0.3 in that order

```
TOT_MST : 16.353974; TOT_ST : 16.321743; TOT_ST/TOT_MST :  0.9980291
```

Figure 3.1f

CONCLUSION

Computer graphics is used today in many different areas of industry, business, education, entertainment, and even in the home. The list of applications is enormous and growing rapidly with the development of technology. In this thesis an interactive computer graphics package was developed for a particular application; namely, the purpose of solving certain geometric problems in the Euclidean plane. These are of practical importance because in many applications real physical objects or data or some variables are commonly mapped onto a set of points in the plane. Thus by simply visualizing them and dynamically varying pictures on the screen, interactive computer graphics provides a useful means of communicating information.

The application program in our Geometric Graphics Package contains not only general-purpose graphics subroutines for creating arbitrary views of two-dimensional objects, but it also has a number of functions for our specific purpose. These functions are to find useful geometric structures arising from the given point set efficiently. They are the Voronoi diagram, the Delaunay triangulation, and the Euclidean Minimal Spanning Tree. We have mentioned their numerous practical applications as well as problems in computational geometry.

The Geometric Graphics Package provides an interface between the application program and graphics hardware. And it defines the interface to be independent of the specific hardware available so that the application program is portable. Also higher level functions that use our Geometric Graphics Package functions as subroutines are easily developed to serve the needs of specific application areas.

Our specific application using the Geometric Graphics Package was to develop a heuristic algorithm for finding the Euclidean Steiner Minimal Tree, known as Steiner's Problem. We have developed an iterative heuristic algorithm using a fixed tension model. At present, the heuristic is in an experimental stage. By visualizing the behavior of the heuristic we were able to draw many inferences needed for refining the heuristic in the future studies and to enhance our understanding of the complexity of the problem. Also the function finding the Euclidean Minimal Spanning Tree in our graphics package played an important role in the heuristic algorithm. Future studies on improving the heuristic will explore the approach utilizing the Voronoi diagram and the Delaunay triangulation, which are available in our Geometric Graphics Package.

BIBLIOGRAPHY

[Aho (74)] Aho, A.V., Hopcraft, J.E., and Ullman, J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

[Bergeron (78)] Bergeron, R.D., Bono, P., and Foley, J.D., "Graphics Programming Using the Core System," Computing Surveys, 10(4), 389-443, 1978.

[Boyce (75)] Boyce, W.M. and Seery, J.B., Steiner 72, An Improved Version of Cockayne and Schiller's Program STEINER for the Minimal Network Problem, Technical Report 35. Computing Science, Bell Laboratories, 1975.

[Chang (72)] Chang, S.K., "The Generation of Minimal Trees with a Steiner Topology," JACM, 19, 699-711, 1972.

[Chang (76)] Chang, S.K., "A Model for Distributed Computer System Design," IEEE SYST. M., 6, 344-359, 1976.

[Chung (78)] Chung, F.R.K. and Hwang, F.K., "A Lower Bound for the Steiner Tree Problem," SIAM J. Appl. Math., 34, 27-36, 1978.

[Courant (41)] Courant, R and Robbins, H, What is Mathematics?, Oxford University Press, 1941.

[Coxeter (61)] Coxeter, H.S.M., Introduction to Geometry, John Wiley, 1961.

[Dasarathy (80)] Dasarathy, B. and White, L.J., "A Maximin Location Problem," Operat. Res., 28, 1385-1401, 1980.

[Du (82)] Du, D.Z., Yae, E.Y., and Hwang, F.K., "A Short Proof of a Result of Pollark on Steiner Minimal Trees," J. Comb. Theory, Series A, 97, 396-400, 1982.

[Foley (82)] Foley, J.D. and Van Dam, A., Fundamentals of Interactive Computer Graphics, Addison-Wesley, 1982.

[Garey (77a)] Garey, M.R., Graham, R.L., and Johnson, D.S., "The Complexity of Computing Steiner Minimal Trees," SIAM J. Appl. Math., 32, 835-859, 1977.

[Garey (77b)] Garey, M.R. and Johnson, D.S., "The Rectilinear Steiner Tree Problem is NP-Complete," SIAM J. Appl. Math., 32, 827-834, 1977.

[Gilbert (68)] Gilbert, E.N. and Pollark, H.O., "Steiner Minimal Trees," SIAM J. Appl. Math., 16, 1-29, 1968.

[Graham (76)] Graham, R.L. and Hwang, F.K., "A Remark on Steiner Minimal Trees," Bull. Inst. Math. Acad. Sinica, 4, 177-182, 1976.

[Harary (71)] Harary, F., Graph Theory, Addison-Wesley, 1971.

[Hwang (76)] Hwang, F.K., "On Steiner Minimal Trees with Rectilinear distance," SIAM J. Appl. Math., 30, 104-114, 1976.

[Hwang (79)] Hwang, F.K., "An O(nlogn) Algorithm for Rectilinear Minimal Spanning Trees," JACM, 26, 177-182, 1979.

[Kuhn (75)] Kuhn, H.W., "Steiner's Problem Revisited," in Studies in Mathematics, Vol. 10, Studies on Optimization, G.B. Dantzig and B.C. Eaves, Eds., Math. Assoc. AM., 1975.

[Loberman (57)] Loberman, H. and Weinberger, A., "Formal Procedures for Connecting Terminals with a Minimum Total Wire Length," JACM, 4, 428-437, 1957.

[Matula (80)] Matula, D.W. and-Sokal, R.R., "Properties of Gabriel Graphs Relavent to Geographic Variation Research and the Clustering of Points in the Plane," Geog. Anal. 12, 205-222, 1980.

[Melzak (61)] Melzak, Z.A., "On the Problem of Steiner," Canad. Math. Bull., 4, 143-148, 1961.

[Miehle (58)] Miehle, W., "Link-Length Minimization in Networks," Operat. Res., 6, 232-43, 1958.

[Osteen (74)] Osteen, R.E. and Lin, P.P., "Picture Skeletons Based on Eccentricities of Points of Minimal Spanning Trees, SIAM J. Comput., 3, 23-40, 1974.

[Shamos (75)] Shamos, M.I. and Hoey, D., Closet-Point Problems, Sixteenth Annual IEEE Symposium on Foundations of Computer Science, 151-162, 1975.

[Shamos (78)] Shamos, M.I., "Computational Geometry," Ph.D thesis, Yale University, 1978.

[Smith (81)] Smith, J.M., Lee, D.T., and Liebman, J.D., "An O(nlogn) Heuristic for Steiner Minimal Tree Problems on the Euclidean Metric," Networks, 11, 23-29, 1981.

[Soukup (75)] Soukup, J., "On Minimum Cost Networks with Nonlinear Costs," SIAM J. Appl. Math., 29, 571-581, 1975.

[Supowit (81)] Supowit, K.J., "Topics in Computational Geometry," Ph. D. Thesis, University of Illinois, 1981.

[Toussaint (80)] Toussaint, G.T., "The Relative Neighborhood Graph of a Finite Planar Set," Patt. Recog., 12, 261-268, 1980.

[Urquhart (82)] Urquhart, R., "Graph Theoretical Clustering Based on Limited Neighbourhood Sets," Patt. Recog. 15, 173-187, 1982.

[Werner (69)] Werner, C., "Networks of Minimum Length," Canad. Geog. 13, 47-69, 1969.

[Zahn (71)] Zahn, C.T., "Graph-Theoretical Methods for Detecting and describing Gestalt Clusters," Patt. Recog., 15, 173-187, 1982.

APPENDIX

IMPLEMENTATION OF THE GEOMETRIC GRAPHICS PACKAGE

The package was written in the C language for the VAX in Computer Science Department at Oregon State University. The graphics hardware used was GIGI made by DEC. The GIGI is a raster-scan display terminal which supports a graphics language called ReGIS. There are a set of ReGIS commands that allows you to control the video monitor screen and draw pictures on the screen with lines, curves, and circles. Also ReGIS provides commands to include text characters in pictures.

Some of the ReGIS commands used in the package are the commands for:

(1) writing text strings with different colors,

(2) drawing lines with different patterns,

(3) drawing circles with shading option on,

(4) reporting screen locations to a host program, and

(5) erasing the entire screen or the part of screen.

Their usages were respectively:

(1) displaying the menus and the screen-feedbacks,

(2) drawing edges in a graph or showing the window,

(3) drawing points or vertices in a graph,

(4) specifying the location of a point to be added or deleted and the location of a new window, and

(5) erasing the screen or the menus or the screen-feedbacks.