A Clustering-Based Algorithm for the Rectilinear Steiner Tree Problem

by

Yusufali Karim Karimjee

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

June, 1994

A CLUSTERING-BASED ALGORITHM FOR THE RECTILINEAR STEINER TREE PROBLEM

BY

YUSUFALI KARIM KARIMJEE

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

JUNE 1994

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN, SAUDI ARABIA

This thesis, written by

Yusufali Karim Karimjee

under the direction of his Thesis Advisor, and approved by his Thesis committee, has been presented to and accepted by the Dean, College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Thesis Committee:

[illegible]

Prof. Meng Er (Chairman)

[illegible]

Dr. Rafik Braham (Member)

Kanaan Faisal

Dr. Kanaan Faisal (Member)

[illegible]

Dr. M. Al-Mulhem
Department Chairman

[illegible]

Dr. Ala H. Al-Rabeh
Dean, College of Graduate Studies

Date: 31·7·94

الدراسات العليا
COLLEGE OF GRADUATE STUDIES

123

Dedicated to

My Parents,

Brothers, Sister

and twin who has returned...

Acknowledgements

The acknowledgement is to King Fahd University of Petroleum and Minerals who have supported the research. Also I would like to express my deepest gratitude to my advisor, Prof. Meng C. Er who has taught me many things while working on this research. My sincere appreciation goes to the committee members, Dr. R. Braham and Dr. K. Faisal, who have been very cooperative and suggestive throughout the research period. Further I thank the department chairman, Dr. M. Al-Mulhem, for availing the use of the facilities in the department. My collegues and friends who have made my stay in this place enjoyable, also deserve a mention here. I also thank Mr. Y. Garout and Mr. A. Lafi who translated the abstract to arabic and Mr. Abdallah Abbas for typing it out. Thanks go to Mr. Asjad Khan for keeping the NeXT lab open during holidays. My profuse appreciation is towards Sayedna Muhammed Burhanuddin who gave the consent to pursue the further degree and my family who have beared my continued absence from home.

Contents

123

123

List of Figures

123

123

List of Tables

123

Abstract

Name: Yusufali Karim Karimjee

Title: A Clustering-Based Algorithm for the Rectilinear Steiner Tree Problem

Major Field: Computer Science

Date of Degree: 1994

The Steiner Tree Problem is the determination of the shortest connecting length between a given set of points and additional points. The Rectilinear Steiner Tree Problem is the same as above but the connecting lines are confined to the horizontal or vertical lines only (L_1 metric). A cluster is a set of points that 'influence' each other locally. This research gives a new algorithm based on a clustering metric that determines much less steiner points locally in a cluster as well as those external to it. The algorithm matches the worst-case time complexity of $O(nlogn)$ of previous authors while giving better average results then theirs. In addition, a Neural Solution to the Steiner problem in Networks, which is a graph-theoretic representation of the problem, is explored and its limitations presented.

Master of Science Degree

King Fahd University of Petroleum and Minerals

Dhahran, Saudi Arabia

June, 1994

123

خلاصة الرسالة

اسم الطالـــب : يوسف علي كريم كريمجي

عنوان الرسالة : خوارزم يعتمد على العناقيد لحل مسألة شجرة شتاينر ذات الخطوط المستقيمة

التخصـــص : علوم الحاسب الآلي والمعلومات

تاريخ الشهادة : يونية ١٩٩٤ م

إن مشكلة شجرة شتاينر هي تحديد أقصر مسافة للربط بين مجموعة نقاط معطاة وأخرى اضافية. أما مشكلة شجرة شتاينر ذات الخطوط المستقيمة فهي شبيهة بسابقتها إلا أن خطوط الربط أفقية أو عمودية . العنقود هو مجموعة من النقاط التى تؤثر على بعض البعض محليًا . هذا البحث يقدم خوارزمًا جديدًا يعتمد على نظام عنقودي يحدد عدد أقل من نقاط شتاينر المحلية والخارجية .

إن دقة الخوارزم فى أسوأ حالاته بدرجة (ن . لو (ن)) من الباحثين السابقين بينما تعطي نتائج أفضل في المعدل . إضافة إلى ذلك ، فإن عصبيًا لمشكلة ستاينر في الشبكات والذي هو تمثيل نظري - رسومي للمشكلة ، قد بَحِث وعرضت مقيداته .

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران ، المملكة العربية السعودية

يونية ١٩٩٤م

123

Chapter 1

Introduction

The Steiner problem has been studied for over three decades and has recieved attention recently due to its appearance in a number of applications. These include planning problems in transportation networks, in communication cable laying, building systems, printed circuit boards, VLSI and in genetics [HR92].

Consider the elementary problem of finding a point p in a triangle with vertices **a, b** and **c** such that it minimizes the sum of its distances $\overline{pa} + \overline{pb} + \overline{pc}$. This point is a unique point and is known as the *steiner point* and named after J. Steiner who first stated it for the three point case in the early eighteenth century [BG89]. Figure 1.1a shows the points and 1.1b, the 'Napolean' (equilateral) triangles that are constructed on the lines connecting these points. These triangles are circumscribed in circles that provide the chords which intersect at the steiner point in Figure 1.2.

If an angle of a triangle is $\geq 2\pi/3$ then p is a vertex, otherwise p lies inside the

123

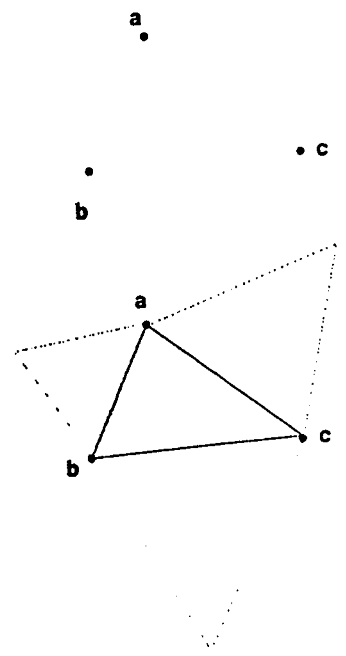


Figure 1.1: (a) 3 points, (b) Equilateral triangles on the connecting lines

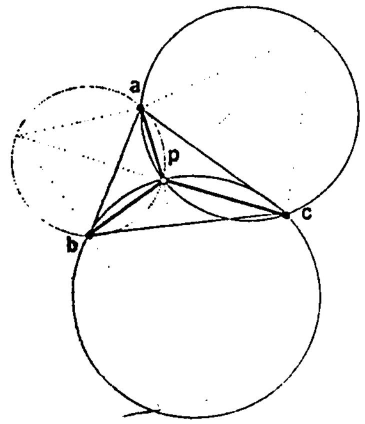


Figure 1.2: Determined Steiner Point

123

triangle with lines connecting to all the other three points that have mutual angles subtending at $2\pi/3$ [Mel61]. The distance to be minimized is Euclidean (L_2 metric):

$$\overline{pa} = ((x_p - x_a)^2 + (y_p - y_a)^2)^{\frac{1}{2}}$$

This can be generalized to n points in a plane, $n \geq 3$ and the problem then becomes:

S_n To find the construction for the shortest tree whose points contain these n points - this is the Euclidean Steiner Tree Problem (EST).

Figure 1.3 shows a construction starting from points A and B to determine steiner points for a set {A, B, C, D, E, F, G}. Z refers to one of these points and S to the additional point. The steiner tree T can be constructed from a finite sequence of such constructions and has the following characteristics [Mel61]:

1. T has the n given points and additional k points $s_1, s_2, \ldots, s_k$.

2. T is not self-intersecting.

3. $w(s_i) = 3$, $1 \leq i \leq k$, where w(r) represents the degree of point r.

4. each s_i, $1 \leq i \leq k$ is the S-point of the triangle $\{S_i\}$.

5. $0 \leq k \leq n - 2$.

123

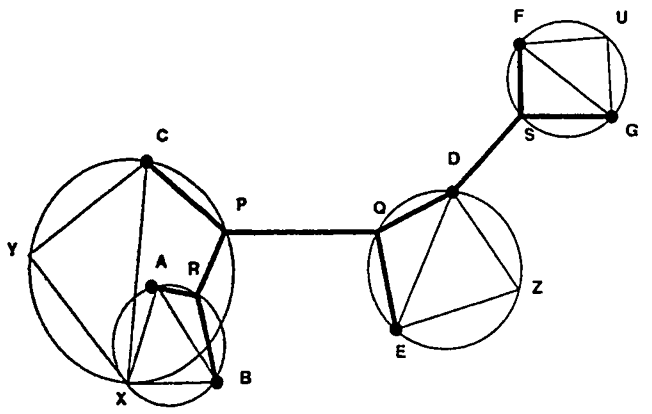


Figure 1.3: A construction for the Euclidean Steiner Problem

123

Figure 1.4a shows an arrangement of points (black), b. the shortest connection possible with no additional points. On introducing a single point (white) there is no change in length as shown in c, and d shows the change in length when two points are inserted. However, if the arrangement is as in e, the minimal length is obtained. On adding an additional point, no change is observed as in f, and in other cases as in g, the length increases. Thus it is shown that while the minimum length is desired, the number of additional points is also of importance.

The steiner problem when confined to a grid where lines can be horizontal or vertical only, is referred to as the Rectilinear Steiner Tree Problem (RST) or the Geometric Steiner Problem. The length is the Manhattan (or the city-block):

$$\overline{pa} = | x_p - x_a | + | y_p - y_a |$$

The analogous conditions for validity are:

1. $w(z) = 3$ or 4
2. $1 \leq w(z_i) \leq 4,\ 1 \leq i \leq n$
3. $0 \leq s \leq n - 2$

This problem is common in printed circuit board technology and in VLSI. In both, the wire length between components to be connected is desired to be minimised. Figure 1.5 shows an example. In another application in cable or conduit laying in building design, where the rectilinear measure is desired for installation

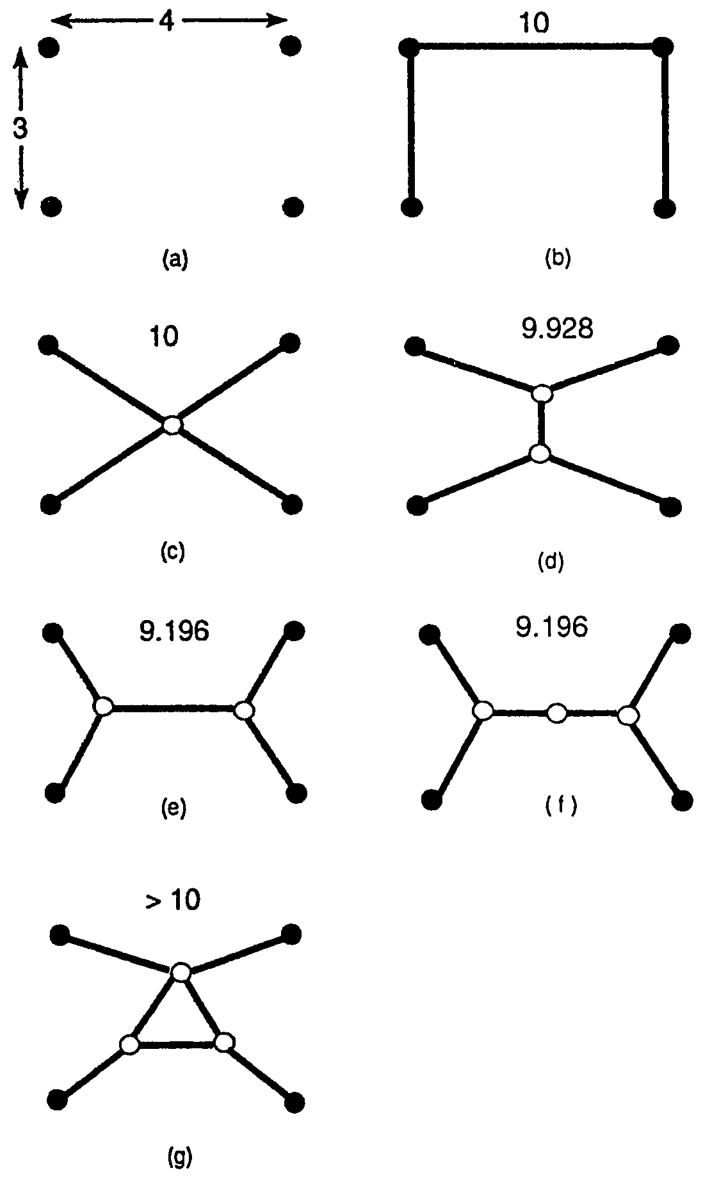


Figure 1.4: Examples (a) The four points (b) Shortest connection with no additional points (c) One steiner point (d) Two steiner points (e) Two steiner points with different topology (f) Three steiner points (g) Three steiner points with different topology

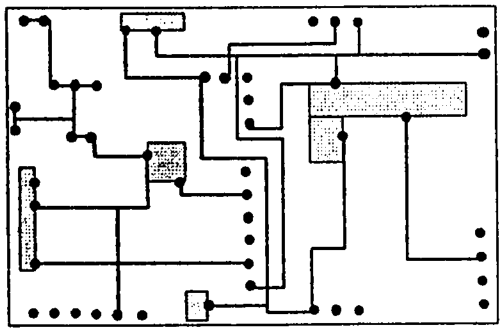


Figure 1.5: An example of a Circuit Board

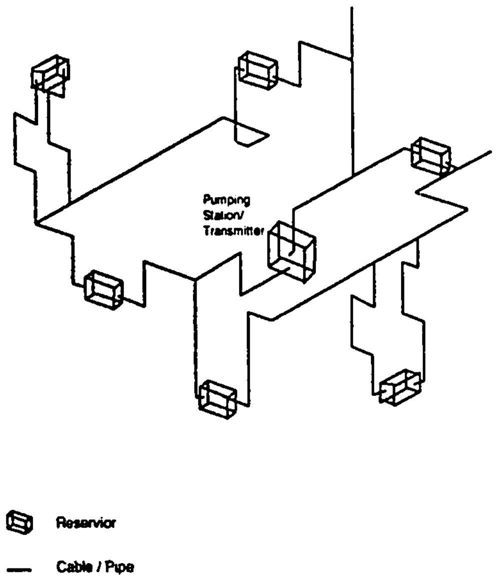


Figure 1.6: An example of cable/pipe system in a building

of these systems, like in figure 1.6, the difference in cost of a spanning tree of the points and that of a steiner cost is substantial.

The above problem has been found to be in the class of NP problems [GJ77] [GGJ77]. Thus it is desirable that suitable heuristics be researched for. It is the purpose of this thesis to determine a fast and good approximate solution to the rectilinear steiner tree problem. The rest of the thesis is organized as follows : Chapter 2 gives the literature survey on the Steiner Problem in Networks, which is a different formulation of the general problem, the specific Rectilinear Steiner Tree Problem and the Neural Network applications to optimization problems to date. Chapter 3 outlines the background of point orientations and clustering in the rectilinear grid. Chapter 4 presents the algorithm and its time complexity. The algorithm is further improved in its time complexity and its details are also illustrated. Chapter 6 gives the proposed neural network representation and presents the results obtained. It also discusses the problems encountered. Chapter 7 concludes the work and indicates future work directions.

Chapter 2

Literature Survey

The literature survey first discusses the relation of the graphical approach to the Rectilinear Steiner Tree Problem, and then in two parts, presents the research to date. In the last part we give the survey for the Neural Network applications to optimization problems which are related to this problem.

Research on the Steiner Tree Problem has essentially followed two tracts, one on the Euclidean Problem and the other on the Rectlinear Problem. Each has special aspects that require a different approach. The common problems are

1. How to determine the steiner points and how many are determined
2. To choose from these, those that contribute to the minimum cost.

In order to facilitate representation, the Graphical Steiner Tree Problem or the

Steiner Problem in Networks (SPN) was formulated. This is the graph-theoretic approach to the steiner problem. Here the points which are given belong to a subgraph of vertices, say Z, and the additional vertices to the set S, both composing a larger set of vertices V in graph G. These are then interconnected by arcs whose length is given by the cost function c : E $\rightarrow$ R+ [Hak71]. The aim is then to minimize the cost of the tree of that graph. More formally, to find:

$$min_{V_\delta \subset V - Z} \; c(T(Z \cup V_\delta))$$

where the minimum is taken over all subsets $V_\delta \subseteq V - Z$ and T(Z), and T(V) represents the respective minimum weight spanning tree of G with respect to Z. This problem is then a network optimization problem, frequently occuring in transport planning as a choice problem [MW84]. Figure 2.1a shows an example and 2.1b its solution.

Both the Euclidean and the Rectilinear problem can be mapped to this representation. Earlier research had been directed at finding a solution using the above representation and had been concerned with (2) of above only. The first part outlines these algorithms. Those that deal with (1) and (2) of the above are discussed in the second part.

Hanaan [Han66], presented in the earliest of the papers on this subject, the

123

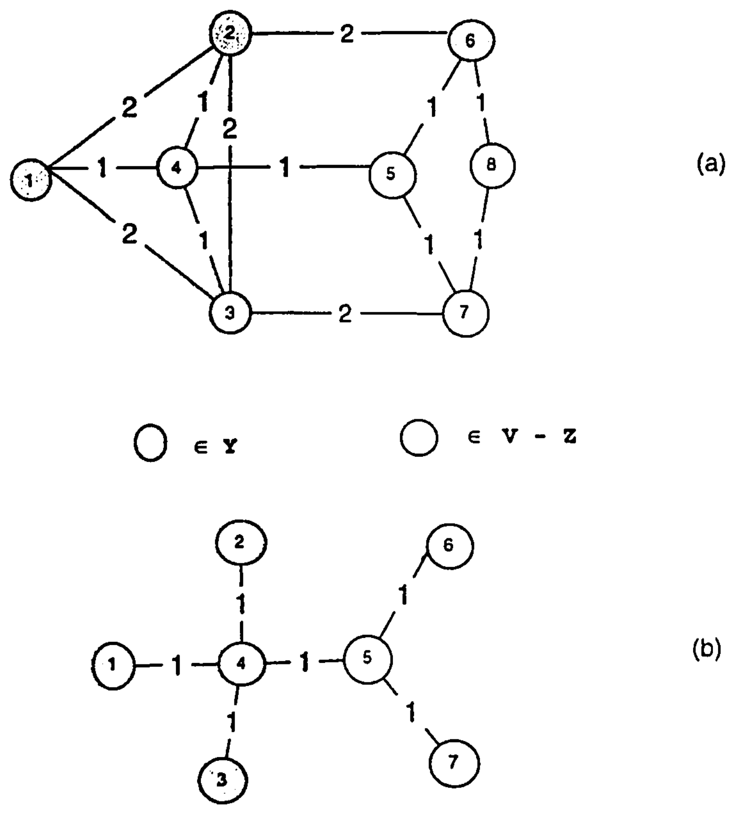


Figure 2.1: (a) A Weighted Graph (b) Its Steiner Tree

method to determine the steiner points in the Rectilinear metric. This is to extend the parallels of the axis on each point in both dimensions and marking the intersection points. Such a vertex set generated for the set S, is of order n^2 and increases the computation complexity considerably. Thus it is desired that fewer vertices be generated. Recent research concentrates on this and we mention this later.

2.1 The Steiner Problem in Networks

We present briefly the conventional exact algorithms, the heuristics, and the recent new methods that have been researched.

2.1.1 Conventional Exact Algorithms

A survey of the SPN appears in Winter [Win87]. An updated survey then given by Richards and Hwang [HR92] discussed the complete steiner problem. We outline the formulation of the exact algorithms which are all exponential in time complexity.

1. **Spanning Tree Enumeration Algorithm (STEA)**

   The orignal simple formulation was by Hakimi [Hak71] which determines the minimum tree by enumerating MST's of G induced by subsets of W of V such that $Z \subseteq W \subseteq V$.

   In another formulation by Balakrishnan and Patel [BP87], all S-vertices are linked by zero cost edges to an additional vertex 0, and any Z-vertex is labelled as 1 and also connected by the zero-cost edge to vertex 0. The minimum cost tree is then that formed which includes vertex 1 and the Z-vertices and where all S-vertices in that solution (adjacent to vertex 0) having degree 1. The determination of the tree is through enumeration of spanning trees of this G containing edge 0-1 in order of increasing cost with the first tree as the solution having the above condition.

2. **Topology Enumeration Algorithm (TEA)**

   Hakimi [Hak71] defines another method with the following definition of a psuedo-adjacent edge i-j as :

   (a) when deg(i) = deg(j) = 1 and path p_{ij} contains S-vertices only then only one of them has degree greater than 2, while the remaining must have degree 2.

   (b) either deg(i) = 1 or deg(j) = 1 and the path p_{ij} contains S-vertices with degree 2.

   The algorithm recursively computes the minimum cost tree by determining the psuedo-adjacency of i-j for all vertices in G.

3. **Dynamic Programming (DP)**

   Dreyfus and Wagner [DW72] present the principle of optimality for the problem and give a dynamic programming formulation of the problem. Y is the non-empty subset of Z, i $\in$ V-Y and $T_{Y \cup i}$ is the Steiner Minimal Tree (SMT) for Y ∪ i. $T_i(Y)$ is the union of two SMTs, one spanning $X \cup i$ and the other Y-X∪ i, where $\Phi \subset X \subset Y$. The tree is obtained by minimizing over all choices of X:

$$c(T_i(Y)) = min_{\Phi \subset X \subset Y}\{c(T_{X \cup i}), + c(T_{(Y-X) \cup i})\}$$

where $T_i(Y)$ is the union of $T_{X\cup i}$ and $T_{(Y-X)\cup i}$ for which $c(T_i(Y))$ was attained. With $i \in V$, $T_i(Y)$ can be determined for every two-vertex subset Y of Z and every $i \notin Y$. The possible configuration for $c(T_{Y \cup i})$ are :

(i) deg(i) > 1 then $T_{Y\cup i} = T_i(Y)$

(ii) deg(i) = 1. $T_{Y\cup i}$ must be a union of a shortest path from i to some $k \in S$ of degree at least 3 with $T_k(Y)$ or a union of shortest path from i to some $k \in Y$ with T_Y and the following cost minimization:

$$c(T_{Y\cup i}) = min\{c(T_i(Y)), min_{k\in Y}\{d_{i,k} + c(T_k(Y))\}\}$$

4. **Branch and Bound (BB)**

The formulation was developed by Shore et al [SFG82] and is presented here. Each set F_i, a feasible solution, has an included edge set IN_i and an excluded set OUT_i, and when F_i remains unfathomed, it is split into F_j and F_k with $IN_j = IN_i \cup \{e\}, OUT_j = OUT_i, IN_k = IN_i, OUT_k = OUT_i \cup \{e\}$ for some edge $e \in E - (IN_i \cup OUT_i)$. The subset F_j is examined initially and when found or determined that it is not in F_j, F_k is checked, and then backtracking to F_i. The determination is through the bound checking, where an upper bound is

123

the sum of edges in IN_i and the tree spanning Z_i in G_i, and the lower bound is the minimum cost edge $k \in Z_i$ incident to k in G_i (∞ if no edge exists). The minimum is over all T_i found in F_i together with edges in IN_i during this search which gives the solution.

5. **Integer Programming (LP)**

Here the problem is stated as a 0-1 Linear Programming problem and cast into various forms as under:

(a) **Set Covering Algorithm (SCA)**

A partition W,$\overline{W}$ of V such that $Z \cap W \neq \Phi$ and $Z \cap \overline{W} \neq \Phi$ is a cut-set and are enumerated $C_1, \ldots C_q$. Edges of G $e_1, \ldots e_m$ with the matrix A $=(a_{ij})$ are defined as :

$$a_{ij} = \begin{cases} 1 & if\ e_j \in C_i \\ 0 & otherwise \end{cases}$$

for i = $1, 2, \ldots, q$ and j = $1.2. \ldots, m$. The SCA is defined as :

$$min \sum_{j=1}^{m} c_j x_j$$

subject to

$$\sum_{j=1}^{m} a_{ij} x_j \geq 1, \quad i = 1.2. \ldots . q$$

$$x_j \in \{0.1\}. \quad j = 1.2. \ldots . m$$

(b) **Lagrangean Relaxation Algorithm (LRA)** The problem is formulated for $Z_i = Z$ and $G_i = G$. Let $Z_i = Z - \{1\}$; x_{ij} be a binary variable

123

for edge (i,j) $\in$ E; for i,j,k, (i,j) $\in E, k \in Z_i$; y_{ij}^k denotes the amount of commodity k on edge (i,j) in direction i to j. The problem is:

$$min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

subject to

$$\begin{array}{ll}
x_{ij} \geq y_{ij}^k + y_{ji}^k & \forall (i,j) \in E, \forall k \in Z_1 \\
\sum_{(i,j)\in E} y_{1j}^k \geq 1 & \forall k \in Z_1 \\
\sum_{(h,k)\in E} y_{hk}^k \geq 1 & \forall k \in Z_1 \\
\sum_{(i,j)\in E} y_{ij}^k - \sum_{(h,i)\in E} y_{hi}^k \geq 0 & \forall k \in Z_1, \forall i \in V - \{i,k\} \\
p-1 \leq \sum_{(i,j)\in E} x_{ij} \leq n-1 & \\
x_{ij} \in \{0,1\} & \forall (i,j) \in E \\
y_{ij}^k \geq 0 & \forall i,j : (i,j) \in E, \forall k \in z_i
\end{array}$$

The first inequality is relaxed in the Lagrangean fashion with multipliers u_{ijk} ($u_{ijk} \geq 0 \; \forall (i,j) \in E, k \in Z_1$) :

$$min \sum_{(i,j)\in E} (c_{ij} - \sum_{k\in Z_1} u_{ijk})x_{ij} + \sum_{k\in Z_i} \sum_{(i,j)\in E} u_{ijk}(y_{ij}^k + y_{ji}^k)$$

which forms the lower bound on the minimization and is solved for the optimal solution.

2.1.2 Heuristics

1. **Minimum Path Heuristic (MPH)**

Initially a tree T_1 of G having only a single vertex i of Z is made. $k = 1$ and

$Z_k = i$. Additional vertices $i \in Z - Z_k$ are determined by that being closest to T_k which forms T_{k+i}. This is repeated until $k = p$ and T_p is the solution. Time takes $O(pn^2)$ and the solution is bounded by :

$$c(T_p)/c(T_Z) \leq 2 - \frac{2}{p}$$

2. **Distance Network Heuristic (DNH)**

   First a MST is determined for G_Z. This is improved upon by replacing each edge in T by the minimum corresponding edge in G (through V-Z). A MST of T_Z is then determined for G_Z. Thereafter, deletion of those V-Z vertices having a degree of 1. The resulting tree is the solution. The complexity and bound are exactly similar to the previous one.

3. **Average Distance Heuristic (ADH)**

   A list L = $\{T_1, T_2, \ldots, T_k\}$ is composed, initially of Z-vertices. Using a function $f : V \rightarrow R$:

$$f(i) = \begin{cases} d(i, T_2) & if\ i \in T_1 \\ min_{2 \leq r \leq k}\{\sum_{j=1}^{r} d(i, T_j)/(r-1)\} & otherwise \end{cases}$$

   The vertices are labelled in non-decreasing order of their distance from i and $d(i, T_1) \leq d(i, T_2) \leq \ldots \leq d(i, T_k)$. Choosing $p \in V$ and $f(p) \leq f(i) \forall i \in V$ with $d(p, T_1) \leq \ldots \leq d(p, T_k)$, then while $p \in T_1$ join T_1 and T_2 by minimum edge in G. Hence two subtrees of L are selected and joined by the minimum cost path, and this is repeated until L just remains with a single tree spanning all Z-vertices. The complexity of this is $O(n^3)$ with no bound given.

4. **Other heuristics related to the LP problem**

   Here various improvements to the LP problem are suggested through removal of redundant elements and inclusion of new lower bounds [Win87].

2.1.3 New Methods

Here we discuss briefly the new approaches in combinatorial optimization and how they have been applied to the steiner problem.

1. **Simulated Annealing**

   Using the approach from statistical mechanics, a simulated annealing schedule is followed to obtain a better solution from an initial solution. In [Dow91], this procedure is applied to SPN where the following are defined:

   (a) *Neighborhood Structure* is the set of steiner trees in which all steiner vertices have degree 2 or more.

   (b) *Exchange Heuristic* is the exchange of all paths with *key* paths that are the paths existing in the optimal solution. The process is either 1-opt, 2-opt or generalized to k-opt where the constants represent the number of paths considered at a time. In [Dow91], only the first two are considered experimentally.

(c) *Objective Function* is the cost function of that graph.

The exchange for the path is accepted if the cost is reduced depending on the energy distribution defined on the cost function at that temperature. The sequence is iterated until no further change is observed in the cost reduction. The authors state that the algorithm produces results varying around 5% from the optimum.

In another application [OG91], two simulated annealing algorithms are compared for performance when applied to the directed steiner tree. The first is the static schedule SCA (Static Cooling Algorithm) which is performed depending on the neighborhood size which is fixed at the begining of the algorithm. An increase in cost is accepted depending on the ratio of the accepted moves over the generated moves. The temperature decrement is fixed and determines the length of chains to be evaluated. The iterations terminate when the ratio is less than a predetermined constant (after a number of trials). The second algorithm is the Dynamic Cooling Algorithm (DCA) and uses a running ratio which is the running average of the decrease of the cost function. Here when the cost changes more at a certain temperature, more iterations are performed.

The terminating stage is when no further change is possible. Both methods, as stated by the authors, give good results at around 3% of optimum and at only a few instances does the SCA give better reults than DCA.

2. **Genetic Algorithm**

   Based on the dynamics of natural population genetics, this approach is an adaptive search procedure and was first applied to the Euclidean Steiner Problem by Hesser et. al.[HMS89]. The SPN problem appears in [KRSS93]. Here the chromosome is encoded with the vertices of the graph and with the initial population as the vertices of an MST. With the crossover parameter set at 0.5, the selection based on the roulette and the mutation set at 0.02, the new generations are produced that replaces the old. This gives better solution each time and the terminating condition was set at an upper bound on the number of iterations or the optimal solution, whichever was earlier. Results give 70% of the time optimal solutions and around 7% off the optimal in other cases.

2.2 Rectilinear Steiner Tree Problem

We outline here the literature survey on the Rectilinear Steiner problem and present their various limitations where appropriate. For convenience, we represent the set of points given to be Z with $\mid Z \mid = n$, and the additional (steiner) points S which are to be determined. A *topology* constitutes an arrangement of points on the grid.

MST refers to the Minimum Spanning Tree, RST refers to the Rectilinear Steiner Tree.

In Hanaan [Han66], besides his dimension reduction method, he presented a way to determine the steiner point for three points, namely that the coordinates of the steiner point, if present, is the median of the ordinates of each dimension (in plane). He also introduced the concept of the *enclosing rectangle* and derived the upper bound for four and five points, which is the sum of the length and twice the width of the enclosing rectangle.

Hwang [Hwa76] proves the ratio $L_s/L_m \geq \frac{2}{3}$, where L_s is the length of the steiner tree and L_m is the length of the minimum rectilinear spanning tree. This was in response to the Gilbert-Polak Conjecture in the Euclidean Plane [GP68]. He used the constraints of orientation of the points and the placement of this chain of points in the enclosing rectangle. Another proof appears in [Sal92] but much simpler. Here, the characterization of full topologies which have all $n-2$ steiner points, is considered and shown to reach the bound of $\frac{2}{3}$. A generalization of the bound to any dimension appeared separately in [Syn91].

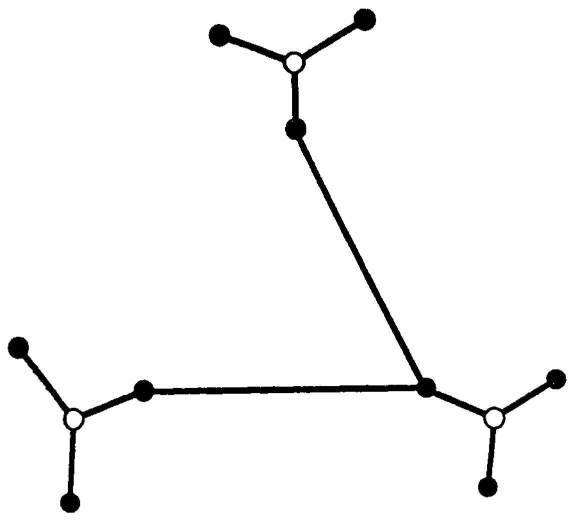
A proof of the NP-completeness of the rectilinear steiner tree problem appears in [GJ77]. The authors map the decision problem into the 3-XAT (exact 3-node cover

123

problem) which is also NP-complete. Thus solutions to the problem will have to be heuristic in nature and research has been directed towards obtaining good heuristics.

The earliest heuristic developed was by Lee, Bose and Hwang [LBH76], which has complexity $O(n^2)$. This algorithm, for the single net case, computes the rectilinear steiner tree for 3 points and then inserts each 3-point topology by the virtue of being the shortest and being nearest to the current tree. The algorithm is greedy in nature and Figure 2.2 shows where this algorithm omits the crucial steiner point for the optimal tree.

Most other algorithms developed somehow use the rectilinear minimum spanning tree algorithm which is either the Prim's or Kruskal's MST algorithm [HS78] with the rectilinear metric used. Prim's algorithm has complexity $O(n^2)$ and Kruskal's has $O(e\, log e)$ where e is the number of edges. Hwang [Hwa79a] gave an $O(n\, log n)$ algorithm to generate the rectilinear minimum spanning tree based on the derivation of the Voronoi Diagram in the rectilinear grid. Later in [Hwa79b], he gave the $O(n\, log n)$ algorithm to determine the rectilinear steiner tree. He used the strategy of Lee, Bose & Hwang [LBH76] to improve the RMST by considering only the three points lying on the edge of the RMST and determining the shortest edge of the three points that could reduce the RMST length. We discuss later in chapter 4 about their strategy.



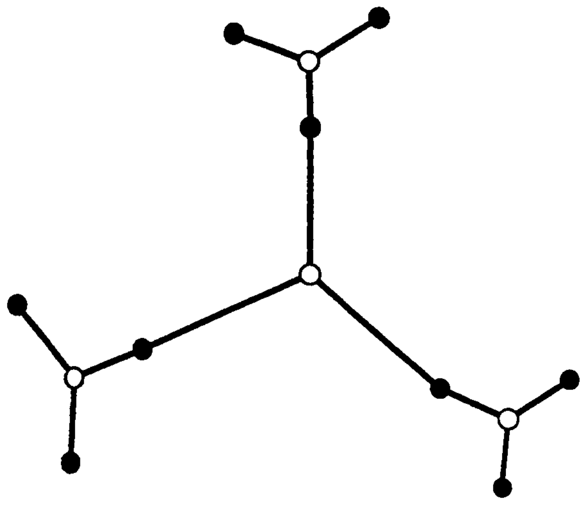


Figure 2.2: (a) Minimum Spanning Tree (b) Steiner Minimal Tree

Smith, Lee & Liebman [SLL80] gave a different algorithm from Hwang [Hwa79b] with a worst cast time complexity of $O(n\,logn)$. They used the dual of the Voronoi Diagram, the Delaunay Triangulation along which the RMST was constructed. They then determined the steiner points on edges of the RMST by considering the nearest traingle or nearest two triangles. We discuss this later too.

Komlos and Shing [KS85] give a probabilistic algorithm such that with probability 1 - $o(1)$, $RST \geq \frac{\sqrt{n}}{5}$. They partition the points into equal point quadrants and apply the Dreyfus and Wagner [DW72] approach for the steiner point determination. Time complexity is $O(t\,3^t)$ where t is the number of points in a quadrant .

Hwang, Vijayanan and Wong [HVW90] give two algorithms. They define a separable MST as one having a non-overlaping staircase layout of the edges. After computing this in $O(n^2)$ from the given set of points, the first algorithm determines a L-shaped arrangement for optimal steiner tree if the separable MST exists. The second algorithm places Z-shaped edges in polynomial time to obtain an optimal solution from the separable MST with worst case time complexity of $O(n \times l^6_{max})$. where l_{max} is the maximum length of the edge computed on the length of the grid.

The paper of Kahng and Robbins [KR92a] gives an on-line algorithm which introduces single steiner points where the reduction of the MST length is greatest. The complexity is $O(n^3)$. The authors also present the performance bound of the algorithm of $\leq \frac{4}{3}$ where a $\frac{3}{2}$ is expected in the optimal case.

An MST-based algorithm of Beasely [Bea92]. considers four adjacent vertices and computes the optimal steiner tree for these points. These sets of 4 points are chosen along the MST and their steiner points computed. Only those points that reduce the overall cost of the tree after previous addition of the steiner point are included. This process is repeated until there are no more improvements. It is clearly seen that by computing along the MST, the crucial steiner points between topologies as stated in the earlier paragraph is omitted. However, they mention an improvement of 10 - 12% in $O(n^2logn)$ worst case time and has also been run for 10000 vertices.

An enlightening paper by Kahng and Robbins [KR92b] shows the topology where the upper bound attainable by all MST-based approaches is close to $\frac{3}{2}$ for large n. They present an example, Figure 2.3a, where its best configuration that could be obtained for all MST-based algorithms is as Figure 2.3b. Its optimal solution is given as Figure 2.3c. They also generalize this to higher dimensions where the cost of optimal steiner tree is at most (2 d - 1) / d. They give a counter example to the claim of optimality for separable MST of [HVW90], given here as Figure 2.4. Starting out with a MST is a reason for non-optimality for the algorithms in [LBH76]

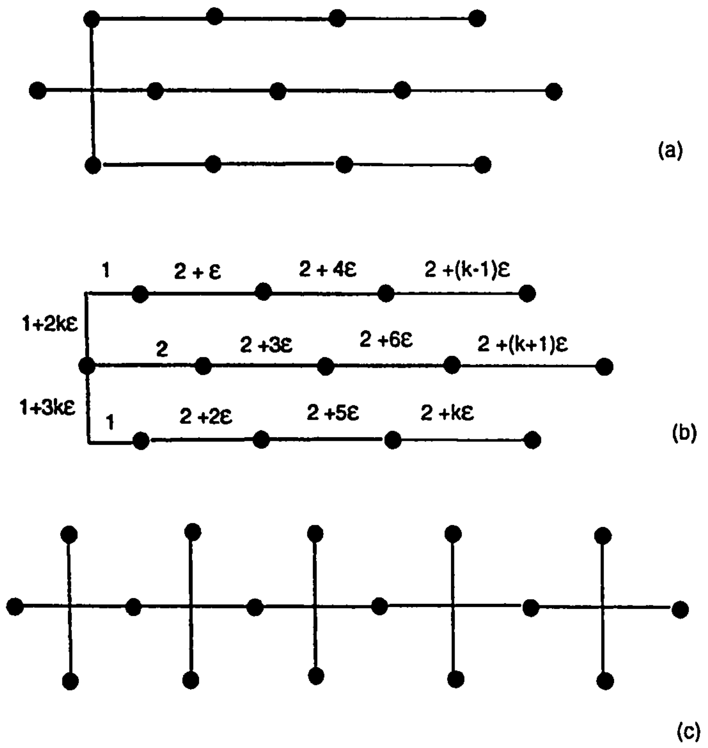


Figure 2.3: (a) Best Obtainable Solution (b) Greedy Solution (c) Optimal Solution

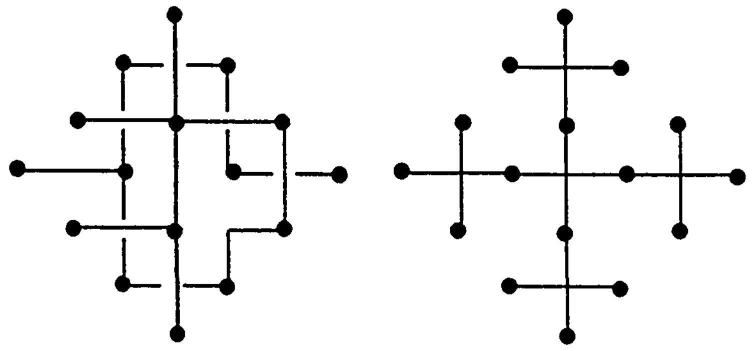


Figure 2.4: (a) A Non-Separable MST (b) Optimal Tree

and [Hwa79b].

The algorithm by Richards [Ric89] gives a new method in the plane-sweep sense. By starting from a furthest point at the bottom, up, left or right position in the quadrant, steiner trees are computed and placed near to each other and this progresses towards the other end. The complexity is reduced by discarding earlier points by virtue of they being far or 'covered' by the former points. This results in a $O(nlogn)$ time algorithm with much less average running time. It is seen that the algorithm only uses the local constraints and thus may miss the global optimum solution. It gives around 5% improvement.

The work by Sarrafzadeh and Wong [SW92] considers the case for λ-geometry where orientations with angles of $i\pi/\lambda$ are only allowed. They give a divide and conquer approach which is exponential in the subproblem since they choose an arbitrary size of points (k = 2, 4, 8) for computing the exact steiner tree in the rectilinear geometry ($\lambda = 2$). They generalize this to any geometry and present an upper bound on its length to the optimal. They give their results for the $\lambda = 2$ geometry of a 10.7% improvement.

Other related work which uses the rectilinear problem as a subproblem is in Performance Oriented Rectilinear Steiner Problem in circuit board technology. This considers in addition to minimizing the length, the timing delay in the point connections, which are input separately and do not depend on the length but type of connections. Lim, Cheng and Wu [LCW93] provide an algorithm that grows the steiner tree an edge at a time along the Hanaan grid depending on a experimentally determined score. Other work by Hong et. al [ea93] considers source-to-sink connections with timing delay in addition to the wire length minimization. They minimize length by a force directed approach along the axes that determines point of growth. Both the papers used small sets of points.

2.3 Neural Network Applications

We give here the literature survey of Neural Network applications to Optimization which are related to this problem.

Hopfield and Tank [HT85] developed the first neural network model for optimization and applied it to the famous Travelling Salesman Problem(TSP). Thereafter, an avalanche of neural network applications have followed.

Looi [Loo92] presents a survey on various other applications of neural networks to NP problems. He also gives the update on the various modified forms of the Hopfield model which have been applied to the TSP and other neural models as well.

Taligarni and Page [TP87] gave applications to constraint satisfaction problems of the Non-attacking Queens and the Graph Isomorphism problem, the later requiring some pre-processing. Ramajuan and Saddayapan [RS88] give a strategy to map optimization problems to neural networks and formulate a number of graph problems that could be solved. Taligarni, Christ and Page [TCP91] give an application of a three dimensional neural network to the Weapon-Target Assignment Problem. A host of other applications appear in Takefuji's book [Tak92], where he and his collegues have applied the Hopfield model, and variations of it using the McCulloch-Pitts neurons and the McCulloch-Pitts hysterisis neurons. Most of these problems are direct applications of the Hopfield network, where the constraints are directly representable in the synaptic connections as modelled by the dynamic equations and no pre-processing is required. This is also true for the graph problems cited earlier and the Maximum Clique problem as given by Jagorta [Jag92] and [Lin93]. An identical application and developed almost at the same time is of Burke [Bur92] that solves for the Maximum Independent Set, which is the dual problem of Maximum Clique in graph theory.

Particularly of interest are neural network applications to the problems in VLSI. While closely related to the rectilinear steiner problem, Circuit Partitioning seeks to place circuit modules/components such that the external wires connecting these have minimal length. Yih and Mazumder [YM89] consider the connections for two components where the partitioning is of nearly equal size. The net produces suitable results. Shen, Gan and Yao [SGY92] give a Self-Organizing Feature Map using fuzzy values for partitioning. This approach gives better results.

Module Orientation concerns the placing of components on the surface of the chip including any rotations that could be made such that the wire length between the components is minimised. Hadas and Lin [LL89] present the orientation and rotation problem separately and get good results. Rao and Patnik [RP92] describe a similar approach but consider also the component (cell) overlap and area of component.

Channel Routing considers the interconnections of points (terminals) on opposite sides of rectangular channels where the wire length has to be minimised. This is a 2-point case and Shih and Feng [SF91] proposed a method which use the *conflict graph* (consisting of horizontal and vertical segments which are permitted) for constraint mapping. With a similar idea, Shih, Chang and Feng [SCF91] present their version considering in addition the wire overlap that could result between the rows/columns of components on the grid.

The only work which takes the global wire length into consideration is this problem and the first approach has been given by Kahng [Kah91]. He uses an elastic-net

123

method, where the points on the border of the enclosing rectangle are gradually deleted in a parallel fashion until the bare skeleton remains. This is the resultant steiner tree and the procedure is analogous to the bubble shrinking principle. He reports around 10% improvement with no details on how his dynamical equations are modelled and his termination procedure, but mentions a time of around 7000 iterations by his simulation.

Chapter 3

The Clustering-Based Algorithm

It was mentioned earlier that determination of steiner points constitute a major part of computation of the algorithms researched so far. We consider the case of minimizing the generation of steiner points. Some definitions are presented first. All distance measures are in the L_1 metric.

3.1 Point Orientations

Definition:

The *Minimum Enclosing Rectangle* (MER) with respect to a point is a rectangle composed from a point and its two nearest neighbours. A steiner point, if it exists, in a MER composed from $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ is the point having the coordinates median (x_1, x_2, x_3), median (y_1, y_2, y_3).

Definition:

A degenerate MER (Bounding Box) is a rectangle formed with 2 end-points of a line.

See Figure 3.1 for illustrations.

Lemma 1. *In a MER, at least one point must be at the corner.*

Proof Only two points are necessary to define a rectangle, and the nearest two points located at its opposite corner define the minimum rectangle so formed. In case there are three points, two points define the two borders of the rectangle and the third must define the other two sides which is possible only by being at the other corner of that MER. Four orthagonal points define a MER by each being at its corner.

Lemma 2. *Any other point, say x, on the MER (p, q, r) with respect to a point p, must lie on the opposite corner to this point defined by the region outspan as* $d(p, x) > max\{d(p, q), d(p, r)\}$.

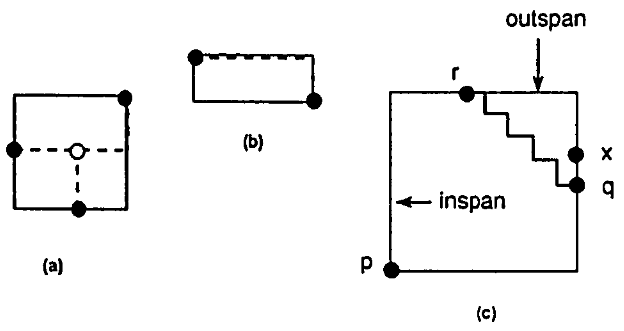


Figure 3.1: (a) MER, (b) Degenerate MER, (c) outspan of MER

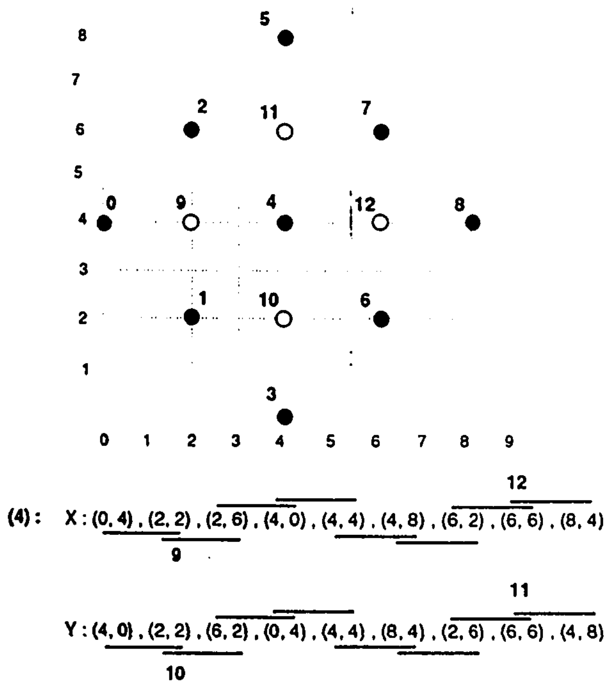

0 1 2 3 4 5 6 7 8 9 10 11 12
(4): X : (0, 4) , (2, 2) , (2, 6) , (4, 0) , (4, 4) , (4, 8) , (6, 2) , (6, 6) , (8, 4)
Y : (4, 0) , (2, 2) , (6, 2) , (0, 4) , (4, 4) , (8, 4) , (2, 6) , (6, 6) , (4, 8)

Figure 3.2: Equidistant points in a cluster

Proof If a fourth point is introduced in the interior of the MER, it is always possible to shrink the borders of the MER by leaving out a non-corner point of the previously defining border to form a new border defining a smaller MER. If the point is placed on any border of the MER, the same applies, except when x is placed in the region $d(p,x) > max\{d(p,q), d(p,r)\}$ where it does not form a neighbour to the point p. Figure 3.1c shows the outspan so defined.

Definition: A Cluster is a collection of MERs, a degenerate MER or a set of collinear lines.

We first consider the case of computing steiner points in the cluster and later give details for this cluster determination. We have the following cases for the construction of a MER of a point

1. A single nearest point and one or more nearer equidistant points

2. Two or more near equidistant points

In both cases, many MERs will be generated. For the case (1), the MER is constructed by having this point with the nearest neighbour, as a pair, and the third point being one of the equidistant points. Additional MERs are constructed using the same pair and each of the equidistant points in turn. for all such points determined. For the case (2), we have to choose a pair of points from the equidistant

123

points, with this point such that a MER is formed. An exhaustive enumeration would lead to a quadratic complexity in the size of the cluster. We consider choosing a pair in succession after the equidistant points are sorted in the X-ordinate as major and then Y-ordinate as minor, and separately in Y-ordinate major and then X-ordinate minor. We present the scenario for the 8 equidistant points in a cluster in Figure 3.2.

Lemma 3. *Given a set of equidistant points. in a cluster. from a point in question, all steiner points can be located with respect to this point by sorting the equidistant points in the X-ordinate as major. then Y-ordinate as minor. and separately sorting in Y-ordinate as major and X-ordinate as minor and constructing the MERs with this point and two points in succession from this two lists.*

Proof By sorting in the X-ordinate as major and Y-ordinate as minor. points close to each other are aligned on the X-axis such that by choosing the pair of points in succession, the MER so formed locates the steiner point. The same is true for the points sorted in the Y-ordinate as major and then X-ordinate as minor which would allign points on the Y-axis.

We refer to the given points as Z-points, black in the figures. and the additional points as S-points, white in the figures.

123

It is seen that when the MERs are constructed with respect to each point, they would overlap in either one Z-point, or two Z-points. We consider each in turn. For the one Z-point overlap, simple cases are shown in Figure 3.3a and b that compose MERs joining at either the corner point or at the boundary. For the two Z-points case, examples in Figures 3.3c and d, 3.4 and 3.5 - 3.6 show the various orientations of the MERs. In 3.3c a unique steiner point is created, while in 3.3d, an additional point is found. In Figure 3.4, when MERs are constructed from equidistant points, four steiner points are generated. However, only two are actually used in the construction of the minimum length tree as shown in the figures at the bottom, and these can be any two along the vertical or horizontal. In Figure 3.5, MERs constructed with respect to each point do not generate all the necessary steiner points. Further MER generation, this time with the two steiner points and a Z-point composing the inner enclosing rectangle, generates the additional steiner point.

We examine how this additional steiner points can be found. It is seen that when a MST is composed after the construction of the MERs, for the case (1) from above, Figures 3.3 a and b, and the case of Figure 3.3 c, complete solutions are obtained i.e steiner points will have degree 3. In Figure 3.3d, only one of the steiner points as selected by the MST algorithm will have degree 3, being chosen arbitrarily, and the other will have degree 2. In case of Figure 3.4, two steiner points will be required, and will be definitely chosen by the MST algorithm, these being on the same side of the smaller enclosing rectangle so formed. In Figure 3.5, two cases result, and we

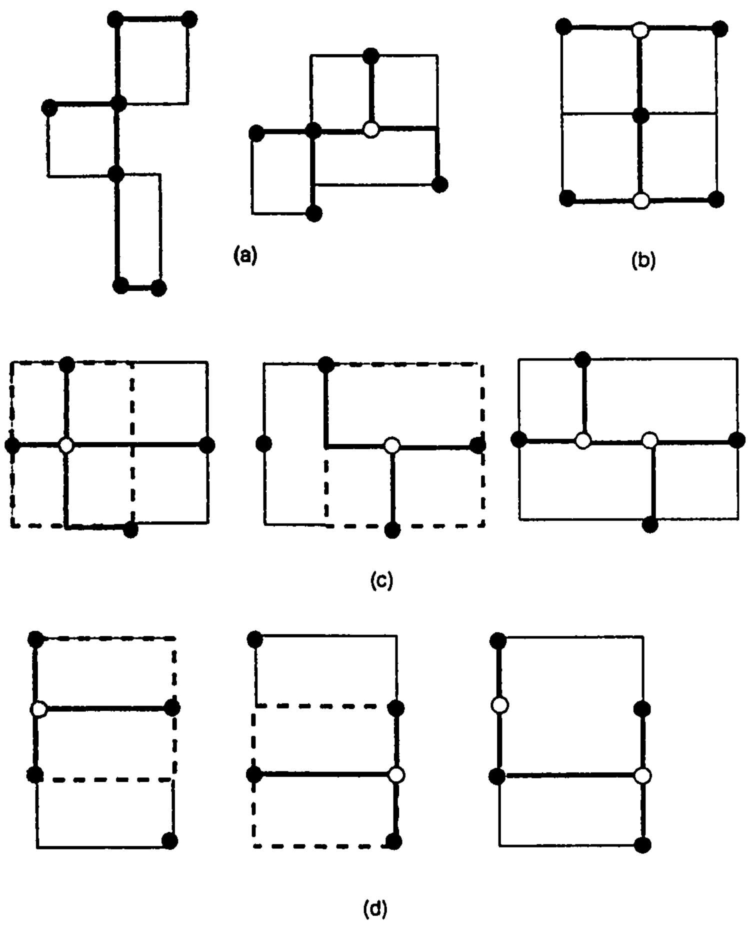


Figure 3.3: Overlapping MERs (a) On Z-points (b) With Z-points on boundary (c) Unique solution by MER, (d) Steiner points of degree 3 and 2

123

show these as Figures 3.6d and e respectively.

Our procedure for the additional steiner points determination (MST improvement) checks for degree 2 vertices of Z that have edges to steiner points only or the points connecting to the degree 3 steiner vertices and determines if a smaller MER can be formed that generates a new steiner point. We show this MER so formed respectively for Figures 3.5 a-d. For (a), point 2 has degree 2 and connects to 5 and 6, both steiner vertices. The MER formed from 2, 6 and 5 generates the additional steiner point, Figure 3.6 a. For (b), 5 and 6 are both degree 3 steiner points and taking (2, 6, 1) and (5. 3; 4) respectively for the MERs generates two additional steiner points, Figure 3.6 b and Figure 3.6 c . Here, once again, only two steiner points at one side of the smaller rectangle will be selected by the MST algorithm, either as Figure 3.6d or e. An improvement can thus be obtained by enforcing such a procedure in every cluster.

Theorem 1. *The construction of the MERs with repect to every point in a cluster and their improvements determine all the locations of the steiner points.*

Proof We prove by induction. For the three-point case the steiner point is located from the definition of the MER. If we consider an arrangement of points in a cluster and add one more point, a pair or a MER, we show that the necessary steiner points are easily located and this would then work for any number of points inserted. For

123

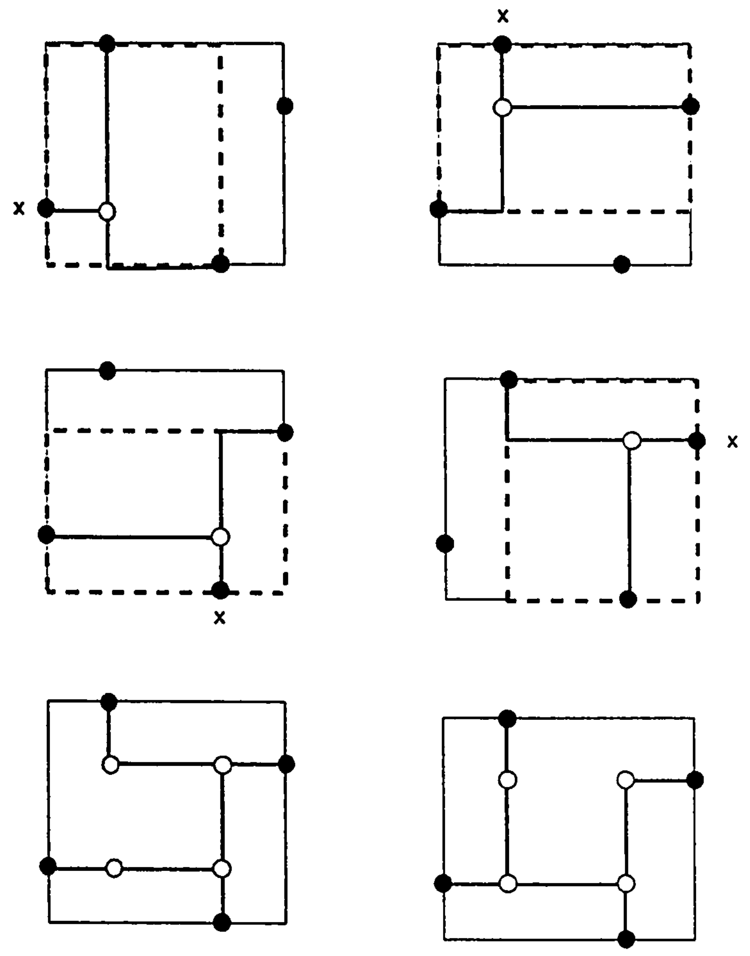


Figure 3.4: Two steiner points of degree 3

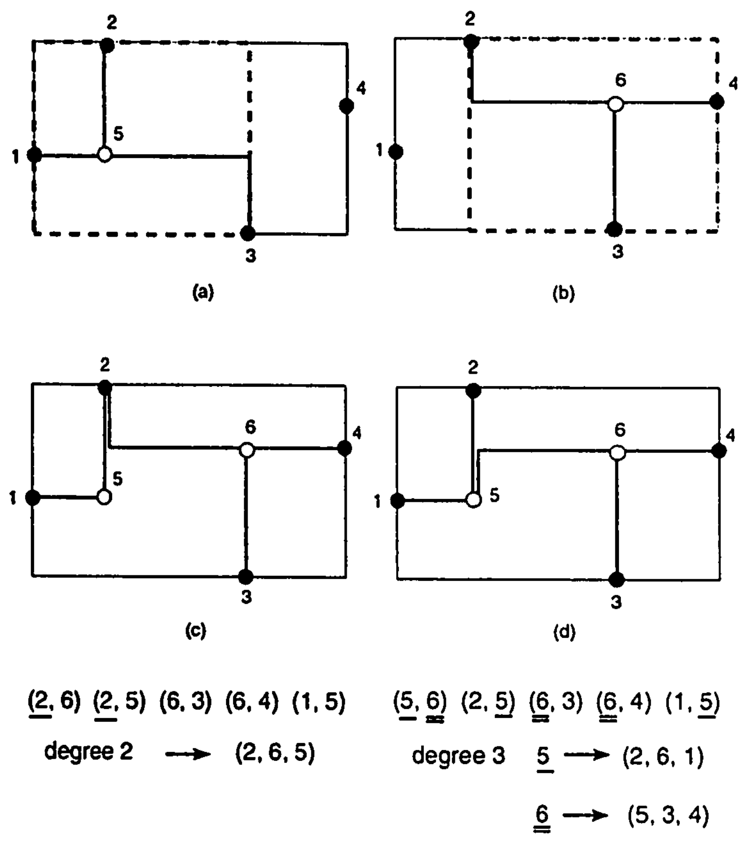


Figure 3.5: MST improvement

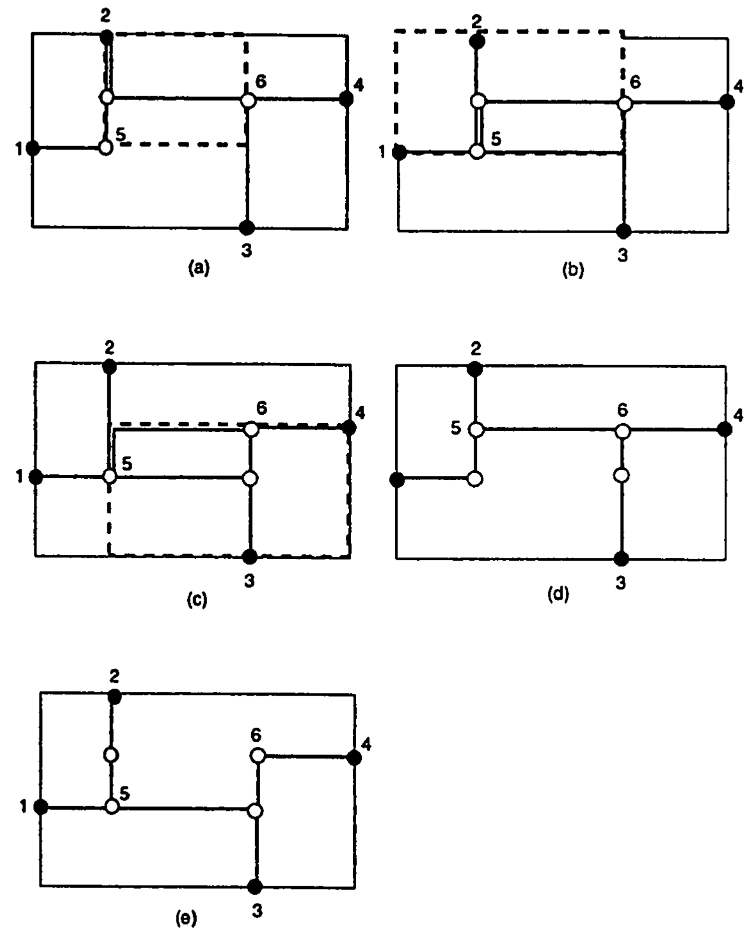


Figure 3.6: Result of the MST improvement

the single point case, this can be done by inserting a point in the interior of an existing MER, on the boundary, or outside. In the first case a smaller MER is formed excluding the further point, and another MER forms to incorporate this excluded point (which may or maynot generate a steiner point). In the second case, if the point is placed on the outspan, by lemma 2, no change takes place, but if placed ouside it (nearer to the corner point) a smaller MER results, with the excluded point forming its own MER. In the last case, a MER may be formed at the cluster boundary or just an edge added with no steiner points determined.

Now we consider for two points. When both are inserted in the interior of a MER, smaller MERs will result which may be overlapping in one Z-vertex, 2 Z-vertices or could be joined by a direct line. If both are inserted on the boundaries of a MER, then the same applies, or if placed on the outspan, one followed by the other, the later being placed in the firsts' outspan, no change results in the MER with respect to that point. If placed outside or adjacent to the cluster boundary, a new MER forms with the closest point on the cluster boundary, or joins through a collinear line. In case the two points are inserted so that they become cocircular to two points on the periphery of a cluster, the MST improvement will determine the steiner points.

Finally, for the 3-point insertion, if already forming a MER, or collinear, then these join by a Z-vertex. If inserted separately, then either the single point case arises or the previous case for two points, or both in either order. In the case that the three points are equidistant to each other and to a point or points on the periphery of the cluster, then by lemma 3, the steiner points can be determined. Thus for any number of points inserted, this would generate steiner points in the cluster and this completes the proof. Δ

The above can be alternatively verified by considering the overlapping MERs meeting at a Z-point or at the border of the MER. In both, the shortest edge consists of that from the steiner point to the Z-point where the join occurs or if at the boundary, the nearest point on the touching boundary. In the case where the shortest edge is from the steiner point to another steiner point in an overlapping MER, the MST improvement determines the shortest edge for this connection. Thus in a cluster, the minimum tree can be obtained by connecting the steiner points generated with the given Z points.

3.2 Cluster Generation

The determination of the cluster is now examined. Here unlike most clustering algorithms which require the determination of the center of clusters, their deviation from these or their approximates, we choose a 'nearness' distance metric. This is an agglomerative algorithm where, points lying within or on the boundary of this measure from any other point, lie in the same cluster. Examples of such measures are the Nearest-Neighbour, which partitions the points such that each point has its nearest neighbour in the same cluster, and the Mutual Nearest Neighbours, where points belong to the same cluster if and only if its nearest neighbour has its nearest neighbour as this point [Mur83]. A measure we define is the Nearest-Two Neighbour. This measure is the average of the length of the steiner tree in one MER with respect to each point i.e the average of the closest two points from every point. However, our algorithm is general enough, and a metric can also be experimentally be determined or be dependent on the distribution of points if known *a priori*. The reason for such an exercise is to locate steiner points that lie inbetween the clusters as that of Figure 2.2 and which most algorithms bypass. To determine these points we use an analogous construction to the MER, but this time inbetween the clusters. Using a greedy approach :

Determine for every point in a cluster, two nearest different clusters. Compute its MST length by taking the two lines ending in different clusters. We do so for all such points in the cluster, since it is not known which cluster is nearest to which point in the cluster, and then determine the minimal such MST. We also compute the minimum length edge from different points to different two closest clusters and compare this sum with the minimal lengths of MSTs found previously. If the former is smaller or equal to the later, we do not generate the MER, otherwise the steiner point is determined through the MER construction. In case there are multiple such minimal MSTs, their respective MERs are computed and their steiner points determined.

We examine in detail the point connections that are possible between the clusters.

1. **Z-Z connection.** For the two cluster case, there is a point to point connection and since this would be the shortest edge when chosen, it can be in any orientation. See Figure 3.7.

   When there are three clusters and all connecting with Z-vertices, then the MER can be constructed with each point in a different cluster to locate the steiner point. There will be cases where more than one point from a nearby cluster is closest to the point from another cluster (due to orientation of the points in that cluster), and these would generate additional points. Taking the case for four clusters, a case shown in Figure 3.7 does not locate all the steiner points, and hence like before, a MST improvement would be necessary. The case is similar to generating MERs for additional steiner points and would work for any orientation. Thus, this could be generalized to more than four clusters.

2. **S-Z connection.** In the two cluster case, the only possible case is the direct connection since the S-point in one cluster would be already of degree 3 and an edge could only be added, if coming from a Z-point, in a direct line of approach to this S-point where no edge exists. See Figure 3.8a. For the three

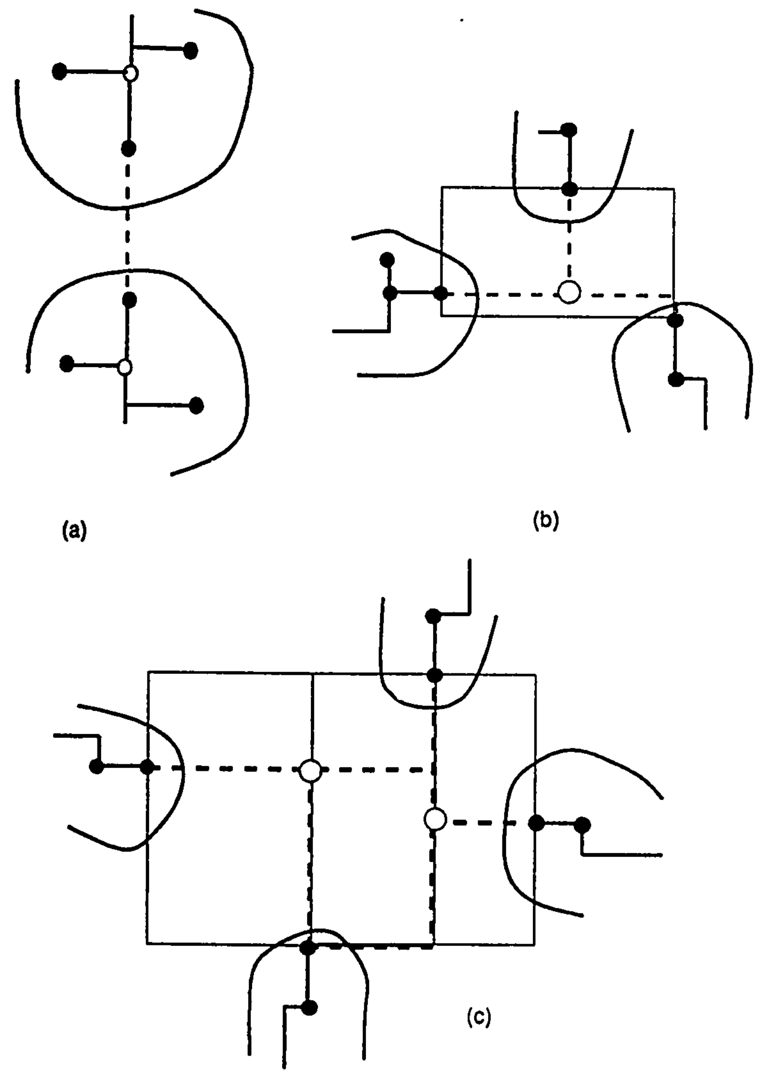


Figure 3.7: Z-Z connections (a) Two Cluster case (b) Three Cluster case

cluster case, considering each cluster in turn and constructing the MER if the condition is met, or multiple MERs in case there are equidistant points produces the desired steiner points. A case is shown in Figure 3.8b. Here also, the solution determined by the MST algorithm would select only one steiner point with degree 3, two steiner points with degree 2 and one with degree 1. A case where the MST length between clusters is less than the steiner length, such as in Figure 3.8c, no MERs are constructed by the greedy procedure - thus the MST algorithm will determine the optimal length in this case. In cases where more than one point is closest, additional steiner points would be generated and it is left to the final MST algorithm to determine suitable connections. The same can be applied to the four cluster case and larger.

3. S-S connection. For the S-S connections in two clusters, Figure 3.9a shows that such a case will always have a similar length line from the Z-point to either a Z-point or a steiner point and forms the case for the Z-S connection. The three cluster case is given in Figure 3.9b, where due to equidistant points, many steiner points are generated. Here, it is seen that only one will have degree 3 when the MST algorithm is finally executed, the rest having degree 2. Figure 3.9c shows the case for four clusters which has the same characteristics, and thus can be generalized for larger number of clusters.

It is seen in the above configurations that points that could be connected between

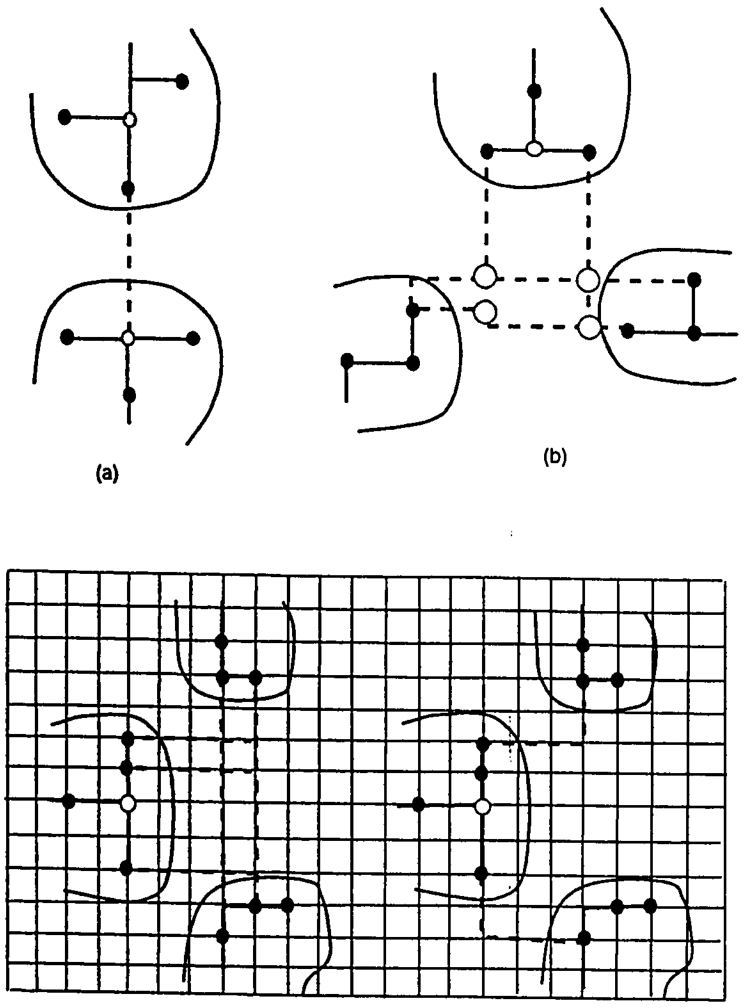
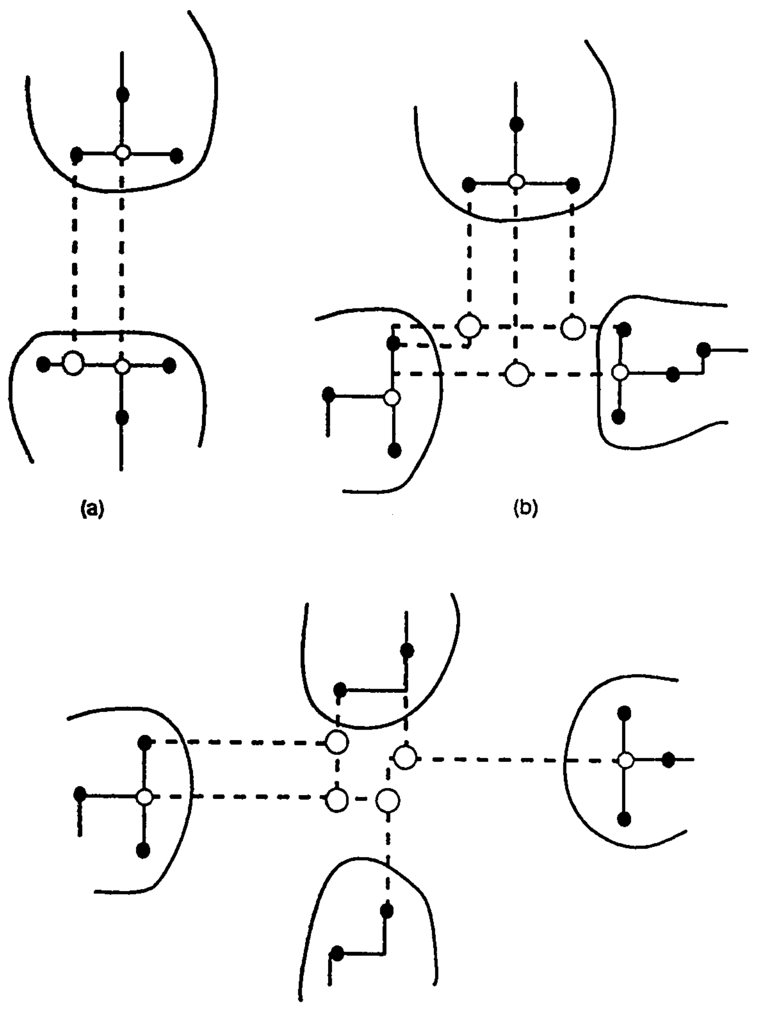


Figure 3.8: Z-S connections. (a) Two Cluster case, (b) Three Cluster case that generates Steiner points (c) Three Cluster case which does not generate Steiner points



(a) (b) (c)

Figure 3.9: S-S connections. (a) Two Cluster case. (b) Three Cluster case. (c) Four Cluster case

clusters have steiner points introduced by either the MER construction or after the MST improvement. The final MST algorithm then chooses sufficient number of points that would have degree 3 in the final topology. The steiner points with degree 2 are then eliminated.

Chapter 4

Complexity Analysis and Improvements

In this chapter, we first give the complexity analysis of the algorithm and then discuss how we can improve the time complexity while outlining the shortcomings of the previous algorithms that have used such procedures.

4.1 Complexity Analysis

We first give the psuedocode for the main algorithm :

```
   CLUSTERING-BASED ALGORITHM

1.  NEAREST_NEIGHBOUR_METRIC()
2.  CLUSTER()
3.  COMPUTE_MERS()
4.  GREEDY_CLUSTER()
```

```
5.   MST_IMPROV()
6.   MST_FINAL()
```

The input is given as a set V of N points and is labelled by positive numbers in increasing order of input. All distance measures are in the L_1 metric. We analyze each procedure in turn :

NEAREST_NEIGHBOUR_METRIC()

```
1.    $sum = 0$
2.    for $\forall$ i $\in$ V do
2.1.      find a nearest point in th $L_1$ metric from the set $V - \{i\}$
          and accumulate the $sum$
3.    compute the average $d = \lceil sum/N \rceil$
```

Computing the Nearest Neighbour or the Nearest-Two Neigbour requires examining all the points in the set, for each point in question. Time complexity is $O(n^2)$ and space $O(1)$.

CLUSTER()

```
1.    assign point 1 to cluster 1 and make K = 1
2.    assign 1 to Marked_counter
3.    for all points within or at distance $d$ from 1, put them in a queue
4.    while queue $\neq \emptyset$ do
4.1.      pick $Q \in$ queue
4.2.      mark $Q$ and increment Marked_counter
4.3.      determine all unmarked points within or at distance $d$ from $Q$
          and not already in queue
4.4.      insert in queue if any
5.    if Marked_counter $\neq$ N
5.1.      pick an unmarked point and determine all points within or at distance $d$
          form this point
```

```
5.2.        if none determined increment K and assign K to this point
5.3.        else assign the current cluster number K to it
             and insert the points in queue
5.4.        mark this point
5.5.        goto step 4.
6.     else return
```

Lines 1-2 take constant time. Line 3 takes $O(n)$ time. The lines in loop of 4 take constant time, except 4.3 which requires a scan of all points. Line 5 block takes also constant time with exception of 5.1 that may require a scan of all points. In the worst case, if points are at an increasing distance from each other and placed at opposite extremes of the point set in succession, then 5.5 will branch to 4 for a factor of n points, giving the bound of $O(n^2)$. Space complexity for queue and counters is $O(n)$.

```
COMPUTE_MERS()

1.     for ∀ i ∈ V do
1.1.        determine nearest two points, and all such points if equidistant and
             in the same cluster.
1.2.          if there are more than two points, then copy these and sort in
                  major X, minor Y, and the other copy in major Y, minor X and take
                  two points in succession to form an MER with $i$, and obtain steiner
                  points if any generated
1.3.          else construct MER with $i$ and obtain steiner points
```

Line 1.1 scans the entire set in $O(n)$ time and sorting $O(n\, log n)$ if many equidistant points, but constant time if only two points are determined. This is repeated for all points in the set thus taking a time of $O(n^2)$ worst case. If a factor of the points generate equidistant points, like in a 'star', then only the inverse factor of that points would have such a case, thus binding the time complexity to $O(n\, logn)$. Space complexity is at most $O(n)$ for storing the equidistant points.

```
GREEDY_CLUSTER()

1.      for each I = 1..K do
1.1.        for each $j \in$ I do
1.1.1.          locate two closest clusters and compute the MST formed by
                the two points each from different cluster with $j$.
                If there are equidistant points store them in a list
1.2.        determine the closest two clusters from $p, q \in$ I with $p \neq q$ and
            compute the sum of their lengths
1.3.        compute RST of the points in step 1.1.1, and for all in the list if many
1.4.        if RST < MST accept the steiner point, otherwise discard it
```

The greedy procedure locates steiner points from closest two points for a point in each cluster only if the length of the connections with the point is less than the external MST length of the three clusters. Due to the nested loops it takes $O(n^2)$ in the worst case. Space complexity is a constant since not many clusters have equidistant points.

```
MST_IMPROV()

1.    use Prim's MST algorithm on a point set $\leq 2N$
2.    locate degree 2 points and compute MERs
3.    locate degree 3 steiner points and compute MERs
4.    update MST
```

Prim's algorithm takes $O(n^2)$ time with space complexity of $O(n^2)$. All the other computations fall under this bound.

```
MST_FINAL()
1.    compute the final MST
2.    locate degree 2 steiner points and eliminate them from the adjacency list
```

The final MST computation and the removal of steiner points of degree 2 also fall under the time bound of $O(n^2)$.

Thus the overall complexity of this algorithm is $O(n^2)$. The space complexity is overall $O(n^2)$.

4.2 The Delaunay Triangulation and its Usage

The computation of the Nearest Neighbours is a common problem in Computational Geometry and falls under the general case of All Nearest Neighbours as a proximity problem. The powerful algorithm of the Voronoi Diagram forms the basis for all proximity problems and in this case, the Delaunay Triangulation which is the dual of the Voronoi Diagram, has been employed by previous researchers [Hwa79b], [SLL80]. This structure gives an improved algorithm for the MST, reducing the time complexity to $O(n)$. However, this is due to preprocessing, which falls under an upper bound on the construction of the structure itself. A construction is shown in Figure 4.1.

Hwang [Hwa79a] gave the first algorithm which uses the Voronoi Diagram in the

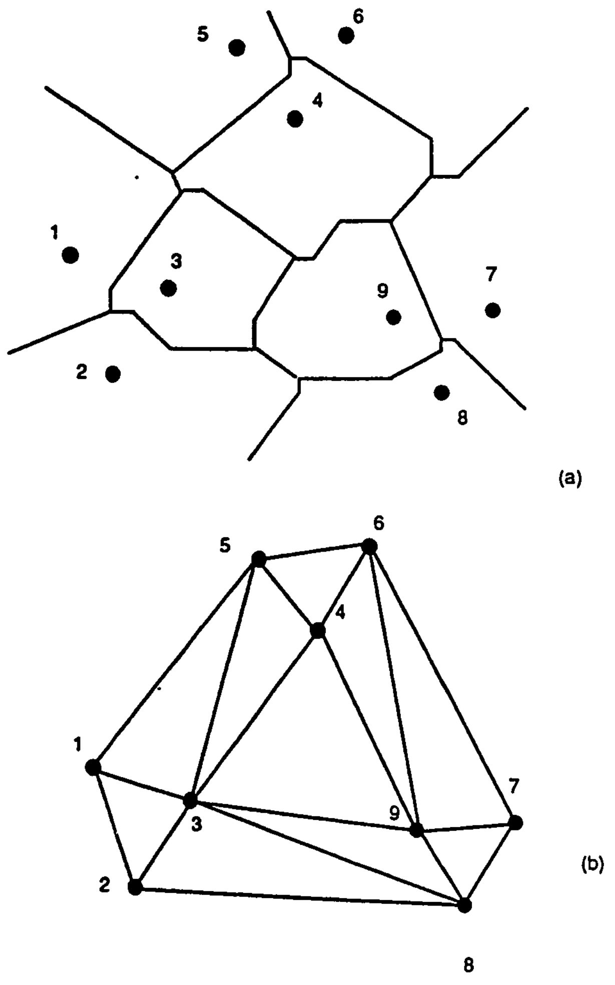


Figure 4.1: (a) The Voronoi Diagram (b) The Delaunay Triangulation

L_1 metric. He however uses the structure to compute the rectilinear MST (RMST) first, then resorts to improving the edge lengths by considering the triple of points along the RMST edge to determine the steiner points. Another work of Smith, Lee and Liebmann [SLL80] begins with the Delaunay Triangulation, computes the RMST and and also resorts to improve the construction by picking on the triangles with the shortest edge, and further, by a pair of triangles if any other improvement is possible.

We note that by starting out with a RMST, the chances of obtaining the shortest edge in the final solution is reduced, particularly if there are edges with equidistant lengths. The RMST algorithm being greedy in nature, sufers from being mislead as indicated by Kahng [KR92b]. We show the cases in Figure 2.2.

We use the Delaunay Triangulation procedure to improve our time complexity. Most of the time in our algorithm is used in locating the Nearest Neighbours, and we utilise the structure of the triangulation to restrict our search procedure. We present the new algorithm as:

```
NEW_CB_ALGORITHM

1.  DELAUNAY_TRIANGULATION()
2.  NEAREST_NEIGHBOUR_METRIC()
3.  CLUSTER()
4.  COMPUTE_MERS()
5.  GREEDY_CLUSTER()
6.  MST_IMPROV()
7.  MST_FINAL()
```

We consider the details in turn :

DELAUNAY_TRIANGULATION() computes the triangulation of the point set in the L_2 metric using the algorithm of Lee and Schachter [LB80]. It takes as input the point list and constructs a double-linked list for each point as DT(v_i) in the point set $v \in V$, which is the set of points having an edge to v_i in the triangulation. The output is this set of double-linked lists. The algorithm is recursive and has time complexity $O(n\,logn)$ and space complexity of $O(n)$.

The NEAREST_NEIGHBOUR_METRIC() takes linear time since we only have to examine the linked list once. The Nearest Neighbour (or Nearest Two Neighbours) are determined passing through each points list DT(v_i), in one scan while determining the nearest neighbour(s) and in the end computing its average over the N points. Space use is of $O(1)$.

In CLUSTER(), we have the same procedure as our previous algorithm but instead of scanning all the point set we only look for the points in each points' linked list for the points not exceeding the distance d to determine the next point. However, while we do this, we also use a list to mark points already processed and also when deleting from the queue, we maintain a cluster label list with respect to cluster number. since it is known that the average number of neighbours is six [PS85] to a point, a result of Euler, the queue length is a constant and hence overall time complexity is

123

linear. Space complexity additionally turns out to be linear too.

For COMPUTE_MERS(), we again use the linked list DT(v_i) to determine the nearest two neighbours, and all such neighbours, provided they are in the same cluter in order to compute the MERs. We store the new steiner points in the same double-linked list, while updating the cluster labels. The time complexity is thus linear. Since a maximum of $n - 2$ steiner points can be obtained, additional linear space is used.

In GREEDY_CLUSTER(), we begin at the cluster label list and for each point in the same cluster, we determine two closest different clusters from the adjacency list and compute its sum. We do so for all such points in the cluster, and then find the minimal such sum. Then the list is searched again, this time for the closest cluster different from this cluster number and also for the next closer cluster number but not from the same point as the closest one. We then determine if the RMST length from this set of points is larger than the steiner tree length from the MER construction with the previous set of points. If this is so, then the steiner point generated is inserted in the first point of the previous set. If there were many such minimal sums, they are first checked in the respective adjacency lists of the point sets before insertion so as to avoid duplication. In each scan there are only a constant number of comparisons, and hence the time complexity is linear. Space complexity is linear since a factor of the point set may determine steiner points.

MST_IMPROV() generates a degree list which is updated when an edge is joined to form a tree. This is done as the MST is constructed using the algorithm of Cheriton and Tarjan [CT76], which uses a queue to store singleton trees initially which are then combined at each stage depending on their established priority. Then the MST improvement is done where the MERs are composed in one scan with the steiner points of degree 3 or with those of Z-points and steiner points and inserted in the respective linked list. The time bound is of $O(n\,logn)$ for the MST construction and its improvement. Space complexity may result in worst case $O(n)$.

The final MST_FINAL() algorithm first computes the MST, the second time, and also maintains the adjacency list. It then deletes steiner points of degree 2 by examining the degree list for such cases. The algorithm results in the worst case time complexity of $O(n\,logn)$. Space has linear complexity.

From the above, it is seen that the overall time complexity of the algorithm is bounded by the complexity of the Delaunay Triangulation construction and the MST construction, and is thus $O(n\,logn)$. The space complexity turns out to be $O(n)$.

123

Chapter 5

Computational Results and Limitations

5.1 Computational Results

To test the algorithm, we generated random points uniformly distributed on a M x M grid. The algorithm is coded in C and run on a NeXT Computer with 16Mb main memory and a 68030 Motorola microprocessor. The Clustering metric chosen were the Nearest-Two Neighbours. whose results are given in Table 5.1. and the Nearest-Neighbour metric, given in Table 5.2. In both $\overline{d}$ is the average distance metric and K the number of clusters determined. We present the results of our runs separately. They were tested for the same problems. Each row gives the results for the average values made for 500 samples.

No. of Points	Grid	d_{2NN}	K_{2NN}	$\frac{MST-RST}{MST}$ %	maximum	minimum
10	50 X 50	15	3	9.09	19.87	0.00
20	"	10	9	8.97	16.59	2.34
40	"	7	15	8.91	14.42	4.44
60	"	6	26	8.71	13.04	4.86
80	"	5	27	8.56	11.61	5.12
100	"	4	43	8.46	12.03	4.76
10	100 X 100	28	4	8.99	19.56	0.71
20	"	19	12	9.14	15.38	3.17
40	"	13	17	9.09	13.78	4.41
60	"	11	27	9.09	11.57	5.56
80	"	9	37	9.09	12.52	5.07
100	"	8	43	8.95	12.31	5.54
250	"	5	107	8.81	10.84	6.85
500	"	4	158	8.10	9.77	6.48

Table 5.1: Results of Nearest-Two Neighbour metric

No. of Points	Grid	d_{NN}	K_{NN}	$\frac{MST-RST}{MST}$ %	maximum	minimum
10	50 X 50	10	5	9.22	19.87	0.92
20	"	8	11	9.12	15.70	2.78
40	"	5	24	9.10	14.42	4.69
60	"	4	33	8.94	13.06	4.89
80	"	5	44	8.82	11.86	5.88
100	"	3	55	8.79	12.36	5.70
10	100 X 100	26	5	9.00	20.00	1.80
20	"	15	9	9.31	15.65	2.22
40	"	11	24	9.18	13.78	3.85
60	"	9	35	9.22	13.45	5.86
80	"	7	48	9.25	13.01	5.49
100	"	6	58	9.12	12.39	5.86
250	"	4	142	9.02	11.07	7.06
500	"	3	256	8.63	9.94	7.03

Table 5.2: Results of Nearest Neighbour metric

Authors	Complexity	No. of points	$\frac{MST-RST}{MST}$ %
Branch & Bound	exponential	< 225	11.9
L & Z shaped MST	$O(n \times l^6_{max})$	100	10.2
Hierarchical-ST	$O(f(k)\, logn)$	100	9.1 - 10.2
Kruskal-based	$O(n^3)$	-	8.5
Iterated Steiner	$O(n^3)$	< 40	10.9
D.Richards	$O(n\, logn)$	10 - 500	< 5.3
Delanuay Triang.	$O(n\, logn)$	< 40	< 8.6
Clustering-Based(Ours)	$O(n\, logn)$	10 - 500	8.63 - 9.31

Table 5.3: Comparison of worst-case time complexity and average ratios from different authors

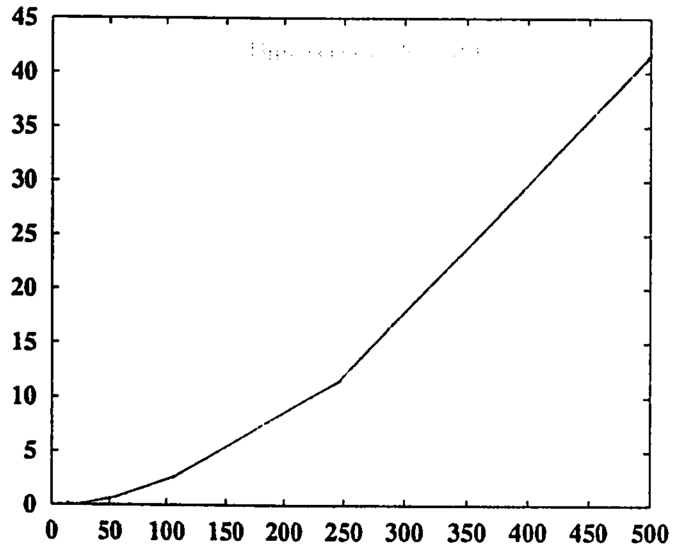


Figure 5.1: Time(secs) Vs. Number of Points for the $O(n^2)$ algorithm (on NeXT)

We compare the results to those in literature in Table 5.3. Our algorithm has matched the previous $O(n\, logn)$ worst-case bound algorithms with better average ratios then theirs. Figure 5.1 shows the running time in seconds against number of points for the $O(n^2)$ algorithm on the NeXT machine.

5.2 Limitations

It is noted that the algorithm still resorts to the RMST procedure for connecting the determined steiner points. Although reducing the greedy effect, it does not completely remove it. This is particularly the case when equidistant points are encountered. Figure 5.2a shows the case in point. Steiner points are determined through the sorting procedure and then the MER construction of successive pair of points with respect to point (2,2).

It is noted that the choice of the starting line matters, and in Figures 5.2b and c show suboptimal solutions, while d shows the optimal required. This is due to no priority given to the selection of point connections in the final RMST construction procedure. However, such occurances are rare in random points and thus the results do indicate so.

123

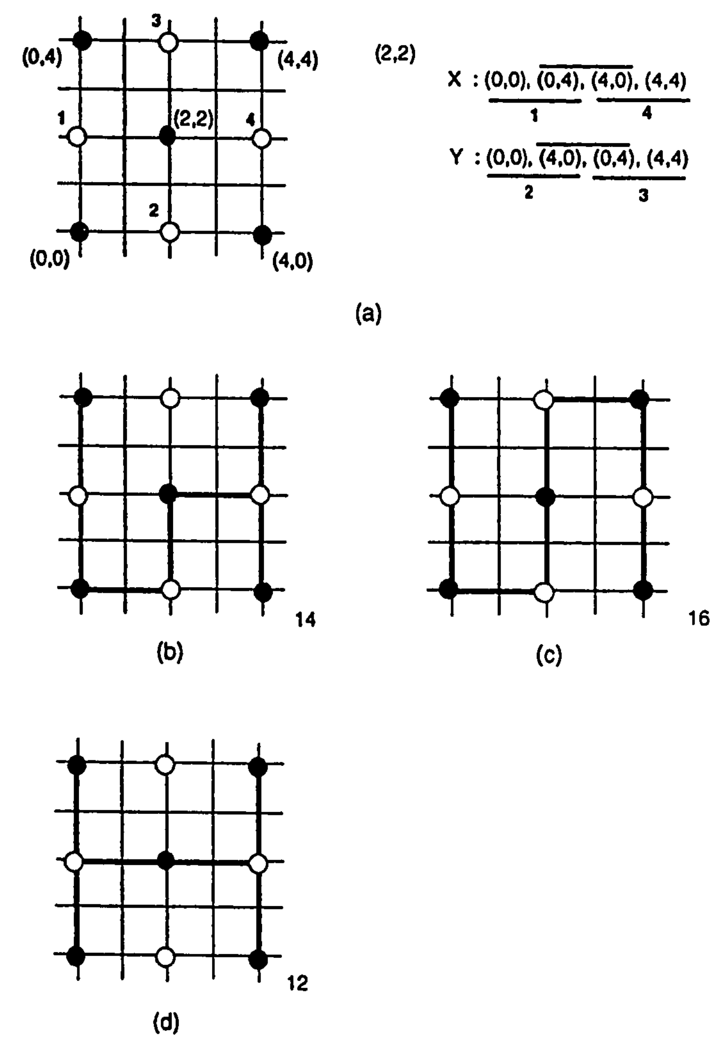

(0,4)
3
(4,4)
(2,2)
1
(2,2)
4
2
(0,0)
(4,0)
X : (0,0), (0,4), (4,0), (4,4)
1
4
Y : (0,0), (4,0), (0,4), (4,4)
2
3
(a)
14
(b)
16
(c)
12
(d)

Figure 5.2: Effect of the greedy procedure on equidistant steiner points

Chapter 6

A Neural Model for the Steiner Problem in Graphs

In this chapter we attempt to formulate the Steiner Problem in Networks for a neural network implementation. We first give an introduction to the Hopfield model and give our problem representation for it. We then present the results and the problems encountered.

6.1 The Hopfield Model

This is a general class of a neural model where neurons are fully connected and have a threshold function. Figure 6.1a shows such a single layer network. To use this in optimization, an energy function is derived which corresponds to the objective function of the entities in the problem and also accounts for the constraints of the problem. These constraints are constructed such that when they act upon the inputs to the problem in a dynamic fashion, the output so generated reflects the minimization of the objective function. The quadratic form of the energy function is represented as :

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} I_iV_i$$

where V is the state of the network, T is the connection matrix (synapse) between the neurons V_i and V_j, and I is the input bias. See Figure 6.2 for a circuit diagram. Each neuron has $\sum_{j\neq i}^{N} T_{ij}V_j + I_i$ as input and its state changes according to

$$V_i \rightarrow 0 \quad if \sum_{j\neq i}^{N} T_{ij}V_j + I_i < U_i$$

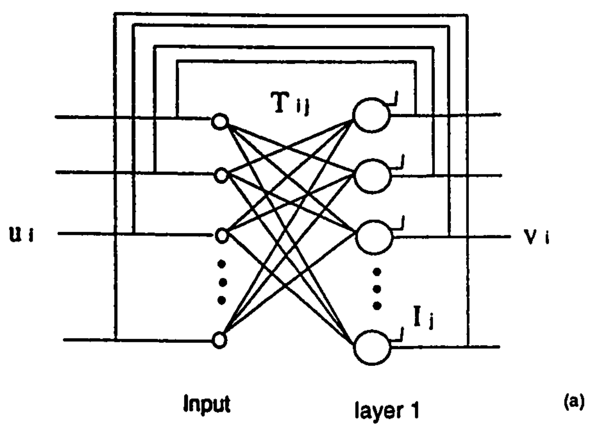
$$V_i \rightarrow 1 \quad if \sum_{j\neq i}^{N} T_{ij}V_j + I_i > U_i$$

where U_i is the threshold condition for the ith unit.

Such a system with symmetric connections and non-negative elements on the diagonal [Hop82] would always converge to a stable state which would be a local minimum of the energy. This is the binary model, and its continuous analogous model has the following form:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} T_{ij}v_iv_j + \frac{1}{\lambda}\int_0^{v_i} g_i'(v)dv - \sum_{i=1}^{N} I_iv_i$$

Here $R_{ij} = |\frac{1}{T_{ij}}|$ is the input resistance to I_i and $g_i'(v) = u$ is the inverse of the



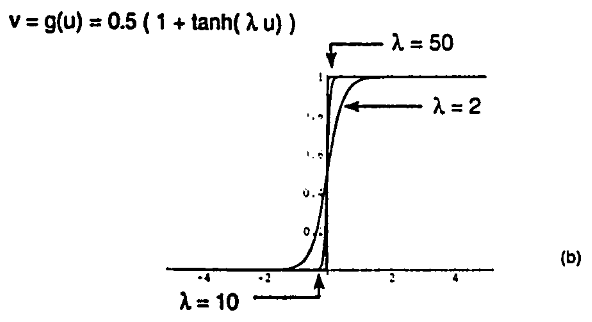


Figure 6.1: (a) Single Layer Recurrent Network (b) Sigmoid function of each unit

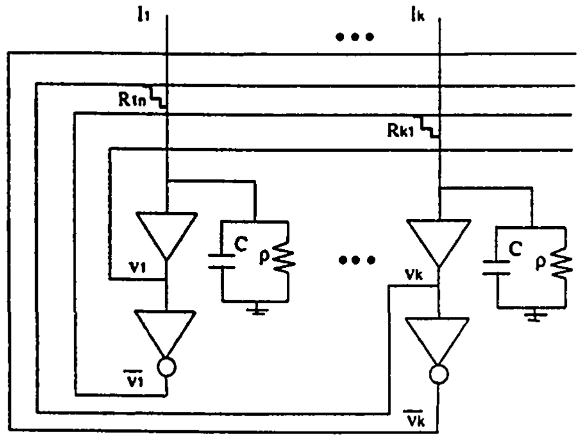


Figure 6.2: The Circuit Diagram of a Hopfield Network

123

sigmoid function

$$v_i = g(\lambda u_i) = \frac{1}{2}(\,1 + tanh(\lambda u_i)\,)$$

where λ is the gain parameter which is shown in Figure 6.1b.

The evolution dynamics of the network is governed by the equations :

$$\frac{du_i}{dt} = \sum_{j \neq i}^{N} T_{ij} v_j - \frac{u_i}{\tau} + I_i$$

which we iterate as

$$\Delta u_i(t+1) = \Delta u_i(t) + \Delta u_i$$

It has been shown by Hopfield [Hop84] that this system of equations models a Lyapunov function and takes the form :

$$\frac{dE}{dt} = -\sum \frac{dv_i}{dt}\left(\sum_{j=1}^{N} T_{ij} v_j - \frac{u_i}{\tau} + I_i\right)$$

which decreases as the system evolves and convergence is guaranteed to a local minimum. The updates can be either in an asynchronous manner, where different neurons update in different order, or in a synchronous manner where neurons are all updated at the same time. Either of the ways can be used and in most applications it does not give significant differences in results.

6.2 Preprocessing Procedures

We adhere to the convention of representing the given nodes as Z-nodes and the additional nodes as S-nodes. In order to represent the constraints, we use the following information which we avail to the network:

1. **Single Source Shortest Paths** : Every node should know its distance to all the other Z-nodes. An S-node in the final solution should have the minimum distance to all the other Z-nodes. We use the Floyd-Warshall algorithm of complexity $O(n^3)$ to obtain the All Pairs Shortest Paths (the Single Source Shortest Path when executed for every node). This information is available to every neuron.

2. **Removed Edge Shortest Paths** : This procedure computes the difference in shortest path lengths of each node to every other Z-node from that value when each edge to a node is removed. More formally, if node i has edge x, then its sum of distance to all the Z-nodes is say, *sum1*, and when x is removed, the nodes distance to all the other Z-nodes is say, *sum2*, then this routine should return the difference *sum1 - sum2*. If the difference is positive, it means that when the edge is removed the node is disconnected (i.e its distance to all the other nodes is 0) and thus this edge is crucial. If the difference is negative then, it also shows that the edge is important since the cost increases when the edge is removed. If the difference is zero, then this edge has no effect, and can be deleted without any change in cost. The algorithm is a repeated application of the Floyd-Warshalls algorithm for each edge tested with the resulting complexity of $O(n^3m)$.

6.3 A Representation Scheme

Given a graph $G(V,E)$ with $|V| = N$ and $|E| = M$ and its vertices and edges are in some lexicographic order, we utilise a $M \times N$ array of neurons for this problem. We represent this as a 2D array (transpose of the incidence matrix) where the rows represent the labeled edges, and the columns, the labelled universe of vertices. Neurons with output 0, indicate absence of the edge, and 1 its presence. The final solution should give a minimum number of 1s to reflect the minimum number and hence the cost of edges. Figure 6.3 shows an example and its optimal solution is as Figure 6.4. We use the double subscripts to represent the vector V as a matrix.

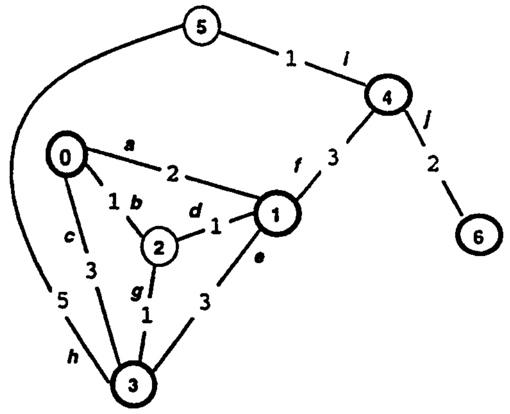
To determine the Z-nodes from the universe of vertices, every neuron must have the information that the column it is in represents either a Z-node or a S-node. This can be provided by a bit field indicating a Z-node (1) or a S-node (0).

The energy function for this problem is given as :

$$\begin{aligned} E = {} & P\sum_{x=1}^{M}\sum_{i=1}^{N} v_{xi} Row(x) + Q\sum_{x=1}^{M}\sum_{i=1}^{N} v_{xi}(1 - zvtx(i)).Col(i) + R\sum_{x=1}^{M}\sum_{i=1}^{N} v_{xi} RemSP(x) \\ & + F\sum_{x=1}^{M}\sum_{i=1}^{N} v_{xi} ShPath(i) + B\sum_{x=1}^{M}\sum_{i=1}^{N} v_{xi} Cost(x) + C\sum_{x=1}^{M}\sum_{i=1}^{N} v_{xi} zvtx(x) \\ & + A\sum_{x=1}^{M}\sum_{i=1}^{N}\sum_{j=1_{j\neq i}}^{N} v_{xi} \end{aligned}$$

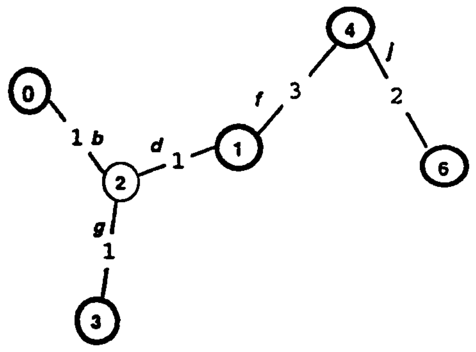
This can be written as :

$$E = -\frac{1}{2}(\frac{A}{2})\sum_{x=1}^{M}\sum_{i=1}^{N}\sum_{j\neq i}^{N} v_{xi}v_{xj} - \sum_{x=1}^{M}\sum_{i=1}^{N}(P.Row(x) + Q.(1 - zvtx(i)).Col(i)$$



	0	1	2	3	4	5	6
a	1	1	0	0	0	0	0
b	1	0	1	0	0	0	0
c	1	0	0	1	0	0	0
d	0	1	1	0	0	0	0
e	0	1	0	1	0	0	0
f	0	1	0	0	1	0	0
g	0	0	1	1	0	0	0
h	0	0	0	1	0	1	0
i	0	0	0	0	1	1	0
j	0	0	0	0	1	0	1

Figure 6.3: (a) The Labelled Graph (b) Matrix representation



	0	1	2	3	4	5	6
a	0	0	0	0	0	0	0
b	1	0	1	0	0	0	0
c	0	0	0	0	0	0	0
d	0	1	1	0	0	0	0
e	0	0	0	0	0	0	0
f	0	1	0	0	1	0	0
g	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0
j	0	0	0	0	1	0	1

Figure 6.4: (a) Shortest Connecting Graph (b) Matrix of the solution

$$+F.ShPath(i) + R.RemSP(x) + B.Cost(x) - C.zvtx(i)\ )\ v_{xi}$$

The first part composes the connection matrix and the second part the external input. The first two terms in the second part are dynamic inputs, i.e they are a function of the outputs, and always decrease. The other two terms are static (constant), and always decrease the energy. The last term increases E but is never greater than the total sum combined of all others. The evolution of the system is then modelled as

$$\Delta u = (\sum_{j=1}^{N} T_{ij} v_j - \frac{u_{xi}}{\tau} + I_i)\,\Delta t$$

which we iterate as $u_i(t+1) = u_i(t) + \Delta u_i$. On making appropriate substitution, we get the form :

$$\Delta u_{xi} = [\ -\frac{u_{xi}}{\tau} - A\sum_{y=1}^{N}\sum_{j\neq i}^{M} v_{yj} - P.Row(x) - Q.(1 - zvtx(i)).Col(i) - F.ShPath(i)$$
$$-R.RemSP(x) - B.Cost(x) + C.zvtx(i)\ ]\ \Delta t$$

We now explain this in detail. The first term is due to the time constant of the circuit (the damping term), and the second one inhibits all the other nodes and edges in attempt to minimize the overall number of 1s in the solution. The third and fourth terms are 'syntatic' constraints, in that they only facilitate effective representation of the graph problem. The fifth term acts as inhibition to the nodes in attempt to eliminate those nodes that have large path lengths to all the other Z-nodes. The sixth term inhibits those edges that do not change the overall length when removed.

The last term excites those nodes that have to be there for a valid solution. We give the energy equation details as under :

1.

$$E_1 = P \sum_{x=1}^{N} \sum_{i=1}^{M} V_{xi} Row(x)$$

This term is zero when there are only two 1's in a row to depict an edge, otherwise it increases the energy. The function Row(x) checks to see if row x has only two 1's and returns 1 if it does not.

$$Row(x) = \begin{cases} 0 & if\ \sum_{i=1}^{N} v_{xi} = 2 \\ 1 & otherwise \end{cases}$$

2.

$$E_2 = Q \sum_{x=1}^{M} \sum_{i=1}^{N} v_{xi} (1 - zvtx(i)).Col(i)$$

This term increases the energy if the S-nodes have only one edge (degree 1), otherwise it remains zero. This is to remove 'dangling' edges in the graph.

$$Col(i) = \begin{cases} 1 & if\ \sum_{y=1}^{M} v_{yi} = 1 \\ 0 & otherwise \end{cases}$$

3.

$$E_3 = F \sum_{x=1}^{M} \sum_{i=1}^{N} v_{xi} ShPath(i)$$

Energy is desired to be minimized when the shortest paths are sought from each node to all the other Z-nodes. This is effectively the case for any S-node that exists in the optimal solution. The function *ShPath* gives the sum of distances as computed in the preprocessing stage.

4.

$$E_4 = R\sum_{x=1}^{M}\sum_{i=1}^{N} v_{xi} RemSP(x)$$

The energy increases if RemSP(x) returns 1 due to the fact that the removal of edge x does not effect the cost. Thus such unecessary edges are minimized. RemSP(x) returns zero if edge is either very crucial or just as important (due to the difference being either positive or negative respectively).

5.

$$E_5 = B\sum_{x=1}^{M}\sum_{i=1}^{N} v_{xi} Cost(x)$$

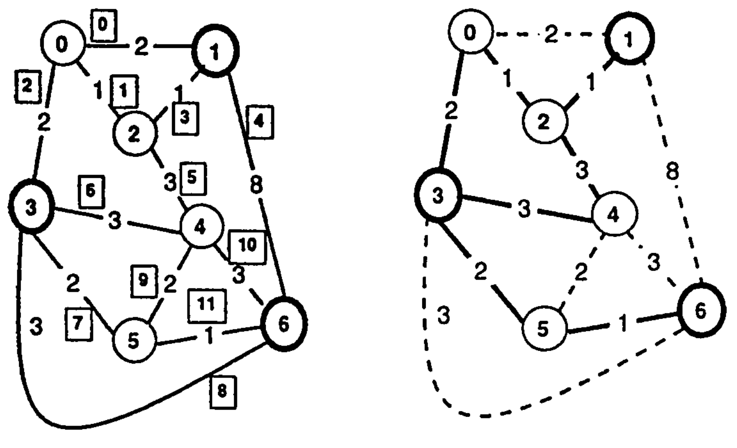
Those edges having high cost increase the energy and thus through minimization, less costly edges are desired to be determined.

6.

$$E_6 = C.zvtx(i)$$

We seek to enforce Z-nodes to be present in the final solution in order that a valid solution be generated. This external bias is positive and increases the energy but only marginally since it has influence only when no other inhibitions are significant.

An example showing the computation of the RemSP values is shown in Figure 6.5. Edges are labelled in boxes and tabled in the first column. The second column



(a)

(b)

E	sum of shortest paths on end nodes *sum1*		sum of shortest paths without edge on end nodes *sum2*		Difference of *sum1 - sum2*	
0	9	11	9	11	0	0
1	9	10	9	12	0	-2
2	9	7	16	10	-7	-3
3	7	10	11	12	0	-2
4	11	10	11	10	0	0
5	10	10	10	13	0	-3
6	7	10	7	11	0	-1
7	7	9	7	11	0	-2
8	7	10	7	10	0	0
9	10	9	10	9	0	0
10	10	10	10	10	0	0
11	9	10	13	10	-4	0

(c)

Figure 6.5: (a) Graph with labelled edges (b) After RemSP (c) Table of values

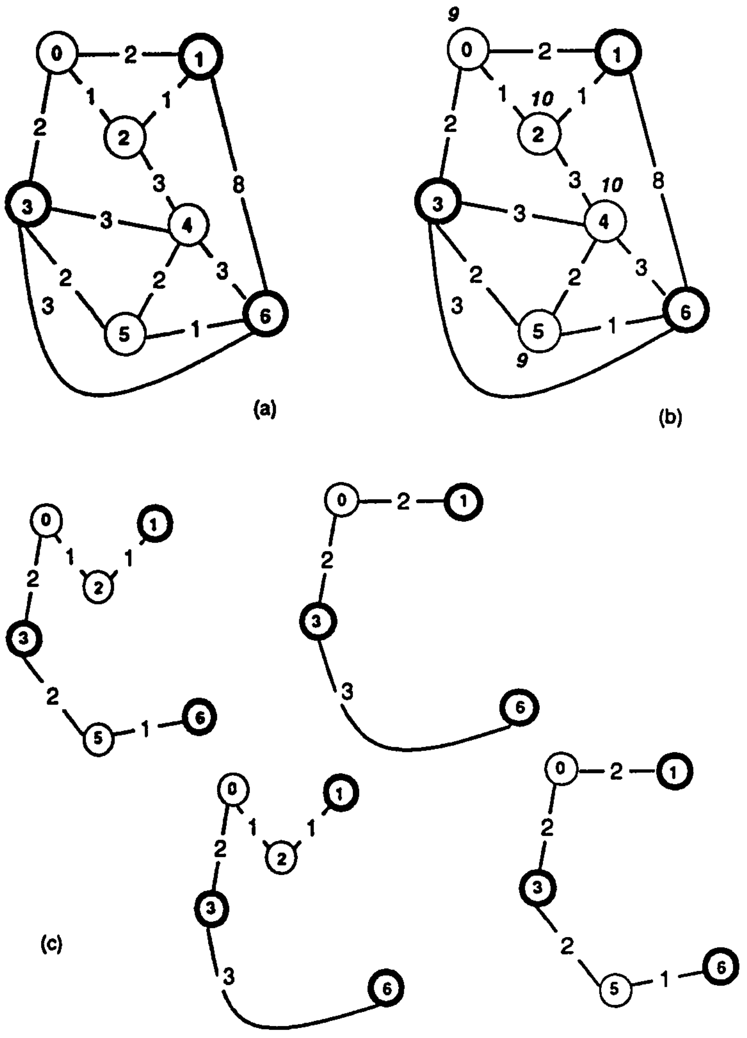
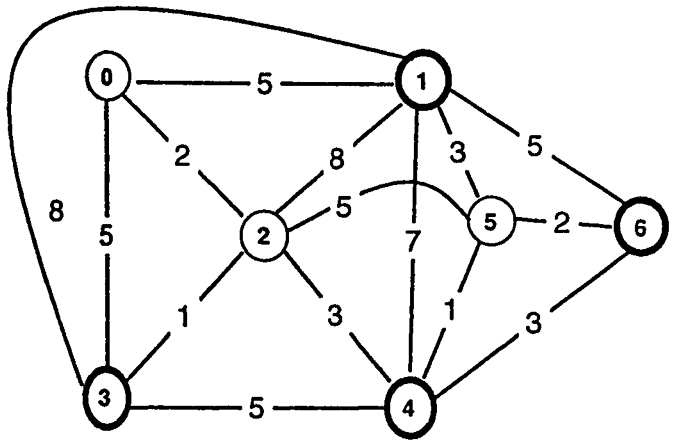


Figure 6.6: (a) The Graph (b)Graph with sum of shortest paths to Z-nodes (c) The Solutions

in the table computes the sum of the shortest path to all Z-nodes from each of the end nodes of the corresponding edge. The third column does the same thing but without the edge. The difference is given in the last column and the corresponding edge that has its difference value zero is portrayed as a dashed edge in the illustration. These are the edges that get inhibition in the network so that they are eliminated. An example where the Cost and ShPath function values differ in terms of locating suitable edges is shown in Figure 6.6. Here the main confusion comes from paths that are equal in costs. Edge 3-6 has a direct edge of cost 3 and a path via node 5 of cost 3. However, the network determines one of the solutions correctly since larger edge costs are eliminated first. The network gives the first graph in Figure 6.6e as the solution.

Figure 6.7 shows how the shortest paths alone do not help in determining all the nodes that will be there, and the RemSP procedure shown in Figure 6.8 determines vaguely the necessary edges. The edge 0-2, is not determined by the RemSP procedure but the shortest path weight removes it on node 0. The edge 2-5 is also not determined by RemSP, but is removed due to its cost being high as is edge 1-6. Edge 2-3 however, determined by RemSp to be deleted, appears in the solution due to its low cost. Only the cost function effectively deletes the high-cost edges.

Figure 6.9 denotes how the RemSP procedure does no edge deletion at all, and we must resort to the shortest path procedure which determines the node that could be included and thus directs search into better solutions. We mention that no



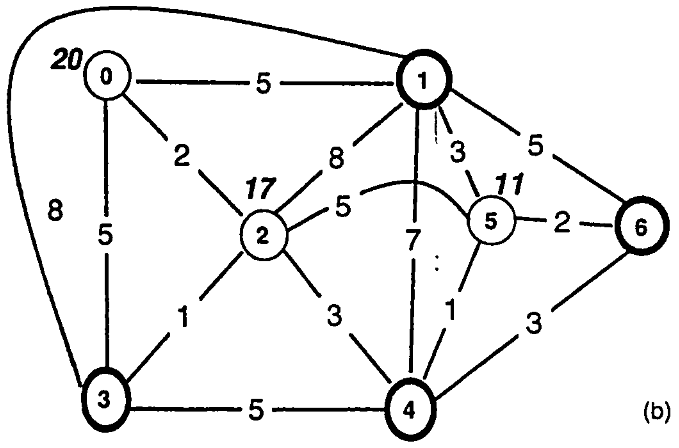
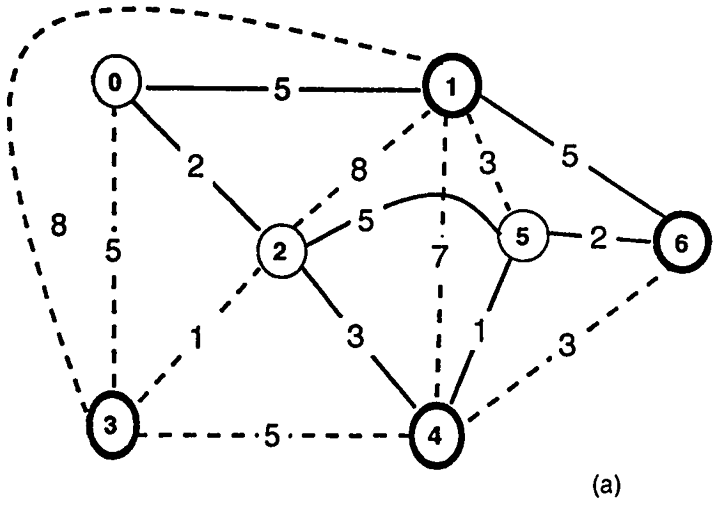


Figure 6.7: (a) Graph with labelled edges (b) Shortest Path to all Z-nodes



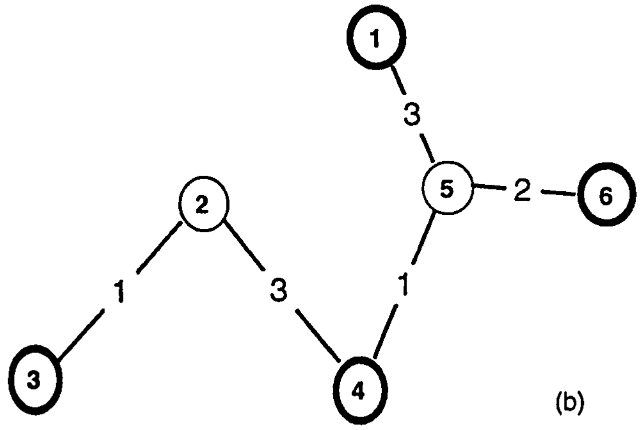


Figure 6.8: (a) Graph with RemSP computation (b) The Solution

single input determines the complete solution and thus a collection of such inputs is desirable.

It is noted that the dynamic equations have more of inhibition than excitation and the energy would not settle to a value until all edges have been deleted, making the change in energy zero. Thus, the termination of the iterations should be externally activated upon some condition. For a valid solution, since it is required that the shortest connecting network must contain the given vertices, whenever a Z-node just begins to be eliminated, we stop at that point. This can be detected through a summation of the outputs in each column of the matrix for each of the Z-node to determine its absence. A trigger can then be incorporated to stop the iteration. However, in order not to loose the previous neuron outputs, they should be stored in a register as the solution. Figure 6.10 gives the layout of the network and the trigger for the Z-nodes as of Figure 6.4.

6.4 Computational Results and Problems

We have tested the network on some random graphs with a varying number of Z-nodes and S-nodes (the sum being under 13). The results are depicted in Table 6.1. The parameters A, B, C, F, P, Q, R were empirically determined and are different for different size problems.

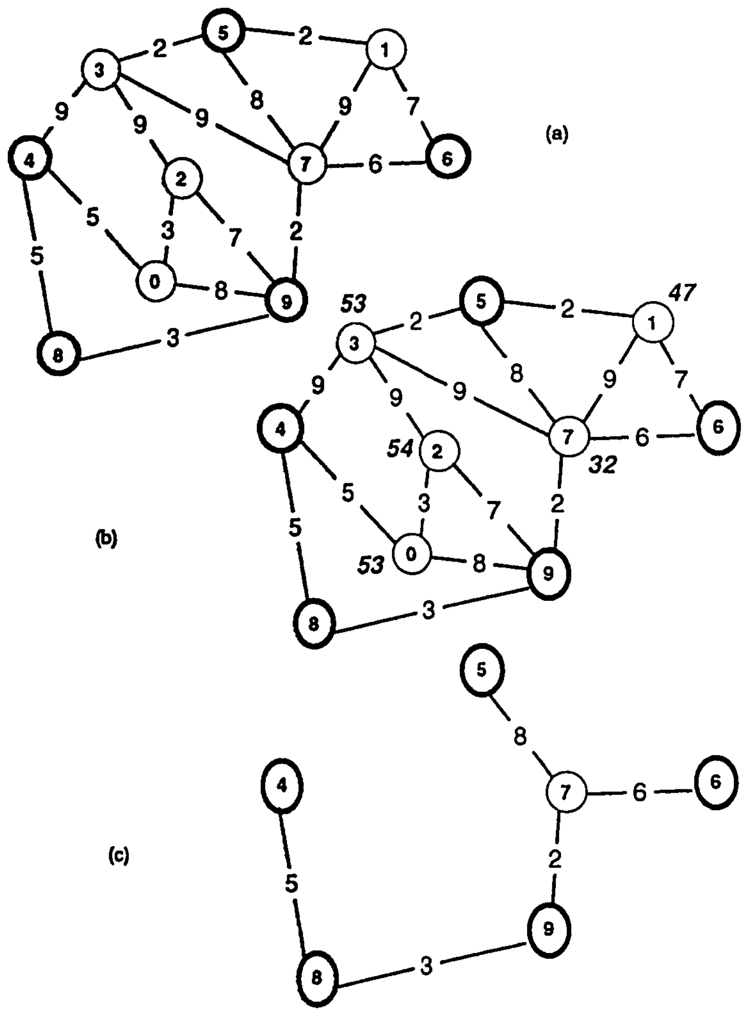


Figure 6.9: (a) Graph with labelled edges(b) After RemSP (no change) and ShPath weights on nodes (c) Optimal solution by network

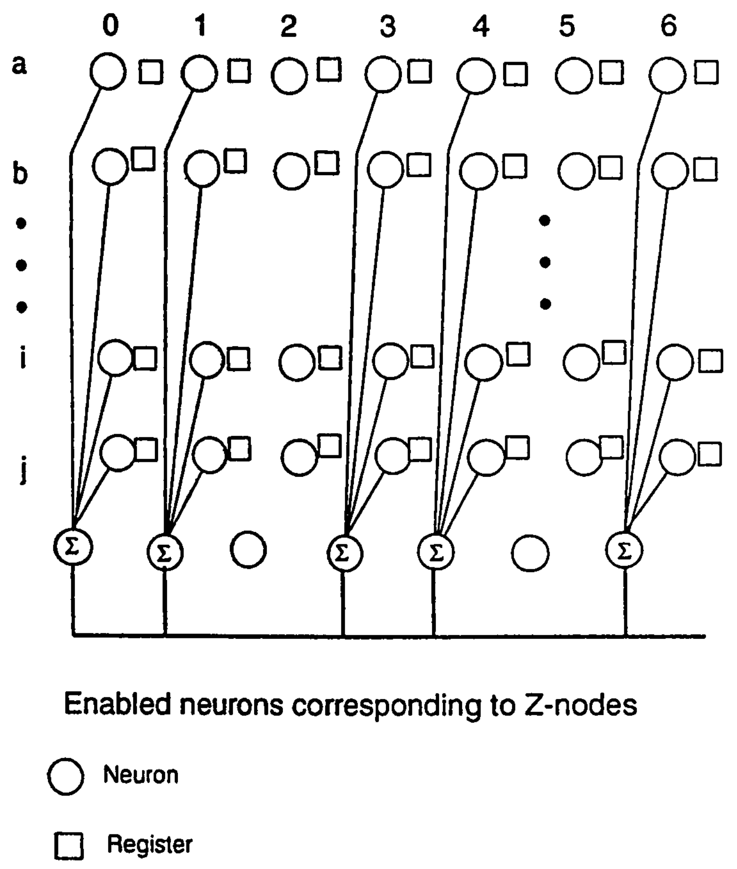


Figure 6.10: Terminating the Network

We note that the network does not have any global information that would help in the choice of edges in a dynamic way. Also the case of avoiding multiple paths is not understood. Figure 6.11 shows the case where the alternate path is not discarded for the graph of Figure 2.1. The problem is aggravated by the paths being not edge-disjoint and at times having many cycles. Another problem is of obtaining fragmented subgraphs of the solution. It is not known how to connect these in the optimal way. Figure 6.15a shows the graph and in b, the effect of ShPath (weights on nodes) and RemSP (dashed edge). The possible solutions are in 6.15c and that obtained by the network is d. Figure 6.13 shows a plot of the energy with number of iterations. Sometimes we get oscillations in the energy values, although it decreases until net termination as shown in Figure 6.14.

We introduce a post processing stage of MST construction. This removes the cycle but in a greedy fashion. Figure 6.12 shows the solution strategy. However, when the neural output contains forests, an MST to connect these introduces complications (in a greedy sense) and would lead to non-optimal solutions.

Z-nodes	Edges	No. of nodes	optimal	A, B, C, F, P, Q, R
0.4	0.4	10	2	0.5, 8, 2, 2, 700, 700, 1, 30
0.4	0.3	5	4	0.5, 6, 2, 2, 200, 200, 1, 25
mixed ratios		15	1	different values

Table 6.1: Results of the Neural solution to the Steiner Problem in Graphs

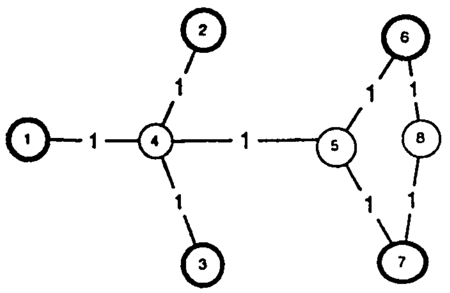


Figure 6.11: Alternate paths in solution

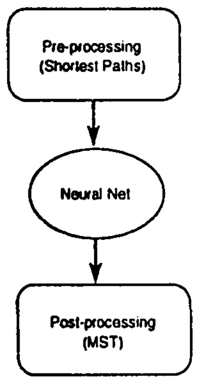


Figure 6.12: Solution Flowchart

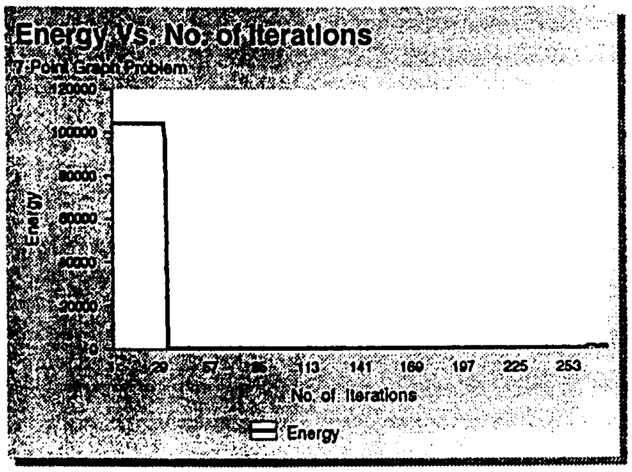


Figure 6.13: Energy against No. of iterations for a 7-point graph problem

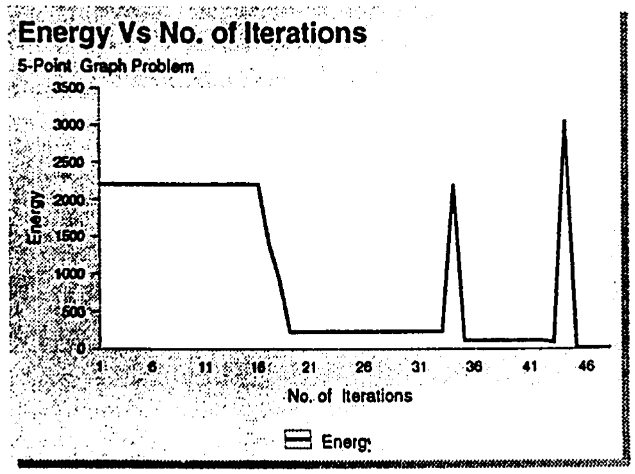


Figure 6.14: Energy against No. of iterations for a 5-point graph problem

123

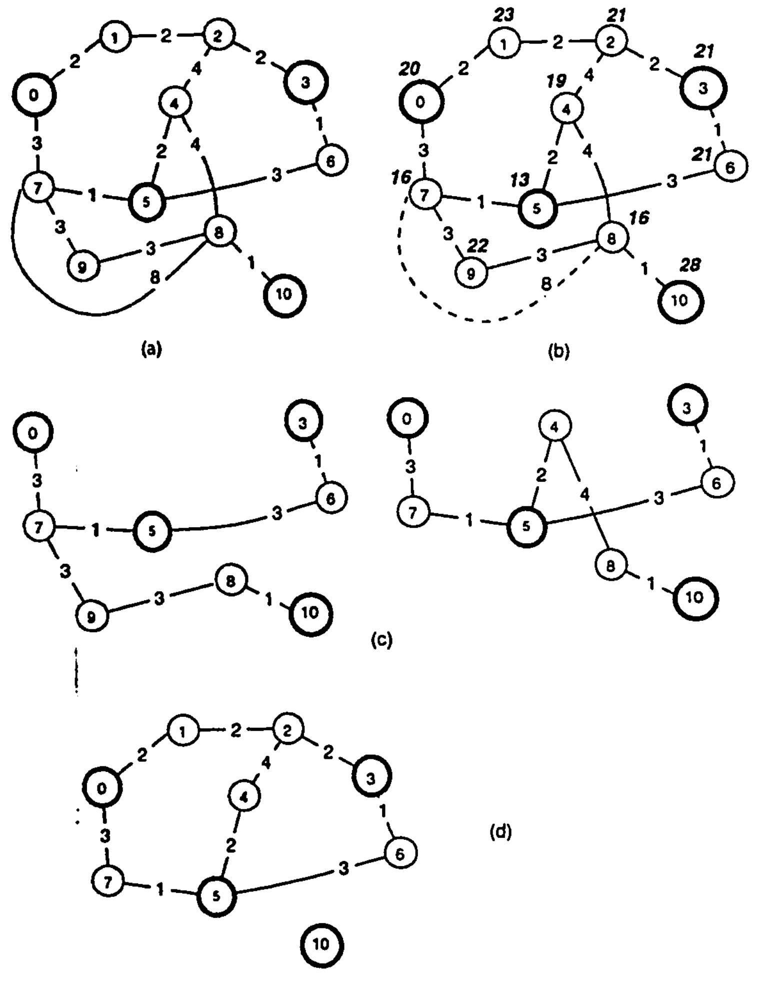


Figure 6.15: (a) The Graph (b) with ShPath and RemSP (c) optimal solutions (d) neural solution

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This last chapter concludes the research with the contribution this work has made. In addition the neural solution and its extensions are discussed.

For the Rectilinear Steiner Tree Problem, this research has shown that by considering only the relation of each point and its closest neighbours, it is possible to determine all the steiner points that would reduce the length to a minimum in a localized area. These closest neighbours compose the Minimum Enclosing Rectangles, which we have defined, and are constructed with respect to each point in the local area. The local area is the region where all such points 'influence' each other and is effectively its cluster. Steiner points exterior to this cluster are determined in a greedy way and only if they reduce the inter-cluster distance between nearest points on the periphery of the nearby clusters. Thus considerably less steiner points are generated than that given by [Han66]. Other methods of [Ric89], [SLL80] and [Hwa79b] do not generate all the necessary points. In addition this work has examined in detail the point connections that are possible between clusters and defined the types of connections.

The clustering is performed using a 'nearness' metric, and the algorithm presented is tested for two different metrics, the Nearest-Two Neighbours and the Nearest-Neighbour. Results for random uniformly distributed points show that the Nearest-Neighbour gives slightly better results. When compared to the work done by previous authors, this algorithm matches the best worst-case time complexity of $O(n \log n)$ with the average ratio of our results better than the results of the similar worst-case time complexity.

A neural network solution to the Steiner Problem in Networks is also presented. The model uses the continuous Hopfield recurrent network for optimization. A representation scheme is formulated, using pre and post processing. The network does not compare favourably with other optimization techniques, and has various limitations.

Recent research of Bhaumik [Bha94] which was brought to our attention by R. Braham, has tackled the Euclidean Steiner Tree problem using the asynchronous update of neurons. His network determines steiner points through an ordering of points that provides the constraints which are mapped onto the energy equation.

This approach is bottom-up, and the solution is determined from all the negated energy states, one that has the maximum energy and hence that will produce the near minimum connecting length. Our neural approach tackles the generalized Steiner Problem in Networks in a top down fashion and has been tested for general random graphs.

7.2 Future Work

This work can be extended in the following ways:

1. A parallel version of the Clustering Algorithm could be derived, particularly with the recent advances of computing the Voronoi Diagram in parallel.

2. The neural formulation could be improved using other information derived from some properties predetermined from special types of graphs.

3. Obtaining neural solutions to the polynomial-time algoritms of the Single Source Shortest Paths and the Minimum Spanning Tree which we use in pre and post processing. These problems are of interest in their own right.

Bibliography

[Bea92] J. Beasely. A heuristic for euclidean and rectilinear steiner problems. *Euro. J. Opl. Res.*, 58:284–292, 1992.

[BG89] M. Bern and R. Graham. The shortest network problem. *Scientific American*, 205, January 1989.

[Bha94] J. B. Bhaumik. A neural network for the steiner minimal tree problem. *Biological Cybernetics*, 70(5):485–494, April 1994.

[BP87] A. Balakrishan and N. Patel. Problem reduction methods and a tree generation algorithm for the steiner network problem in networks. *Networks*, 17(1):65–85, 1987.

[Bur92] L. Burke. A neural design for solution of the maximum independent set problem. *Euro. J. Opl. Res.*, 62(1):186–193, 1992.

[CT76] D. Cheriton and R. Tarjan. Finding minimum spanning trees. *SIAM J. Comp*, 5(4):724–741, December 1976.

123

[Dow91] K. Dowsland. Hill-climbing, simulated annealing and the steiner problem in graphs. *Eng. Optz.*, 17:91–107, 1991.

[DW72] S. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1:195–207, 1972.

[ea93] Hong et al. Performance driven stiener tree algorithms for global routing. *Proc 30th ACM/IEEE Design Automation Conference*, pages 177–181, 1993.

[GGJ77] M. Garey, R. Graham, and D. Johnson. The complexity of computing steiner minimal trees. *SIAM J. Appl. Math*, 32(4):835–859, June 1977.

[GJ77] M. Garey and D. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM J. Appl. Math*, 32(4):826–834, June 1977.

[GP68] E. Gilbert and H. Pollack. Steiner minimal trees. *SIAM J. Appl. Math.*, 16(1):1–29, 1968.

[Hak71] S. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1(2):113–133. 1971.

[Han66] M. Hanan. On steiner's problem with rectilinear distance. *SIAM J. Appl. Math.*, 14(2):255–265, March 1966.

123

[HMS89] J. Hesser, R. Manner, and O. Stucky. Optimization of steiner trees using genetic algorithms. *Proceedings of Genetic Algorithms*, pages 231 236. June 1989.

[Hop82] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. of Sci. USA*, 79(8):2554–2558, April 1982.

[Hop84] J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci.*, 81(10):3088–3092, May 1984.

[HR92] F. Hwang and D. Richards. Steiner tree problems. *Networks*. 22(1):55 89. 1992.

[HS78] E. Horowitz and S.Sahni. *Computer Algorithms Fundamentals*. Computer Science Press, 1978.

[HT85] J. Hopfield and D. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141 152. 1985.

[HVW90] J. Ho, G. Vijayan, and C. Wong. New algorithms for the rectilinear steiner tree problem. *IEEE Tansactions on CAD*, 9(2):185–193, February 1990.

[Hwa76] F. K. Hwang. On the steiner minimal trees with rectilinear distance. *SIAM J. Appl. Math.*, 30(1):104–114, January 1976.

[Hwa79a] F. K. Hwang. An o(n log n) algorithm for rectilinear minimal spanning trees. *J. ACM*, 26(2):177–182, April 1979.

[Hwa79b] F. K. Hwang. An o(n log n) algorithm for suboptimal rectilinear steiner trees. *IEEE Trans. Circs. & Sys.*, VCAS- 26(1):75–77, January 1979.

[Jag92] A. Jagota. Efficiently approximatting max-clique in hopfield-style network. *Proceedings of IJCNN*, 2:248–253, 1992.

[Kah91] A. Kahng. A steiner tree construction for vlsi routing. *Proceedings of IJCNN, Seattle*, 1:133–139, 1991.

[KR92a] A. Kahng and G. Robbins. A new class of iterative steiner tree heuristics with good performance. *IEEE Trans. on CAD*. 11(7):893 901. July 1992.

[KR92b] A. Kahng and G. Robbins. On performance bounds for a class of rectilinear steiner tree heuristics in arbitrary dimensions. *IEEE Trans. on CAD*, 11(11):1462–1465, November 1992.

[KRSS93] A. Kapsalis. V. Rayward-Smith, and G. Smith. Solving the graphical steiner tree problem using genetic algorithms. *J. Opl Res. Soc.* 44(4):397 406, 1993.

[KS85] J. Komlos and T. Shing. Probabilistic partitioning algorithms for the rectilinear steiner problem. *Networks*, 15(4):413–423, 1985.

123

[LB80] D.T. Lee and B.J.Schachter. Two algorithms for constructing a delaunay triangulation. *Intl. J. of Computer and Inform. Sci.*, 9(3):219 242, 1980.

[LBH76] J. Lee, N. Bose, and F. Hwang. Use of suboptimal routing in rectilinear metric. *IEEE Trans on Circuits & Systems*, 23(7):470–476, July 1976.

[LCW93] A. Lim, S. Chen, and C. Wu. Performance oriented rectilinear steiner trees. *Proc. 30th ACM/IEEE Design Automation Conference*, pages 171–176, 1993.

[Lin93] F. Lin. A parallel computation network for the maximum clique problem. *IEEE Symposium on Circuits & Systems*, 3:2549–2552, 1993.

[LL89] R. Libeskind and C. Liu. Solutions to the module orientation and rotation problems by neural computation networks. *Proc. 26th ACM/IEEE Design Automation Conference*, pages 400–405, 1989.

[Loo92] C. Looi. Neural network methods in combinatorial optimization. *Computers Oper. Res.*, 19(3):191–208, 1992.

[Mel61] Z. A. Melzak. On the problem of steiner. *Canadian Math. Bull.*, 32(4):143–148, May 1961.

[Mur83] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4), 1983.

[MW84] T. Magnanti and R. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1 55. February 1984.

[OG91] L. Osborne and B. Gillett. A comparison of two simulated annealing algorithms applied to the directed steiner problem on networks. *ORSA Journal on Comput.*, 3(3):213–225, 1991.

[PS85] F. Perparata and M.I. Shamos. *Computational Geometry : An Introduction.* Springer Verlag, 1985.

[Ric89] D. Richards. Fast heuristics for rectilinear steiner trees. *Algorithmica*, 4:191–207, 1989.

[RP92] D. Rao and L. Patnaik. Circuit layout through an analogy with neural networks. *Computer-Aided Design*, 24(5):251–257, 1992.

[RS88] J. Ramanujam and P. Sadayappan. Optimization by neural networks. *Proceedings of IJCNN*, 2:325–332, 1988.

[Sal92] J. Salowe. A simple proof of the planar rectilinear steiner ratio. *Operations Research Letters*, 12(4):271–274, 1992.

[SCF91] P. Shin, K. Cheng, and W. Feng. Neural computation network for global routing. *CAD*, 23(8):539–547, October 1991.

[SF91] P. Shih and W. Feng. A neural network approach to channel routing. *IEEE Intl. Symposium on Circuits & Systems*, 2:1593 1596, 1991.

[SFG82] M. Shore, L. R. Foulds, and R. Gibbons. An algorithm for the steiner problem in graphs. *Networks*, 12(3):322–333, 1982.

[SGY92] T. Sen, J. Gan, and L. Yao. Application of fuzzy neural computing for partitioning circuits. *Proc. IEEE Custom Integrated Circuits Conference*, pages 5.3.1–5.3.4, 1992.

[SLL80] J. Smith, D.T. Lee, and J. Liebman. An o(nlog n) heuristic algorithm for the rectilinear steiner minimal tree problem. *Eng. Optz.*, 4(4):179-192, 1980.

[SW92] M. Sarrafzadeh and C. Wong. Hierarchical steiner tree construction in uniform orientations. *IEEE Trans. on CAD*, 11(9):1095–1103, September 1992.

[Syn91] T. Synder. Lower bounds for rectilinear steiner trees in bounded space. *Inform. Process. Lett.*, 37(2):71 74, January 1991.

[Tak92] Y. Takefuji. *Neural Network Parallel Computing*. Kluwer Academic Publ.. 1992.

[TCP91] G. Tagliarini, J. Christ, and E. Page. Optimization using neural networks. *IEEE Trans. on Computers*, 40(12):1347 1358. December 1991.

[TP87] G. Tagilarini and E. Page. Solving constraint satisfaction problems with neural networks. *Proceedings of IJCNN*, 3:741–747, 1987.

[Win87] P. Winter. Steiner problem in networks: A survey. *Networks*, 17(2):129–167, 1987.

[YM89] J. Yin and P. Mazumder. A neural network design for circuit partitioning. *Proceedings 26th ACM/IEEE Design Automation Conference*, 1(26.2):406–411, 1989.

Vita

- Born in Mombasa, Kenya.
- Obtained B.S in Computer Engineering from Middle East Technical University, Ankara, Turkey -1991.
- Worked at Galaksy Bilgisayar, Ankara as System Analyst- 1991
- Obtained M.S in Computer Science from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia -1994.