An Enhanced Convex-Elastic Net Technique for Solving E-TSP

by

Tareq Al-Sayed Ahmad Muhammad Al-Maghrabi

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

March, 1996

AN ENHANCED CONVEX-ELASTIC NET TECHNIQUE FOR SOLVING E-TSP

BY

Tareq Al-Sayed Ahmad Muhammad Al-Maghrabi

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

March 1996

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN 31261, SAUDI ARABIA

COLLEGE OF GRADUATE STUDIES

This thesis, written by **Tareq Al-Sayed Ahmad Muhammad Al-Maghrabi** under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of the College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE.

Thesis Committee:

Dr. Muhammed Al-Mulhem (Thesis Advisor)

Dr. Mostapha Aref (Member)

Dr. Rafik Braham (Member)

Dr. Muhammed Al-Suwaiyel (Member)

Department Chairman

Dean, College of Graduate Studies

10.4.96

Date

Dedicated to My Parents

ACKNOWLEDGMENT

Acknowledgment is due to King Fahd University of Petroleum and Minerals for support of this research.

I wish to express my deep appreciation to Dr. Muhammed Al-Mulhem who served as my thesis advisor, and whose help, direction and encouragement made this work an enjoyable experience and a possible achievement. Acknowledgment is also due to other members of my thesis committee Dr. Mostapha Aref, Dr. Rafik Braham, and Dr. Muhammed Al-Suwaiyel.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

Name : Tareq Al-Sayed Ahmad Muhammad Al-Maghrabi
Title : An Enhanced Convex-Elastic Net Technique for Solving E-TSP
Major Field : Computer Science
Date Of Degree: March, 1996

The Euclidean Traveling Salesman Problem (E-TSP) is one of the important NP-complete problems that has many applications in real life. In this thesis, we propose an *Enhanced Convex-Elastic Net* technique that combines an adaptive-type neural network model and an iterative algorithm for solving E-TSP. The main advantage of this technique is that it can be implemented in parallel architecture which is potentially faster than sequential architecture. The simulation results performed have shown that the proposed technique yields a sub-optimal solution that compares favorably with several known techniques such as *Guilty Net, Hopfield and Tank Model, Elastic Net,* and *Simulated Annealing.*

Master Of Science Degree
King Fahd University Of Petroleum And Minerals
Dhahran, Saudi Arabia
March, 1996

بسم الله الرحمن الرحيم

خلاصة الرسالة

الإسم : طارق بن السيد أحمد بن محمد المغربي.
عنوان الرسالة : الشبكة المرنة ذات التحدب المغلق لحل مشكلة مندوب المبيعات في البعد الثنائي.
التخصص : علوم الحاسب الآلي.
تاريخ الشهادة : شوال ١٤١٦ هـ.

تعتبر مشكلة مندوب المبيعات المتجول في سطح مستوي من المشاكل التي يصعب حلها بالرغم من التطبيقات العديدة لهذه المشكلة في الحياة العملية. وفي هذه الاطروحة نقدم طريقة الشبكة المرنة ذات التحدب المغلق لحل هذه المشكلة. وتعتمد هذه الطريقة الجديدة على استخدام نوع من الشبكات المرنة بالإضافة إلى اسلوب تقليدي لإيجاد أقصر الطرق لحل المشكلة.

ويمكن تطبيق الطريقة الجديدة بطريقة متزامنة على أجهزة الحاسب الآلي مما يعطي هذه الطريقة سرعة عالية جدا مقارنة بالطرق التقليدية. كما تشير نتائج هذه الدراسة بأن الحل الذي يتم إيجاده بالطريقة الجديدة ينافس العديد من الحلول التي يمكن إيجادها بالطرق الأخرى.

درجة الماجستير في العلوم
جامعة الملك فهد للبترول والمعادن
الظهران - المملكة العربية السعودية
شوال ١٤١٦ هـ

CHAPTER 1

INTRODUCTION

1.1 Statement of the Problem

The Traveling Salesman Problem is a topic of current concern, but it is not a new topic. Nearly six decades ago, the TSP appeared and was discussed informally among mathematicians at mathematics meetings. In 1930, the Viennese mathematician Karl Menger made the first statement of TSP. In 1934, Merrill Flood defined the customary TSP statement as the closed minimum tour that a salesman must go through [30].

The Traveling Salesman Problem may be defined as follows. Let G = (V, A) be a graph where V is a set of vertices and A is a set of arcs between vertices with non negative costs. The TSP consists of finding a tour that passes through every vertex exactly once and is of minimum length [43]. The Euclidean Traveling Salesman Problem is a special case of the TSP with triangular inequality.

The Euclidean Traveling Salesman Problem (E-TSP) is to find a closed tour of minimum length through points that are given in the two-dimensional space where the distances between cities are computed according to the Euclidean metric $\sqrt{(X_1-X_2)^2+(Y_1-Y_2)^2}$ for two cities (X_1, Y_1) and (X_2, Y_2). E-TSP can be defined formally as follows. Let G =(V, A) be a graph where **V** is a set of **N** points in the plane, **A** is a set of edges between these points, and the Euclidean distance for each edge between point **i** and **j** is C_{ij}. The E-TSP consists of determining a tour of minimum length passing through each point once and only once.[44]

1.2 Thesis Objective and Scope of Work

The Traveling Salesman Problem (TSP) is an NP-complete optimization problem for which there is no polynomial time algorithm exist and the only known solution requires an amount of time that grows exponentially with the problem size. In addition, traditional techniques for solving optimization problems fail to provide a satisfactory solution for large-scale problems. For these reasons, many researchers are motivated to explore the massive parallelism and robustness of neural networks to solve optimization problems in order to reduce their large search space.

There has been some interest in recent years in using other approaches with neural networks as a hybrid technique for solving optimization problems [49,60,61]. Motivations for this include improving performance of the model and pushing the model toward feasible solutions; since neural networks may not guarantee feasibility of solutions generated [1,37]. In this thesis, this idea is considered for the Euclidean Traveling Salesman Problem (E-TSP). The proposed approach combines a modified version of a known technique called *Elastic Net* and an iterative improvement approach to generate an efficient algorithm for the Euclidean Traveling Salesman Problem.

1.3 Motivations and TSP Applications

Both the TSP and the E-TSP are NP-complete problems [34]. In an instance with N cities, there are $(N!/2N)$ distinct tours [39]. It is unlikely that we can find a polynomial time algorithm for solving this problem exactly. Finding an efficient algorithm for the E-TSP that approaches the global minimum solution is a challenging task, since E-TSP contains several local minimums that can mislead most algorithms [31].

The TSP is an important problem because it represents a wide class of NP-complete problems and many real-world permutation problems can be formulated as instances of it. Practical applications for the TSP include, among many others, the following: [44]

1) *Vehicle Routing Problems (VRPs)*: VRP is a distribution problem in which vehicles are required to visit a given geographically scattered customers in order to fulfill the customers needs in a given time. Examples for VRPs include collection of mail from mail boxes, coins from telephone boxes, tour all houses by a doctor, preventive maintenance inspection tours, scheduling of airline crews, and movement of a robot arm.

2) *Board Wiring*: In computers and digital systems, the pins are located at some positions where these pins have to be interconnected by wires such that each pin is attached to two wires and the total length for wiring is minimized. This formulation is similar to the Euclidean Traveling Salesman Problem.

3) *Cutting Wallpaper*: Given a single large roll of paper with a pattern that repeats each unit interval, we need to cut **N** sheets of paper (based on orders), such that the total waste is minimized. Sheet **i** has a starting point for the pattern S_i and ending point F_i where $0 \leq S_i$ and $F_i \leq 1$. Therefore, the wasted paper when sheet **j** is cut from the roll immediately after sheet i is $C_{ij} = (S_j - F_i)$ (mode 1). The cutting wallpaper problem can be defined as **N+1** city TSP, when a dummy sheet, **0,** is used with the same starting and finishing points S_0 and F_0.

4) *Hole Punching*: The problem of hole punching is a well-known problem that occurs in printed boards and metallic sheets manufacturing [46]. In this context, the drilling path used can significantly affect the cost of the

production. Hole punching is a TSP where the path that visits all nodes once and of shortest length should be found.

5) *Job Sequencing*: In this problem, N jobs are to be performed sequentially on a single machine such that the total time is reduced. Let the time needed to execute job j immediately after job i be C_{ij}. This problem can be formulated as a TSP if a dummy job is introduced.

1.4 Approaches to Solve the TSP

The Traveling Salesman Problem is a classical combinatorial optimization problem, which is simple to state but difficult to solve. Strategies to solve the TSP can be grouped into the following three categories: [47]

1) *Heuristic Algorithms*: Several heuristics are used to achieve near optimal solutions, for example *Nearest Neighbor heuristic* [3], *Lin Kernighan Algorithm* [35,58], and *Christofides' Approximation Algorithm* that constructs a tour of length no more than 50% more than the optimal tour length [44].

2) *Statistical Algorithms*: Statistical mechanics techniques can solve the TSP approximately. For example, the *Simulated Annealing* algorithm has been applied to solve the TSP with satisfactory solution [40], but it requires a lot of computation time [16] and parameters setting of the energy function used for each problem size [32].

3) *Neural Networks*: Several neural network models have been used to solve the TSP as an inherently parallel heuristic [4,5,38].

1.5 The Proposed Approach to Solve the E-TSP

In this thesis, we introduce a new hybrid technique for solving the Euclidean Traveling Salesman Problem. Its aim is to show the potential for integrating operation research and neural networks. The neural networks approach is used for the Euclidean TSP, because it is more suitable for problems that have an underlying geometric structure [24].

The technique for solving E-TSP is developed using two phases as shown in Figure 1. In the first phase, the *Convex-Elastic Net* is used as a global search technique that generates an initial tour for the second phase by enforcing the E-TSP constraints instead of transforming them into penalties. In the second phase, the tour found is tuned further and iteratively improved by making local changes to it. The two introduced techniques are combined into a new approach called the *Enhanced Convex-Elastic Net* (ECEN).

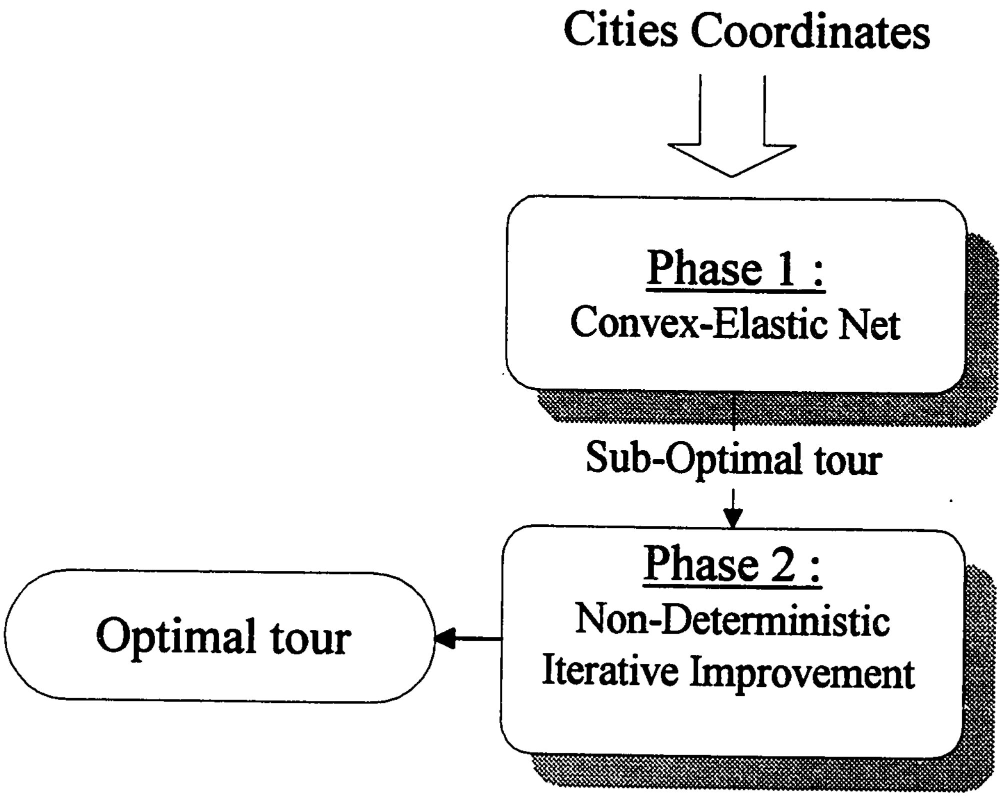


FIGURE 1: Block diagram for the enhanced convex-elastic net approach

1.6 Thesis Organization

This thesis includes five chapters. In Chapter 2, a comprehensive review of the various ways of mapping the Traveling Salesman Problem onto the neural network models is given. Then, the proposed *Enhanced Convex-Elastic Net* (ECEN) technique for solving E-TSP is detailed in Chapter 3. In Chapter 4, experimental results are given to demonstrate that the proposed algorithm can get quite close to the optimal solution of the E-TSP. Finally, Chapter 5 includes the summary, conclusions and recommendation for future work.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Many of the optimization problems which are required to be solved in practice are NP-complete (e.g. Traveling Salesman [23], Steiner Minimal Tree [33] and Bin Packing problems). Finding an efficient algorithm for solving such problems in polynomial time for all input instances is almost impossible. With the motivation of modern neural techniques, several neural network models have been proposed for solving complicated optimization problems like the Traveling Salesman Problem successfully [11,19,49,65]. In this chapter, some neural-based techniques for solving TSP are reviewed.

There are three basic approaches of using neural networks for solving the TSP. First, the neural network can be non-adaptive by deriving it from the TSP statement without any ability to learn from the presented instances of the problem (e.g. *Hopfield and Tank Model* [62]). Second, the neural network can be adaptive so that neurons compete to become active under certain conditions (e.g. *Guilty Net*, and *Elastic Net*). Third, the neural network can be designed using another traditional technique such as a hybrid approach (e.g. *Hopfield and Tank Model with Genetic Breeding for Control Parameters*).

This chapter is organized as follows. In Section 2, we describe the *Hopfield and Tank* non-adaptive approach for solving the TSP. In Section 3, we present two adaptive approaches for solving the TSP namely *Elastic Net*, and *Guilty Net*. In Section 4, several hybrid approaches for solving TSP based on neural networks are presented.

2.2 Non-Adaptive Approaches for solving TSP

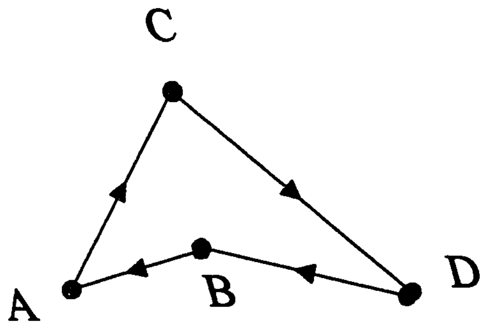
In the non-adaptive approach, the neural network is not trained on the TSP instances but derived from them. In this section, the *Hopfield and Tank* non-adaptive model is discussed and some enhancements to this model are presented.

2.2.1 The Hopfield and Tank Model

Hopfield and Tank were the first to suggest a neural network approach to solve combinatorial optimization problems, like TSP, in 1985 [55]. The *Hopfield Neural Network* is used as a minimization machine for an energy function that represents a combinatorial optimization problem. Hopfield has shown that for a network with symmetric connections and non-negative elements on the diagonal of the matrix, the network will always converge by performing gradient descent to a stable state which is a local minimum of the energy function.

Hopfield and Tank Model for solving the TSP can be described as follows [2,47]. A matrix of neurons is used to represent the solution for the problem. Therefore, solving the TSP requires $O(N^2)$ neurons and $O(N^4)$ connections between them [6]. In this representation, the neuron at the **x**-th row and the **i**-th column is 1 only when city **x** is visited at the **i**-th position on the tour; otherwise this neuron gets the value 0. In this way, the final configuration of the network can be interpreted as a solution to the Traveling Salesman Problem as shown in Figure 2 [26].

Each pair of neurons in the matrix of neurons are connected by a link, and a weight is associated with each link. These weights between neurons are derived from the distances between the cities to choose tours of shorter lengths as well as to prevent infeasible tours.



Position on the tour

City	1	2	3	4
A	1	0	0	0
B	0	0	0	1
C	0	1	0	0
D	0	0	1	0

FIGURE 2: Neural network representation for TSP tour

When neuron **i** is activated, a signal is sent along the connection from neuron **i** to each other neuron **j** with weight T_{ij}. If the weight between a pair of neurons is negative (inhibition), both units will have an inhibitory connector. Therefore, both neurons cannot be active at the same time. However, if two neurons are connected by a positive weight (excitation), then both neurons can be active at the same time. After that, the energy function for the TSP (E_{TSP}) is formulated such that its minimum corresponds to the optimal solution [38].

$$E_{TSP} = \frac{A}{2}\sum_{x}^{N}\sum_{i}^{N}\sum_{j\neq i}^{N} V_{xi} * V_{xj} + \frac{B}{2}\sum_{x}^{N}\sum_{i}^{N}\sum_{y\neq x}^{N} V_{xi} * V_{yi} + \frac{C}{2}\left(\sum_{x}^{N}\sum_{i}^{N} V_{xi} - N\right)^2 + \frac{D}{2}\sum_{x}^{N}\sum_{i}^{N}\sum_{y\neq x}^{N} d_{xy} * V_{xi}(V_{y,i+1} + V_{y,i-1}) \quad (1)$$

The first three terms in the TSP energy function represent the following constraints:

1) Each city has no more than one position on the tour.
2) No position in the tour is assigned to more than one city.
3) Exactly **N** entries in the **V** array are of length **N**.

The last term in the TSP energy function represents the length of the tour to be minimized. When all the constraints are satisfied, the first three terms in the energy function will be zero and the optimal tour is found when the last term is minimized where the parameters **A**, **B**, **C**, and **D** in equation (1) are used to adjust the relative weights of the different parts of the energy function. Next, the connection weights are derived by comparing the TSP energy function with the following general Hopfield energy function.

$$E = -\frac{1}{2}\sum_{x=1}^{N}\sum_{i=1}^{N}\sum_{y=1}^{N}\sum_{j=1}^{N} T_{xiyj} * V_{xi}V_{yj} - \sum_{x=1}^{N}\sum_{i=1}^{N} V_{xi} * I_{xi} \quad (2)$$

Therefore, the connection weights and the external input (bias) are given by the following equation.

$$T_{xiyj} = -A\alpha_{xy}(1-\alpha_{ij}) - B\alpha_{ij}(1-\alpha_{xy}) - C - D\, d_{xy}(\alpha_{j,i+1} + \alpha_{j,i-1})$$
$$I_{xi} = C * Ne \quad ; \; Ne > N \; ; \qquad \alpha_{ij} = \begin{cases} 1 & ; \text{if } i = j \\ 0 & ; \text{otherwise} \end{cases} \tag{3}$$

After building the network, each neuron is initialized to **1/N** ± small random perturbation and updated randomly to minimize the energy function. Each neuron is updated over time using the following three equations of motion.

$$U_{xi}^{t+1} = U_{xi}^{t} + \delta_t \frac{dU_{xi}^{t}}{dt} \tag{4}$$
$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - A\sum_{j\neq i} V_{xj} - B\sum_{y\neq x} V_{yi} - C\Big(\sum_{x}\sum_{j} V_{xj} - N\Big) - D\sum_{y\neq x} d_{xy}(V_{y,i+1} + V_{y,i-1}) \tag{5}$$
$$V_{xi} = g(U_{xi}) \frac{1}{2}\Big[1 + \tanh\Big(\frac{U_{xi}}{U_o}\Big)\Big]. \tag{6}$$

The U_o is a constant, δ_t is a small fraction of τ, and U_{xi} is the input to neuron **xi**. The states of neurons are updated as follows: If the activation value for a neuron is greater than its threshold, then the neuron will be set to **1**. However, if the activation value for a neuron is less than its threshold, then the neuron will be set to **0**. Otherwise, the neuron will not change its state.

The *Hopfield and Tank Model* for solving TSP can be described by the following algorithm: [20]

1. Initialize activation of all units, U_{xi}.

```
2. Initialize time interval dt, and the constant $U_0$ to small values.
3. Repeat
     FOR i = 1 to $N^2$ DO
     {  Select a neuron, $U_{xi}$, at random
        Update the activation for the selected neuron (Equation 5):
          $U_{xi}^{New} = U_{xi}^{Old} + dt\left[-U_{xi}^{Old} - A\sum_{j \neq i} V_{xj} - B\sum_{y \neq x} V_{yi} - C\left(\sum_{x}\sum_{j} V_{xj} - N\right) - D\sum_{y \neq x} d_{x,y}(V_{y,i+1} + V_{y,i-1})\right]$
        Apply the output function (Equation 6):
          $V_{xi} = \frac{1}{2}\left[1 + \tanh\left(\frac{U_{xi}}{U_0}\right)\right]$
     }
   Until the activation $U_{xi}$ for all neurons are stable.
```

It is found that for 10 city TSP 80% are feasible solutions, half of them are optimal tours [64]. One problem with the *Hopfield and Tank Model* is that only problems with up to 30 cities, which is a very small problem size, could be solved [53]. Also, the success rate for finding feasible solutions scales poorly as the size of the TSP increases [31].

2.2.2 Variations of the Hopfield and Tank Model

There are several problems with the *Hopfield and Tank Model*. First, this model has the disadvantage that the generation of a new solution requires many iterations, and often configurations of infeasible solutions have to be visited in between. Second, the TSP is not solved in space of $O(N!)$ but in $O(2^{N^2})$ where many configurations are infeasible solutions. For this reason, the *Hopfeild and*

Tank Model produces illegal tours most of the time [51]. Another problem with the *Hopfield and Tank Model* is that it is heavily dependent on the choice of parameters and there is no systematic way to determine the coefficients for the TSP energy function [1,17,57]. Also, only a local minimum is found and the *Hopfield and Tank Model* is sensitive to the initial state.

These difficulties in the *Hopfield and Tank Model* have led the researchers to try overcoming these problems by introducing several modifications to the model. These enhancements to the *Hopfield and Tank Model* can be classified into five main categories:

1) Using a different representation for the TSP.
2) Using a modified energy function.
3) Estimating good control parameters for the energy function.
4) Starting from a different initial configuration.
5) Using a different method for updating neurons.

In 1990, A. Joppe, H. Cardon, and J. Bioch proposed a modification to the *Hopfield and Tank Model* by representing the TSP based on the adjacency of the cities in the tour rather than the positioning of the cities [36]. This representation for the TSP simplifies the energy function and decreases the convergence time of the network to a feasible solution. Another modification to the *Hopfield Tank Model* that ensures feasibility of solutions is proposed by L. Carralho and V. Barbosa [10], where a threshold is used in representing the TSP, instead of synaptic weights only.

Harolid Szu has improved the energy function of the *Hopfield and Tank Model* for solving the TSP. This modification is performed by replacing the third term in the energy function of the *Hopfield and Tank Model* (Equation 1) by the following term.

$$\frac{C}{2}\left\{\sum_{x}\left[1-\sum_{i}V_{xi}\right]^{2}+\sum_{i}\left[1-\sum_{x}V_{xi}\right]^{2}\right\} \qquad (7)$$

The new term represents that "Exactly one unit should be on in each raw and each column". Also, Szu updates neurons in his network simultaneously unlike the *Hopfield and Tank Model.* [20]

In [37], Kamgar-Parsi and Kamgar-Parsi demonstrate a relationship between the control parameters for the *Hopfield and Tank Model* that ensures the stability of valid solutions. Also, this technique can test the given control parameters for dynamic stability by using a set of inequalities between the control parameters. Another way to select the control parameters for the *Hopfield and Tank Model* is based on the theory of Variable Structure System (VSS) [48]. The main advantage of using the theory of VSS is that it will help the *Hopfield and Tank Model* in escaping local minimums in the solution.

Different starting configurations were used to improve the convergence speed for the *Hopfield and Tank Model.* For example, heuristic seeding technique where cities that are close in distance are forced to be adjacent in the tour can be used. Another initial configuration that is suggested by S. Hedge and W. Levy is to initialize all activation values of neurons to 0.5 [55].

In 1994, F. Shiratani and K. Yamamoto proposed a new method for updating the neurons in the *Hopfield and Tank Model* namely the *Block Sequential Mode* [59]. The simulation results showed that for 30 cities TSP the *Block Sequential Mode* gave better tours than the synchronous or asynchronous model. Various other modifications of the *Hopfield and Tank Model* have also been explored; see e.g., [13,55,63].

2.3 Adaptive Approaches for Solving TSP

Another group of neural networks to solve the TSP use the adaptive structure. Adaptive neural networks are those models that can adjust their connections weights. In the following subsections, two adaptive neural networks are presented, namely *Elastic Net* and *Guilty Net*. Both of these adaptive neural networks form a tour of the cities in response to the problem data.

2.3.1 Elastic Net

In 1987, Durbin and Willshaw developed an *Elastic Net* algorithm for the TSP independent of the parameters [18]. The main difference between the *Hopfield and Tank Model* and the *Elastic Net* is that the *Elastic Net* enforces the TSP constraints, while the *Hopfield and Tank Model* converts these constraints into penalties [55]. As a result, the *Elastic Net* produces shorter tours than the *Hopfield and Tank Model*, and it scales well with the problem size [37]. The *Elastic Net* is a geometric technique for solving TSP in which the initial path is gradually updated to produce the final tour. For this reason, the *Elastic Net* cannot be used for solving TSP with non-Euclidean distances. In this net, there are **m** points laying on an imaginary ring or rubber band located at the center of the cities, where **m** is greater than the number of cities, **N**. Each city is mapped to a point in the ring, where neighboring points on the ring are mapped as close as possible on the plane. Points move to minimize distances between the cities and the points as well as the ring length. This can be achieved through minimizing the following energy function.

$$E(\{Y_j\},K) = -A * K\sum_i \log\sum_j e^{-|X_i - Y_j|/(2K^2)} + B\sum_j \{Y_j - Y_{j+1}\}^2 \tag{8}$$

Let $\mathbf{X_i}$ be the position of a city, $\mathbf{Y_j}$ be the point on the path, and let the parameters **A** and **B** determine the relative strength of the forces from the points on the ring and from the cities. Then, during the points movements process, two forces are applied: one for minimizing the length of the ring and the other for minimizing the distances between the cities and the points on the rubber band (See Figure 3). As a result, the points on the imaginary ring are updated at each iteration as follows.

$$\Delta Y_j = -A\sum_i W_{ij}(X_i - Y_j) + B * K(Y_{j+1} + Y_{j-1} - 2Y_j)$$

$$\text{where } W_{ij} = \frac{e^{-|X_i - Y_j|^2/(2K^2)}}{\sum_k e^{-|X_i - Y_k|^2/(2K^2)}} \tag{9}$$

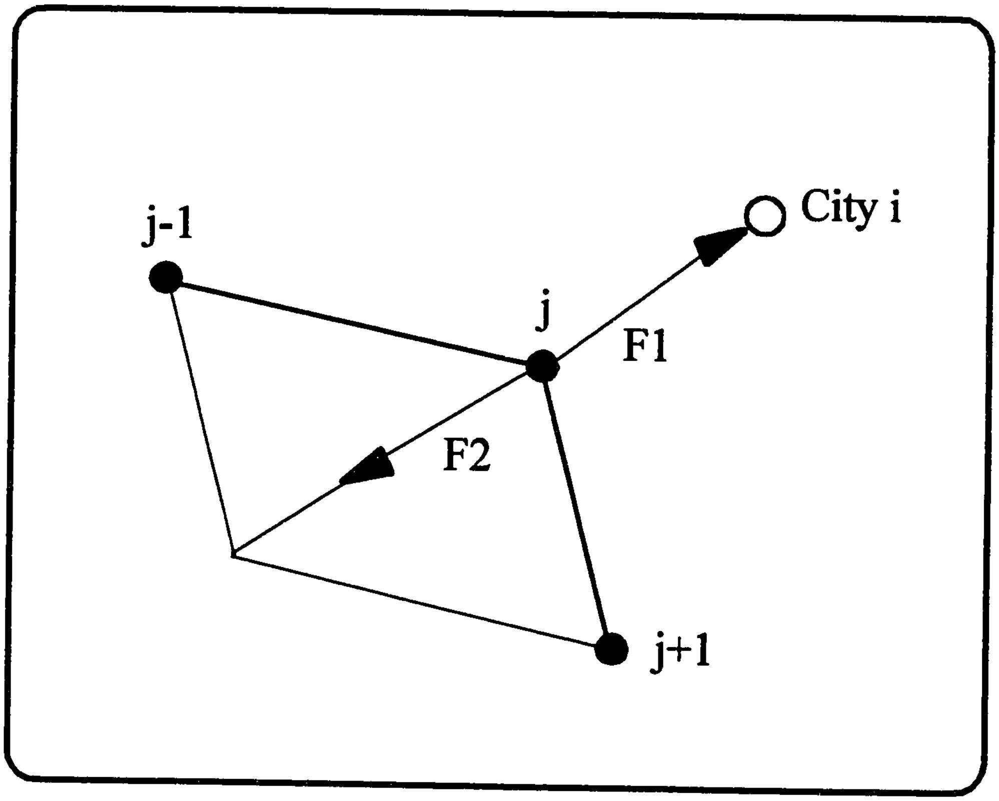


FIGURE 3: Forces on the ring point j for the elastic net

In equation (9), W_{ij} measures the influence of city i on point j and K is a scale parameter that gradually reduces to minimize the energy function. The first term in the updating function, equation (9), drives the points on the ring towards the cities, so that for each city i there is at least one ring point j within distance K. The second term, on the other hand, keeps neighboring points together to produce a shorter tour. The success of the *Elastic Net* model may be due to the way it imposes its constraints as shown in Figure 4.

The number of connections required by the *Elastic Net* is of order N^2 for N cities E-TSP which is less than N^4 connections required by the *Hopfield and Tank Model.* The simulations results conducted by Durbin and Willshaw showed that for 30 cities generated randomly within a unit square, the *Elastic Net* generates tours shorter than the *Hopfield and Tank Model* by 19%. Also, for 50 cities the *Elastic Net* can find a tour of length 6.02 which is longer than the tour found by *Simulated Annealing* by 1% only.[18]

After the success of the Durbin and Willshaw *Elastic Net* for E-TSP, several modifications to that model have been developed. For example, M. Boerest and L. Carralho proposed a filtering mechanism for the *Elastic Net,* called the γ-filter for γ in $\{0,1\}$, that displaces points as a result of some cities which exert a significant influence on them. This technique reduces the running time of the *Elastic Net* without reducing the quality of the solutions obtained.[5]

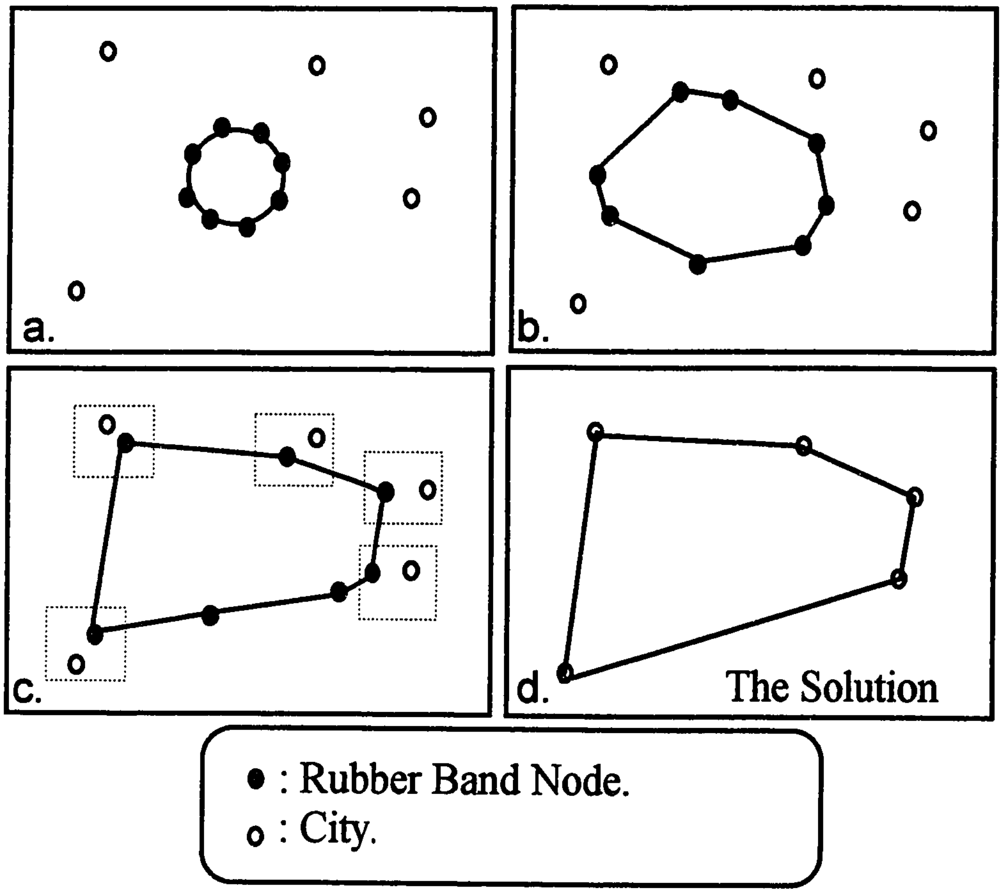


FIGURE 4: Evolution of the elastic net over time (a, b, c, and d)

Another modification to the *Elastic Net* is introduced by J. Platt and A. Barr in 1987, which is called the *Basic Differential Multiplier Method* (BDMM) [54]. The BDMM can satisfy all the TSP constraints exactly by creating forces that apply the constraints over time using a neural network that estimates Lagrange multipliers. This neural network is used where neurons represent positions on the plane and the rubber band minimizes its length, which is given by the following equation.

$$\sum_{i}^{N}(X_{i+1}-X_{i})^2-(Y_{i+1}-Y_{i})^2 \tag{10}$$

The rubber band length, equation (10), is subject to the constraint that the rubber band points must lie on the cities as given by the following equation.

$$k(X^{*}-X_{C})=0 \text{ and } k(Y^{*}-Y_{C})=0 \tag{11}$$

The pair (X^{*},Y^{*}) represents the cities coordinates, (X_{C},Y_{C}) represents the closest point to the city, and **k** is a strength parameter. For 120 city TSP, the J. Platt and A. Barr technique generates a tour length that is 4-8% longer than that yielded by *Simulated Annealing* [55].

2.3.2 Guilty Net

The *Guilty Net*, which was proposed by L. Burke and P. Damany in 1992, is a competitive learning system [8,9]. In this technique, coordinates of cities are presented to the network successively to initiate the competition phase, where the most appropriate group for a city is chosen to adjust its weight. The *Guilty Net* technique can be described by the following steps:

Step 1. Start with a random order of cities, and let the number of winnings for each city, Win(i), equal zero.

Step 2. A city X_m is selected and presented to the network. After that, the competition phase takes place where the winning node moves towards the city it claims. The winning neuron is selected based on the following equation.

$$Net(j) = |X_m - W_j| + K \frac{Win(j)}{1 + \sum_{kk} Win(kk)} \tag{12}$$

The first term in equation (12) represents the Euclidean distance between the weight and the presented city coordinate X_m. The second term is used so that neurons that are selected previously will have less probability for winning again.

Step 3. The weight for the winning neuron i and its neighbors will be updated using the following equation.

$$W_{kj}^{new} = W_{kj}^{old} + \alpha\, F(d_{ij})(X_{km} - W_{kj}^{old}) \quad ; \text{for node } j \tag{13}$$

where $1 \le k \le d$, $1 \le j \le N$, and learning rate α

The neighborhood function, $F(d_{ij})$, that decreases when the distance between i and j increases, is given by the following equation.

$$F(d_{ij}) = \begin{cases} 1 - \dfrac{d_{ij}}{N/2} & ;\ \text{when } d_{ij} < N/2 \\ 0 & ;\ \text{otherwise} \end{cases} \qquad (14)$$

$$d_{ij} = \min\{|i-j|, N-|i-j|\}$$

Step 4. Increment m by 1.

IF m < N+1 THEN go to step 2.

IF m = N+1 THEN let B = 1.1*B and m = 1.

Step 5. If all nodes are within a very small distance from cities they claim, then the network has converged. Otherwise, go to step 2.

An important feature of the *Guilty Net* is that, if processing must be terminated prematurely, the system can provide a feasible sub-optimal solution. Also, the *Guilty Net* has a simple neural structure where a fixed number of output nodes that equals the number of cities are used. The main difficulty of this model is that setting learning rates parameters depends on the problem size and the distribution of cities. Simulation results have shown that the *Guilty Net* produces tours that exceed the theoretical optimal tour length by 14% for 10-100 cities [8].

2.4 Hybrid Approaches for Solving TSP

There has been some interest in recent years in using other approaches with neural networks as a hybrid technique for solving optimization problems. In the following subsections three hybrid approaches for solving TSP are discussed, namely *Hopfield and Tank Model with Genetic Breeding for Control Parameters*, *a Neural Network with Coordinate Newton Updating Method*, and *Parallel Mean Field Annealing Neural Network.*

2.4.1 Hopfield and Tank with Genetic Breeding

Several studies have shown that the *Hopfield and Tank Model* is very sensitive to the parameters **A**, **B**, **C**, and **D** of the energy function and setting these parameters is not an easy task. For this reason, a genetic algorithm is used with the *Hopfield and Tank Network* to breed an effective set of control parameters for the *Hopfield and Tank Network.* This technique was proposed by W. Lai and G. Coghill in 1992 to provide a way for selecting good control parameters for the *Hopfield and Tank Model* [42]. In this technique, each possible set of control parameters is represented as a chromosome of binary bits.

The genetic breeding algorithm for the control parameters can be described by the following steps. First, the population that consists of several chromosomes is created randomly. After that, two chromosomes from the current population are selected randomly and combined by crossover and/or mutation operators. The crossover operator is performed often by swapping the values in the selected chromosomes over a crossover point as shown in Figure 5. However, the mutation operator is performed with a small probability by flipping the bits of the selected chromosome.

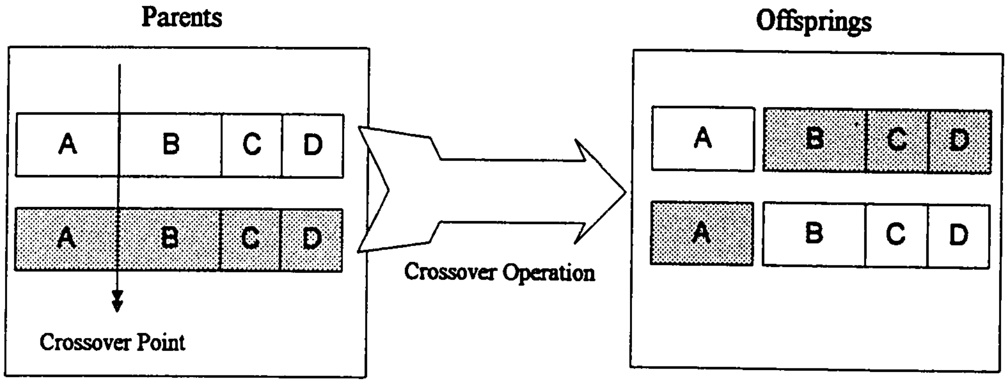


FIGURE 5: The crossover operator for two chromosomes

After that, the generated chromosomes are added to the current population by replacing the worst two chromosomes by the generated two. A good evaluation function for selecting the worst two chromosomes from the population is the energy function itself. The recombination operators are applied repeatedly until good control parameters are found. In this case, the *Hopfield and Tank Neural Network* is used to find an optimal solution for the given TSP instance.

Table 1 presents the simulation results for 10 cities using the *Genetic Breeding Method for the Control Parameters of the Hopfield and Tank Model.* This table shows that the *Genetic Breeding Algorithm* has a very clear effect on the *Hopfield and Tank Model* and it increases the number of feasible solutions generated from 72% to 99.7%.

2.4.2 Neural Network with Coordinate Newton Updating

A hybrid neural network method for solving many optimization problems based on the *Coordinate Newton Method* was suggested by K. Sun and H. Fu in 1993 [60]. An important feature for K. Sun and H. Fu method is that it provides a systematic way for constructing an energy function to represent an optimization problem. This method can find feasible solutions for the TSP, although its performance is worse than the *Elastic Net* method. This neural network consists of a constraint network and a goal network.

	% of feasible Solutions	Best Length	Optimal Tour
Hopfield and Tank Model	72.0 %	2.95	2.69
Hopfield with Genetic Breeding	99.7 %	2.95	2.69

TABLE 1: Simulation results for the Hopfield model with genetic breeding

The constraint network represents the constraints of the TSP and computes the updating values $\{\Delta X_1, \ldots, \Delta X_n\}$ of each neuron such that, the energy function **E** converges and satisfies all the constraints of the problem. The goal network directs the net towards the convergence by finding an optimal value for the cost function

The K. Sun and H. Fu method can be described by the following steps (See Figure 6). First, the TSP is transformed to an energy function that should be minimized. This function consists of two parts: cost and constraints. This can be achieved by representing the TSP as a set of logical expressions that will be transformed into algebraic expressions by replacing TRUE with 1, FALSE with 0, **NOT** X with (1 - X), and (X **AND** Y) with (X * Y). After that, the energy function is derived based on the algebraic equations that represent the cost function and the constraints.

Let C_{xi} represent whether a city **x** is being visited at time **i** during a tour. The TSP contains three constraints. The first constraint is that "Each city **i** must be visited exactly once". This constraint can be represented by the following logical expressions.

$C_{i1} \vee C_{i2} \vee \ldots \vee C_{iN} = \text{True}$ (at least one C_{ik} is true; $1 \le k \le N$) (15)

$C_{i1} \wedge C_{i2} = \text{False}, C_{i1} \wedge C_{i3} = \text{False}, \ldots$ (no more than one C_{ik} is true; $1 \le k \le N$) (16)

The second constraint is that "A salesman arrives at only one city at any time **j** during the tour". This constraint can be represented by the following logical expressions.

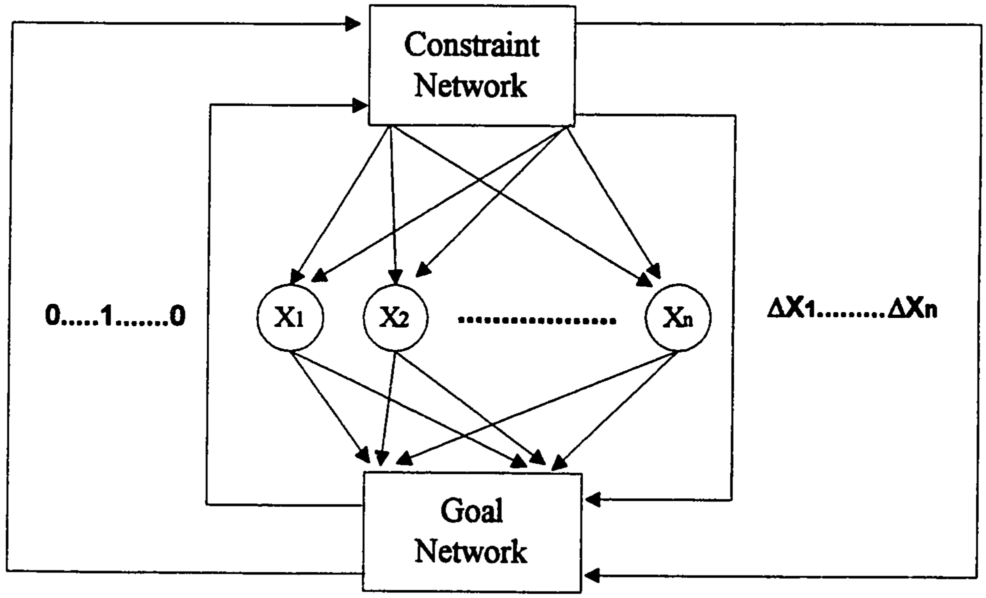


FIGURE 6: The neural network with the coordinate Newton method

$$C_{1i} \vee C_{2i} \vee \ldots\ldots \vee C_{Ni} = \text{True} \qquad \text{(at least one } C_{ki} \text{ is true ; } 1 \le k \le N) \qquad (17)$$

$$C_{1i} \wedge C_{2i} = \text{False}, C_{1i} \wedge C_{3i} = \text{False}, \ldots \qquad \text{(no more than one } C_{ki} \text{ is true ; } 1 \le k \le N) \qquad (18)$$

The cost criterion for the TSP that should be minimized is represented as follows.

$$\text{Cost} = \sum_{x}^{N} \sum_{y}^{N} \sum_{i}^{N} [d_{xy}(C_{xi} \wedge (C_{yi+1} \vee C_{yi-1}))] \qquad (19)$$

After that, the constraints expressions (15), (16), (17), and (18) are transformed into the following algebraic expressions, respectively.

$$(1 - C_{i1})(1 - C_{i2}) \;\ldots\ldots\; (1 - C_{iN}) = 0 \qquad (20)$$

$$C_{i1}\, C_{i2} = 0, C_{i1}\, C_{i3} = 0, \;\ldots\ldots \qquad (21)$$

$$(1 - C_{1i})(1 - C_{2i}) \;\ldots\ldots\; (1 - C_{Ni}) = 0 \qquad (22)$$

$$C_{1i}\, C_{2i} = 0, C_{1i}\, C_{3i} = 0, \;\ldots\ldots \qquad (23)$$

While the cost function (19) is converted to the following algebraic expression.

$$\text{Cost} = \sum_{x}^{N} \sum_{y}^{N} \sum_{i}^{N} [d_{xy} C_{xi}(C_{yi+1} + C_{yi-1} - C_{yi+1} C_{yi-1})] \qquad (24)$$

Therefore, the energy function can be formulated from the cost function and the squared errors as given by the following equation.

$$E = A\sum_{i=1}^{M}\left(t_i - a_i\right)^2 + B * \mathrm{Cost} \qquad (25)$$

Here M is the number of equations in (20), (21), (22), and (23), t_i is the target value for each equation (right hand side), and a_i is the iteration value for each equation (left hand side). All the constraints of the TSP are satisfied when the squared error term, first term of equation (25), equals zero.

The second step in the K. Sun and H. Fu method is to apply the *Coordinate Newton Updating Algorithm* that minimizes **E** until the energy function converges to a stable state. This is achieved by three stages. In the first stage, the Constraint Net computes the updating values, ΔC_{xi}, using the *Coordinate Newton Method* as given by the following equation.

$$\Delta C_{xi} = \left(-\frac{\frac{dE_s}{dC_{xi}}}{\frac{d^2E_s}{dC_{xi}^2}} \right) \quad ; \text{for each neuron } C_{xi} \qquad (26)$$

In the second stage, the Goal Net computes the partial derivatives of the cost function as follows.

$$\left(\frac{d\mathrm{Cost}}{dC_{xi}} = \sum_{y=1}^{N}\sum_{x \neq y}^{N} d_{xy}\left(C_{yi+1} + C_{yi-1} - C_{yi+1}C_{yi-1}\right) \right) \quad ; \text{for each variable } C_{xi} \qquad (27)$$

In the third stage, the Goal Net finds the maximum updating value $\Delta C^{*} = \mathrm{Max}(\{|\Delta C_{xi}| \,;\, 1 \le i \le N\})$; where the set of variables **V** is defined as $V = \{C_{xj} \mid |\Delta C_{xi}| = \Delta C^{*} \,;\, \text{for all } j\}$. A neuron C_{xk} from **V** is selected such that

$C_{xk} = \mathrm{Min}\{ \left(\frac{d\,Cost}{d\,C_{xi}}\right)$; for all i in V} and C_{xk} is updated using the following equation.

$$C_{xk}^{t+1} = C_{xk}^{t} + \Delta C_{xk}^{t} \tag{28}$$

The *Neural Network with the Coordinate Newton Method* provides a systematic transformation method for representing optimization problems as an energy function. Also, it does not require adjustments for the parameters of the energy function. For these reasons, several local minimums can be skipped using the *Newton Coordinate Method* which has less computation time than the standard *Hill-Climbing* technique.

2.4.3 Parallel Mean Field Annealing Neural Network

One way to allow the *Hopfield and Tank TSP Network* to escape local minimums is to use both the *Hopfield and Tank Model* and the *Simulated Annealing* technique [32,39] as a hybrid technique called the *Parallel Mean Field Annealing* [61,66]. In the *Parallel Mean Field Annealing* technique, the energy function for the TSP is replaced by a simpler energy function.

$$E_{TSP} = \sum_{x}^{N}\left[\alpha\sum_{y\neq x}^{N}\sum_{i}^{N}(d_{xy} - d_{max})V_{xi}V_{y,i+1} + \left(1.0 - \sum_{y\neq x}^{N}\sum_{i}^{N}V_{xi}V_{y,i+1}\right)^{2}\right] \tag{29}$$

The first double summation term of equation (29) will minimize the tour length. On the other hand, the second term imposes the TSP constraint that is, two cities cannot have the same tour position and represent one path from a city to another. The $\mathbf{d_{max}}$ is a parameter that should be grater than the maximum distance between any two cities in the given TSP instance.

The *Parallel Mean Field Annealing* neural network that can be applied to binary as well as continuous problems can be described by the following algorithm:

```
1. Initialize all neurons to some values and the temperature value T.
2. Repeat   { as T decreases, E decreases until E becomes constant }
    2.1. Select a city X randomly.
    2.2. Disturb output value $V_{xi}$
            $V_{xi}$ is the probability of finding city x in the tour position i
    2.3. Calculate $\Delta E$ using the proposed energy function (Equation 29)
    2.4. IF $\Delta E \leq 0$ THEN accept with Pr(accept) = 1
          ELSE accept with Pr(accept) = exp(-$\Delta E/T$)
    2.5. Compute the mean field $E_{xi}$ for the neurons.
            $E_{xi} = d_{max} \sum_{y \neq x}^{N} V_{xi} + \sum_{y \neq x}^{N} d_{xy} (V_{y,i+1} + V_{y,i-1})$
   Until a fixed point is found.
3. IF ( T reaches the final temperature ) THEN
          Stop
   ELSE
          Decrease the temperature (e.g. $T_n = 0.9 * T_{n-1}$).
          Go to step 2
   ENDIF
```

The *Parallel Mean Field Annealing Network* has been applied on 10 and 30 cities TSPs where all tours found are feasible solutions. It is found that the *Parallel Mean Field Annealing* technique converges rapidly to an optimal solution. For 30 cities TSPs, the tour length was 12% longer than the tour found by *Simulated Annealing* [61].

CHAPTER 3

THE ENHANCED CONVEX-ELASTIC NET ALGORITHM

Many local search algorithms have been developed to solve the E-TSP. However, local search algorithms often get stuck at local minimums [29]. To reduce the effect of the local minimum problem, we introduced two search techniques: a *Convex-Elastic Net* (CEN), and a *Non-Deterministic Iterative Improvement* (NII) technique. The CEN is a global search technique that generates an initial tour that can be used by a local search technique. The NII approach is a local search technique that improves the tour found over time and local minimums are escaped through the noise added to the cost function. The two introduced techniques are combined into a new hybrid approach called the *Enhanced Convex-Elastic Net* (ECEN).

The proposed technique for solving E-TSP is developed using two phases as shown in Figure 7. In the first phase, the *Convex-Elastic Net* algorithm is used as a global search technique that generates an initial tour for the second phase by enforcing the E-TSP constraints instead of transforming them into penalties. In the second phase, the tour found is tuned further and iteratively improved by making local changes to it.

3.1 The Convex-Elastic Net Phase

In 1956, Merrill Flood showed a general property for any optimal tour of the E-TSP **[43]**. This property is that vertices located on the convex hull should be visited as they appear on the convex hull as shown in Figure 8. We propose a new technique that combines this property and the *Elastic Net* approach **[18]**. The convex hull for a set of points, S, can be found using *Graham Scan* technique [14]. In this technique, points are sorted by polar angle in counterclockwise order around the point that has the minimum y-coordinate. While the angle formed by any three consecutive points $\{P_{i-1}, P_i, P_{i+1}\}$ is not a left turn, the point P_i is removed from the set S. When the algorithm terminates, the set S contains exactly the vertices of the convex hull.

A set of N cities, where each city i is denoted by its two coordinates in the plane (C_{ix}, C_{iy}), is given. A solution for the E-TSP in our approach consists of a tour that is represented by a set of nodes joined together in one dimensional ring or rubber band. Each node j on the ring is characterized by two dimension coordinates (W_{jx}, W_{jy}). Each node on the ring is also related to its two neighbors.

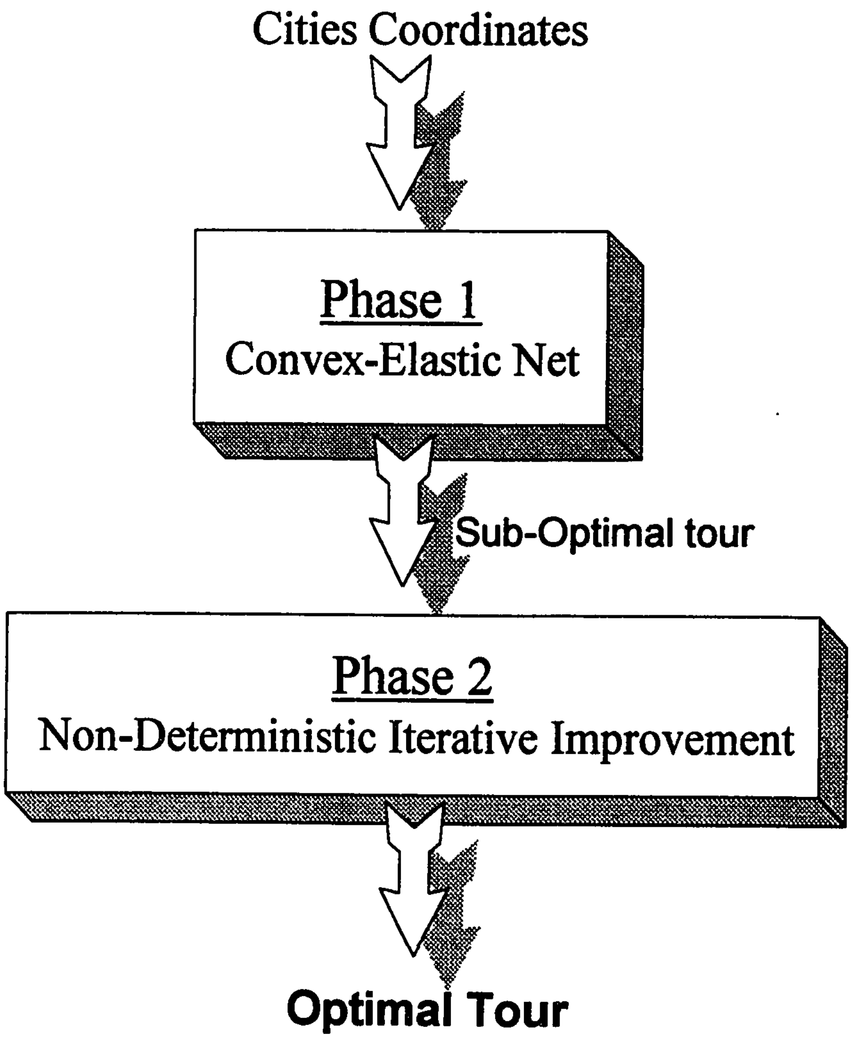
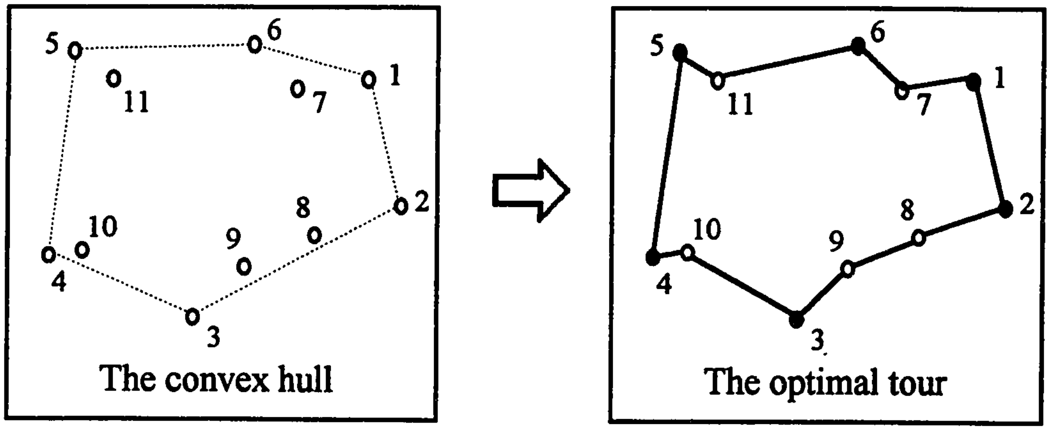


FIGURE 7: The enhanced convex-elastic net phases


The convex hull

The optimal tour

FIGURE 8: A known property of the E-TSP

At the beginning of the process, the proposed approach (CEN) creates an initial tour as a rubber band that is originally shaped as a convex hull for the given N cities using *Graham Scan* technique. The number of nodes grows subsequently according to a node creation and selection process. Since more than one rubber band node can be attached to a city, there should be more rubber band nodes than cities. On the other hand, a large number of nodes will increase the computation time for the CEN algorithm. For this reason, the number of rubber band nodes in the CEN algorithm is restricted to be less than or equal to twice the number of cities. The nodes of the rubber band are free to move in the plane through an iterative process. Eventually, an actual solution is obtained when every city has caught one node of the rubber band.

In each iteration, the influence of all cities that are not on the rubber band on the nodes of the rubber band is computed and the nodes are displaced accordingly as shown in Figure 9. In Figure 10, one force that keeps the rubber band nodes together (F1) is applied on each node and another force (F2) pulls each node towards a city such that the tour is stretched and more cities are introduced to the rubber band.

The CEN is a projection for a set of two-dimensional cities coordinates in the plane to a set of positions on a convex hull. So, each city is mapped to a node on the convex hull and neighborhood relationships between cities on the rubber band are preserved. This is similar to many problems naturally solved in the brain. For example, the connections between the cells of the vertebrate retina and the visual area of the brain (optic tectum) must preserve the correct relationship between different parts of the scene observed [18].

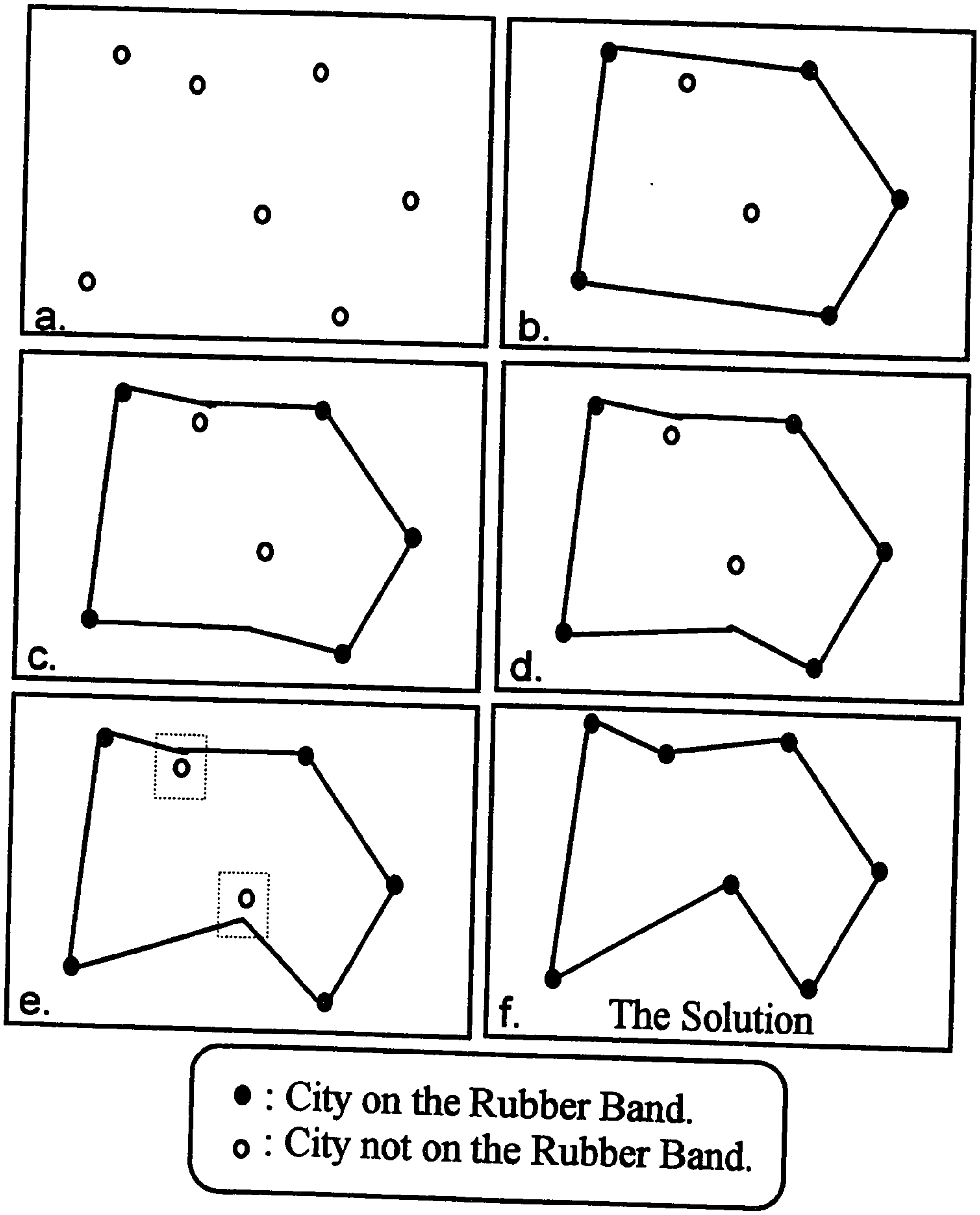

a.
b.
c.
d.
e.
f.
The Solution
● : City on the Rubber Band.
o : City not on the Rubber Band.

FIGURE 9: Evolution of the convex-elastic net over time

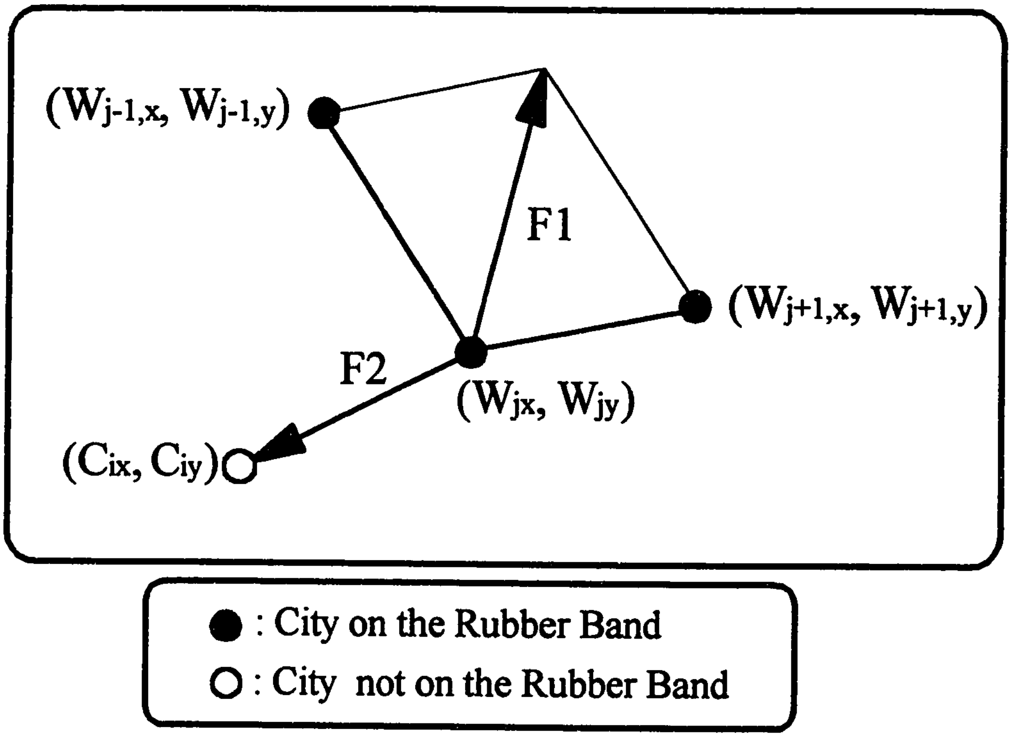


FIGURE 10: Forces on the rubber band for the CEN

The *Convex-Elastic Net* is an adaptive neural-based technique that is derived from the *Kohonen's Self-Organizing Feature Maps Approach* **[22,56]**. Both of these techniques learn to organize themselves without being taught as unsupervised learning networks. In the CEN technique, the two-dimensional coordinates of cities are the input patterns and the similarity criterion is the Euclidean nearness. The CEN consists of two layers network: an input layer and an output layer as shown in Figure 11. Each neuron in the input layer is connected to each neuron in the output layer by a connection with an associated weight. The input layer is represented as a two-dimensional array of cities coordinates. The output layer is represented as one dimensional array of closed rubber band nodes. The CEN model has **N** neurons and M*N connections, where **M** is the maximum number of rubber band nodes and $M \leq 2N$. Therefore, the number of connections is $M * N \leq 2N^2$.

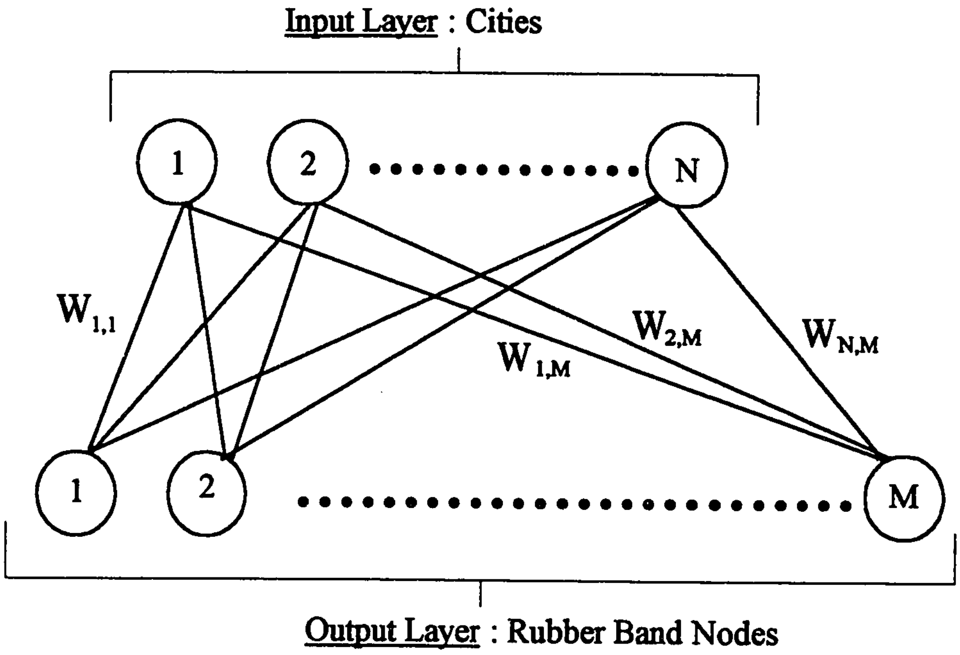


FIGURE 11: The structure of the convex-elastic net model

At each iteration, cities not on the rubber band are presented to the CEN network. As a result, a competition learning phase is initiated. The competition learning phase consists of presenting patterns to the network one at a time to train it. The best training patterns for the E-TSP are the coordinates of cities themselves. The winning node is found by making the following computations. First, the Euclidean distance between the input pattern (city) and the weight vector associated with each output node is found. Then, the output node with the minimum Euclidean distance is the winning node. The Euclidean distance is the square root of the sum of the squares of the differences between each input vector component and its associated weight vector component. After that, the winning node is updated by adjusting its weight similar to the *Kohonen's Self-Organizing Feature Maps* [20]. Once all cities are sufficiently near a rubber band node, the tour formed by the rubber band is a sub-optimal solution for the given instance of the E-TSP.

The *Convex-Elastic Net* (CEN) algorithm can be described by the following steps:

Input : Set of N cities coordinates in two-dimensional plane.
Output : Optimal tour through the N cities.

Step 1. Find the Convex Hull:

Find the convex hull using the *Graham Scan* algorithm.

Step 2. Initialize the Weights:

Initialize the weights for the rubber band nodes as the coordinates of the convex hull vertices as follows:

```
FOR each vertex i on the convex hull DO
{ $W_{ix} = C_{ix}$
  $W_{iy} = C_{iy}$
}
```

Where

(W_{ix}, W_{iy}) : The connection weight for the i-th node on the rubber band.

(C_{ix}, C_{iy}) : The coordinates of the i-th city on the convex hull.

Step 3. Competition Learning Phase:

Present each city, i, not on the rubber band to the net.

3.1. Find the nearest point j on the rubber band:

The nearest point to the i-th city is the point (X^*, Y^*). This point has two cases. First, point (X^*, Y^*) is located on one of the nodes of the rubber band. In this case, point (X^*, Y^*) can be found using the following equation.

$$NET_k = MIN\{D_{ij}\} \quad ; \text{where } 1 \le j \le \text{Number\_rubberband\_nodes} \qquad (30)$$

$$(X^*, Y^*) = (W_{kx}, W_{ky})$$

Where

NET_k : The smallest distance between city i and all nodes of the rubber band.

D_{ij} : The Euclidean distance between the i-th city and the j-th weight which is equal to $\sqrt{(C_{ix}-W_{jx})^2+(C_{iy}-W_{jy})^2}$.

(W_{kx}, W_{ky}) : The coordinates of the k-th node on the rubber band. It is considered as the weight of the k-th node.

Second, point (X^*, Y^*) is on the nearest edge of the rubber band to the i-th city (C_{ix}, C_{iy}). In this case, point (X^*, Y^*) is located between two nodes of the rubber band (W_{jx}, W_{jy}) and $(W_{j+1,x}, W_{j+1,y})$ and on the line connecting these two nodes, as shown in Figure 12.

1) The first line passes through the nodes (W_{jx}, W_{jy}) and $(W_{j+1,x}, W_{j+1,y})$ as shown in Figure 12. This line has the following line equation.

$$\frac{Y-W_{jy}}{X-W_{jx}}=\frac{W_{j+1,y}-W_{jy}}{W_{j+1,x}-W_{jx}} \tag{31}$$

$$\Rightarrow Y=\frac{(W_{j+1,y}-W_{jy})}{(W_{j+1,x}-W_{jx})}*(X-W_{jx})+W_{jy} \tag{32}$$

2) The second line passing through the i-th city (C_{ix}, C_{iy}) and is perpendicular to the line connecting (W_{jx}, W_{jy}) and $(W_{j+1,x}, W_{j+1,y})$, as shown in Figure 12. The slope of this line is $-\frac{(W_{j+1,x}-W_{jx})}{(W_{j+1,y}-W_{jy})}$.

$\Rightarrow$ The second line has the following equation.

$$Y=-\frac{(W_{j+1,x}-W_{jx})}{(W_{j+1,y}-W_{jy})}*(X-C_{ix})+C_{iy} \tag{33}$$

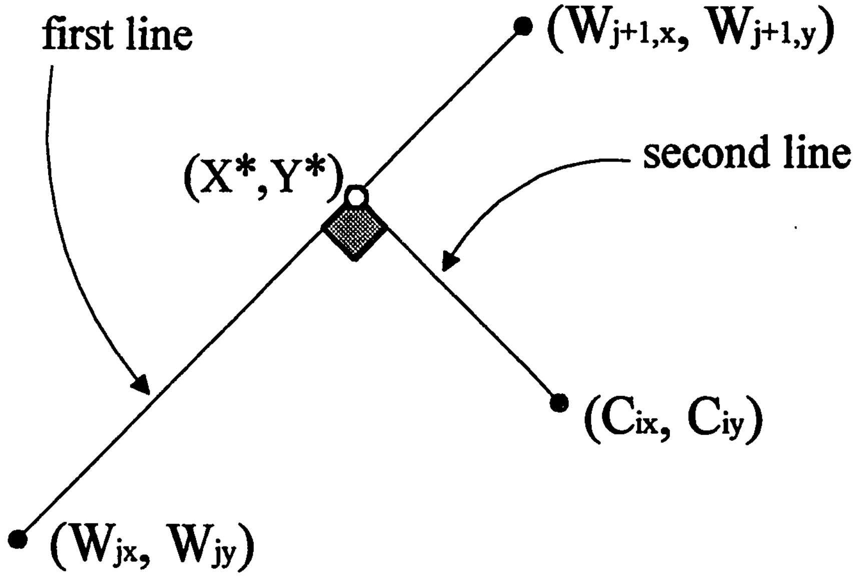


FIGURE 12: The nearest point between a point and a line

$$\Rightarrow X = -\frac{(W_{j+1,y} - W_{jy})}{(W_{j+1,x} - W_{jx})} * (Y - C_{iy}) + C_{ix} \tag{34}$$

The two lines intersect at point (X^*, Y^*). Substituting equation (32) in equation (34) and replacing (X, Y) by (X^*, Y^*) will give X^*.

$$X^* = -\frac{(W_{j+1,y} - W_{jy})^2}{(W_{j+1,x} - W_{jx})^2} * (X^* - W_{jx}) + \frac{(W_{j+1,y} - W_{jy})}{(W_{j+1,x} - W_{jx})} * (C_{iy} - W_{jy}) + C_{ix} \tag{35}$$

$$\begin{aligned} \Rightarrow (W_{j+1,x} - W_{jx})^2 * X^* = {} & -(W_{j+1,y} - W_{jy})^2 * (X^* - W_{jx}) + (W_{j+1,x} - W_{jx})^2 * C_{ix} + \\ & (W_{j+1,y} - W_{jy}) * (W_{j+1,x} - W_{jx}) * (C_{iy} - W_{jy}) \end{aligned} \tag{36}$$

$$X^* = \frac{(W_{j+1,y} - W_{jy})^2 * W_{jx} + (W_{j+1,y} - W_{jy}) * (W_{j+1,x} - W_{jx}) * (C_{iy} - W_{jy}) + (W_{j+1,x} - W_{jx})^2 * C_{ix}}{(W_{j+1,y} - W_{jy})^2 + (W_{j+1,x} - W_{jx})^2}$$

Substituting equation (34) in equation (32) and replacing (X, Y) by (X^*, Y^*) will give Y^*.

$$Y^* = -\frac{(W_{j+1,y} - W_{jy})^2}{(W_{j+1,x} - W_{jx})^2} * (Y^* - C_{iy}) + \frac{(W_{j+1,y} - W_{jy})}{(W_{j+1,x} - W_{jx})} * (C_{ix} - W_{jx}) + W_{jy} \tag{37}$$

$$\begin{aligned} \Rightarrow (W_{j+1,y} - W_{jy})^2 * Y^* = {} & (W_{j+1,y} - W_{jy})^2 * (C_{iy} - Y^*) + (W_{j+1,x} - W_{jx})^2 * W_{jy} + \\ & (W_{j+1,y} - W_{jy}) * (W_{j+1,x} - W_{jx}) * (C_{ix} - W_{jx}) \end{aligned} \tag{38}$$

$$Y^* = \frac{(W_{j+1,y} - W_{jy})^2 * C_{iy} + (W_{j+1,y} - W_{jy}) * (W_{j+1,x} - W_{jx}) * (C_{ix} - W_{jx}) + (W_{j+1,x} - W_{jx})^2 * W_{jy}}{(W_{j+1,y} - W_{jy})^2 + (W_{j+1,x} - W_{jx})^2}$$

3.2. Creating / Deleting Nodes on the Rubber Band:

```
IF ( point j is not a node on the rubber band ) OR
   ( point j is a city on the rubber band ) THEN
BEGIN
   IF (number of rubber band nodes ≥ 2N ) THEN
      Delete from the rubber band the node that has the least winning rate.
   Create a new node j with coordinates $(X^*, Y^*)$.
   Initialize the winning rrate for the new node to zero
ENDIF
```

At this point, the winning neuron (X^*, Y^*) is selected. This neuron is clearly the nearest point on the rubber band to the i-th non-rubber band city.

3.3. Update Weights for the Winning Node j:

Using the following neuron updating rule, the weight for node j is updated. The node that wins the competition (node j) moves towards the city.

$$W_{jx}^{new} = W_{jx}^{old} + L * \Delta W_{jx} \tag{39}$$

$$W_{jy}^{new} = W_{jy}^{old} + L * \Delta W_{jy} \tag{40}$$

$$\Delta W_{jx} = A * F_{1,x} + B * \frac{F_{2,x}}{|F_2|} = A * (W_{j+1,x} + W_{j-1,x} - 2 * W_{jx}) + B * (\frac{C_{ix} - W_{jx}}{D_{ij}}) \tag{41}$$

$$\Delta W_{jy} = A * F_{1,y} + B * \frac{F_{2,y}}{|F_2|} = A * (W_{j+1,y} + W_{j-1,y} - 2 * W_{jy}) + B * (\frac{C_{iy} - W_{jy}}{D_{ij}}) \tag{42}$$

Where

(W_{jx}, W_{jy}) : The coordinates of the j-th node on the rubber band. It is considered as the connection weight between city **i** and node **j**.

L : The learning rate for the neurons. It is set to some small fraction, for example .02.

F1: The force that is applied on each rubber band node to keep neighboring nodes together to produce a shorter tour. This force has two components $F_{1,x}$ and $F_{1,y}$ as shown in Figure 10.

F2 : The force that is applied on each rubber band node to pull the rubber band node towards a city. This force has two components $F_{2,x}$ and $F_{2,y}$ as shown in Figure 10.

A : Parameter for the first force (F1) that keeps neighboring nodes together to produce a shorter tour.

B : Parameter for the second force (F2) that pulls the rubber band node towards a city. The two parameters **A** and **B** should be adjusted relative to each others. For example, initialized **A** and **B** to 1. After each complete presentation of cities to the system, decrease parameter **A** (or increase parameter **B**) by a small fraction in the range (.01-.05). This will move more nodes towards cities and push the system towards the final solution where each city is attached to a rubber band node. Other possible values for parameters **A** and **B** can be found experimentally.

(C_{ix}, C_{iy}) : The coordinates of the i-th city.

D_{ij} : The Euclidean distance between the **i**-th city and the **j**-th node on the rubber band which is equals to $\sqrt{(C_{ix} - W_{jx})^2 + (C_{iy} - W_{jy})^2}$.

Also, the winning rate for the winning node, **j**, is incremented by one. The main purpose of the winning rate for each rubber band node is to keep the number of rubber band nodes limited to 2N at most. This will be achieved by deleting the node that has the least winning rate before creating a new node (when the number of nodes equals to 2N).

3.4. Checking if a Rubber Band Node Near a City:

IF ($D_{ij} \leq$ a threshold) THEN

Let City **i** be a node on the rubber band.

Adjusting the threshold value for the CEN is an important stage in finding a good solution for the E-TSP. Choosing a small threshold value can increase the running time for the algorithm drastically, while a large threshold value can give inefficient tours. Through experimentation, it is found that good threshold values should be no more than 1% of the longest distance between cities in a given instance. In our implementation for the ECEN the given E-TSP instance is scaled to one unit square. As a result, the threshold used is .01.

Step 4. Decrement the Parameter A:

When parameter A in equations (41) and (42) is decreased, the force that pulls the rubber band nodes together is reduced relative to the second force that moves the nodes towards the cities.

Step 5. Check the Termination Condition:

```
IF (there is a city not attached to a rubber band node) THEN
    Go to Step 3
```

Step 6. Return the formed tour:

The number of rubber band nodes is greater than or equal to the number of cities. For this reason, the tour formed by the order of the rubber band nodes $\{(W_{1x}, W_{1y}), (W_{2x}, W_{2y}), \ldots, (W_{2N,x}, W_{2N,y})\}$ will not be a valid tour for the E-TSP. To get the valid tour for the problem, each city should be associated with a single node on the rubber band. Simply, every city is associated with the nearest node on the rubber band. The tour formed by the associated rubber band nodes only is the solution for the problem.

Figure 13 shows the CEN algorithm described previously in a block diagram form.

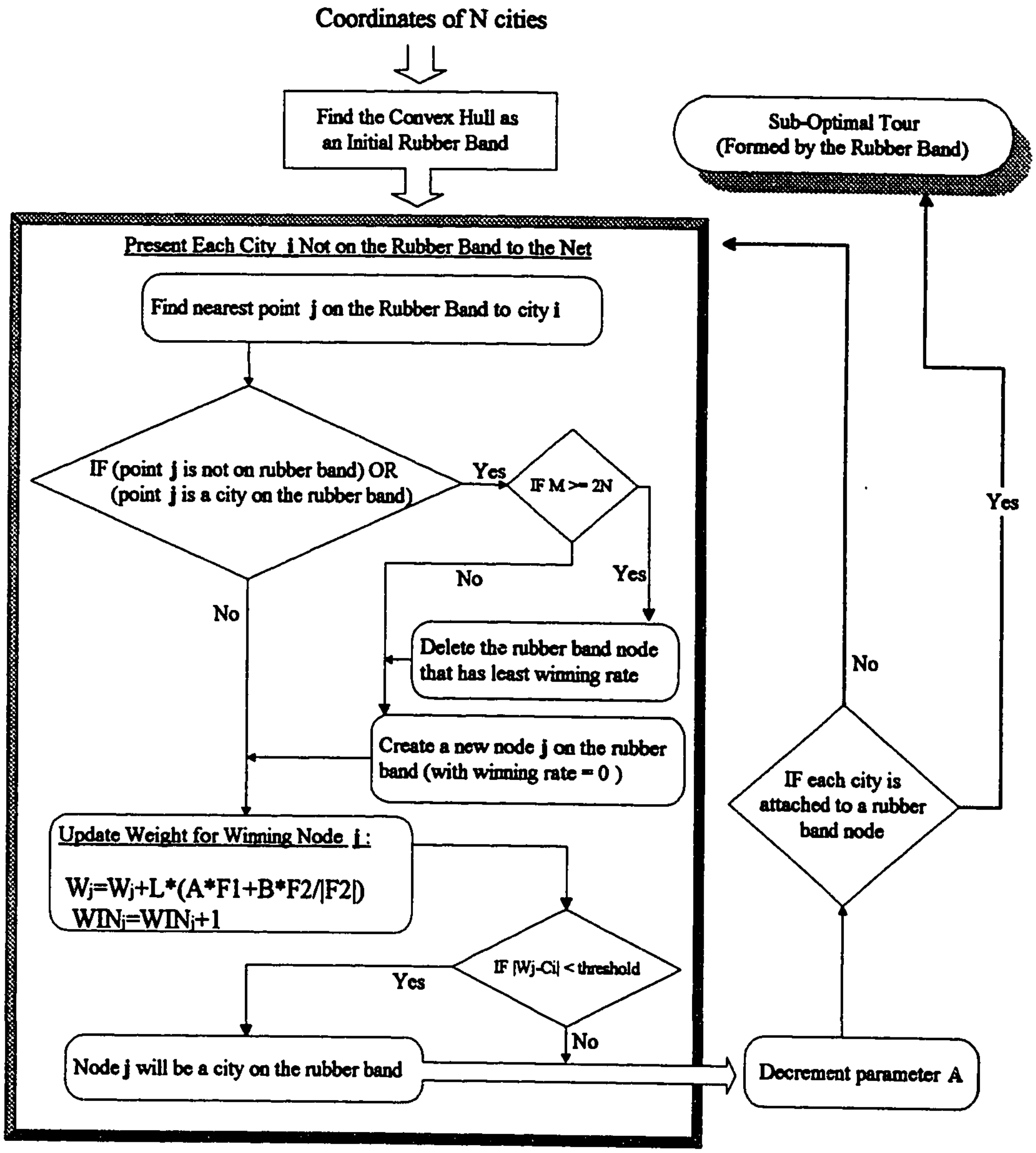

Coordinates of N cities
Find the Convex Hull as an Initial Rubber Band
Sub-Optimal Tour (Formed by the Rubber Band)
Present Each City i Not on the Rubber Band to the Net
Find nearest point j on the Rubber Band to city i
IF (point j is not on rubber band) OR (point j is a city on the rubber band)
Yes
IF M >= 2N
No
Yes
No
Delete the rubber band node that has least winning rate
Create a new node j on the rubber band (with winning rate = 0)
Update Weight for Winning Node j :
Wj=Wj+L*(A*F1+B*F2/|F2|)
WINj=WINj+1
IF |Wj-Ci| < threshold
Yes
No
Node j will be a city on the rubber band
Decrement parameter A
IF each city is attached to a rubber band node
No
Yes

FIGURE 13: Block diagram for the convex-elastic net algorithm

3.2 The Non-Deterministic Iterative Improvement Phase

The *Non-Deterministic Iterative Improvement Approach* (NII) is developed to enhance the tour produced by the *Convex-Elastic Net* phase. To construct the NII algorithm, we need a mean of representing the tour and a mean of generating rearrangement of the tour. Each tour can be described by a permutation list of numbers from 1 to N, which represents the cities. A powerful set of rearrangement operators consists of an operator for removing loops from a tour and another one for changing the cities positions in a tour. These operators are simple rearrangement operators that are derived from two known heuristics, namely *2-Optimal* (2-Opt) [55] and the *Point Heuristic* [41]. These operators are used here to remove the intersecting paths from the tour produced by the *Convex-Elastic Net* phase.

The *2-Opt Heuristic* [55] is developed by Lin Kernighan and used by many algorithms like *Genetic* and *Simulating Annealing* techniques. This heuristic locally modifies a given tour by replacing two edges in the current tour by another two that reduce the tour length. This process continues until the tour cannot be improved any further. The *2-Opt Heuristic* can be described by the following steps:

1) Consider an initial random tour **T** for the given cities.
2) Delete two edges from the tour **T**. Then, reverse one of the resulting paths and reconnect them to get a new tour **T`**.
3) IF Length(T`) < Length(T) THEN T = T`.
4) Repeat steps 2 and 3 until no improvement can be found to **T**.

The *Point Heuristic* [41], used by Patrick Krolak, removes a city from the tour and inserts it between another two adjacent cities provided that it reduces the tour length. This heuristic can be described by the following steps:

1) Consider an initial random tour **T** for the given cities.
2) Delete a city from the tour **T**. Then, insert it between two adjacent cities to get a new tour **T**`.
3) IF Length(T`) < Length(T) THEN T = T`.
4) Repeat steps 2 and 3 until no improvement can be found to **T**.

The NII algorithm starts with a given tour. Then, one of the two rearrangement operators (step 2 from *2-Opt* or *Point Heuristic*) is applied to the current tour to get N new tours. Next, one of the new tours is selected based on its probability of selection. After that, the selected tour becomes the current tour. This process is continued until no further improvements can be found to the current tour. The NII algorithm can be described by the following steps:

Step 1. Given an initial tour T from the first phase (CEN). This tour is represented as one dimensional array of size N that shows the order of visiting cities. Where N is the number of cities.

Step 2. Consider N possible changes $\{V_1, V_2, ..., V_n\}$ to the current tour T to get N new tours $\{T_1, T_2, ..., T_n\}$. These possible changes to the current tour (V[i]) can be found as follows.

Let V[i] = random number in the range (1, 2, ..., N) and V[i] $\neq$ i; for $1 \leq i \leq N$

```
IF (iteration is odd) THEN        {apply the 2-Opt Heuristic operator}
    V[i] = k
      where k represents the change in the current tour T that results when the
      subtour (T[i], T[i+1], ..., T[k]) is reversed.
ELSE                              {apply the Point Heuristic operator}
    V[i] = k
      where k represents the change in the current tour T that results when
      city T[k] is inserted between the adjacent cities T[i-1] and T[i].
ENDIF
```

Step 3. Compute the probabilities of selection, **Pr[i]**, for each new tour T_i depending on the change in the tour length (V[i] = k) as follows:

$$\Pr[i] = \begin{cases} \dfrac{\Delta T_i}{\sum_{u=1}^{N} \Delta T_u} & ; \text{if } \Delta T_i < 0 \\ 0 & ; \text{otherwise} \end{cases} \qquad ; \text{where } 1 \le i \le N \qquad (43)$$

Where

ΔT_i = The difference in length between the new tour T_i and current tour T.

= tour length(T_i) - tour length(T) + noise

$$= \text{noise} + \begin{cases} d(i,k+1) + d(k,i-1) - d(i-1,i) - d(k,k+1) & ; \text{if iteration is odd} \\ d(i,k) + d(k,i+1) + d(k-1,k+1) - d(i,k+1) - d(k,k-1) - d(k,k+1) & ; \text{otherwise} \end{cases}$$

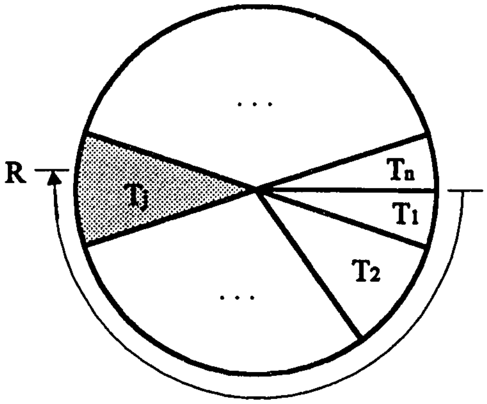
Where

noise = d_{max} * random number in the range (-.03 to .03).

d_{max} = the maximum distance between any two cities in a given E-TSP.

d(i, k) = the Euclidean distance between city i and city k.

Step 4. Select a tour, T_j, using roulette wheel [21] as follows. First, probabilities of selection for the **N** new tours $\{T_1, T_2, \ldots, T_n\}$ are represented in a wheel. Then, a random number, **R**, between 0 and 1 is generated. Next, the area associated with **R** in the wheel is selected as the next tour ($T = T_j$) as shown below.



Step 5. Repeat steps 2, 3 and 4 until no further improvement to the current tour **T** can be found.

Step 6. Return the current tour **T** as the optimal solution.

Figure 14 shows the block diagram for the NII algorithm.

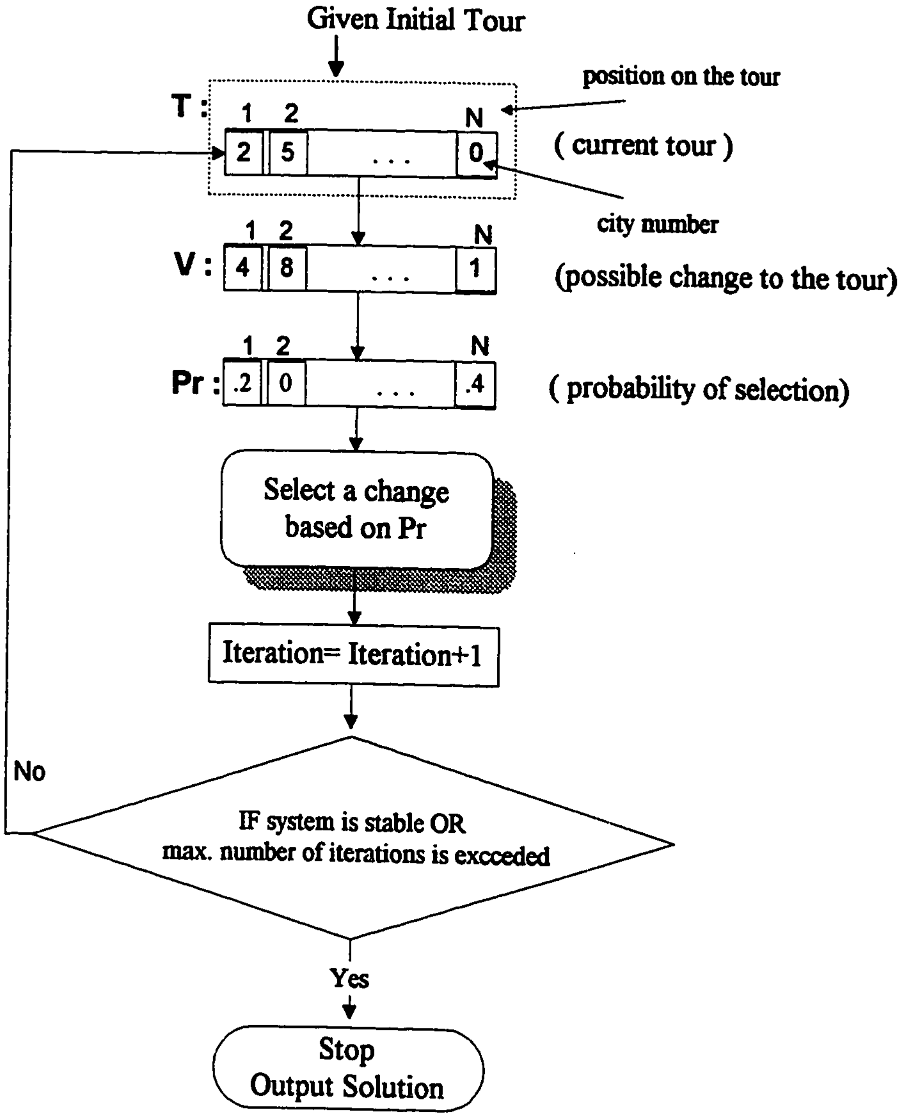
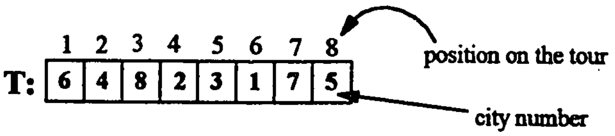
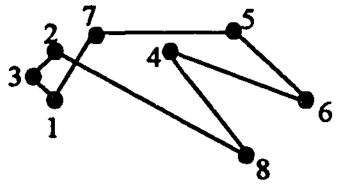


FIGURE 14: Block diagram for the NII algorithm

The following example demonstrates how the NII technique works. Given a tour T from the first phase (CEN).



	1	2	3	4	5	6	7	8
T:	6	4	8	2	3	1	7	5



A random number from 1 to 8 is generated for each position on the tour. This will give the following tour changes.

	1	2	3	4	5	6	7	8
V :	3	4	5	⑥	6	7	8	2

Assuming the iteration number is odd, then we use the *2-Opt Heuristic* operator. For example, V[4] = 6 means that the current tour can be changed by reversing the sub-tour {T[4], T[5], T[6]}. Then, the probability of selection for each change {Pr[1], Pr[2], ..., Pr[8]} is computed using equation (43). For example $\Pr[1] = \frac{\Delta T_1}{\sum_{u=1}^{8} \Delta T_u} = \frac{\text{noise} + d(1,4) + d(3,8) - d(8,1) - d(3,4)}{\Delta T_1 + \Delta T_2 + \ldots + \Delta T_8} = 0.2$.

	1	2	3	4	5	6	7	8
Pr :	.2	0	.1	.5	0	0	0	.2

Next, the roulette wheel is used to represent the probabilities of selection. Then, a random number in the range (0.0-1.0) is generated, say 0.45. Then, using the roulette wheel V[4] is selected because the probabilities of V[1]+V[2]+V[3]+V[4] = 0.2+0+0.1+0.5 = 0.8 ≥ 0.45, as shown in Figure 15.

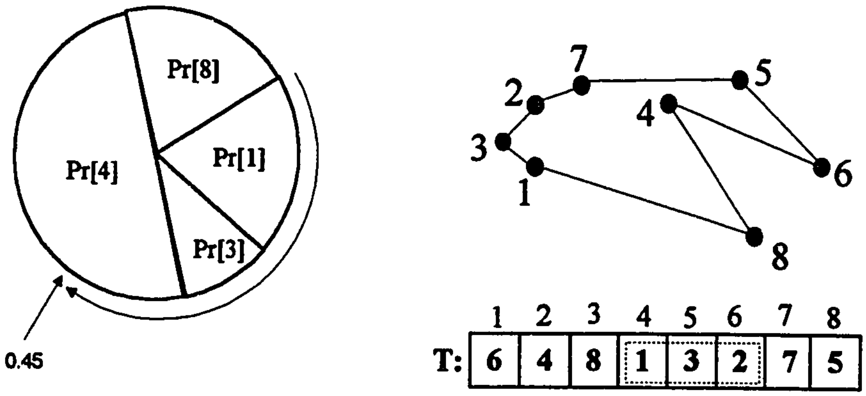


FIGURE 15: Example for roulette wheel selection

The same process is repeated. However, when the iteration number is even, the *Point Heuristic* operator is used for changing the current tour. A random number from 1 to 8 is generated for each position on the tour. This will give the following tour changes.

	1	2	3	4	5	6	7	8
V :	5	4	7	3	3	1	②	1

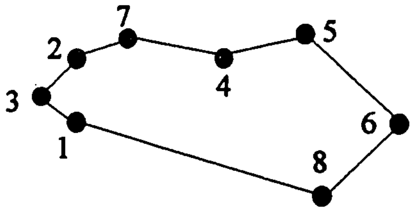
Assuming the iteration number is even, then we use the *Point Heuristic* operator. For example, V[7] = 2 means that the current tour can be changed by inserting city T[2] after city T[7]. Then, the probability of selection for each change {Pr[1], Pr[2], ..., Pr[8]} is computed using equation (43). For example

$$\Pr[3] = \frac{\Delta T_3}{\sum_{u=1}^{8} \Delta T_u} = \frac{\text{noise} + d(3,7) + d(7,4) + d(6,8) - d(3,8) - d(7,6) - d(7,8)}{\Delta T_1 + \Delta T_2 + \ldots + \Delta T_8} = 0.1.$$

	1	2	3	4	5	6	7	8
Pr :	0	.1	.1	0	0	0	.5	.3

Next, the roulette wheel is used to represent the probabilities of selection. Then, a random number in the range (0.0-1.0) is generated, say 0.3. Then, using the roulette wheel the change V[7] is selected because the probabilities of V[1]+V[2]+V[3]+V[4]+V[5]+V[6]+V[7] = 0+0.1+0.1+0+0+0+0.5 = 0.7 ≥ 0.3. The following tour will be the result.

	1	2	3	4	5	6	7	8
T :	6	8	1	3	2	7	④	5



These steps are repeated continuously until the tour cannot be improved further as shown in Figure 16.

3.3 Comparisons between the ECEN and Previous Work

In this section the *Enhanced Convex-Elastic Net* algorithm is compared with other well known algorithms such as *Elastic Net, Lin and Kernighan Heuristic, Jun Gu Local Search Technique, Nearest Neighbor Heuristic, 2-Optimal* and *3-Optimal Heuristics.*

The *Convex-Elastic Net* algorithm uses the same basic idea of the *Elastic Net* algorithm where the cities coordinates are mapped onto a ring of output units to define an ordering of the cities [18], but there are two differences between these two algorithms. First, the *Elastic Net* algorithm starts with a rubber band or ring located at the center of the cities. The *Convex-Elastic Net* algorithm, however, starts with a rubber band that is located at the convex hull of the cities. Second, the *Convex-Elastic Net* algorithm has a different equation for updating the weights representing the rubber band.

Also, there is a difference between the *Convex-Elastic Net* and the *Nearest Neighbor Heuristic.* A path is constructed in the *Nearest Neighbor Heuristic* while in the *Convex-Elastic Net* algorithm a tour, not a path, exists all the time [34].

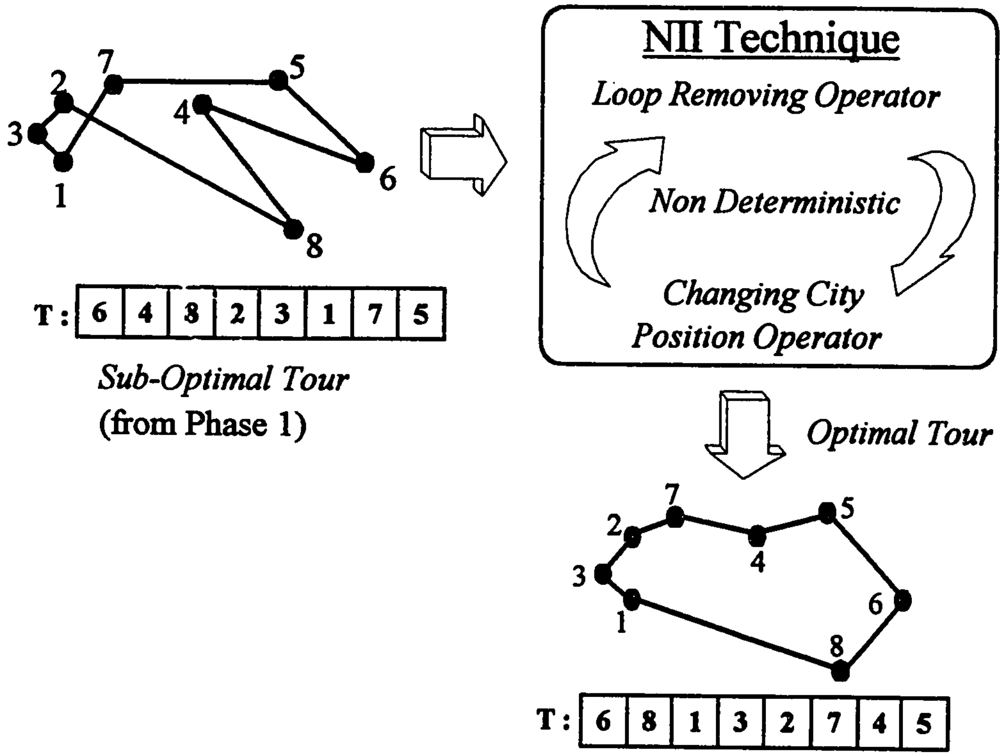


FIGURE 16: Example for the NII algorithm that enhance a given tour

There is a difference between the *Non-Deterministic Iterative Improvement* technique used in the *Enhanced Convex-Elastic Net* and other iterative heuristics, for example *Lin and Kernighan, 2-Opt* and *3-Opt Heuristics.* In the *Lin and Kernighan Heuristic,* deterministic decisions are taken to produce new tours which are very unlikely to escape from the local minimum [55]. On the other hand, the *Non-Deterministic Iterative Improvement* technique can escape several local minimums by accepting unfavorable tours with the random noise generated at each cycle. The NII algorithm uses a different updating rule than that used in *2-Opt* and *3-Opt Heuristics. 2-Opt* and *3-Opt Heuristics* locally modify the current solution by replacing k-arcs in the tour by other k-arcs such that the new tour length is reduced [55]. In contrast, the NII algorithm moves the city from a position on the tour to another.

Both the *Jun Gu Technique* [29] and the NII algorithm are local search techniques that use different approaches to escape the local minimum problem. In the *Jun Gu Technique,* search space smoothing approximates the original space so that some local minimums are skipped. In the NII algorithm, however, the changes in the tour that decrease the tour length mostly are selected with high probability and vice versa. Thus, the probability of being trapped in a local minimum point is decreased and the probability of finding the global minimum is increased.

CHAPTER 4

SIMULATION RESULTS

4.1 Introduction

The proposed algorithm has been implemented in Turbo C 2.0 language and executed on Pentium 75 MHz personal computer. Since worst case analysis for E-TSP is as hard as finding the optimal, we performed probabilistic analysis only.

This chapter is organized as follows. In Sections 2 and 3, the performance of our algorithm is tested against two types of E-TSP instances: Randomly generated cities coordinates within one unit square and instances obtained from the literature with best known optimal tour length. Several test problems in this thesis are taken from the literature in order to compare our algorithm with the experimental results of other researchers. In Section 4, the time and space analysis for the proposed *Enhanced Convex-Elastic Net* algorithm are presented.

4.2 Randomly Generated Instances

In comparing our results with the optimal tour length, the expected length of the optimal tour which is found by Stein in 1977 is used. The Stein formula for N cities distributed uniformly over a unit square is given by the following equation [44].

$$0.765 * \sqrt{N} \leq \text{Optimal tour length} \leq (0.765 + 4/N) * \sqrt{N} \qquad (44)$$

In this thesis, $0.765 * \sqrt{N}$ was taken as the presumed lower bound optimal solution and $(0.765+4/N)*\sqrt{N}$ was taken as the presumed upper bound optimal solution. All the comparisons in the following experiments were based on the percentage difference between the ECEN tour and the theoretical optimal tour length. This percentage difference is defined as follows.

$$\frac{\text{ECEN Algorithm Tour Length} - \text{Optimal Tour Length}}{\text{Optimal Tour Length}} * 100\,\% \qquad (45)$$

The following are the simulation results for 100 E-TSPs using the *Enhanced Convex-Elastic Net* technique where cities coordinates are generated uniformly at a unit square and the average time, tour length and number of iterations are recorded (See Table 2).

Number of Cities	Optimal Tour Length Bounds		Enhanced Convex-Elastic Net			% Difference between ECEN & Optimal Tour Length
	Lower	Upper	Avereg Tour Length	Avereg Time (seconds)	Avereg Iterations	
10	2.42	3.68	2.47	.52	153.2	0-2.0 %
30	4.19	4.92	4.32	1.54	665.2	0-3.1 %
50	5.41	5.98	5.57	3.21	1239.6	0-2.9 %
100	7.65	8.05	8.04	9.02	2479	0-5.1 %
150	9.37	9.70	9.86	17.97	3669.7	1.6-5.2 %
200	10.82	11.10	11.40	28.49	4766.7	2.7-5.4 %
250	12.10	12.35	12.78	43.93	5940	3.5-5.6 %
300	13.25	13.48	14.01	60.85	7047.9	3.9-5.7 %
325	13.79	14.01	14.65	70.23	7607.1	4.6-6.2 %

TABLE 2: The enhanced convex-elastic net technique results for 100 samples

As can be seen from simulation results, the computation times for our algorithm were reasonable. Nevertheless, the computation time for simulating a neural network on a serial computer is not particularly meaningful, because this implementation did not exploit the parallelism of the model. As a result, computation times are usually missing from the research papers of neural networks.

Using the data of Table 2, we can compare the solution found for 100 E-TSPs with the estimated optimal solution as shown in Figure 17. This figure shows the results after applying the first and the second phases of the ECEN algorithm. It indicates that the enhancement technique used in the second phase of the algorithm is essential to get near the optimal solution. Also, it is clear that our proposed approach can discover a solution that is near optimal after examining an extremely small fraction of possible solutions.

Table 3 represents a comparison of several methods for solving the E-TSP [8] with the new approach. It was found that, given some initial random coordinates, the new technique can find a sub-optimal solution.

Figure 18 represents a comparison for the tour length of several techniques for solving the E-TSP including the new method. It is evident that the *Enhanced Convex-Elastic Net* algorithm empirical performance is much better than several known techniques such as *Simulated Annealing* [39], *Guilty Net* [9], *Space Filling Curve*, and *Hopfield and Tank Model* [10] in terms of the average value of the tour length.

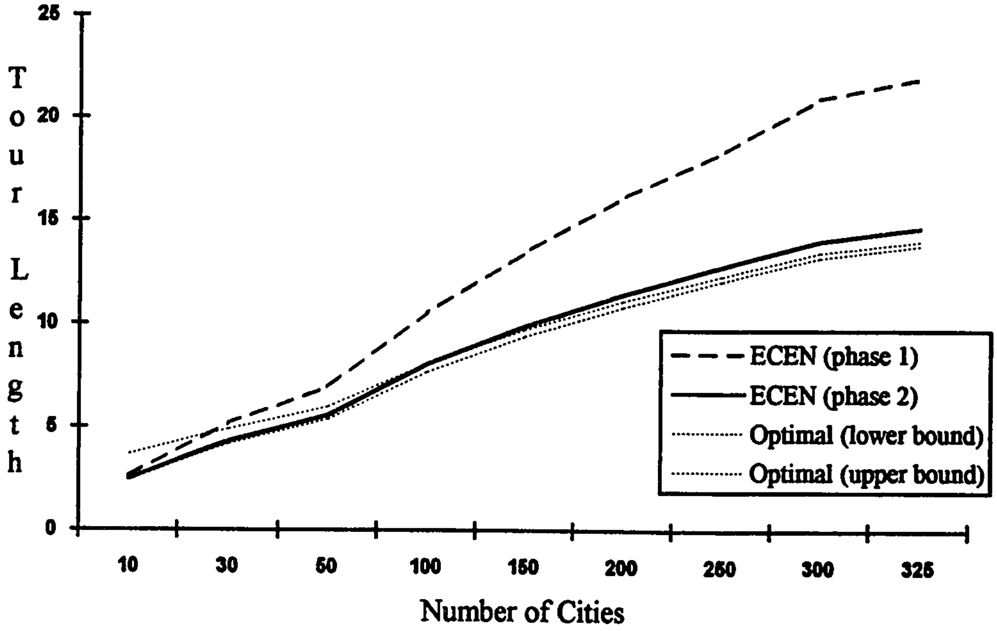


FIGURE 17: Comparison between the ECEN and the optimal solution

	10 Cities	30 Cities	50 Cities	100 Cities
Optimal Tour	2.42-3.68	4.19-4.92	5.41-5.98	7.65-8.05
ECEN	2.47	4.32	5.57	8.04
Space Filling Curve	3.14	5.45	7.03	9.56
Hopfield and Tank	2.92	5.12	6.64	--
Guilty Net	2.78	4.81	6.21	8.81
Simulated Annealing	2.74	4.75	6.13	8.40
Elastic Net	2.45	4.51	5.58	8.23

TABLE 3: Simulation results for the ECEN technique

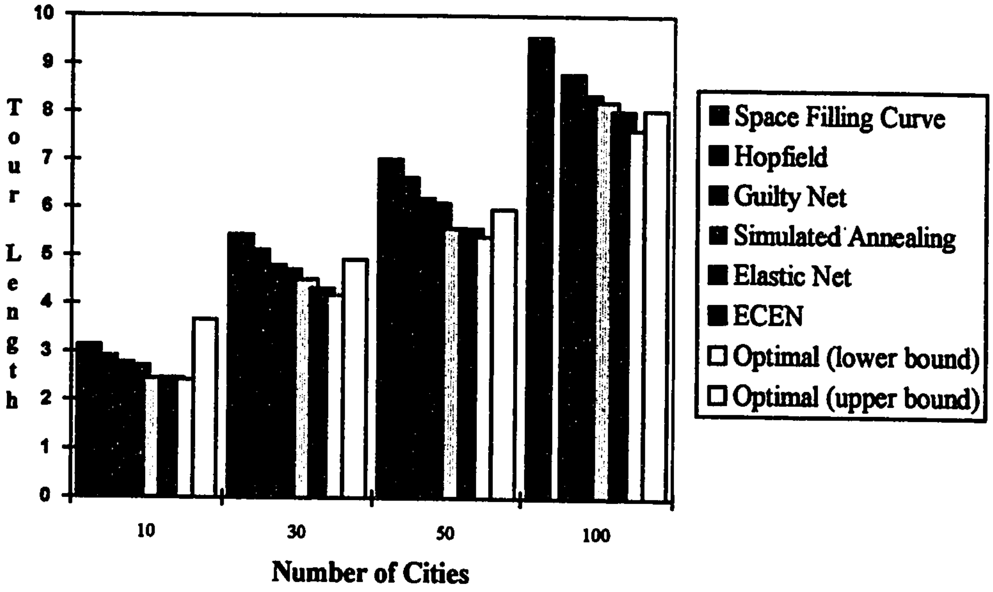


FIGURE 18: Comparison of several methods for solving E-TSP

4.3 Instances Taken from the Literature

In this section, we present five examples taken from the operation research literature which were treated by our computer program. In these examples, the execution time for a trail varies between 8 and 71 seconds for problems ranging from 96 to 318 cities. Since the optimal tour length for each of these problems is known, we observed that our tour was never more than 5% longer than the optimal solution.

In Table 4 we display the results of different algorithms for two 100 city E-TSPs namely GRID100 and KAR100 [25,27]. The GRID100 is a 100 city problem described in Cerny (1985) [12]. In this problem, 100 cities are arranged in a rectangular lattice, such that the nearest distance between any two cities is one unit. As a result, the optimal tour length is clearly 100 units as shown in Figure 19. This optimal tour can be found using *Generalized Insertion Heuristic* or *Simulated Annealing*. After 11 seconds, our ECEN algorithm found the tour given in Figure 20 for this problem.

The KAR100 is a 100 city problem due to Patrick Karolak (1971) with coordinates given in [41] as problem #24. A typical tour for the KAR100 E-TSP found by our *Enhanced Convex-Elastic Net* algorithm is of length 21,622.9 units, that is 1.6% away from the optimal. This algorithm finds this tour in 9 seconds. Figures 21 and 22 show both the optimal tour (21,282 units), as well as, the tour generated by the ECEN algorithm.

Algorithm	GRID100 Problem	KAR100 Problem
ECEN	0.8 %	1.6 %
2-Opt (once)	3.3 %	7.8 %
CCA	6.2 %	1.1 %
CCAO	2.5 %	0.2 %
GENI	0 %	2.4 %
GENIUS	0 %	0 %
Nearest Neighbor	--	16.7 %
Nearest Insertion	--	18.7 %
2-Opt (best of 25 runs)	--	1.11 %

TABLE 4: The difference between the GRID100 & KAR100 and the optimal

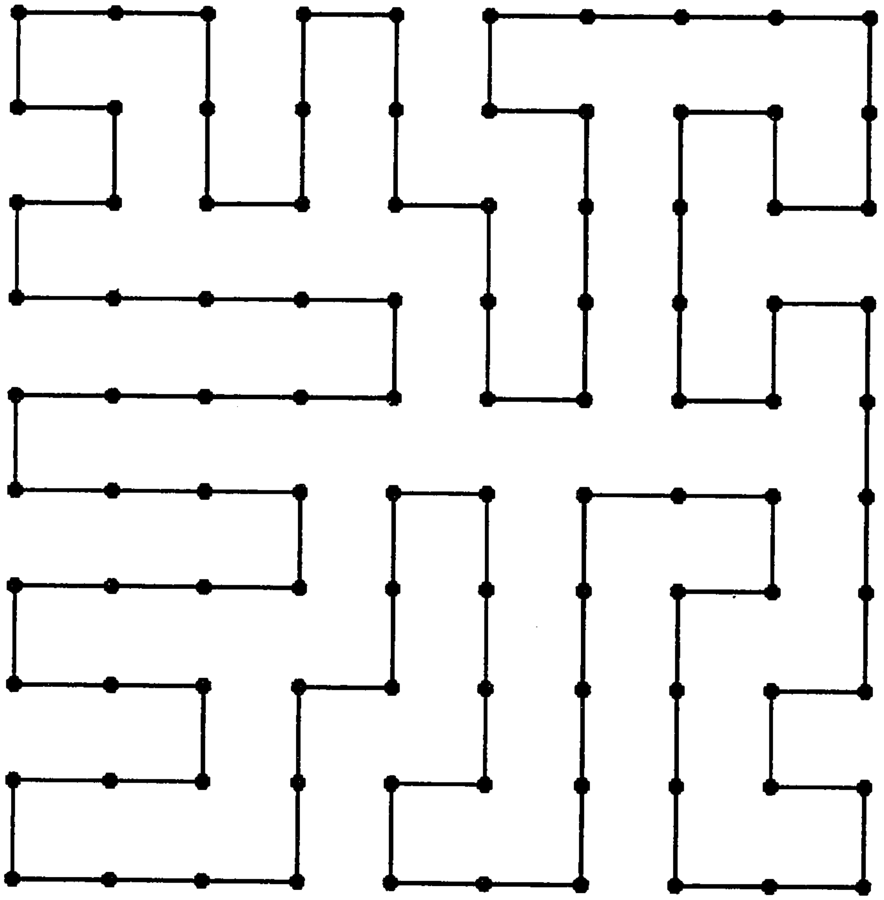


FIGURE 19: The optimal tour for the GRID100 E-TSP

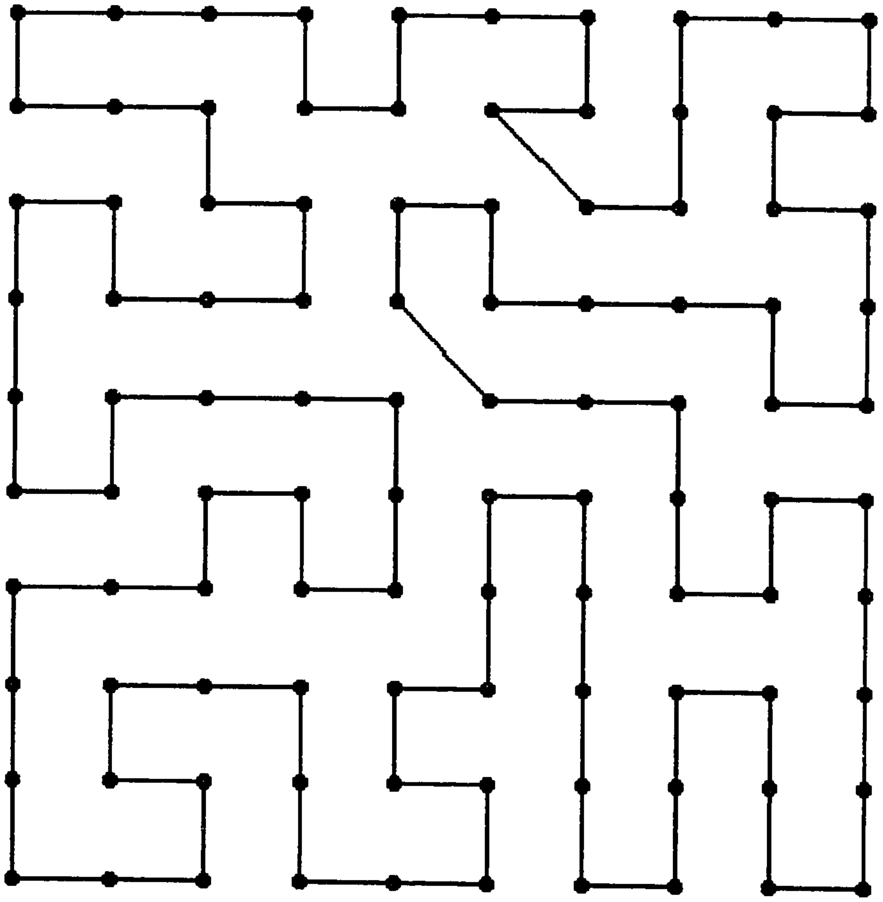


FIGURE 20: The tour found by the ECEN for the GRID100 E-TSP

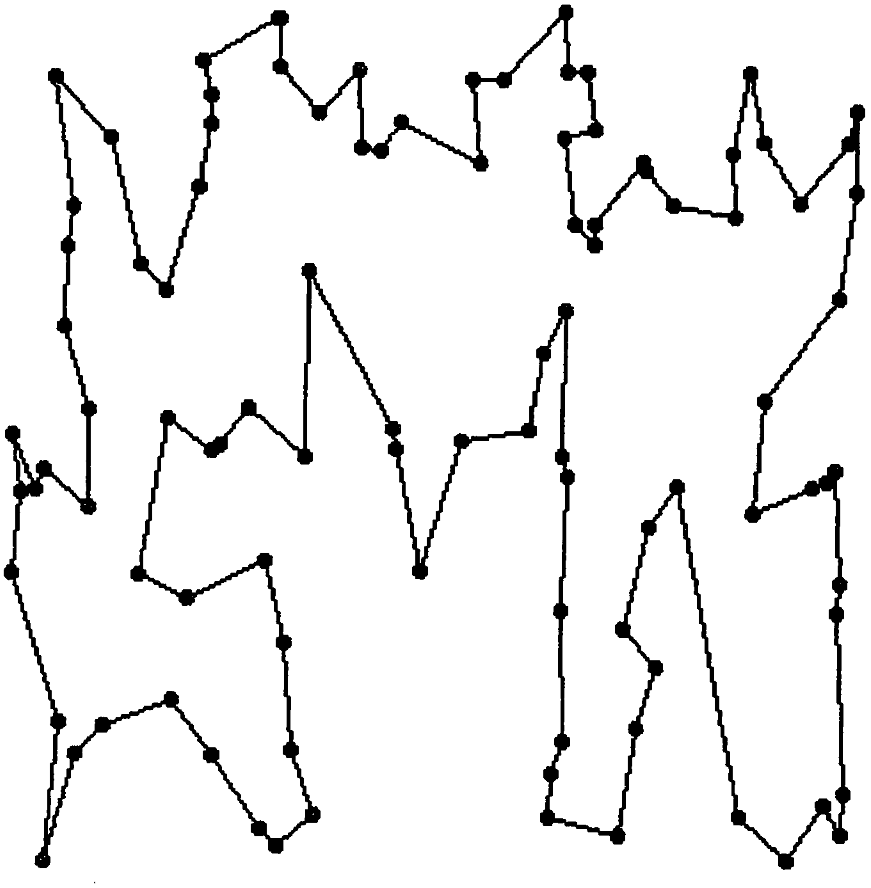


FIGURE 21: The optimal tour for the KAR100 E-TSP

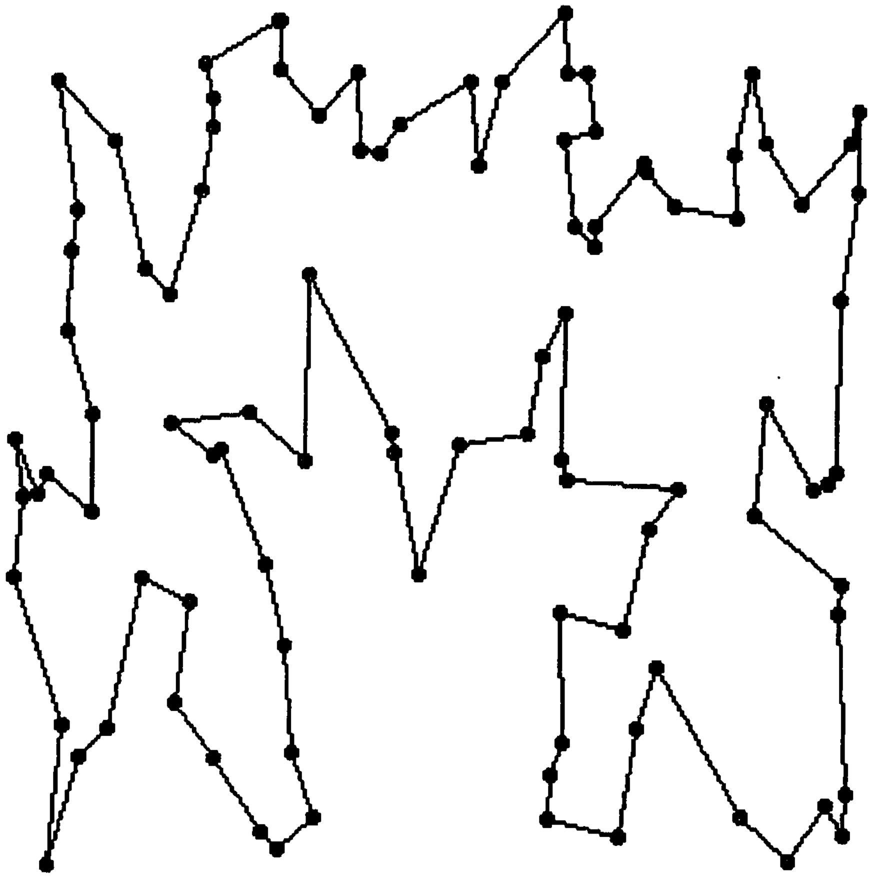


FIGURE 22: The tour found by the ECEN for the KAR100 E-TSP

The LIN318 is the famous 318 city problem due to Lin and Kernighan [45]. Solving this problem using exhaustive search on a computer that can enumerate 10^9 tours per second takes more than 10^{637} years. This problem was solved by several authors as a large scale E-TSP to compare their techniques [15,52]. On the other hand, it is difficult to compare our results with the results given by Lin and Kernighan for the 318 city problem, because *Lin and Kernighan Technique* finds a path instead of a closed tour. Our algorithm, however, generates a closed tour. For this reason, we compare our results with the tour found by the *Most Eccentric Ellipse Choice Mechanism* given in [52] as shown in Figure 23. Using this technique, the best known tour is found after 26.5 minutes on a Univac 1110 machine. However, the ECEN solves the LIN318 problem in 1.2 minutes with a sub-optimal tour of length 46,231.7 units (See Figure 24) that is within 4.7% of the best known tour.

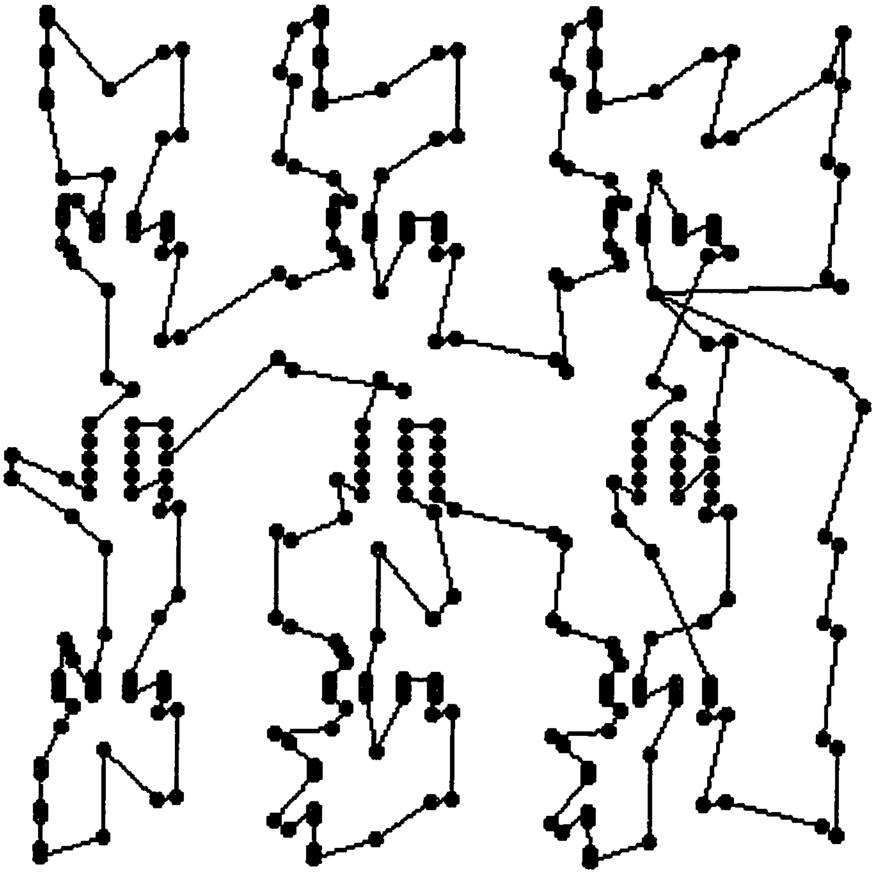


FIGURE 23: The tour for the LIN318 E-TSP using eccentric ellipse choice method

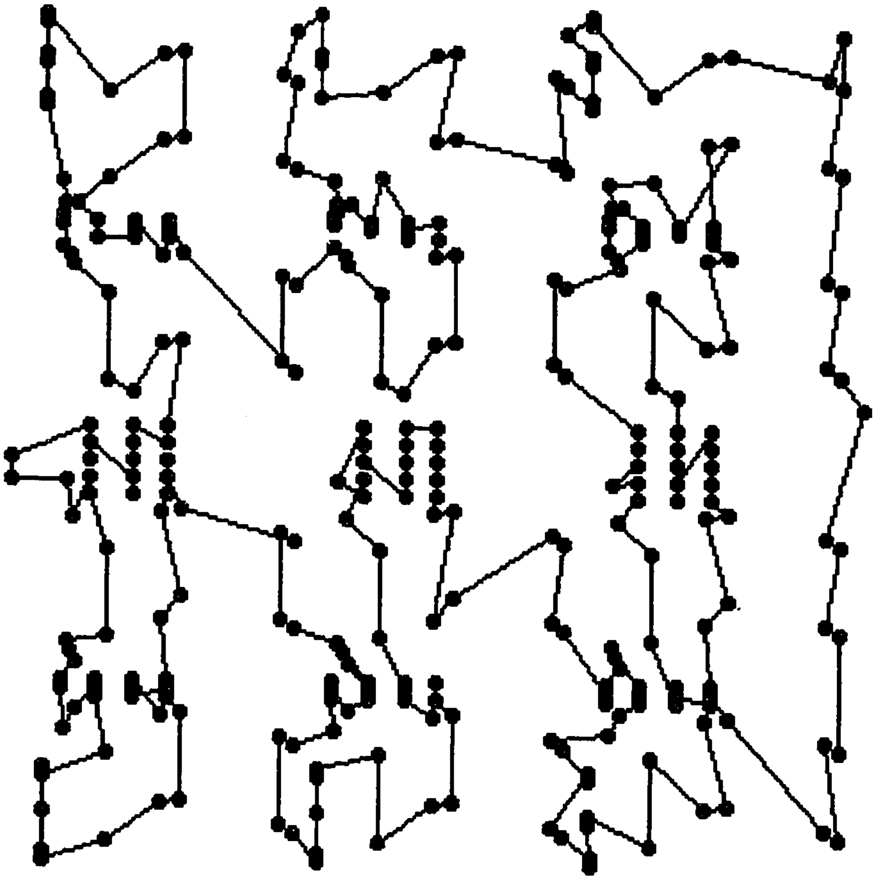


FIGURE 24: The tour found by the ECEN for the LIN318 E-TSP

Two problems based on road map distances are solved by the *Enhanced Convex-Elastic Net* technique using great circle distances that are given by Martin Grotschel (1991) in [28] instead of the Euclidean distances. The first problem is a 96 cities located in Africa and the second problem is a 137 cities located in the United States. The 96 city problem is called GR96 and has an optimal tour of length 55,209 units, while the 137 city problem is called GR137 and has an optimal tour of length 69,853 units as shown in Figures 25 and 26 respectively. These two optimal tours are found using the *City Plane Procedure*.

When the ECEN algorithm is applied on the GR96 and GR137 problems, the tours found are 57,634.1 (Figure 27) and 72,150.4 units (Figure 28), respectively. As can be seen from these results, our approach can produce a tour of length 4.4% longer than the optimal solution for the GR97 problem, and 3.2% longer than the optimal solution for the GR137 problem.

Table 5 shows the simulation results for the five problems taken from the operation research literature. This table indicates that the *Enhanced Convex-Elastic Net* algorithm can find a sub-optimal solution for these problems that is less than 5% from the optimal.

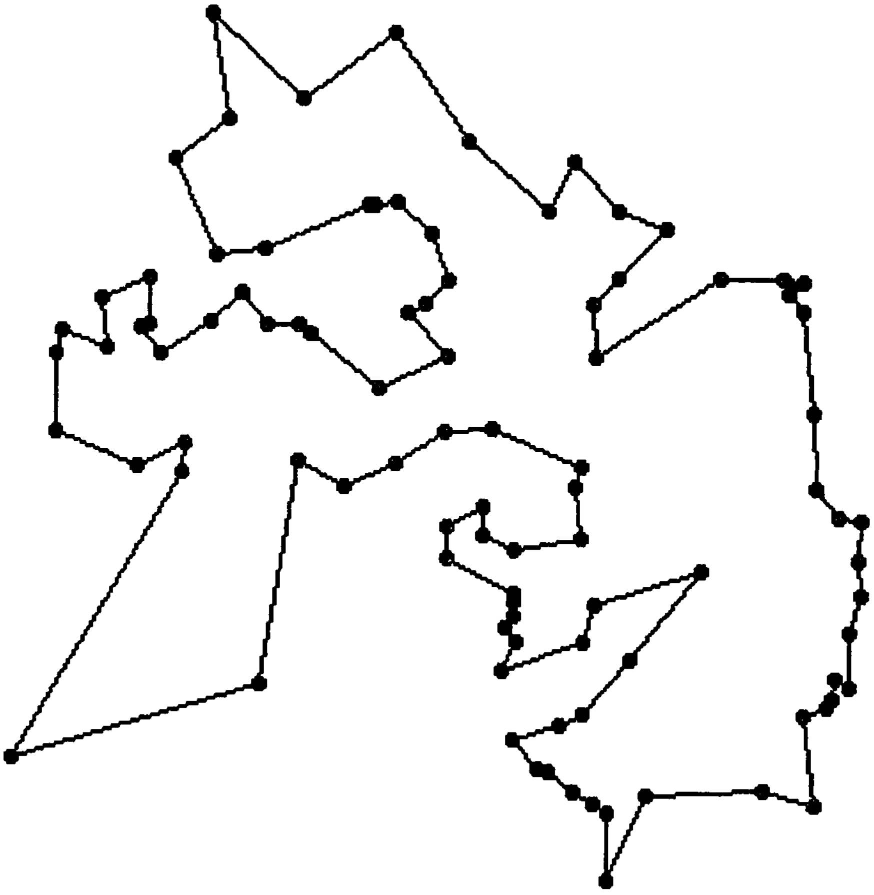


FIGURE 25: The optimal tour for the GR96 E-TSP

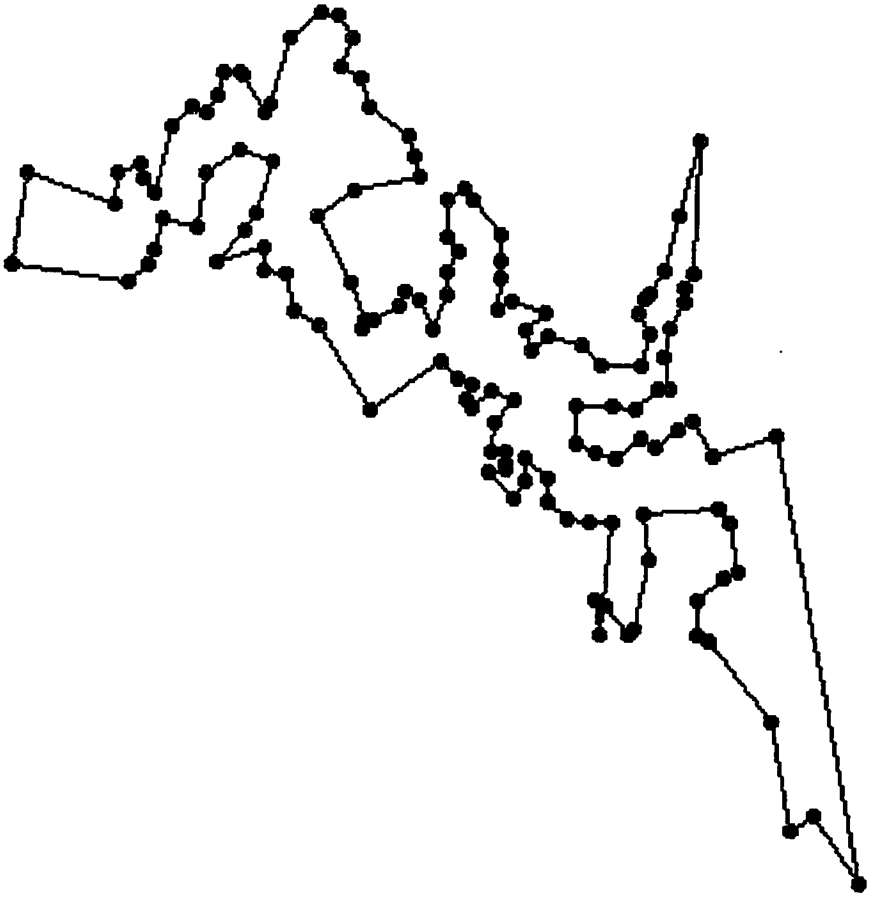


FIGURE 26: The optimal tour for the GR137 E-TSP

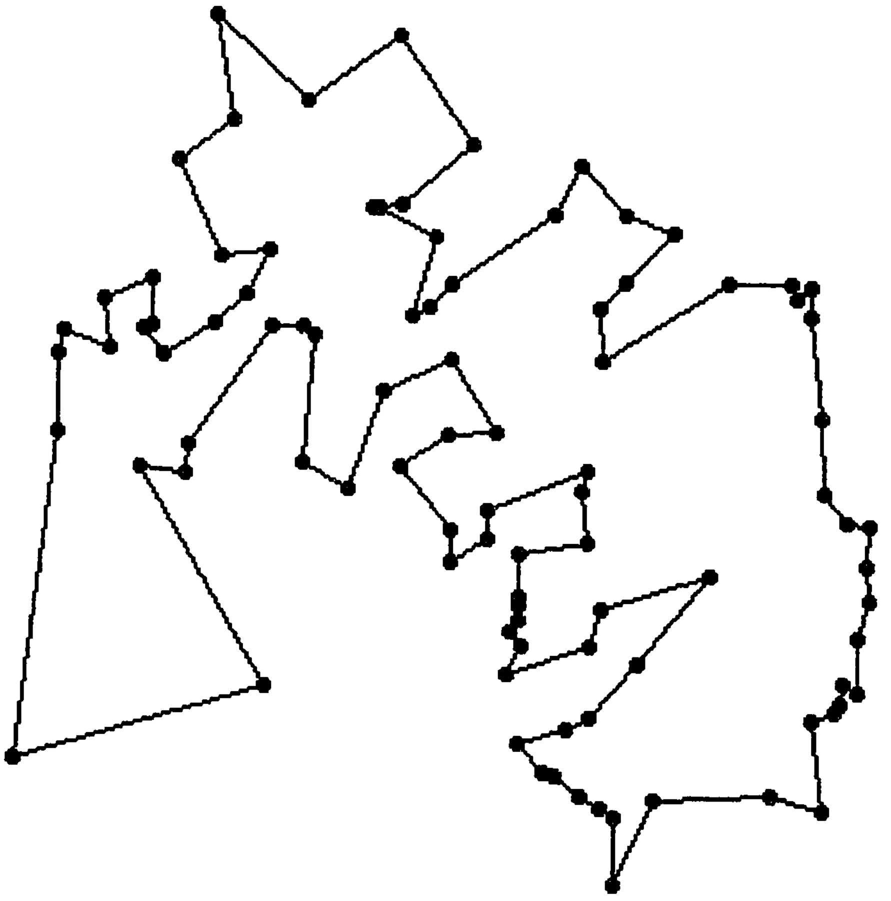


FIGURE 27: The tour found by ECEN for the GR96 E-TSP

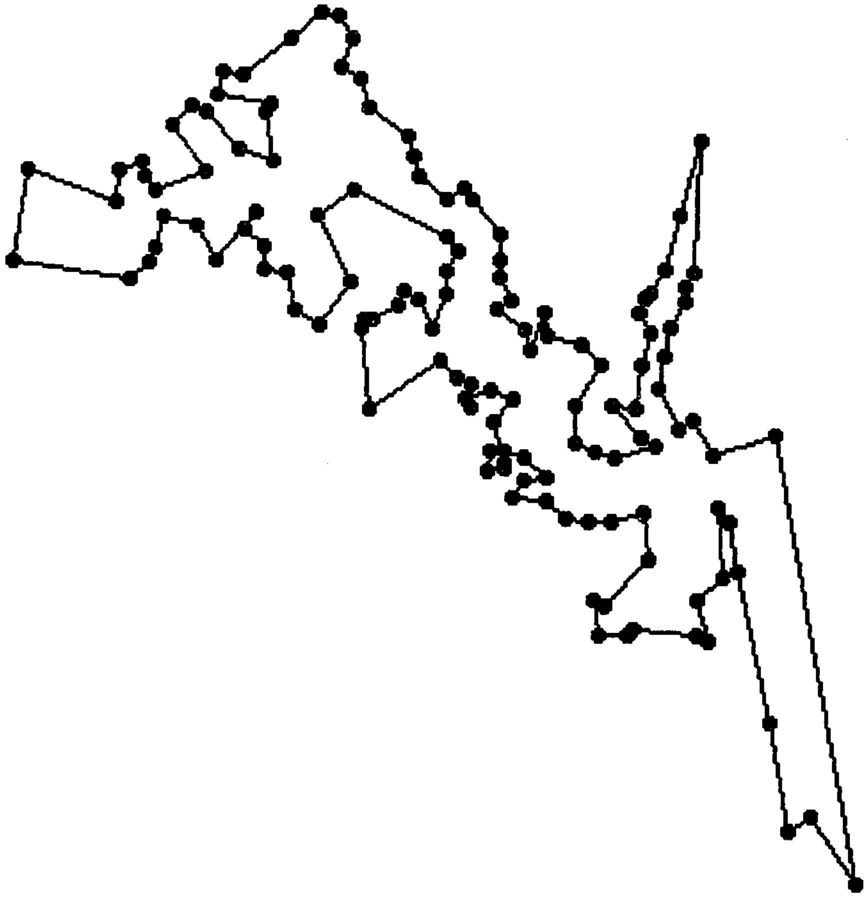


FIGURE 28: The tour found by the ECEN for the GR137 E-TSP

Problem	Number of cities	Optimal / Best Known Tour Length	Enhanced Convex-Elastic Net	
			Tour Length	% Difference
GRID100 [12]	100	100 [25]	100.8	0.8 %
KAR100 [41]	100	21,282 [25]	21,622.9	1.6 %
LIN318 [45]	318	44,169 [52]	46,231.7	4.7 %
GR96 [28]	96	55,209 [28]	57,634.1	4.4 %
GR137 [28]	137	69,853 [28]	72,150.4	3.2 %

TABLE 5: Simulation results for several problems from the OR literature

4.4 Time & Space Analysis

The approach proposed in this thesis (ECEN) consists of two techniques, namely the CEN and NII algorithms. Both techniques cooperate to find a sub-optimal solution for the given E-TSP. For this reason, the time required to solve an E-TSP equals the total time needed by the CEN plus the time needed by the NII algorithm. On the other hand, the space required by the ECEN equals the space needed by the CEN or NII algorithms.

4.4.1. The CEN Time & Space Analysis

The running time of the *Convex-Elastic Net* algorithm can be established as follows. Finding the convex hull takes $O(N \lg N)$ time. Finding the winning neuron for a full presentation of cities takes $O(N^2)$ time. The *Convex-Elastic Net* algorithm total time equals $O(N \lg N) + \text{number of iterations} * O(N^2)$. Since the number of iterations is $O(N)$, its running time is equal to $O(N^3)$.

The *Convex-Elastic Net* algorithm needs $O(N^2)$ space to store the Euclidean distances between cities because this will decrease the computation time of the algorithm and $O(N)$ for the weights array. Thus, $O(N^2)$ total space is needed by the CEN algorithm.

4.4.2. The NII Time & Space Analysis

The running time of the NII algorithm can be established as follows. Finding the possible change in the tour, computing probability and applying a change to the tour based on the probability take $O(N)$ time. Since the number of iterations is $O(N)$, the total running time of the NII algorithm is $O(N^2)$.

The NII algorithm needs 3*N space to store the current tour, find possible changes in the tour and compute probabilities of selection. Thus, $O(N)$ total space is needed by the NII algorithm.

4.4.3. The ECEN Time & Space Analysis

The time required by the ECEN algorithm is equal to $O(N^3)+O(N^2)=O(N^3)$. The space required by the *Enhanced Convex-Elastic Net* algorithm is $O(N^2)+O(N)=O(N^2)$. The time and space analysis for the *Enhanced Convex-Elastic Net* algorithm is briefly presented in Table 6.

	Convex-Elastic Net	NII	Enhanced Convex-Elastic Net
Time	O(N^3)	O(N^2)	O(N^3)
Space	O(N^2)	O(N)	O(N^2)

TABLE 6: Time and space analysis for the enhanced convex-elastic net

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Summary

In this thesis, several ways for mapping the Euclidean Traveling Salesman Problem onto the neural network models have been presented. A modified *Elastic Net* algorithm for finding approximate solutions to the Euclidean Traveling Salesman Problem is developed. This technique is developed using two other techniques: the *Convex-Elastic Net*, and the *Non-Deterministic Iterative Improvement* technique. In the first phase, the *Convex-Elastic Net* algorithm is used as a global search technique that generates an initial tour for the second phase by enforcing the E-TSP constraints. In the second phase, the tour found is tuned further and iteratively improved by making local changes to it. The two techniques introduced are combined into a new approach called the *Enhanced Convex-Elastic Net.*

The major contributions of this thesis include the followings. First, the development of a neural-based algorithm for solving the E-TSP that consists of two other techniques: CEN and NII. Second, the enhancement for the *Elastic Net* technique for the E-TSP by starting with a rubber band located at the convex hull of the cities instead of a ring located at the center of the cities. Third, the development of a non deterministic iterative heuristic that enhances the tour produced by the *Convex-Elastic Net.* Fourth, the developed algorithm (ECEN) is polynomial in terms of time and space.

The experimental results of this study have demonstrated that the developed algorithm can get quite close to the optimal solution of the E-TSP. Also, the results compare favorably with solutions given by both traditional operation research techniques and neural-based techniques reported in the literature.

5.2 Conclusions

In this thesis, we have introduced an *Enhanced Convex-Elastic Net* algorithm for solving the E-TSP that yields feasible solutions and compares favorably with several new approaches for randomly generated instances. This technique appears promising due to the following features:

1) The simulations indicate that the *Enhanced Convex-Elastic Net* algorithm is not infallible, but rather that it gives good solutions in a computationally feasible amount of computer time.

2) Unlike the *Hopfield and Tank Model*, the ECEN does not produce infeasible solutions and there is no need to set the energy function parameters for each problem size [17].

3) It generates sub-optimal solutions for known traditional problems and the results compare favorably with solutions given by traditional operation research techniques such as *Simulated Annealing* and *Space Filling Curve*.

4) It generates sub-optimal problem solutions that are better than the results generated by the neural-based techniques reported in the literature such as *Hopfield and Tank Model* [10], *Guilty Net*, and *Elastic Net*.

5) It scales well with problem size.

6) The ECEN provides a greater degree of robustness or fault tolerance than sequential algorithms because there are many nodes, each with local connections. The damage to a few nodes or links thus need not decrease the overall performance significantly.

7) It can escape from many local minimums by exploring a larger number of solutions. This is similar to the *Simulated Annealing* technique in that it allows local modifications to the tour that increases the length of the tour with some probability of acceptance [50].

In conclusion, we can say that the *Enhanced Convex-Elastic Net* algorithm gives better performance in solving E-TSPs by exploiting the topological order of the cities. For this reason, this method can be extended only to other optimization problems that have path finding nature.

5.3 Recommendations for Future Studies

The *Enhanced Convex-Elastic Net* algorithm proposed has some drawbacks. First, it is a geometrical algorithm that can be extended to more general TSP with cities in any Euclidean space, but not the case where a matrix of cities distances are given. This limitation is due to the use of the *Elastic Net*. For this reason, the ECEN technique is limited to E-TSPs and is not flexible enough to handle general TSPs. Also, the ECEN technique can only accept cities coordinates as an input and cannot accept the matrix for distances between cities.

Based on the above limitations for the *Enhanced Convex-Elastic Net* technique some suggestions should be mentioned. First, future studies can investigate the possiblitt of implementing the *Enhanced Convex-Elastic Net* on parallel architecture which is potentially faster than sequential architecture.

Second, future studies can explore the application of neural networks to other variations of the TSP and other combinatorial optimization problems. Third, future studies should continue in investigating the promising technique that integrates other operation research heuristics and artificial neural networks [7] to provide a solution for optimization problems at speed that was never achieved before.

REFERENCES

[1] Aiyer, Sreeram, Mahesan Niranjan, and Frank Fallside, "A Theoretical Investigation into the Performance of the Hopfield Model". *IEEE Transactions on Neural Networks*, Vol. 1, no. 2, June 1990, pp. 204-215.

[2] Aiyer, S. V. B., M. Niranjan, and F. Fallside, "On the Optimization Properties of the Hopfield Model". *IJCNN*, Vol. I, July 9-13, 1990, pp. 245-248.

[3] Bellmore, M. and G. L. Nemhauser, "The Traveling Salesman Problem: A Survey". *Operations Res.*, Vol. 16, May 1968, pp. 538-558.

[4] Bhide, Shirish, Nigel John, and M. R. Kabuka, "A Boolean Neural Network Approach for the Traveling Salesman Problem". *IEEE Transactions on Computers*, Vol. 42, no. 10, October 1993, pp. 1271-1278.

[5] Boeres, Maria and Luis Carvalho, "A Faster Elastic-Net Algorithm for the Traveling Salesman Problem". *IJCNN*, vol. II, June 7-11, 1992, pp. 215-220.

[6] Bounds, David G., "New Optimization Methods from Physics and Biology". *Nature*, Vol. 329, 17 September 1987, pp. 215-219.

[7] Braham, R. and Ben Daya M., "Neural Networks and Mathematical Programming". *Premiere Journee de l'optimisation des Systems Indutriels (ISOD'95)*, Tunis, 20 July 1995, pp. 29-42.

[8] Burke, Laura and Poulomi Damany, "The Guilty Net For the Traveling Salesman Problem". *Computers Ops Res*, Vol. 19, no. 3/4, 1992, pp. 255-265.

[9] Burke, Laura, "Neural Methods for the Traveling Salesman Problem: Insights from Operations Research". *Neural Networks*, Vol. 7, no. 4, 1994, pp. 681-690.

[10] Carvalho, L. and V. Barbosa, "A TSP Objective Function that Ensures Feasibility at Stable Points". *INNC*, Vol. 1, July 9-13,1990, pp. 249-253.

[11] Cavalieri, S., A. Di Stefano, and O. Mirabella, "Optimal Path Determination in a Graph by Hopfield Neural Network". *Neural Networks*, Vol. 7, no. 2, 1994, pp. 397-404.

[12] Cerny, V., "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm". *Journal of Optimization Theory and Applications*, Vol. 45, No. 1, January 1985, pp. 41-51.

[13] Chu, P. Pong, "A Neural Network for Solving Optimization Problem with Linear Equality Constraints". *Neural Parallel & Scientific Computations 1*, 1993, pp. 71-82.

[14] Cormen, T., C. Leiserson, and R. Rivest, *Introduction to Algorithms*. The MIT Press, McGraw-Hill, 1990, pp. 899-905 & 960.

[15] Crowder, Harlan and Manfred Padberg, "Solving Large-Scale Symmetric Traveling Salesman Problems to Optimality". *Management Science*, Vol. 26, No. 5, May 1980, pp. 495-509.

[16] Davis, Lawrence and Martha Steenstrup, "*Genetic Algorithms and Simulated Annealing*". Morgan Kaufmann Publishers, Inc., Los Altos, California, 1987, pp. 1-11.

[17] Davis, W. Gerald, "Sensitivity Analysis in Neural Net Solutions". *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, no. 5, September/October 1989, pp. 1078-1082.

[18] Durbin, Richard and David Willshaw, "An Analogue approach to the Traveling Salesman Problem Using an Elastic Net Method". *Nature*, Vol. 326,16 April 1987, pp. 689-691.

[19] Fang, Luyuan, William Wilson, and Tao Li, "Mean Field Annealing Neural Net for Quadratic Assignment". *INNC*, Vol. 1, July 9-13,1990, pp. 282-286.

[20] Fausett, Laurene, *Fundamentals of Neural Networks: Architecures, Algorithms, and Applications*. Prentice-Hall, Inc., New Jersey, 1994, Chapter 7.

[21] Fogel, David B., "An Introduction to Simulated Evolutionary Optimization". *IEEE Trans. on Neural Networks*, Vol. 5, No. 1, January 1994, pp. 3-14.

[22] Folk, R. and A. Kartashov, "A Simple Elastic Model for Self-Organizing Topological Mappings". *Networks: Computation in Neural Systems*, Vol. 5, 1994, pp. 369-387.

[23] Gary, Michael R. and David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman & Co., San Francisco, 1979.

[24] Gee, Andrew and Richard Prager, "Limitations of Neural Networks for Solving Traveling Salesman Problems". *IEEE Trans. on Neural Networks*, Vol. 6, No. 1, January 1995, pp. 280-282.

[25] Gendreau, Michel, Alain Hertz, and Gilbert Laporte, "New Insertion and Post Optimization Procedures for the Traveling Salesman Problem". *Operations Research*, Vol. 40, no. 6, November-December 1992, pp. 1086-1094.

[26] Glover, Fred, and Harvey Greenberg, "New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence". *European Journal of Operational Research*, Vol. 39,1989, pp. 119-130.

[27] Golden, B., L. Bodin, T. Doyle, and W. Stewart, "Approximate Traveling Salesman Algorithms". *Operations Research*, Vol. 28, no. 3, Part II, May-June 1980, pp. 694-711.

[28] Grotschel, Martin and Olaf Holland, "Solution of Large-Scale Symmetric Traveling Salesman Problems". *Mathematical Programming*, Vol. 51, 1991, pp. 141-201.

[29] Gu, Jun, "Efficient Local Search With Search Space Smoothing: A Case Study of the Traveling Salesman Problem (TSP)". *IEEE Transaction on Systems, Man, Cybernetics*, Vol. 24, no. 5, May 1994, pp. 728-735.

[30] Held, M., A. Hoffman, E. Johnson, and P. Wolfe, "Aspects of the Traveling Salesman Problem". *IBM J. Res. Develop.*, Vol. 28, No. 4, July 1984, pp. 476-486.

[31] Hertz, John, Anders Krogh, and Richard Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Company 1991.

[32] Igarashi, Harukazu, "An Estimation of Parameters in an Energy Function Used in a Simulated Annealing Method". *IJCNN*, Vol. IV, June 7-11, 1992, pp. 480-485.

[33] Jayadeva, B. Bhaumik, "A neural Network for the Steiner Minimal Tree Problem". *Biological Cybernetics*, Vol. 70, 1994, pp. 485-494.

[34] Johnson, S. David, "Local Optimization and the Traveling Salesman Problem". *Proceedings of the 17th International Colloquium on Automata, Languages and programming*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1990, pp. 446-461.

[35] Johnson, David, "More Approaches to the Traveling Salesman Guide". *Nature*, Vol. 330, 10 December 1987, p. 525.

[36] Joppe, Aart, Jan Bioch and Helmut Cardon, "A Neural Network for Solving Traveling Salesman Problem on the Basis of City Adjacency in the Tour". *IJCNN*, Vol. I, July 9-13, 1990, pp. 254-257

[37] Kamgar-Parsi, Behzad and Behrooz Kamgar-Parsi, "Dynamical Stability and Parameters Selection in Neural Optimization". *IJCNN*, Vol. IV, June 7-11, 1992, pp. 566-571.

[38] Khanna, Tarun, *Foundations of Neural Networks*. Addison-Wesley Publishing Company, 1990, pp. 143-148.

[39] Kirkpatrick, Scott, "Optimization by Simulated Annealing: Quantitative Studies". *Journal of Statistical Physics*, Vol. 34, no. 5/6, 1984, pp. 975-986.

[40] Kirkpatrick, S., C. D. Gelatt, M.P. Vecchi, "Optimization by Simulated Annealing". *Science*, Vol. 220, no. 4598, 13 May 1983, pp. 671-680.

[41] Krolak, Patrick, Wayne Felts and George Marble, "A Man-Machine Approach Towards Solving Traveling Salesman Problem". *Communication of the ACM*, Vol. 14, No. 5, May 1971, pp. 327-334.

[42] Lai, W. K., and G. G. Coghill, "Genetic Breeding of Control Parameters for the Hopfield/Tank Neural Net". *IJCNN*, Vol. IV, June 7-11, 1992, pp. 618-623.

[43] Laporte, Gilbert, "The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms". *European Journal of Operational Research*, Vol. 59, 1992, pp. 231-247.

[44] Lawler, E., J. Lenstra, A. Rinnooy Kan, and D. Shamoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* John Wiley & Sons, New York, 1985.

[45] Lin, S. and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem". *Operation Res.*, Vol. 21, 1973, pp. 498-516.

[46] Litke, John D., "An Improved Solution to the Traveling Salesman Problem with Thousands of Nodes". *Communications of the ACM*, Vol. 27, No. 12, December 1984, pp. 1227-1236.

[47] Looi, Chee-Kit, "Neural Network Methods in Combinatorial Optimization". *Computers Ops. Res.*, Vol. 19, no. 3/4, 1992, pp. 191-208.

[48] Makki, Assaad and Pepe Siy, "Hopfield Neural Networks Control for Optimal Solutions". *IJCNN*, Vol. IV, June 7-11, 1992, pp. 462-467.

[49] Mehta, Shashank and Laszlo Fulop, "A Neural Algorithm to Solve the Graph Matching Problem". *INNC*, Vol. 1, July 9-13,1990, pp. 262-265.

[50] Müller, B. and J. Reinhardt, *Neural Networks: An Introduction*, Springer-Verlag, 1990, pp. 104-112.

[51] Nonaka, Hisanori and Yasuhiro Kobayashi, "Sub-optimal Screening in Optimization by Neural Networks". *IJCNN*, Vol. IV, June 7-11, 1992, pp. 606-611.

[52] Norback, John and Robert Love, "Geometric Approaches to Solving the Traveling Salesman Problem", *Management Science*, Vol. 23, no. 11, July 1977, pp. 1208-1223.

[53] Park, Jeong and Hong Jeong, "Solving the Traveling the Salesman Problem Using an Effective Hopfield Network", *INNC*, Vol. 1, July 9-13,1990, p.291.

[54] Platt, John and Alan Barr, "Constrained Differential Optimization". *Neural Information Systems*, Denever, CO 1987, pp. 612-621.

[55] Potvin, Jean-Yves, "The Traveling Salesman Problem: A Neural Network Perspective". *ORSA Journal on Computing*, Vol. 5, no. 4, Fall 1993, pp. 328-348.

[56] Ritter, Helge, Thomas Martinetz and Klaus Schulten. *Neural Computation and Self-Organizing Maps*. Addison-Wesley Publishing Company, 1992.

[57] Rong, Liu and Liu Ze-min, "Parameters Rules of Hopfield/Tank Model on Solving TSP". *IJCNN*, Vol. IV, June 7-11, 1992, pp. 492-498.

[58] Rosekrantz, Daniel, Richard Stearns, and Philip Lewis II, "An Analysis of Several Heuristics for the Traveling Salesman Problem". *SIAM J. Comput.*, Vol. 6, no. 3, September 1977, pp. 563-581.

[59] Shiratani, Fumiyuki and Kimiaki Yamamoto, "Combinatorial Optimization by Using a Neural Operating in Block-Sequential Mode". *System and Computers in Japan*, Vol. 25, no. 8, 1994, pp. 103-111.

[60] Sun, K.T. and H. C. Fu, "A Hybrid Neural Network Model for Solving Optimization Problems". *IEEE Transactions on Computers*, Vol. 42, no. 2, February 1993, pp. 218-227.

[61] Su Yu, Chong and Won Don Lee, "Parallel Mean Field Annealing Neural Network for Solving Traveling Salesman Problem". *IJCNN*, Vol. IV, June 7-11, 1992, pp. 532-536.

[62] Tagliarini, Gene, Fury Christ and Edward Page, "Optimization Using Neural Networks". *IEEE Trans. on Computers*, Vol. 40, No. 12, December 1991, pp. 1347-1358.

[63] Ueda, Tomoyuki, Kiyoshi Takahashi, Iwao Sasase, and Shinsaku Mori, "Hopfield-Type Neural Networks with Fuzzy Sets to Gather the Convergence Speed". *IJCNN*, Vol. IV, June 7-11, 1992, pp. 624-629.

[64] Wasserman, P. D. and Van Nostrand. *Neural Computing: Theory and Practice*, 1989, pp. 93-112.

[65] Wolfe, William, and others, "Inhibitory Grids and the Assignment Problem". *IEEE Trans. on Neural Networks*, Vol. 4, no. 2, March 1993, pp. 319-331.

[66] Zhang, Zeeman, Nirwan Ansari, Edwin Hou, and Pei-ken Yi, "Multiprocessor Scheduling by Mean Field Theory". *IJCNN*, Vol. IV, June 7-11, 1992, pp. 1271-1278.