Development of an Evolving Neural Network Approach in Unit Commitment of Power System

Man Hong Wong B.Eng.

M.PHIL.

THE HONG KONG

POLYTECHNIC UNIVERSITY

2000

ACKNOWLEDGEMENTS 3

ABSTRACT 4

CHAPTER 1 6

INTRODUCTION 6

1.1 BACKGROUND 6
1.2 PREVIOUS RESEARCH EFFORTS 6
1.3 OBJECTIVE AND SCOPE OF THE STUDY 8
1.4 ORGANIZATION OF THIS REPORT 8
1.5 PUBLICATION OF PAPERS 11
1.6 CONTRIBUTION ON THIS PROJECT 11

CHAPTER 2 13

UNIT COMMITMENT 13

2.1 OBJECTIVE FUNCTION 14
2.2 CONSTRAINTS 16
2.3 ECONOMIC DISPATCH 19

CHAPTER 3 22

DYNAMIC PROGRAMMING 22

3.1 DYNAMIC PROGRAMMING 22
3.2 MATHEMATICAL MODEL OF DP 22

CHAPTER 4 24

ARTIFICIAL NEURAL NETWORK (ANN) 24

4.1 INTRODUCTION 24
4.2 TYPES OF NEURAL NETWORK 25
4.3 BIOLOGICAL NEURAL NETWORK 25
4.4NETWORK ARCHITECTURE 28
4.5 NEURON CHARACTERISTICS 30
4.6 LEARNING 32
4.7 BACKPROPAGATION TRAINING ALGORITHM 35

CHAPTER 5 44

GENETIC ALGORITHMS (GA) 44

5.1 GENETIC ALGORITHMS (GA) 44
5.2 HOW IT WORKS 49
5.3 TERMINOLOGY 51

CHAPTER 6 53

EVOLVING NEURAL NETWORK 53

6.1 GENERATE INITIAL POPULATION POOL 54
6.2 FEED-FORWARD OF NN 55
6.3 CALCULATION OF FITNESS 55

6.4 FITNESS CONVERSION 56
6.5 CROSSOVER AND MUTATION 57
6.6 SECOND LEARNING PHASE 58
CHAPTER 7 59
SOFTWARE OVERVIEW 59
CHAPTER 8 63
IMPLEMENTATION 63
8.1 PARAMETER DETERMINATION 63
8.2 CHAPTER SUMMARY (BEST PARAMETERS) 76
CHAPTER 9 77
RESULTS OF TEST 77
9.1 FURTHER WORK 77
9.2 ERROR VALUE ACCORDING TO THE BEST PARAMETER 79
9.3 CHAPTER SUMMARY 84
CHAPTER 10 85
CONCLUSIONS 85
REFERENCES 86
APPENDIX 1 89
TRAINING FILE 89

Acknowledgements

I would like to use this opportunity to thank my parents and girl friend Kitman Leung for their support and encouragement. I also thank my supervisors Prof. T.S. Chung & Dr Y.K.Wong. Their helpful suggestions and corrections are greatly appreciated.

Abstract

The Unit Commitment (UC) problem involves determining a start-up and shutdown schedule for the generating units over a period of time to meet the forecasted load demand at minimum cost. Intensive research efforts have been made over the past years to achieve a near-optimal solution of the UC problem in an efficient manner. The literature displays a wide range of UC methods with various degrees of accuracy and efficiency to handle difficult constraints and heuristics. The priority list based methods are simple and fast, but highly heuristic. Other approaches are based on dynamic programming and branch-and-bound methods. These methods are general and flexible, but often prone to the curse of dimensionality. More recently, methods such as Benders decomposition and Lagrangian relaxation have been applied to solve the UC problem. The Lagrangian method is efficient and well suited for application on large-scale systems. However, under some circumstances this method requires additional heuristics to improve convergence. In recent years, Artificial Neural Network (ANN) has also been applied to solve the UC problem. However the technique has a number of limitations.

In addition, a training algorithm based on the steepest decent method, such as the back-error-propagation, can be very slow in the vicinity of the final convergence. In the absence of data noise, additional learning takes place in a multilayered perceptron only if new data is introduced that the neural network improperly classifies. The closer the representation comes to the concept, the smaller the chance that this happens. This is a characteristic of the least squares and the steepest-descent techniques.

An alternative to the back-error-propagation is the genetic algorithm (GA) which is an evolutionary algorithm based on the concept of natural selection and genetics. The paradigms of GA were first published by Holland in 1962. During the last few years, GA research has been substantially expanded on Unit Commitment problem. As the genetic algorithm can prevent the local minimum when guided by the parallel evolution, the learning stagnation in neural network can thus be avoided.

In this project we will utilize the GA algorithm to evolve the weight and the interconnection of the neural network and then using the recognition capability of neural network integrated with the trial and error methodology of dynamic programming to solve the unit commitment problem.

Firstly, by using the recognition capability of evolving neural network, a pre-schedule of the commitment schedule is produced by the existing scheduling information for a random load profile. Secondly, the pre-schedule is passed to the dynamic programming algorithm to calculate a proper commitment schedule.

Different approaches of using ENN-DP have been proposed to solve the thermal unit commitment. Performance comparisons of the proposed algorithm with conventional Dynamic Programming approach and Neural Network approach based on its running time and operating cost are given on several cases. From the results shown, ENN-DP has a greater operating time saving than NN-DP. Among the two training options of GA, connection has a better performance. Among the three generation's selection of GA, Roulette Wheel has the best result.

Generally speaking, ENN-DP has a shorter operating time than ANN-DP and DP due to better initiation of weights or optimized number of connections.

Chapter 1

Introduction

1.1 Background

The Unit Commitment (UC) problem involves determining a start-up and shutdown schedule for the generating units over a period of time to meet the forecasted load demand at minimum cost. The commitment schedule must satisfy other constraints such as those on reserve, fuel, and individual units. Mathematically, the UC problem is a mixed-integer one, typically with thousands of 0-1 variables as well as a large and complex set of constraints. The exact solution of such a problem can only be obtained by exhaustive computation, usually at the cost of a prohibitively long computation time.

1.2 Previous research efforts

Intensive research efforts have been made over the past years to achieve a near-optimal solution of the UC problem in an efficient manner. The literature displays a wide range of UC methods with various degrees of accuracy and efficiency to handle difficult constraints and heuristics. The priority list based methods [1] are simple and fast, but highly heuristic. Other approaches are based on dynamic programming [2-3] and branch-and-bound methods [4]. These methods are general and flexible, but often prone to the curse of dimensionality. More recently, methods such as Benders decomposition [5] and Lagrangian relaxation [6] have been applied to solve the UC

problem. The Lagrangian method is efficient and well suited for application on large-scale systems. However, under some circumstances this method requires additional heuristics to improve convergence. In recent years, Artificial Neural Network (ANN) has also been applied to solve the UC problem [7-9]. However the technique has a number of limitations. For example, since the back-error-propagation technique is not designed to be adaptive, all data must be used every time the weights are updated. If a set of old data becomes irrelevant, the NN is retrained by using the entire new data set. Also, when new data is in conflict with old data, the effect of old data cannot be removed unless the NN is retrained without the old data. The importance of some data cannot be easily weighted. In addition, if the size of the NN is not adequately selected, or the convergence criterion is not realistic, thousands of iterations may be required to train a layered preceptron even on a simple problem.

In addition, a training algorithm based on the steepest decent method, such as the back-error-propagation, can be very slow in the vicinity of the final convergence. In the absence of data noise, additional learning takes place in a multilayered perceptron only if new data, improperly classified by the neural network is introduced. The closer the representation comes to the concept, the smaller the chance that this happens. This is a characteristic of the least squares and the steepest-descent techniques.

An alternative to the back-error-propagation is the genetic algorithm (GA) which is an evolutionary algorithm based on the concept of natural selection and genetics. The paradigms of GA were first published by Holland in 1962 [10]. During the last few years, GA research has been substantially expanded on Unit Commitment problem

[11-14]. As the genetic algorithm can prevent the local minimum when guided by the parallel evolution, the learning stagnation in neural network can thus be avoided.

1.3 Objective and scope of the study

In this project we will utilize the GA algorithm to evolve the weight and the interconnection of the neural network and then using the recognition capability of neural network integrated with the trial and error methodology of dynamic programming to solve the unit commitment problem.

Firstly, by using the recognition capability of evolving neural network, a pre-schedule of the commitment schedule is produced by the existing scheduling information for a random load profile. Secondly, the pre-schedule is passed to the dynamic programming algorithm to calculate a proper commitment schedule.

1.4 Organization of this report

This report is organized as follows:

Chapter 1 Introduction

A general description, background and objective of the project are given in this chapter.

Chapter 2 Unit Commitment

In this chapter, we will introduce the theory of Unit Commitment, including the objective functions for the Unit Commitment and the constraints involve.

Chapter 3 Dynamic Programming

In this chapter, we will introduce the basic theory of the Dynamic Programming, the mathematical model and the concept of the theory will also be given.

Chapter 4 Artificial Neural Network

There are seven sections in this chapter. Section one is an introduction to the background of ANN. Section two is a short paragraph to identify different types of ANN used in this project. In section three, biological neural network will be introduced here for easy understanding the concept of ANN. In section four, the architecture and general working principle of ANN will be given. In section five, detailed characteristics of a neuron including the mathematical model and activation function will be described. In section six, the most important part of ANN, will be presented here. Different types of training methods will also be introduced. In section seven, an in-depth description of the training algorithm will be given. It contains the mathematical model of the algorithm and the objective function for minimizing the error value.

Chapter 5 Genetic Algorithm

There are three sections in this chapter. Section one will introduce the background, mechanism and the advantages of using GA. In section two, a paragraph written by Goldberg (the inventor of GA) is quoted for demonstration of the working principle of GA. In section three, a table showing the comparison of natural and GA terminology will be given.

Chapter 6 Evolving Neural Network

A framework for ENN is given in this chapter. The framework contains two learning phases. In the first learning phase, it contains several steps such as the procedures to generate initial population pool, feed forward NN, calculation of fitness, fitness conversion, crossover and mutation etc. The operation of each step described above will be given in the subsection of this chapter. The second learning phase is the back propagation algorithm of neural network typically.

Chapter 7 Software Overview

The software used in this project will be illustrated in this chapter.

Chapter 8 Implementation

The method for the determination of the parameter of ENN will be explained in this chapter.

Chapter 9 Results of test

Based on the data and parameters described in chapter 8, the testing results are presented in this chapter. The results for different approaches are given for comparison.

Chapter 10 Conclusion

The final conclusion of the project and comments for the algorithm will be given in this chapter.

1.5 Publication of Papers

Published

Journal Paper

[P1] M.H.Wong, Y.K.Wong and T.S. Chung, " A Hybrid Artificial Neural Network-Dynamic Programming Approach to Unit Commitment," Transactions of The Hong Kong Institution of Engineers, vol. 5, no.2, Aug 1998, pp. 49-54.

[P2] M.H. Wong, T.S. Chung and Y.K. Wong, "An Evolving Neural Network Approach in Unit Commitment Solution," International Journal in Microprocessors and Microsystems. (Paper accepted for publication in June 2000 and to appear soon)

Conference Paper

[P3] M.H. Wong, T.S. Chung and Y.K.Wong, " Application of Evolving Neural Network to Unit Commitment," Proceedings of The 1998 International Conference on Energy Management and Power Delivery (Singapore), pp.154-159.

[P4] M.H Wong, Y.K. Wong, T.S.Chung, " Parameter Determination of An Evolving Neural Network Approach in Unit Commitment Solution," Proceedings of the 1998 IEEE International Conference on Systems, Man and Cybernetics (USA), SMC'98, pp.1631-1636.

[P5] M.H. Wong, T.S. Chung and Y.K. Wong, "An Evolving Neural Network Approach in Unit Commitment Solution: Theory and Performance Evaluation," Proceedings of 34th Universities Power Engineering Conference of Leicester 99 (England).

Under Evaluation

[U1] M.H. Wong, T.S. Chung and Y.K. Wong, "A Hybrid Evolving Neural Network- Dynamic Programming (ENN-DP) Approach to Unit Commitment," International Journal of Electrical Power and Energy Systems.

1.6 Contribution on this project

On this research project my major contribution are as follows:

1. Apply the evolving neural network algorithm in solving the unit commitment problem [P1], [P3]

2. Define the best parameter of the ENN in solving the unit commitment problem [P4]
3. Comparison of different approach of the selection method of ENN in solving the unit commitment.[P2], [P5], [U1]

Chapter 2

Unit Commitment

Unit Commitment (UC) is the problem of determining the schedule of generating units within a power system subject to device and operating constraints. The decision process selects units to be on or off, the type of fuel, the power generation for each unit, the fuel mixture when applicable, and the reserve margins. [15]

The generic UC problem can be formulated as

Minimize Operational Cost

Subject to:

1. Load constraints
2. Generation constraints
3. Reserve constraints
4. Minimum up-time and down time constraints
5. Crew constraints
6. Unit status

2.1 Objective function

The objective function is operational costs grouped into the following categories:

2.1.1. **Fuel cost**: They are represented by an input/output (I/O) curve that is modeled with a polynomial curve (normally quadratic or reduced cubic), a piecewise constant curve, or a piecewise linear curve. The fuel cost equation for unit i is described by:

$$FCi(Pi)=Fi*Hi(Pi) \qquad \text{(EQ2.1)}$$

where

Pi is the power level (MW)

FCi(Pi) is the fuel cost ($/h)

Fi is the fuel cost ($/MBTU)

Hi(Pi) is the heat rate curve (MBTU/h)

2.1.2. The **start up cost** is described by:

$$ST_i = TS_i F_i + (1- e^{(-Di/ASi)})BS_iF_i + MS_i \qquad \text{(EQ2.2)}$$

where

TS_i is the turbine startup energy (MBTU)

D_i is the number of hours down

AS_i is the boiler cool-down coefficient

BS_i is the boiler startup energy (MBTU)

MS_i is the startup maintenance cost

The function can be expressed as

$$MinC(U) = Min\sum_{i=1}^{N}\sum_{h=1}^{H}(U_{ih}FC_i(P_{ih}) + ST_iU_{ih}(1 - U_{i,h-1}))$$

(EQ2.3)

where

C is the cost function of the UC problem

U_{ih} is the on/off status of unit i at hour, and Uih $\in$ {0,1}

U is the decision matrix of U_{ih} variables for i = 1,...,N

N is the number of thermal generating units

H is the number of hours in the study period

2.2 Constraints

2.2.1. Load balance

$$\sum_{i=1}^{N} U_{ih} P_{ih} = D_h \qquad \text{for h=1,...,H} \tag{EQ2.4}$$

where

Dh is the system load demand at hour h.

2.2.2 Limitations on generator output

$$P_{imin} \le P_{ih} \le P_{imax} \text{ for i = 1,...,N; h=1,...,H} \tag{EQ2.5}$$

where P_{imin} , P_{imax} are minimum and maximum power outputs of unit i.

2.2.3. Spinning reserve

$$Dh(1+r_h\%) \le \sum_{i=1}^{N} U_{ih} P_{i\max} \text{ , and} \tag{EQ2.6}$$

$$\sum_{i=1}^{N} U_{ih} P_{i\min} \le D_h \text{ , for h= 1,...,H} \tag{EQ2.7}$$

where r_h is the system reserve at hour h.

The most nonlinear constraints are the minimum up-time and down-time restrictions. If a unit must be on for a certain number of hours before it can shut off, then a minimum up-time is set. By contrast, minimum down-time is the number of hours a unit mustbe off-line before it can be brought on-line again. Violation of a down time constraint may be alleviated by banking or idling the unit.

2.2.4. Minimum up-time

$$\text{Uih} = 1, \text{ for } \sum_{j=hs}^{h-1} U_{ij} < \text{MUT}_i \quad i=1,...,N \qquad \text{(EQ2.8)}$$

where

h_s is the hour at which unit i is started up,

MUT_i is the minimum up time for unit i.

2.2.5. Minimum down-time

$$\text{Uih} = 0, \text{ for } \sum_{j=hd}^{h-1} (1 - U_{ij}) < \text{MDT}_i \quad i=1,...,N \qquad \text{(EQ2.9)}$$

where

h_d is the hour at which unit i is shut down,

MDT_i is the minimum down time for unit i.

2.2.6. Crew constraints

Crew constraints pertain to the number of units that can be started at the same time in a particular plant. The ramp rate limits restrict the movement of a generating unit between adjacent hours. The upper and lower limits on the generating units force them to operate within their boundaries.

2.2.7 Status

The **status** of each generating unit is one of the following:

- Unit can cycle on or off (available).
- Unit can be outage due to maintenance or other unforeseen problems (Outage)
- Unit can be forced to run (Must-Run)
- Unit can have a fixed output (Fixed)
- Unit that may be brought on-line quickly with no regard to minimum up or down time constraints (Peakers/Gas Turbines)

2.3 Economic Dispatch

Within the Unit Commitment problem, there is a sub-problem of doing the economic dispatch. To determine the economic distribution of load between the various generating units the variable operating costs of the unit must be expressed in terms of the power output. Fuel cost is the principal factor in fossil-fuel plants. A typical input-output curve which is a plot of fuel input for a fossil-fuel plant in British thermal units (Btu) per hour versus power output of the unit in megawatts.

The criterion for distribution of the load between any two units are based on whether increasing the load on one unit as the load is decreased on the other unit by the same amount results in an increase or decrease in total cost. Thus we are concerned with incremental fuel cost, which is determined by the slopes of the input-output curves of the two units. If we express the ordinates of the input-output curve in dollars per hour and let

fi = input to unit i, dollars per hour ($/h)

Pgi = output of unit i, megawatts (MW)

The incremental fuel cost of the unit in dollars per megawatthour is dfi/dPgi, whereas the average fuel cost in the same units is fi/Pgi . Hence, if the input-output curve of Unit i is quadratic, we can write

$$fi = \frac{ai}{2} Pgi^2 + biPgi + ci \quad \$/h \qquad \text{(EQ2.10)}$$

and the unit has incremental fuel cost denoted by λi , which is defined by

$$\lambda i=\frac{dfi}{dPgi}=aiPgi+bi \quad \$/Mwh \qquad (EQ2.11)$$

where ai, bi and ci are constants. The approximate incremental fuel cost at any particular output is the additional cost in dollars per hour to increase the output by 1 MW.

For a plant with two units operating under economic load distribution the λ of the plant equals λi of each unit, and so

$$\lambda=\frac{df1}{dPg1}=a_1P_{g1}+b_1 \qquad \lambda=\frac{df2}{dPg2}=a_2P_{g2}+b_2 \qquad (EQ2.12)$$

Solving for P_{g1} and P_{g2}, we obtain

$$Pg1=\frac{\lambda-b1}{a1} \qquad Pg2=\frac{\lambda-b2}{a2} \qquad (EQ2.13)$$

Adding these results together and then solving for λ give

$$\lambda=\left(\sum_{i=1}^{2}\frac{1}{ai}\right)^{-1}(Pg1+Pg2)+\left(\sum_{i=1}^{2}\frac{1}{ai}\right)^{-1}\left(\sum_{i=1}^{2}\frac{bi}{ai}\right)$$

(EQ2.14)

$$\lambda = a_T P_{gT} + b_T \qquad \text{(EQ2.15)}$$

where $a_T = \left(\sum_{i=1}^{2} \frac{1}{ai} \right)^{-1}$

$$b_T = a_T \left(\sum_{i=1}^{2} \frac{bi}{ai} \right)$$

$P_{gT}=(P_{g1}+P_{g2})$ is the plant output.

If the plant has K units operating on economic dispatch,then

$$a_T = \left(\sum_{i=1}^{K} \frac{1}{ai} \right)^{-1} \qquad \text{(EQ2.16)}$$

$$b_T = a_T \left(\sum_{i=1}^{K} \frac{bi}{ai} \right) \qquad \text{(EQ2.17)}$$

and the total plant output PgT is (Pg1 + Pg2 +Pg3+...+ PgK). The individual output of each of the K units is then calculated from the common value of λ given by EQ 2.15.

Chapter 3

Dynamic Programming

3.1 Dynamic Programming

Dynamic Programming (DP) searches the solution space that consists of the units status for an optimal solution. The search can proceed in a forward or backward direction. For unit commitment problem it is more practical to use the forward search method since the initial conditions are easily specified and the computations can go forward in time as long as required and computer storage is available. The following figure 1 shows a typical DP method. The time periods of the study horizon are known as the stages of the DP problem. Typically each stage represents one hour of operation. The combinations of units within a time period are known as the states of the DP problem. Forward DP finds the most economical schedule by starting at the initial stage accumulating total costs, then backtracking from the combination of least accumulated cost starting at the last stage and ending at the initial stage.

3.2 Mathematical model of DP

$$Fcost\ (K,I) = \min_{\{L\}} [P_{cost}(K,I)+S_{cost}(K-1,L:K,I)+F_{cost}(K-1,L)] \qquad (EQ3.1)$$

where Fcost (K,I) = least total cost to arrive at state(K,I)

$P_{cost}(K,I)$ = production cost for state(K,I)

$S_{cost}(K-1,L:K,I)$ = transition cost from state (K-1,L) to state (K,I)

where state (K,I) is the I th combination in hour K. For the forward dynamic programming approach , we define a strategy as the transition, or path , from one state at a given hour to a state at the next hour.

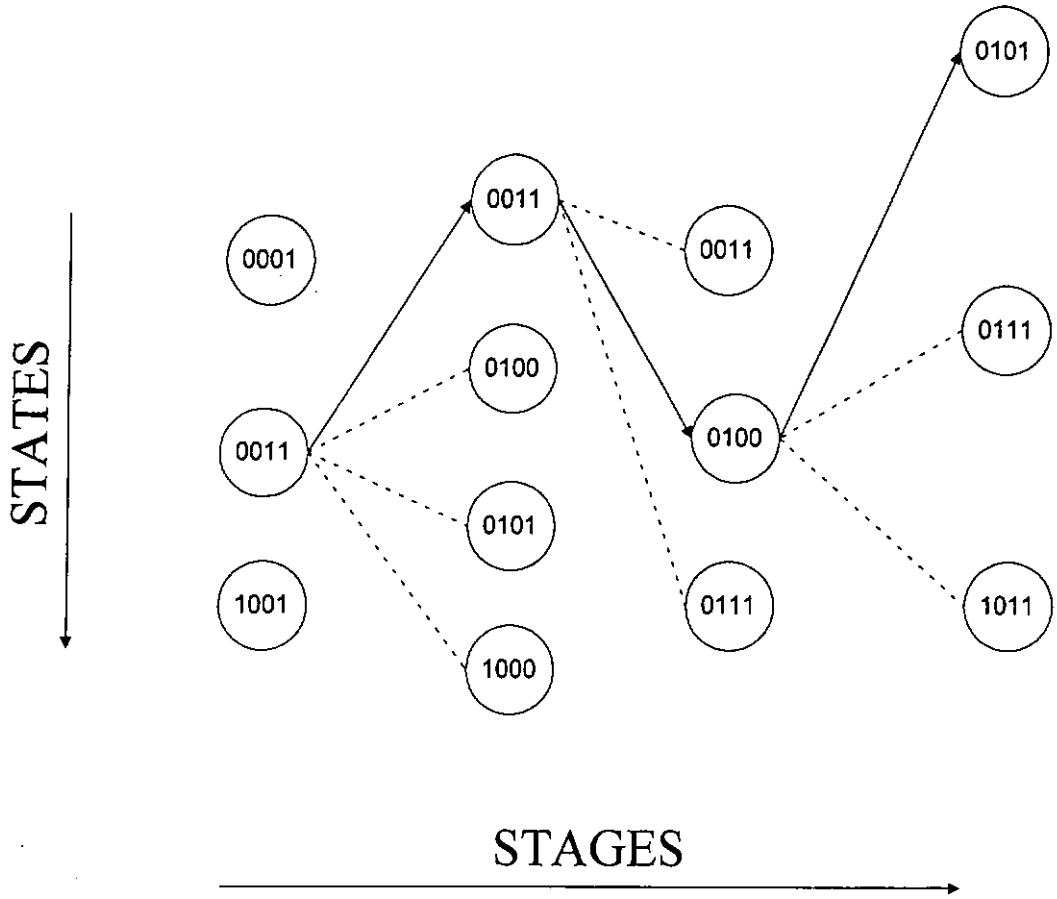


Figure 3.1 Typical DP method for UC

Chapter 4

Artificial Neural Network (ANN)

4.1 Introduction

An artificial neural network is an information processing system that is non-algorithmic, non-digital, and intensely parallel. It consists of a number of very simple and highly interconnected processors called neurodes, which are the analogs of the biological neural cells, or neurons, in the brain [16].

Communication between processing elements occurs along paths of variable connection strengths. By changing the values of these connection strengths, the network can collectively produce complex overall behaviour.

The brain is very good in solving some types of problems, and not so good at others [17]. While we can only muddle our way through simple arithmetic operations "in our head", we have no trouble recognizing faces, picking out objects from cluttered backgrounds, and deciphering speech across noisy phone lines or crowded rooms. Conventional computers can zip through more calculations in a few seconds than we can do in a lifetime, yet are stymied by pattern recognition problems that can easily be solved by a small child. Neural networks are well suited to solving the same types of problems as our brains. A neural network should not be enlisted to perform arithmetic

operations, although in theory that could be done. Rather, neural networks excel at recognition and classification types of problems. Some examples of problems that the neural networks have been applied are radar and sonar signal classification, speech-to-text conversion, medical classification and diagnosis application, loan applications, investing and trading.

4.2 Types of Neural Network

There are many different types of neural networks. Characteristic of each type is a highly parallel processing capability arising from an interconnected network of simple computational elements. The networks differ from one another in architecture and training algorithm [17] .

The type of NN used in this project is called the multilayer feedforward, or so-called backpropagation, network is responsible for most of the successful applications of neural networks and is by far the most commonly used neural network.

4.3 Biological Neural Network

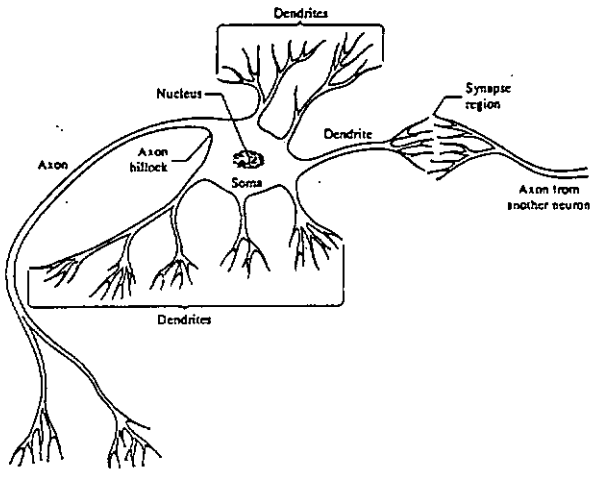


Figure 4.1 A biological neuron

A biological neuron has three types of components that are of particular interest in understanding an artificial neuron: its dendrites, soma and axon. There are many

dendrites receive signals from other neurons. The signals are electric impulses that are transmitted across a synaptic gap by means of a chemical process. The action of the chemical transmitter modifies the incoming signal (typically, by scaling the frequency of the signals that are received) in a manner similar to the action of the weights in an artificial neural network [18].

The soma or cell body sums the incoming signals. When sufficient input is received, the cell fires, that is, it transmits a signal over its axon to other cells. It is often supposed that a cell either fires or does not at any instant of time, so that transmitted signals can be treated as binary. However, the frequency of firing varies and can be viewed as a signal of either greater or lesser magnitude. This corresponds to looking at discrete time steps and summing all activity (signals received or signals sent) at a particular point in time.

The transmission of the signal from a particular neuron is accomplished by an action potential resulting from differential concentrations of ions on either side of the neuron's axon sheath (the brain's "white matter"). The ions most directly involved are potassium sodium and chloride.

A generic biological neuron is shown in Figure 4.l, together with axons from one other neuron (from which the illustrated neuron could receive signals) and dendrites for other neurons (to which the original neuron would send signals). Several key features of the processing elements of artificial neural networks are suggested by the properties of biological neurons, viz., that :

i. The processing element receives many signals.

ii. Signals may be modified by a weight at the receiving synapse.

iii. The processing element sums the weighted inputs.

iv. Under appropriate circumstances (sufficient input), the neuron transmits a single output.

v. The output from a particular neuron may go to many other neurons (the axon branches).

vi. Information processing is local (although other means of transmission, such as the action of hormones, may suggest means of overall process control).

vii. Memory is distributed:

   a. Long-term memory resides in the neurons` synapses or weights.

   b. Short-term memory corresponds to the signals sent by the neurons.

viii. A synapse's strength may be modified by experience.

ix. Neurotransmitters for synapses may be excitatory or inhibitory.

Another important characteristic of biological neural systems is fault tolerance. Biological neural networks are fault tolerant in two aspects. First, we are able to recognize many input signals that are somewhat different from any signal we have seen before. An example of this is our ability to recognize a person in a picture we have not seen before or to recognize a person after a long period of time.

Second, we are able to tolerate damage to the neural system itself. Humans are born with as many as 100 billion neurons. Most of these are in the brain, and most are not replaced when they die. In spite of our continuous loss of neurons, we continue to learn. Even in cases of traumatic neural loss, other neurons can sometimes be trained to take over the functions of the damaged cells.

4.4Network Architecture

Figure 4.2 is a typical back-propagation network with an input layer with N neurons, one hidden layer and one output layer with M neurons. Each layer is fully connected to the succeeding layer. The arrows indicate flow of information. Each neurode in the input layer of the network receives a small piece of the input pattern. Typically these neurodes act merely as "fan-out" neurodes distributing their small pair of the pattern, more or less unchanged, to the neurodes in the middle layer [16].

Each of the middle-layer neurodes thus receives the entire input pattern, but the pattern is modified by its passage through the weighted connections leading to the middle layer. Since the weights on the connections are typically different for each middle-layer neurode, each of these neurodes sees a somewhat different version of the input pattern than its neighbours do. As a result, some combination of middle-layer neurodes will become active to varying degrees, depending on the exact collection of weights on their input connections. This results in a variety of output responses from the middle-layer neurodes, ranging from no output at all to, possible, an extremely strong output.

Normally all the middle-layer neurodes transmit their output signals to all of the neurodes in the output layer, thus, each of the output-layer neurodes receives the complete pattern of output activity from the middle-layer neurodes. However, just as happened with the middle-layer neurodes, this activity pattern is modified by passage through the weighted connections between the middle-layer and output-layer neurodes. So again each neurode in the output layer see somewhat different versions of the middle-layer activity. As with the middle layer, the result is that some

combination of output-layer neurodes becomes active to greater or lesser degree, causing them to output a signal. This pattern of output-layer neurode responses in the network's overall response to the original input pattern stimulus.

The input enter into the input layer then propagated through the hidden layer then reach the output layer. During learning, information is also back-propagated back through the network and used to up date the connection weights.

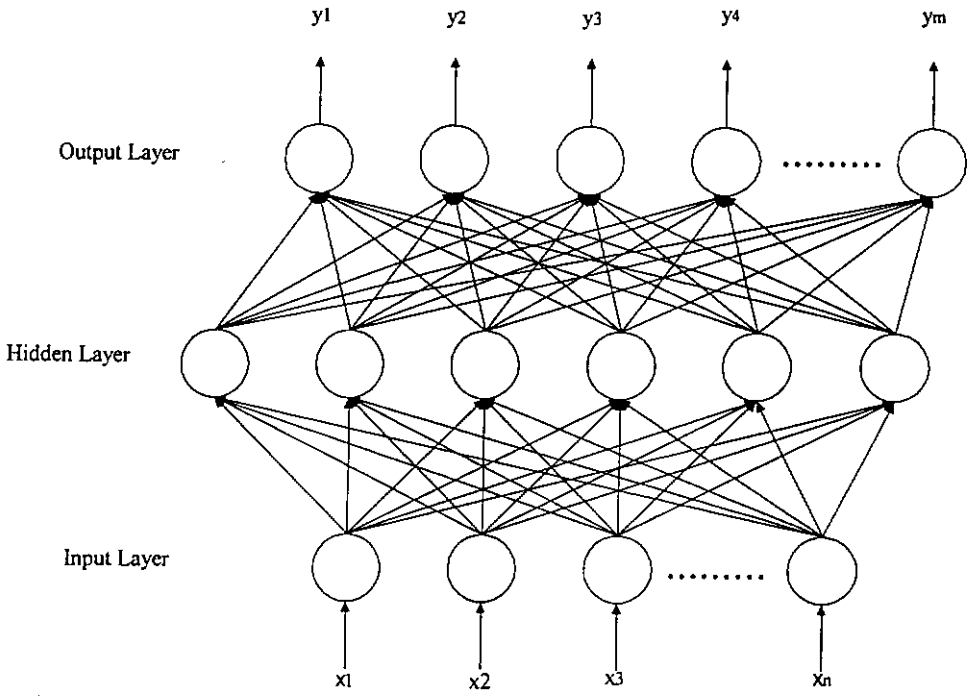


Fig 4.2 Typical one-hidden layer, back-propagation network

4.5 Neuron Characteristics

Fig. 4.3 is the typical back-propagation neuron with N connection from previous layer and one output which connected to the other neuron's input. Each of the input is multiplied by a weight and the products are summed as equation as follow:

$$I_j^{[s]} = \sum_i \left(w_{ji}^{[s]} * x_i^{[s-1]} \right) \qquad \text{(EQ4.1)}$$

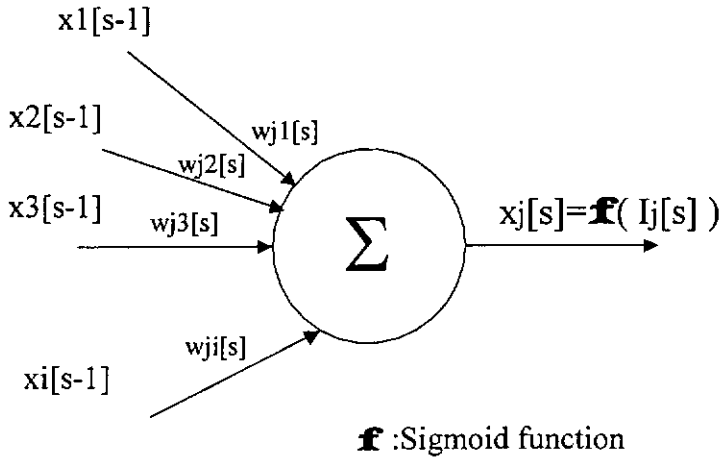


Fig 4.3 Typical back-propagation neuron

Then an activation function is applied to modify the weighted sum. Normally there are eight types of activation functions as shown in tale 4.1[19]. Among the eight types of activation function, the Sigmoid function is the most popular one because it has simple derivative and this function is being used in this project.

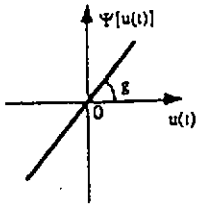
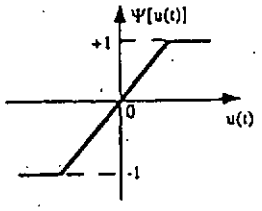
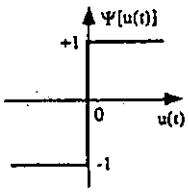
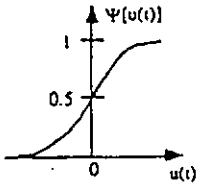
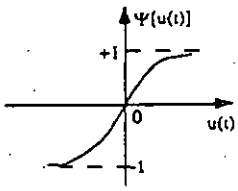
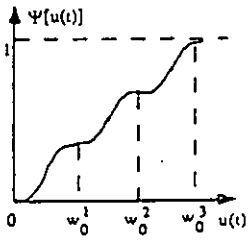
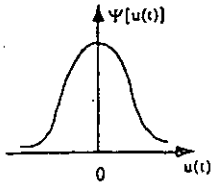
Type	Equation	Functional form
(i) Linear	$\psi[u(t)] = g\,u$, $g > 0$, activation gain	
(ii) Piecewise linear	$\psi[u(t)] = \begin{cases} +1 & \text{if } g\,u > 1, \\ g\,u & \text{if } \lvert g u\rvert < 1, \\ -1 & \text{if } g\,u > -1. \end{cases}$ $g > 0$, activation gain	
(iii) Hard limiter	$\psi[u(t)] = \mathrm{sgn}[u]$	
(iv) Unipolar sigmoidal	$\psi[u(t)] = \dfrac{1}{1+\exp(-g\,u)}$ $g > 0$, activation gain	
(v) Bipolar sigmoidal	$\psi[u(t)] = \tanh[g\,u(t)]$, $g > 0$, activation gain	
(vi) Unipolar multimode sigmoidal	$\psi[u(t)] = \frac{1}{2}\left[1 + \frac{1}{M}\sum_{i=1}^{M} \tanh(g^i(u - w_0^i))\right]$ $g^i > 0$, activation gain	
(vii) Radial basis function (RBF)	$\psi[u(t)] = \exp(u(t))$ $u(t) = \left[\dfrac{-\sum_{i=n}^{M}(w_i(t) - x_i(t))^2}{2\sigma^2}\right]$	
(viii) Maximum	$\psi[u(t)] = \begin{cases} 1 & \text{if } x_n(t) = \mathrm{MAX}\{x_n(t)\}, \\ 0 & \text{otherwise} \end{cases}$ $n \in N$; n = set of all possible winners	

Table 4.1 Activation function

The equation of the sigmoid function is shown as follow:

$$f(z) = (1 + e^{-z})^{-1} \qquad \text{(EQ4.2)}$$

and the derivative of the function is:

$$f'(z) = f(z)[1 - f(z)] \qquad \text{(EQ4.3)}$$

Simple derivative is the key factor for the use of the Sigmoid function in the implementation of backpropagation. Since the algorithm require the function be every where differentiable. Besides, the Sigmoid function have an automatic Gain Control. When the input is small, it provide high gain. In the contrary, when the input is large, the gain become lower. When the input is very large, the output will become saturated.

4.6 Learning

Neural networks learn to solve a problem, they are not programmed to do so. Learning and training are thus fundamental to nearly all neural networks. Learning is achieved not by modifying the neurodes in the network but by modifying the weights on the interconnections in the network [16].

If it is assumed that the transfer function is fixed, then each neuron's output is determined by two things only : the incoming signal and the weights on the input

connections to the neurode. Clearly if the neurode is to learn to respond correctly to a given incoming signal pattern, the only possible element that can be used to improve the neurode's performance is the weight on the connection. Actually, learning in neural networks consists of making systematic changes to these weights in order to improve the network's response performance to acceptable levels.

Training is the procedure by which the network learns; learning is the end result of that procedure. Training is a procedure external to the network; learning is an internal process or activity. Basically, there are three types of training.

4.6.1 Supervised Training

In this technique [16], the network is provided with an input stimulus pattem along with the corresponding desired output pattern. The learning law for such networks typically computes an error - that is, how far from the desired output the network's actual output really is. This error is then used to modify the weights on the interconnections between the neurodes. Usually, there are three different types of supervised training as shown below:

4 .6.1.1 Hebbian Training

If both the input and output stimulus are high, the connection weight on this input path to the processing element is increased or strengthened.

4.6.1.2 Delta Rule Training

Reducing the error between the actual output and desired output by modifying the related connection weights, e.g. backpropagation training.

4.6.1.3 Competitive Training

Processing elements (PE) compete among themselves. The PE with strongest response will be modified itself to match with the input. Another variation will be the winner take all, that is only one winner is allowed per set of points.

4.6.2 Unsupervised Training

This technique is also called self organization. In this procedure, the network is presented only with a series of input patterns and is given no information or feedback at all about its performance levels. Networks that use this kind of training procedure are most often used only for categorization or statistical modeling applications because the network's specific responses cannot be predetermined by the designer [16].

4.6.3 Reinforcement Training .

This is similar to supervised training except that the exact desired output is not provided only a "grade" on how well the network is doing. A measure is used to evaluate the performance of the network in generating outputs from input stimulus. It works like the Competitive Training by allowing the winner to modify itself closer to

the input pattem, while the loser will be penalized by moving away from the input pattern [16]

There are a number of schemes for this kind of training, and they vary from giving merely a brief "you succeeded" or "you failed" message to more informative "too high" or "too low" performance feedback.

4.7 Backpropagation training algorithm

The objective of training the network is to adjust the weight so the application of a set of inputs produces the desired set of outputs. For reasons of brevity, these input-output sets can be referred to as vectors. Training assumes that each input sets is paired with a target vector representing the desired output. Together these are called a training pair. Usually, a network is trained over a number of training pairs.

Before starting the training process, all of the weights must be initialized to small random number. This ensures that the network is not saturated by large values of the weights, and prevents certain other training pathologies. For example, if the weights all start at equal values and the desired performance requires unequal values, the network may not learn.

Training the backpropagation network requires the steps that follow:

1. Select the next training pair from the training set; apply input vector to the network input
2. Calculate the output of the network
3. Calculate the error between the network output and desired output (the target vector from the training pair)

4. Adjust the weights of the network in a way that minimizes the error
5. Repeat steps 1 through 4 for each vector in the training set until the error for the entire set is enough low

In step 1 and 2 above, an input vector is applied and the resulting output is calculated. Calculations are performed on a layer-by layer basis by equation 1 & 2. First the outputs of the neurons in the hidden layer are calculated; these are then used as inputs to output layer; the output layer neuron outputs are calculated and these constitute the network vector.

In step 3, each of the network output is subtracted from its corresponding component of the target vector to produce an error. This error is used in step 4 to adjust the weights of the network, where the polarity and magnitude of the weight changes are determined by the training algorithm.

After enough repetition of these four steps, the error between actual outputs and target output should be reduced to an acceptable value, and the network is said to be trained. At this point, the network is used for recognition and weights are not changed.

It may be seen that step 1 and 2 constitute a forward pass in that the signal propagates from the network input to its output. Step 3 and 4 are a "reverse pass", here, the calculated error signal propagates backward though the network where it is used to adjust weights. These two passes are now expanded and expressed in a somewhat more mathematical form.

4.7.1 Forward Pass

Step 1 and 2 can be expressed in vector form as follows: an input vector X is applied and an output vector Y is produced. The input-target vector pair X and T comes from the training set. The calculation is performed on X to produce the output vector Y.

As we have seen, calculation in multilayer networks is done layer by layer, starting at the layer nearest to the inputs. The weighted sum of its neuron's inputs of each neuron in the first layer is calculated.

The activation function F then squashes this value to produce the output value (OUT) for each neuron in that layer. Once the set of outputs for a layer is found, it serves as input to the next layer. The process is repeated, layer by layer, until the final set of network output is produced.

This process can be stated succinctly in vector notation. The weights between neurons can be considered to be a matrix W. For example, the weight from neuron 8 in layer 2 to neuron 5 in layer 3 is designated w_{85}. Rather than using the summation of products, the weighted sum vector for a layer N may be expressed as the product of X and W. In vector notation N=XW. Applying the sigmoid function F to the weighted sum vector N, component by component, produces the output vector O. Thus, for a given layer, the following expression describes the calculation process:

$$O=F(XW) \qquad \text{(EQ4.4)}$$

The output vector for one layer is the input vector for next, so calculating the outputs of the final layer requires the application of the equation to each layer, from the networks input to its output.

4.7.2 Reverse Pass

4.7.2.1 Adjusting the weight of the output layer:

Because a target value is available for each neuron in the output layer, adjusting the associated weights is easily accomplished using a modification of the Delta rule. Interior layers are referred to as hidden layers, as their outputs have no target values for comparison, hence, training is more complicated.

Figure 4.4 shows the training process for a single weight from neuron p in the hidden layer j to neuron q in the output layer k. The output of a neuron in output layer is subtracted from its target value to produce an ERROR signal. This is multiplied by the derivative of the squashing function [OUT(1-OUT)] calculated for that layer's neuron k, thereby producing the δ value.

δ =OUT(1-OUT)(Target-OUT).....(EQ4.5)

Then δ is multiplied by OUT from a neuron j, the source neuron for the weight in question. This product is in turn multiplied by a training rate coefficient η (typically 0.001 to 1) and the result is added to the weight. An identical process is

performed for each weight proceeding from a neuron in the hidden layer to a neuron in the output layer.

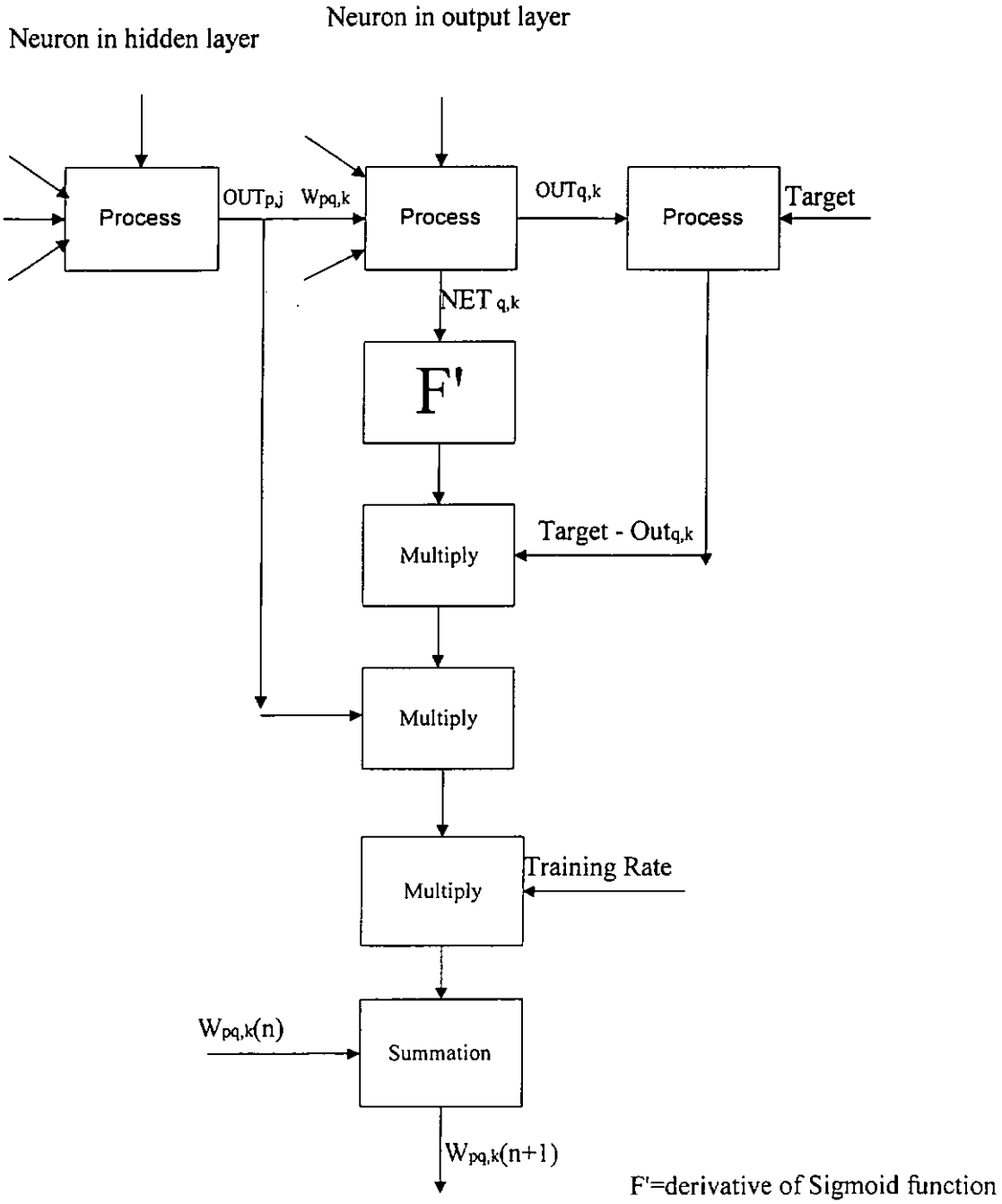


Fig 4.4 Adjusting the weights

The following equations illustrate this calculation:

$$W_{pq,k} = \eta \delta_{q,k} OUT_{p,j}. \qquad \text{(EQ4.6)}$$

$$W_{pq,k}(n+1) = W_{pq,k}(n) + W_{pq,k}$$

(EQ4.7)

Where: $W_{pq,k}(n)$=the value of a weight from neuron p in the hiden layer to neuron q in the out layer at step n (before adjustment); note that the subscript k indicates that the weight is associated with its destination layer.

$W_{pq,k}(n+1)$ = value of the weight at step n+1 (after adjustment)

η= learning rate

$\delta_{q,k}$ = the value of δ for neuron q in the output layer k

OUT_{pj} = the value of OUT for neuron p in the hidden layer j

Note that subscripts p and q refer to a specific neuron; j and k refer to a layer.

4.7.3 Adjusting the weights of the hidden layer

Hidden layers have not target vector, so the training process described above can not be used. This lack of a training target stymied effects to train multilayer networks until backpropagation provided a workable algorithm. Backpropagation trains the

hidden layers by propagating the output error back through the network layer by layer, adjusting weights at each layer.

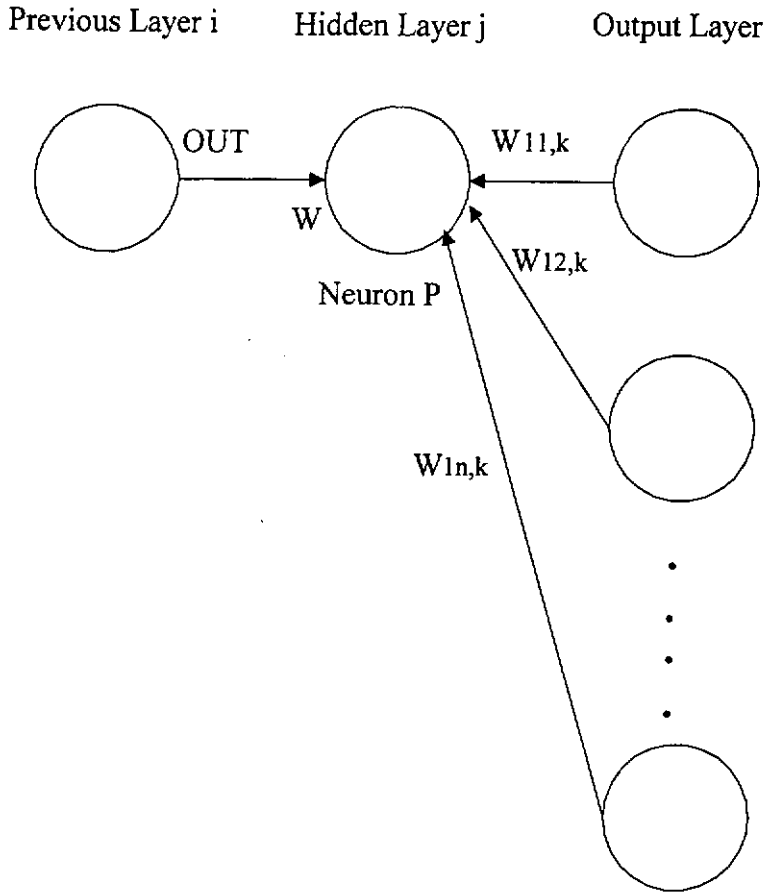


Figure 4.5 Training a Weight in a Hidden Layer

Equations 4.6 and 4.7 are used for all layers, both output and hidden; however, for hidden layer Delta must be generated without benefit of a target vector. Figure 4.4 shows how this is accomplished. First, Delta is calculated for each neuron in the output layer, as in equation 4.5. It is used to adjust the weights feeding into the output layer, then it is propagated back through the same weights to generate a value for Delta for each neuron in the hidden layer. These values of Detla are used, in training,

to adjust the weights of this hidden layer and, in a similar way, are propagated back to the input layer.

Consider a simple neuron in the hidden layer just before the output layer. In the forward pass, this neuron propagates its output value to neurons in the output layer through the interconnecting weights. During training these weigths operate in reverse, passing the value of Delta from the output layer back to the hidden layer. Each of these weights is multiplied by the Delta value of the neuron to which it connects in the output layer. The value of Delta needed for the hidden-layer neuron is produced by summing all such products and multiplying by the derivative of the squashing function:

$$\delta_{p,j} = OUT_{p,j}(1 - OUT_{p,j})(\sum_{q} \delta_{q,k} W_{pq,k}) \qquad \text{(EQ4.8)}$$

With Delta in hand, the weights feeding the first hidden layer can be adjusted using Equation 4.6 and 4.7, modifying indices to indicate the correct layers.

For each neuron in a given hidden layer, Delta must be calculated, and all weights associated with that layer must be adjusted. This is repeated, moving back toward the input layer, until all weights are adjusted.

4.7.4 The Global Error Function

The above discussion has assumed the existence of a global error function without actually specifying it. This fuction is needed to define the local errors at the output

layer so that they can be propagated back through the network. Suppose a vector i is presented at the input edge layer of the network and suppose the desired output d is specified by a teacher. Let o denote the actual output produced by the network with its current set of weights. Then a measure of the error in achieving that desired output is given by

$$E = \sum_{k}((d_k - o_k)^2) \qquad \text{(EQ4.9)}$$

where the subscript k indexes the components of d and o.Here, the raw local error is given by $d_k - o_k$.

Chapter 5

Genetic Algorithms (GA)

5.1 Genetic Algorithms (GA)

Goldberg has defined GA as such "GA are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomised information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old, an occasional new part is tried for good measure. While randomized, GA are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance. Indeed, GA is a newly developed approach to search for optimization of mathematical function. GA are search and optimization methods based on natural evolution and natural selection by which individual of higher fitness have more offspring,

In GA, solutions of optimization problem are regarded as individuals and objective function is regarded as fitness function. Candidates of a final solution are represented by strings of fixed length encoded in consideration with architecture of problem. The

strings are called chromosome. Each individual has one chromosome and forms a population.

Each individual of the population is evaluated by fitness function. The individuals of higher fitness value have more offspring and the next generation is created by the offspring.

GAs have the following features

1) It searches from a population of points, not single point.
2) It uses payoff (fitness or objective functions) information directly for the search direction, not derivatives or other auxiliary knowledge.
3) It uses probabilistic transition rules to select generations, not deterministic rules, so they can search a complicated and uncertain area to find global optimum.

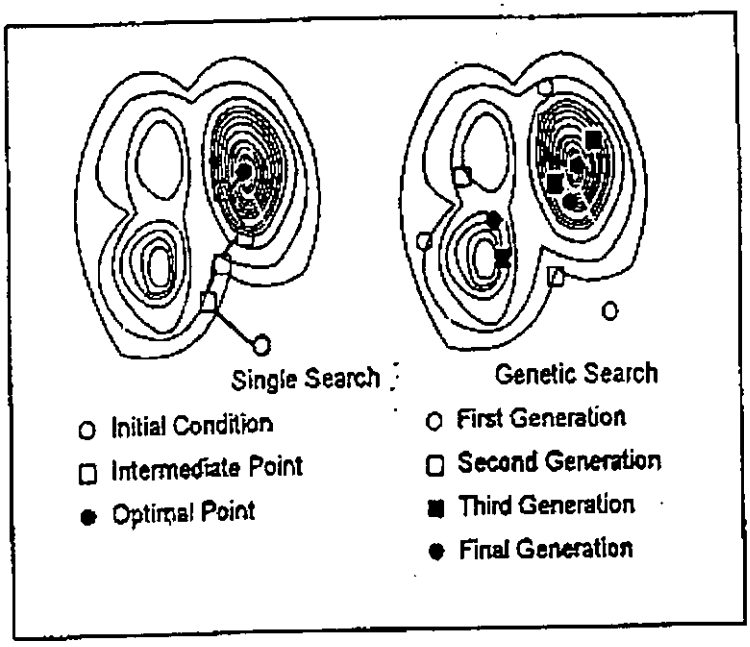


Figure 5.1 Comparison of single search and multiple search [20]

5.1.1 The mechanism of GAs

GA is a population search method. A population of strings are kept in each generation. The next generation is produced by the simulation of natural process of reproduction, gene crossover and mutation.

Selection:

Selection models nature's survival of the fittest principle. The aim is to ensure that fitter strings (solutions) receive a higher number of offspring, thereby getting a higher chance of surviving in the new generation. Selection strategies include Roulette wheel selection , Tournament selection and Ranking selection. The Roulette wheel selection scheme allocates offspring based on the ratio of the fitness value of a string to the average fitness value of the population. Each string is allocated a slot (sector angle = 2 π fitness/ average fitness of a roulette wheel). A string is allocated an offspring if a random number between 0 and 2π falls in the sector corresponding to the string. It is intuitive and easy to implement. In Ranking selection, it rank members based on their fitness within the population, rather than the magnitude of the fitness. The method initially sorts all the elements of the population according to their objective function values, then assigns a maximum number of offspring to the best element of the population and a linearly decreasing count of offspring to the rest of the population. The use of ranking allows the algorithm to control the allocation of the number of offspring to the members of the population and, as a result, the loss of diversity which usually leads to premature convergence can be substantially reduced. Tournament selection is deterministic. Pairs of individuals are picked at random from the population. The one with higher fitness is copied into the mating pool. Larger

tournaments are employed by picking n individuals instead of two (a pair). This increases the selection pressure.

Mutation

Every string in the "mating pool" may be mutated with the given mutation probability. For the string that is undergoing the mutation, a number will be randomly selected from a uniformly distributed [0,l] domain. If this number is less than the mutation probability, the bits in a string will be changed from 1 to 0, or vice versa. The mutation operator produces a new string.

Crossover

The strings in the "mating pool" are grouped in couples. Each couple of strings would swap their bit according to the crossover probability. The uniform crossover is used here which gives every bit in the string the same chance to undergo crossover and therefore is better than one point crossover and two point crossover. The crossover operation would happen, if a number for crossover, also randomly selected from a uniformly distributed [0,l] domain, is less than the given crossover probability. A mask string is set up with randomly selected bits. The bits of the couple strings that are corresponding I's bits of the mask string will swap, while others will stay.

After all strings finish mutation and crossover, a new generation is produced. Each string is decoded back to the control variables to compute its fitness score. The

statistics will compute the new maximum fitness, minimum fitness, sum of fitness and average fitness. The converge condition will be checked. The programme will stop if the condition is met, or otherwise, a new cycle of reproduction, mutation and crossover will start.

5.2 How it works

For the demonstration of the GA we would like to quote the example from the book written by Mr.Goldberg named "Genetic klgorithms in Search, Optimizationk & Machine Learning".

String no.	Initial Population (Randomly Generated)	x value (Unsigned Integer)	f(x) x^2	Pselect fi Σf	Expected Count fi f	Actual count from Roulette wheel	Mating pool after reproduction	Mate (randomly selected)	Crossover site	New population	x value	f(x) x^2
1	0 1 1 0 1	13	169	0.14	0.58	1	0 1 1 0 1	2	4	0 1 1 0 0	12	144
2	1 1 0 0 0	24	576	0.49	1.97	2	1 1 0 0 0	1	4	1 1 0 0 1	25	625
3	0 1 0 0 0	8	64	0.06	0.22	0	1 1 0 0 0	4	2	1 1 0 1 1	27	729
4	1 0 0 1 1	19	361	0.31	1.23	1	1 0 0 1 1	3	2	1 0 0 0 0	16	256
Sum			1170	1.00	4.00	4.0						1754
Average			293	0.25	1.00	1.0						439
Max			576	0.49	1.97	2.0						729

Table 5.1

This example is to maximize the function $f(x)=x^2$, where x is permitted to vary between 0 and 31 (i.e. the variation is between 00000 and 1 1 1 1 1 in binary digit).

In the table note how both the maximal and average performance has improved in the new generation. In simple term, GA efficiently build new solutions from the best partial solution of the previous trials. GA ruthlessly exploit wealth of information by

(1) reproducing high quality notions according to their performance and

(2) crossing these notions with many other high performance notions from other

strings.

Thus, the action of crossover with previous reproduction speculate on new ideas constructed from the high performance building blocks (notions) of past trials. Mutation plays a decidedly secondary role in the operation of GA. Mutation is needed because, even though reproduction and crossover effectively search and recombine extant notions, occasionally they may become overzealous and lose some potentially useful genetic material (l's or 0's at particular location). In artificial genetic systems, the mutation operator protects against irrecoverable loss. In the simple GA mutation is the occasional (with small probability) random alteration of the value of a string position. It is an insurance policy against premature loss of important notions.

Another approach to explain this interesting process is in term of schemata (Similarity templates). A schema (Holland, 1968, 1975) is a similarity template describing a subset of strings with similarities at certain string positions.

For example, the schema *111* describes a subset with four members {01110, 01111, 11110, 11111}. The idea of schema gives us a powerful and compact way to talk about all the well-defined similarities among finite-length strings over a finite alphabet.

Now we consider the effect of reproduction, crossover and mutation on the growth or decay of important schemata from generation to generation .

The effect of reproduction on a particular schema is easy to determine, since more highly fit strings have higher probabilities of selection, on average we give an ever increasing number of samples to the observed best similarity patterns, however, reproduction alone samples no new points in the space. What then happens to a particular schema when crossover is introduced? Crossover leaves a schema unscathed if it does not cut the schema but it may disrupt a schema when it does.
For example, consider two schemata 1***0 and **11*. The first is likely to be disrupted by crossover whereas the second is relatively unlikely to be destroyed. As a result schemata of short defining length are left alone by crossover and reproduced at a good sampling rate by reproduction operator.

Mutation at normal, low rates does not disrupt a particular schema very frequently and we are left with a startling conclusion. Highly fit, short-defining-length schemata (building blocks) are propagated generation to generation by giving exponentially increasing samples to the observed best.

5.3 Terminology

Since GA is rooted in both natural genetics and computer science, the terminology used in GA literature is a mix of natural and artificial.

The following table is the comparison of Natural and GA terminology:

Natural	**GA**
chromosome	string
gene	feature, character, or detector
allele	feature value
locus	string position
genotype	structure
phenotype	parameter set, alternative solution, a decoded structure
epistasis	non-linearity

Chapter 6

Evolving Neural Network

Evolving Neural Network is to use GA to train the NN. Figure 6.1 shows the framework of ENN. Basically, there are two options of GA for training the NN, namely connections and weights. The connections option uses the GA to evolve the interconnect structure of the NN, in which the architecture of the network is fixed. After deciding the connection of the network, it will use back-propagation to train the weight. The weight option uses GA as weight training to find out initial set of good weights of the NN and then uses back-propagation to fine tune the weight of the network.

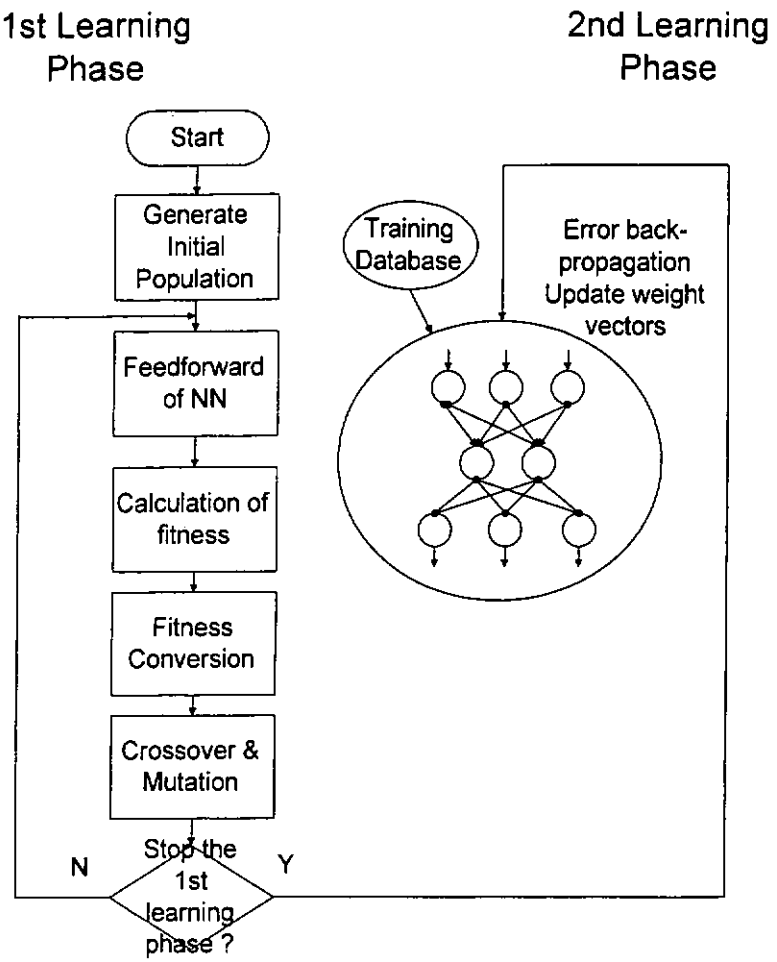


Figure 6.1 Framework of Evolving Neural Network

6.1 Generate Initial Population Pool

In GA, each possible solution is an encoded string of 1s and 0s of a specific length (a member of the population). Sets of such strings (members) form the population and the number of strings is the size of the population. A random number generator is used to generate individuals. The chromosome length of any individual is representing the weight value and topology of the NN.

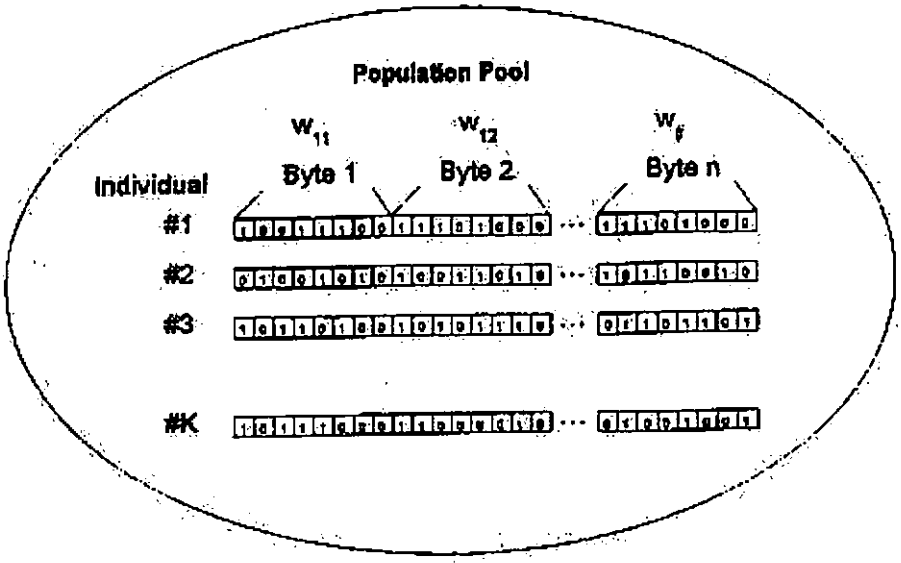


Figure 6.2 Population pool

6.2 Feed-forward of NN

The phenotype is the physical expression of a chromosome where the genotype is the genetic structure of a chromosome. In NN terms, the phenotype is the weight value or the topology of the NN.

In this process the genotype of chromosome is converted to phenotype and the NN will output the value based on this configuration

6.3 Calculation of fitness

The main issue of training is to minimize the objective function as follows:

$$E = \sum_{m}((d_m - y_m)^2) \qquad \text{(EQ6.1)}$$

Where d is the target output

y is the NN output.

The fitness is achieved in a way that smaller value is mapped to larger fitness values.

6.4 Fitness Conversion

The chromosome now expressed with different fitness value, they are then selected by the method described in Chapter 5.1.1 to be a parent. The following figures show the detail of the three selection methods.

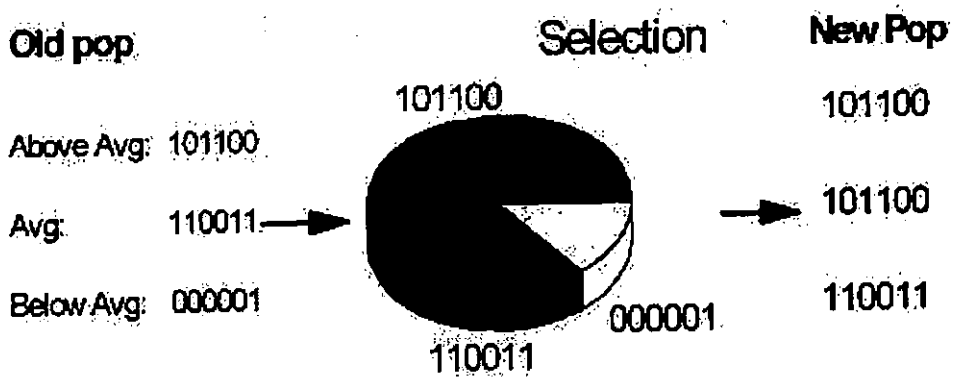


Figure 6.3 Roulette Wheel Selection

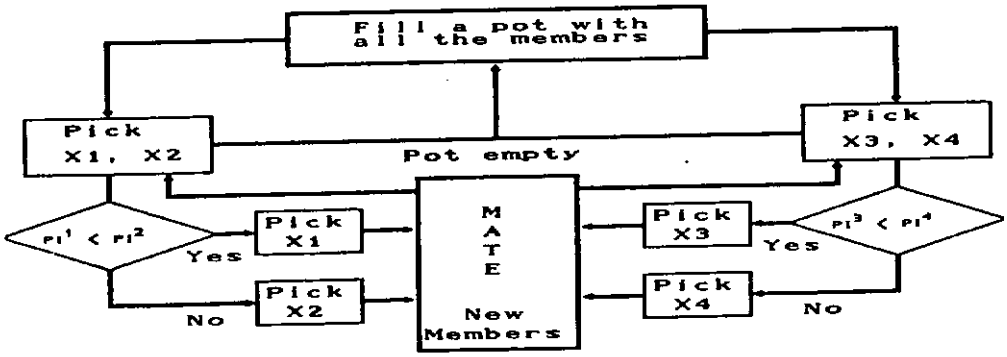


Figure 6.4 Tournament Selection

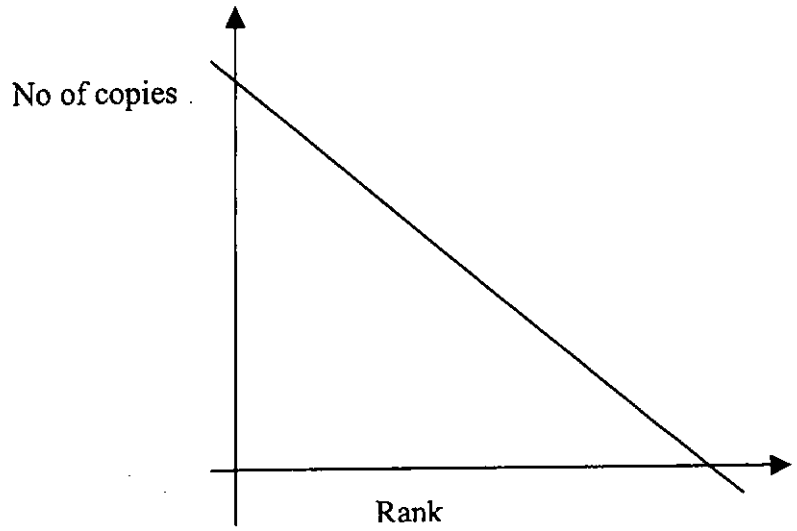


Figure 6.5 Ranking Selection

6.5 Crossover and Mutation

The new population will be generated in this step by applying crossover and mutation on the parent chromosome.

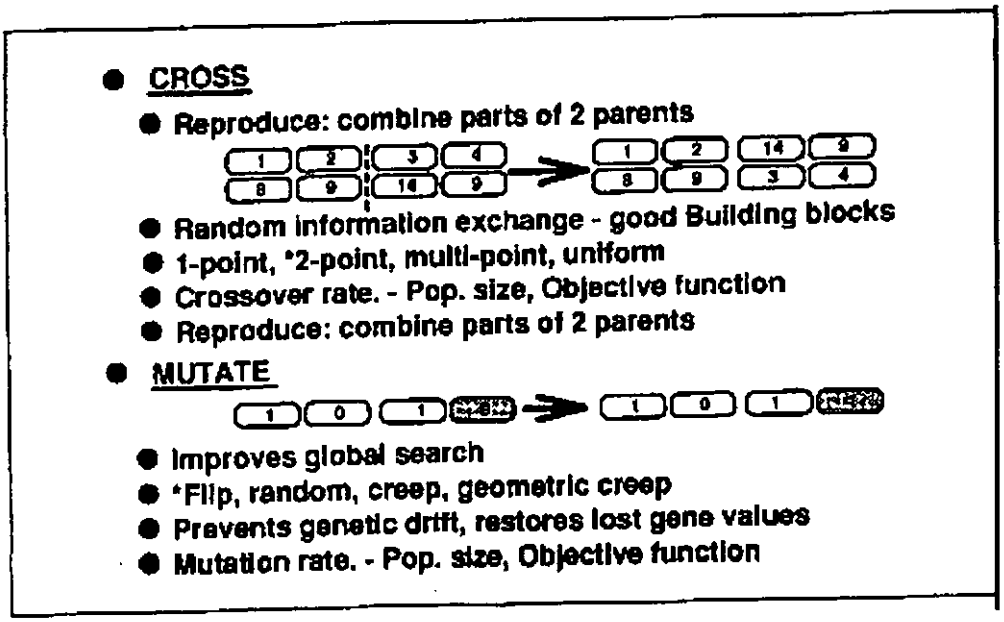


Fig. 6.6 Example of the operation of crossover and mutation

6.6 Second Learning Phase

The second learning phase will start once the stopping criteria in phase one is met. The stopping criteria of phase one are as follows:

Stop after a period of time

Stop when the average is close to the minimum

Stop when there is no improvement in the minimum value

Stop after finding a better solution than the one existed

Stop after finding a desired minimum

In the second phase, NN will use the back propagation to fine-tune the weight of the network.

Chapter 7

Software Overview

In this project a software call Flextool (ENM) will be used for the testing of the algorithm of ANN and ENN. The software is developed by a company called Flexible Intelligence Group, L.L.C. (FIG).

The software is a modular software tool that provides an environment for applying evolving neural network. It includes the Flextool (NN) and Flextool (GA) modules. Version M2.2 is the MATLAB version that provides a total neuro system based design and development environment in MATLAB using graphical menus. The ENM components are provided as .m files. The software is designed to evolve complex system by drawing on the power of MATLAB.

The figures below are the view of the software:

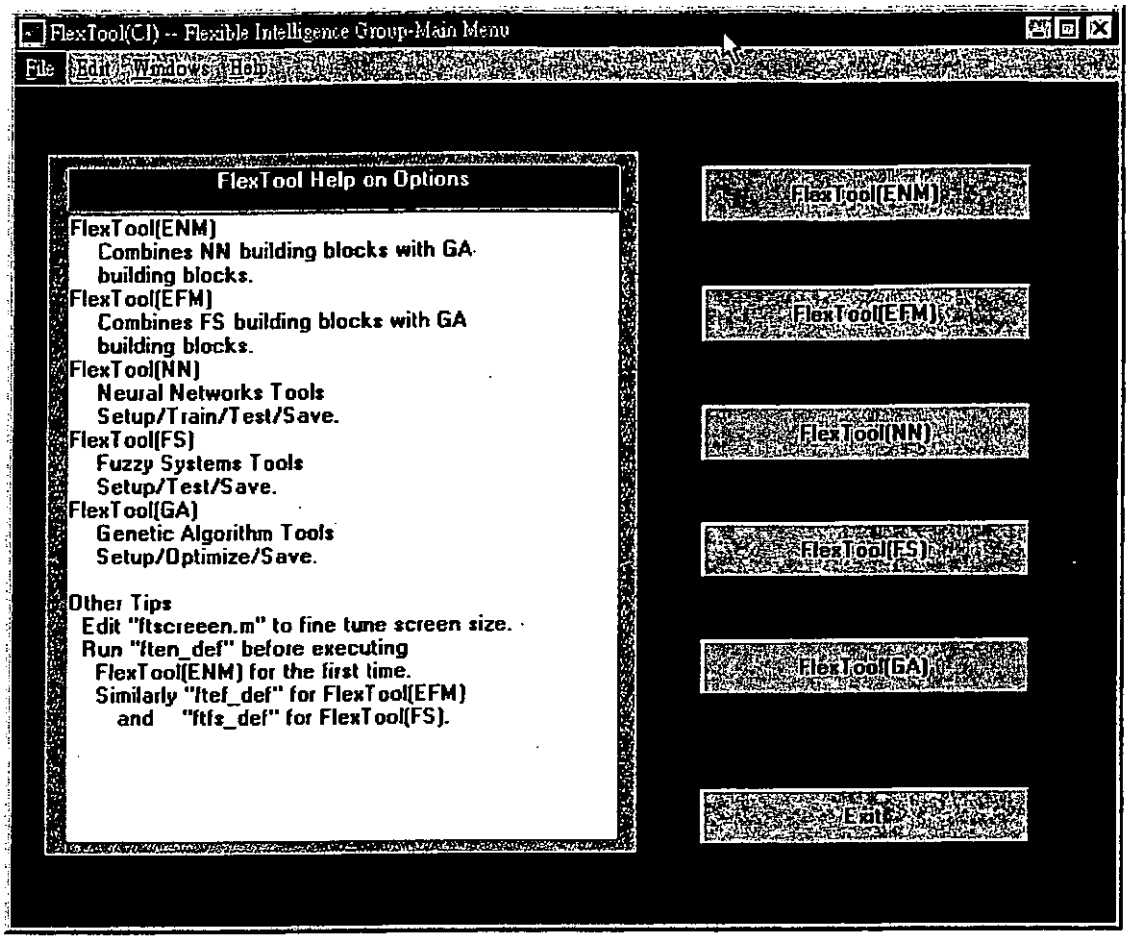


Fig 7.1 Front Page of the program

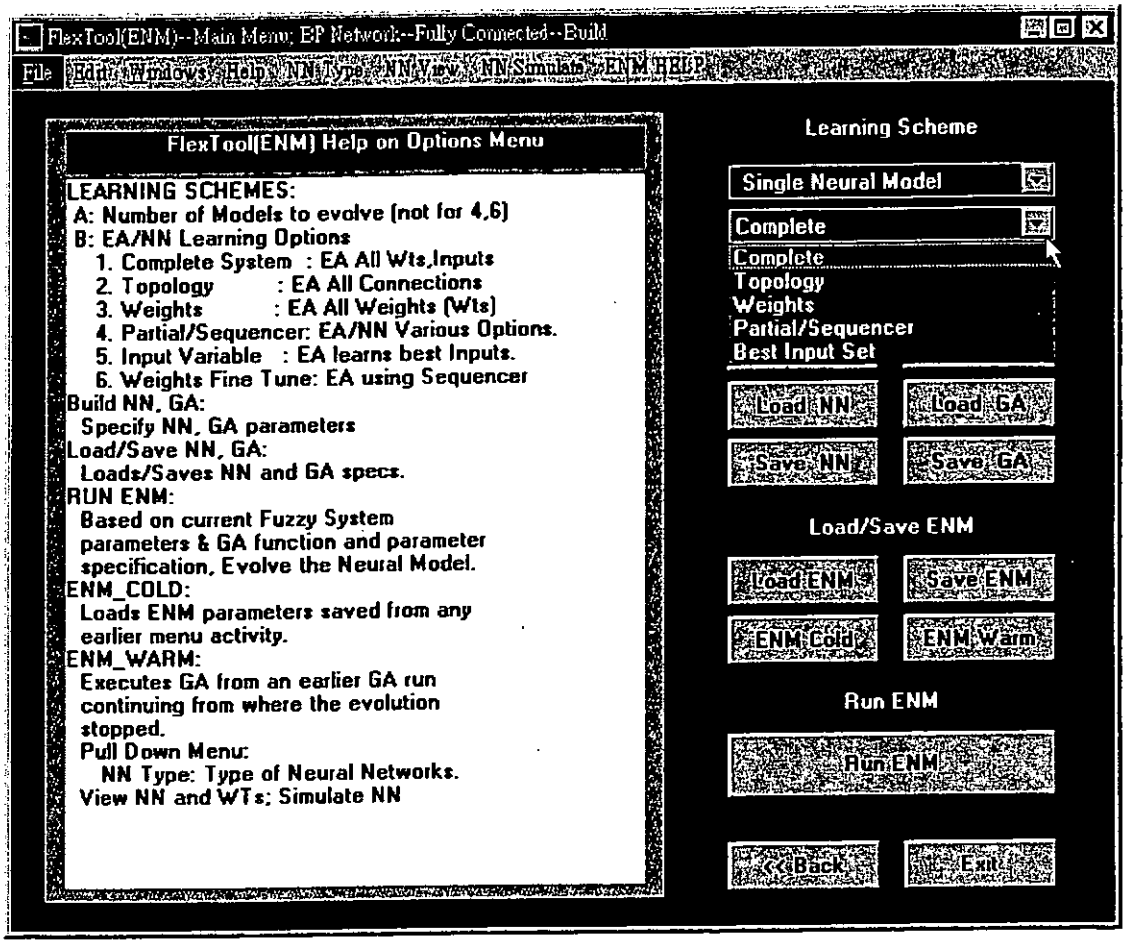


Fig 7.2 Main Menu of the program

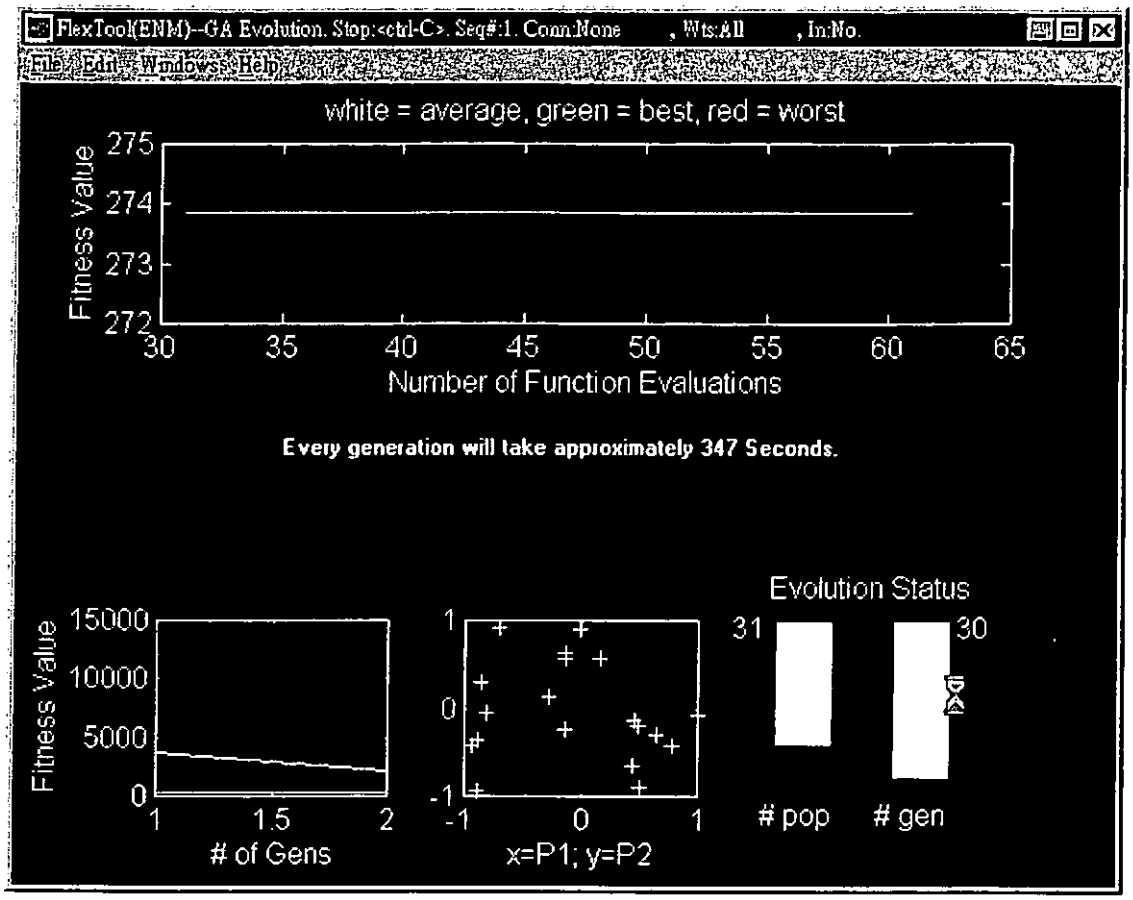


Fig 7.3 Training View of the program

Chapter 8

Implementation

8.1 Parameter Determination

Since behavior of a GA is strongly related to the parameters involved in the algorithms, that is, population size, probability of crossover and probability of mutation. Nevertheless, the method used for selection has a dominant influence on the effectiveness of the algorithm.

An investigation of selection methods is conducted in order to determine their effect on the performance of the GA in terms of their relevant parameters. Three selection methods: Roulette Wheel, Tournament and Ranking as well as two options: Weight and Connection are combined to have six approaches of running the GA and will be listed as follows:

Approach 1: Roulette wheel selection with weight option

Approach 2: Tournament selection with weight option

Approach 3: Ranking selection with weight option

Approach 4: Roulette wheel selection with connection option

Approach 5: Tournament selection with connection option

Approach 6: Ranking selection with connection option

8.1.1 Weight Option

8.1.1.1 Roulette Wheel Selection: Figure 1 to figure 3, shows the behavior of an ENN with roulette wheel selection of weight option. The result shown in figure 8.1 is performed by different population sizes, a probability of cross over Pc = 0.7 and probability of mutation Pm = 0.01. It can be known that the population size of 31 is the best result.

A variation for the same approach is presented in figure 8.2.

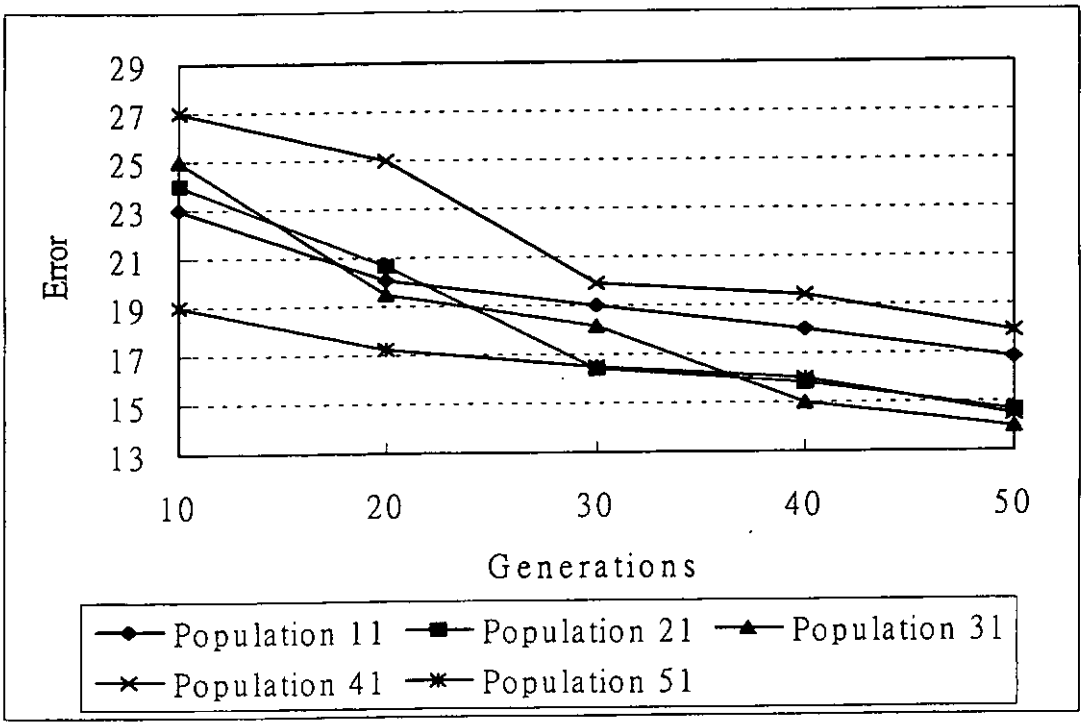


Figure 8.1 Weight option with Roulette Wheel Selection, Pc = 0.7, Pm = 0.01

This time the size of population and the probability of crossover are kept constant (pop size =31, Pc=0.7) whilst different probability of mutation are considered.

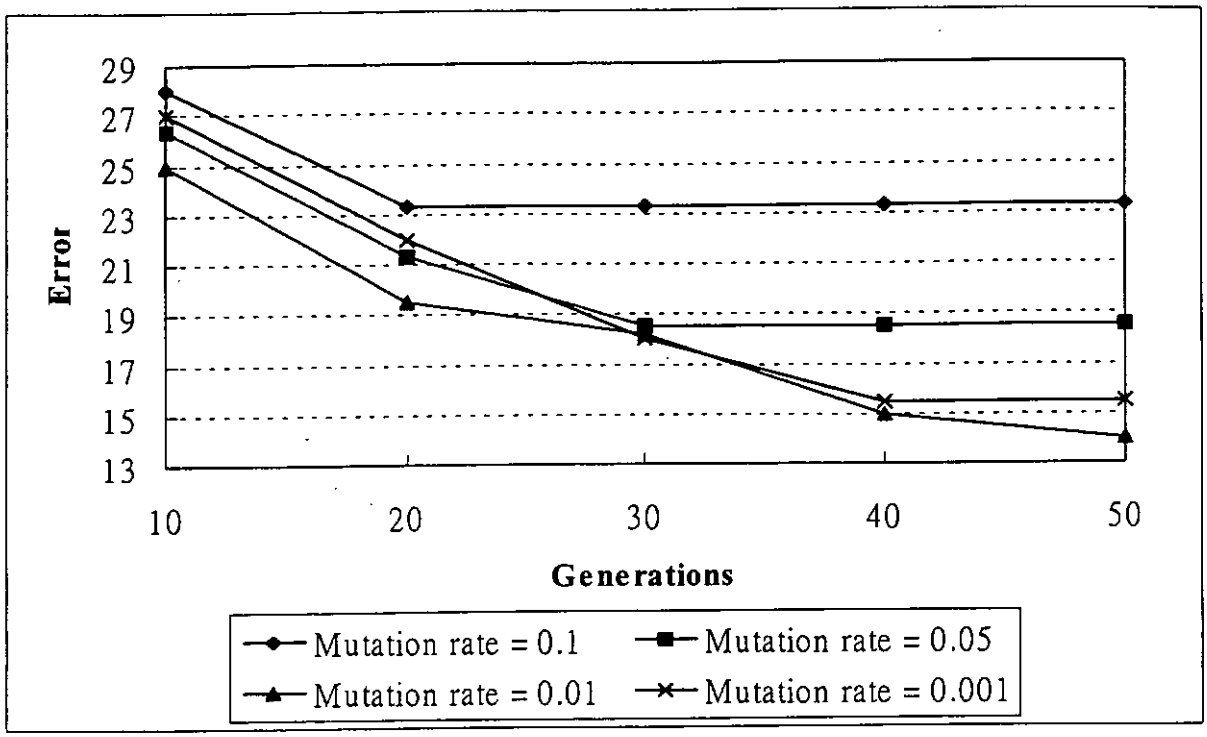


Figure 8.2. Weight option with Roulette Wheel Selection, pop size =31, Pc=0.7

In this time the size of population and the probability of mutation are kept constant (pop size =31, Pm=0.01) whilst different probability of crossover are considered.

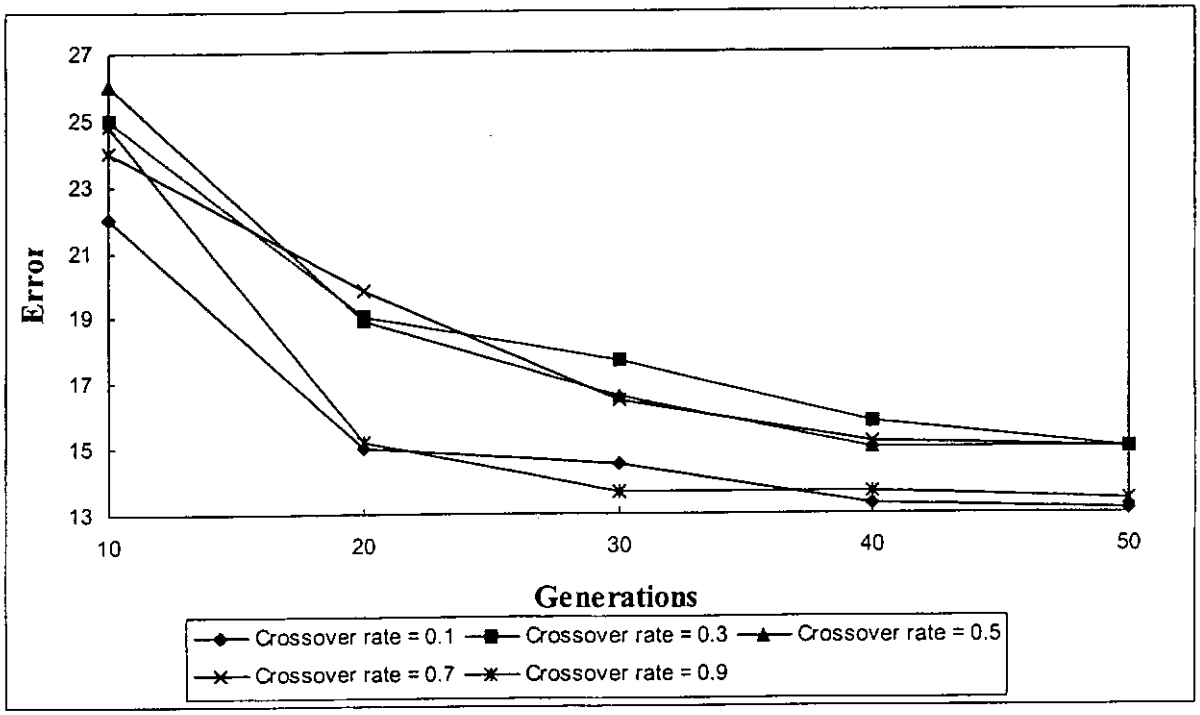


Figure 8.3. Weight option with Roulette Wheel Selection, pop size =31, Pm=0.01

8.1.1.2 Tournament Selection: Figure 4 to figure 6, shows the behavior of an ENN with tournament selection of weight option. The result shown in figure 8.4 is performed by different population sizes, a probability of cross over Pc = 0.7 and probability of mutation Pm = 0.01. It can be known that the population size of 31 is the best result.

A variation for the same approach is presented in figure 8.4.

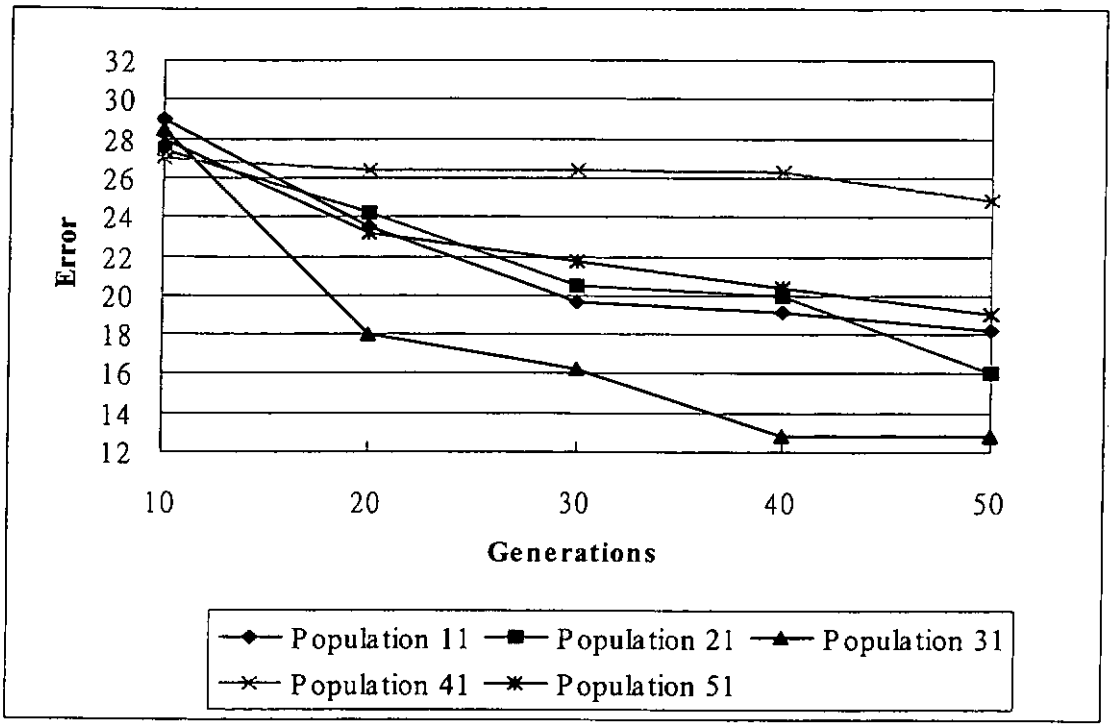


Figure 8.4 Weight option with Tournament Selection, Pc = 0.7, Pm = 0.01

This time the size of population and the probability of crossover are kept constant (pop size =31, Pc=0.7) whilst different probability of mutation are considered.

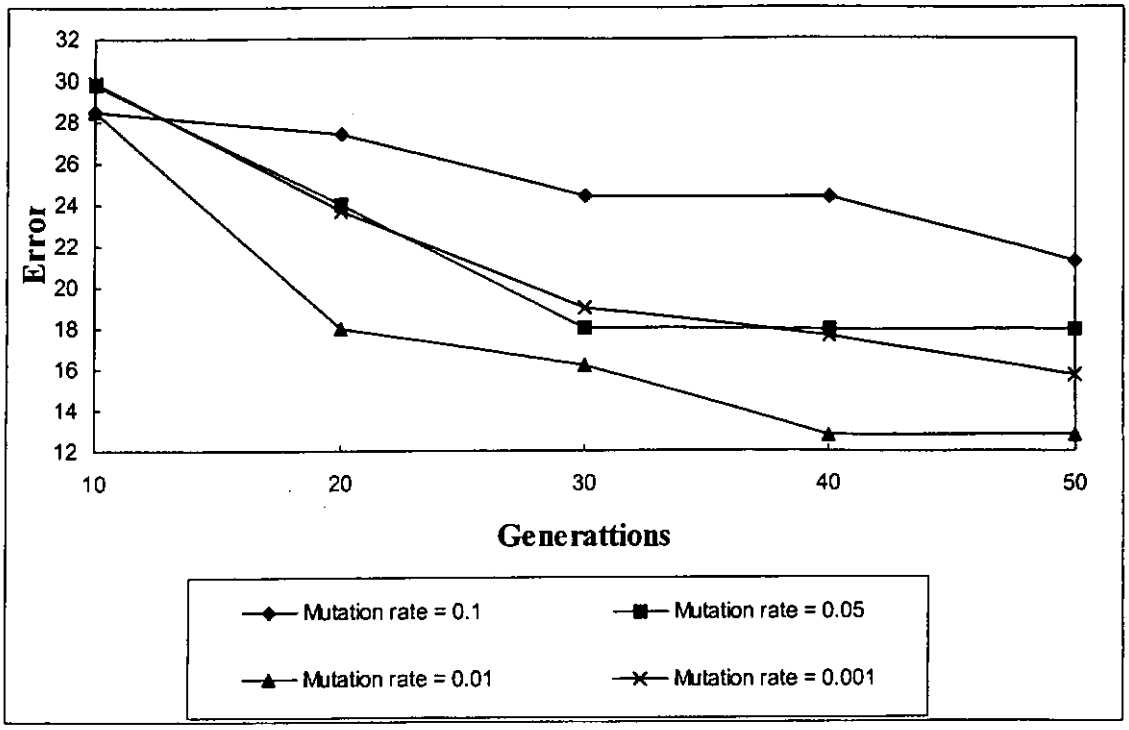


Figure 8.5 Weight option with Tournament Selection, pop size =31, Pc=0.7

In this time the size of population and the probability of mutation are kept constant (pop size =31, Pm=0.01) whilst different probability of crossover are considered.

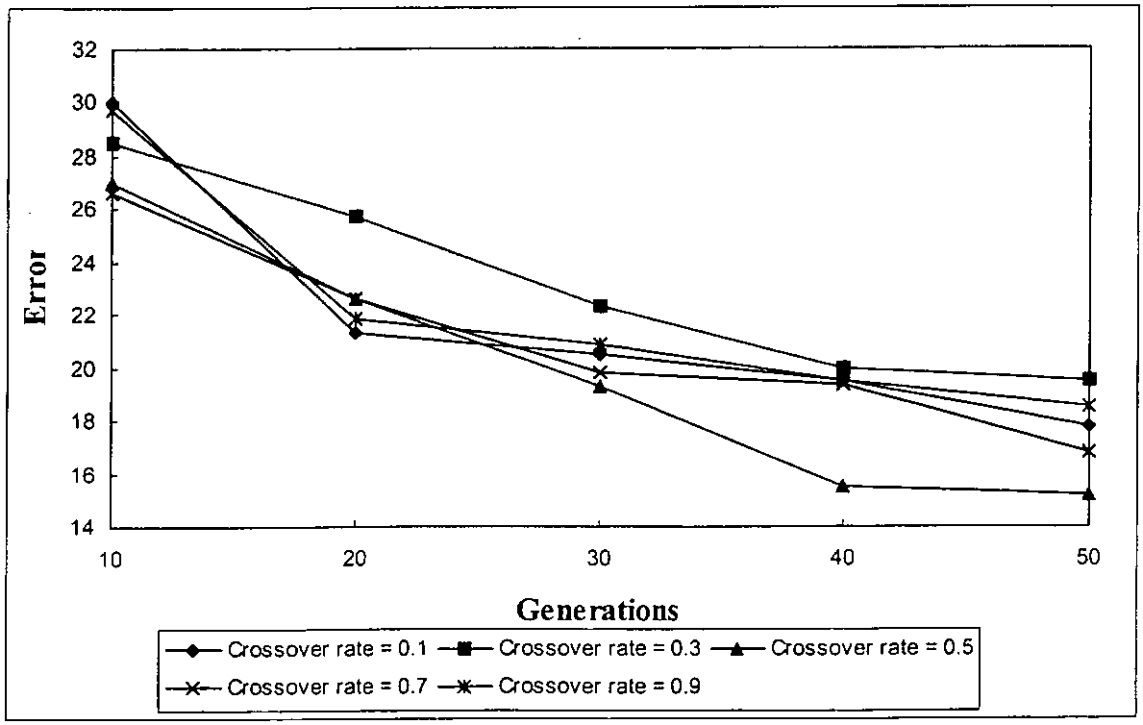


Figure 8.6 Weight option with Tournament Selection, pop size =31, Pm=0.01

8.1.1.3 Ranking Selection: Figure 7 to figure 9, shows the behavior of an ENN with ranking selection of weight option. The result shown in figure 8.7 is performed by different population sizes, a probability of cross over Pc = 0.7 and probability of mutation Pm = 0.01. It can be known that the population size of 41 is the best result. A variation for the same approach is presented in figure 8.7.

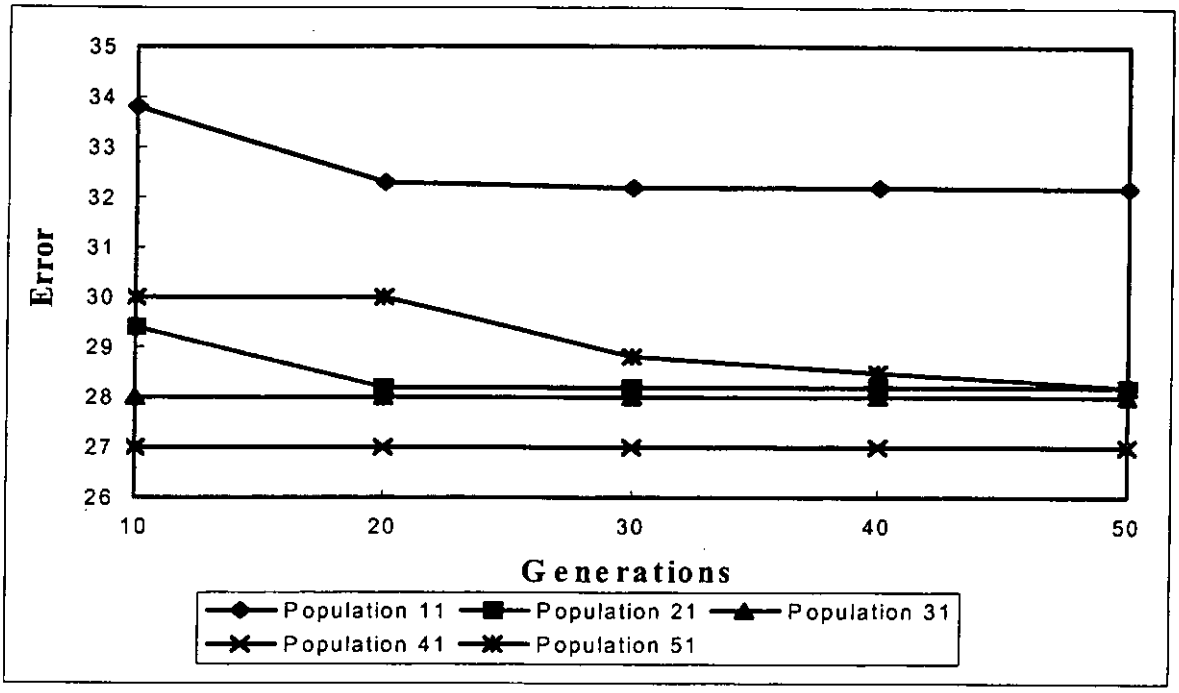


Figure 8.7 Weight option with Ranking Selection, Pc = 0.7, Pm = 0.01

This time the size of population and the probability of crossover are kept constant (pop size =41, Pc=0.7) whilst different probability of mutation are considered.

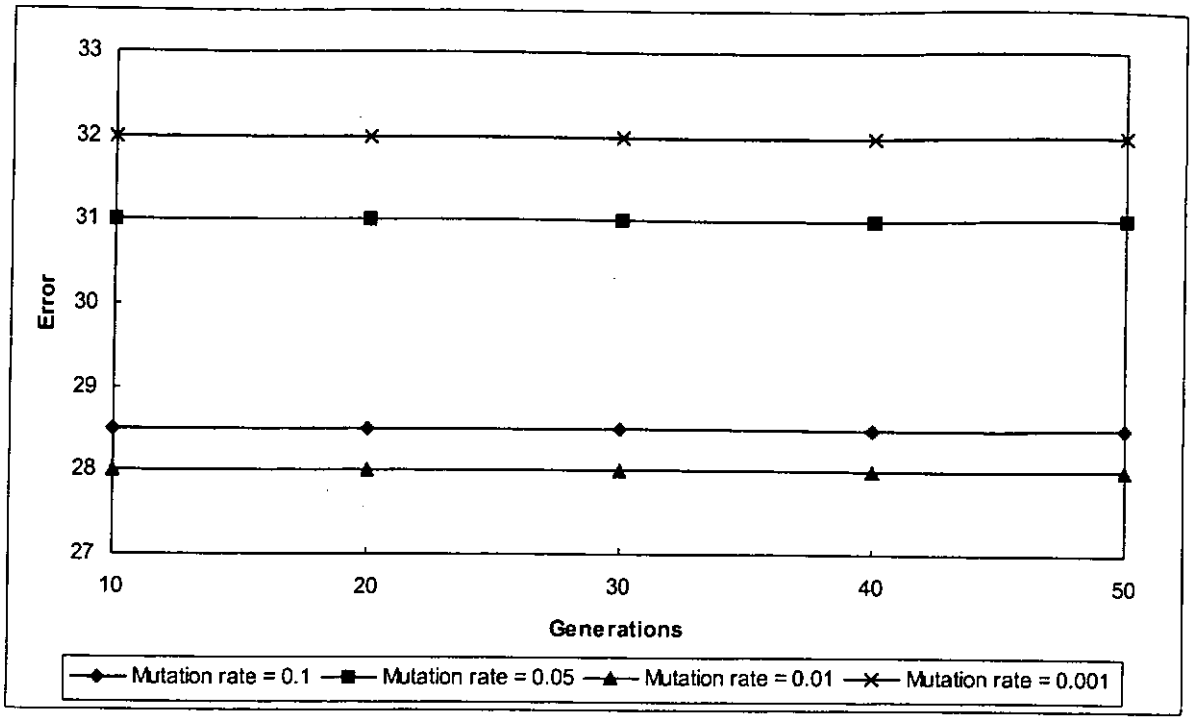


Figure 8.8 Weight option with Ranking Selection, pop size =41, Pc=0.7

In this time the size of population and the probability of mutation are kept constant (pop size =41, Pm=0.01) whilst different probability of crossover are considered.

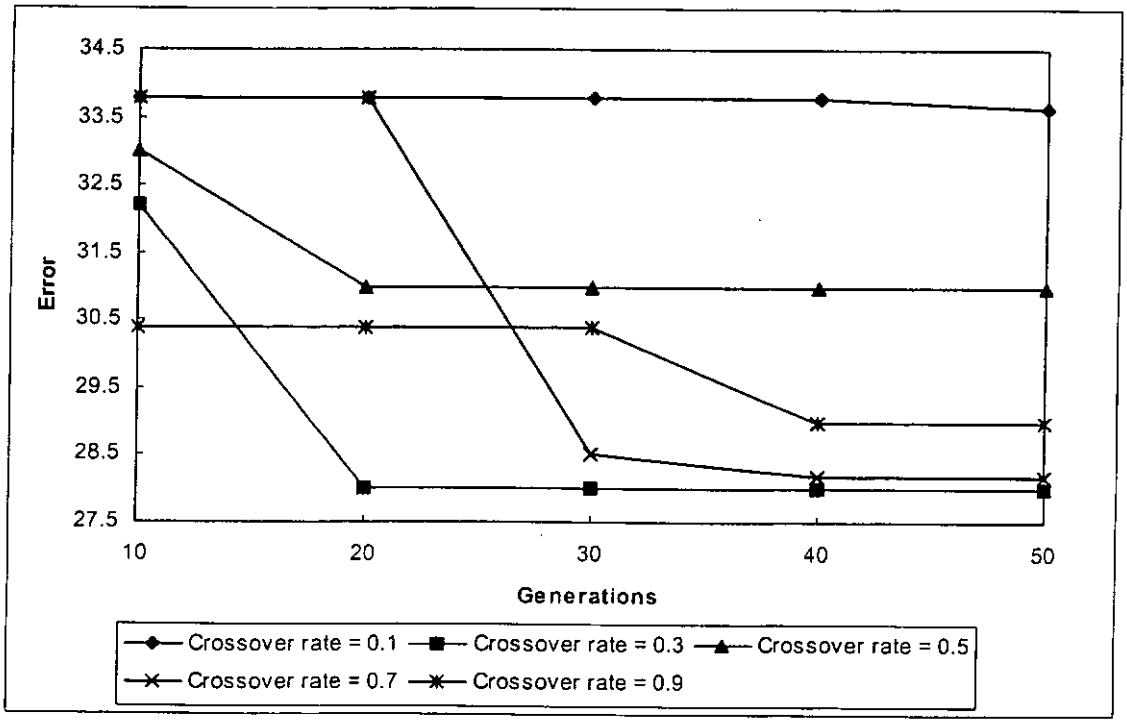


Figure 8.9 Weight option with Ranking Selection, pop size =41, Pm=0.01

8.1.2 Connections Option

8.1.2.1 Roulette Wheel Selection: Figure 10 to figure 12, shows the behavior of an ENN with roulette wheel selection of connection option. The result shown in figure 8.10 is performed by different population sizes, a probability of cross over Pc = 0.7 and probability of mutation Pm = 0.01. It can be known that the population size of 41 is the best result.

A variation for the same approach is presented in figure 8.11.

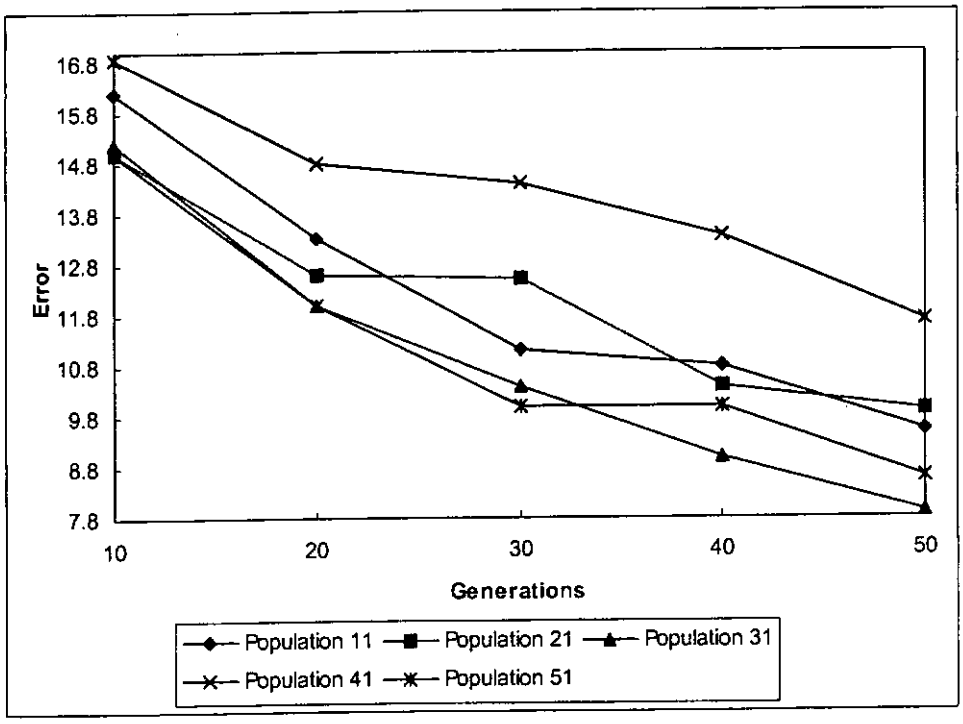


Figure 8.10. Connection option with Roulette Wheel Selection, Pc = 0.7, Pm = 0.01

This time the size of population and the probability of crossover are kept constant (pop size =31, Pc=0.7) whilst different probability of mutation are considered.

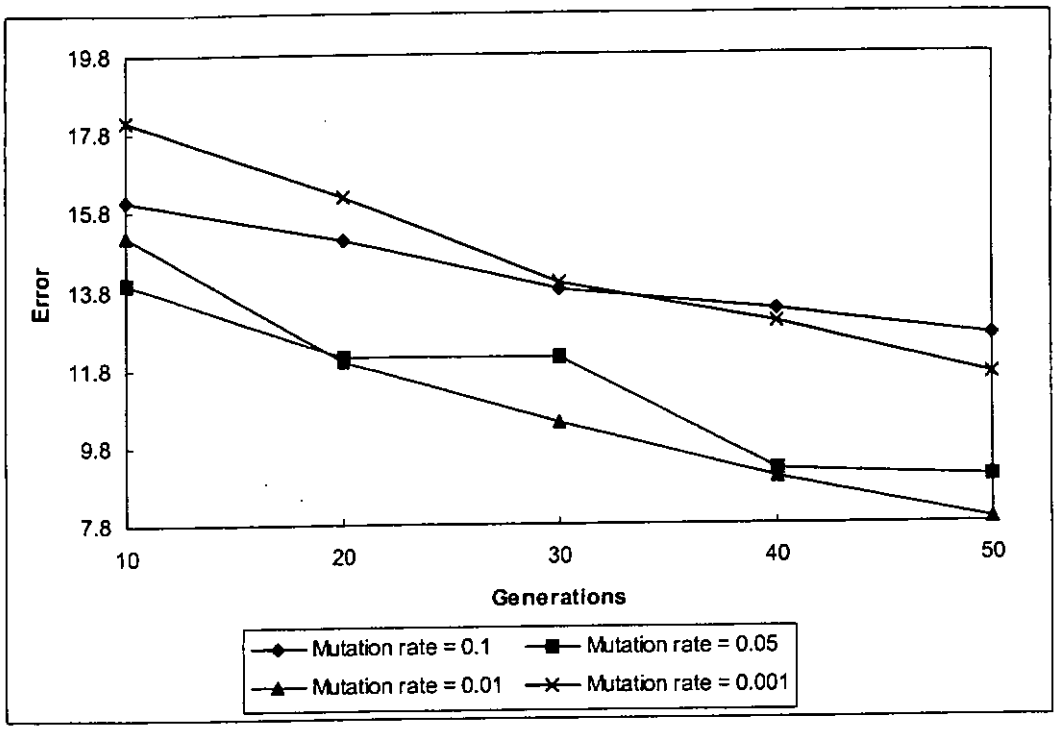


Figure 8.11 Connection option with Roulette Wheel Selection, pop size =31, Pc=0.7

In this time the size of population and the probability of mutation are kept constant (pop size =31, Pm=0.01) whilst different probability of crossover are considered.

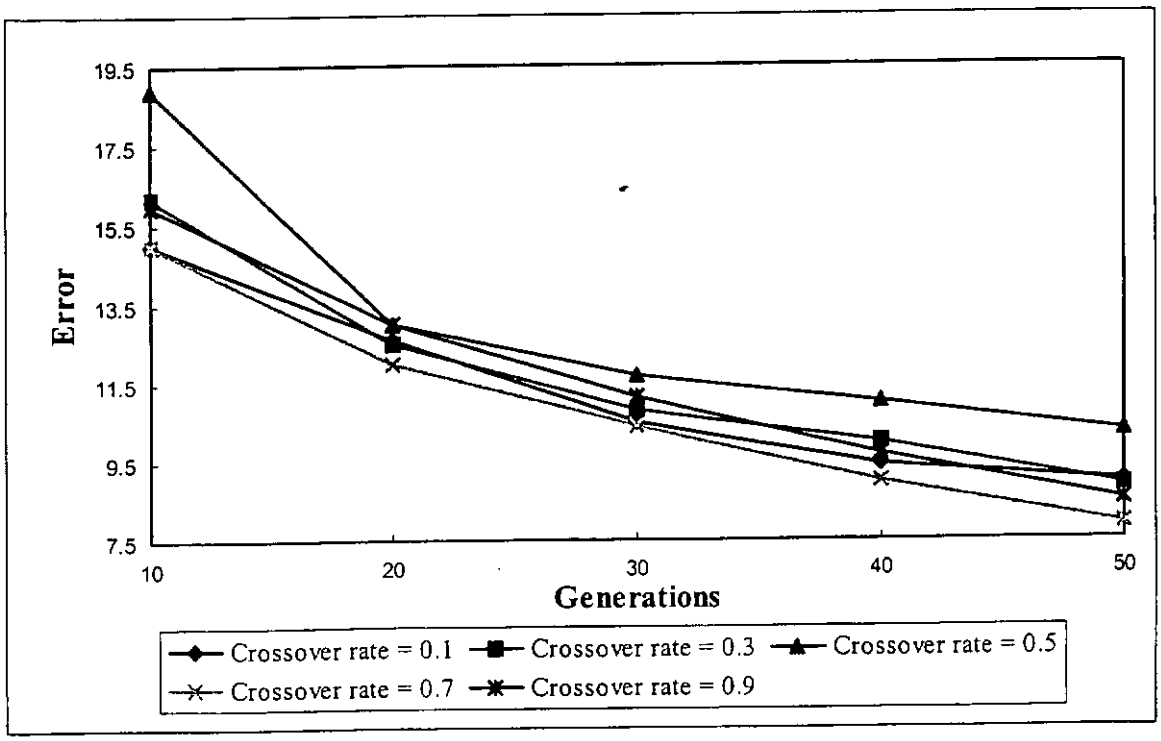


Figure 8.12. Connection option with Roulette Wheel Selection, pop size =31,Pm=0.01

8.1.2.2 Tournament Selection: Figure 13 to figure 15, shows the behavior of an ENN with tournament selection of connection option. The result shown in figure 8.13 is performed by different population sizes, a probability of cross over Pc = 0.7 and probability of mutation Pm = 0.01. It can be known that the population size of 41 is the best result.

A variation for the same approach is presented in figure 8.14.

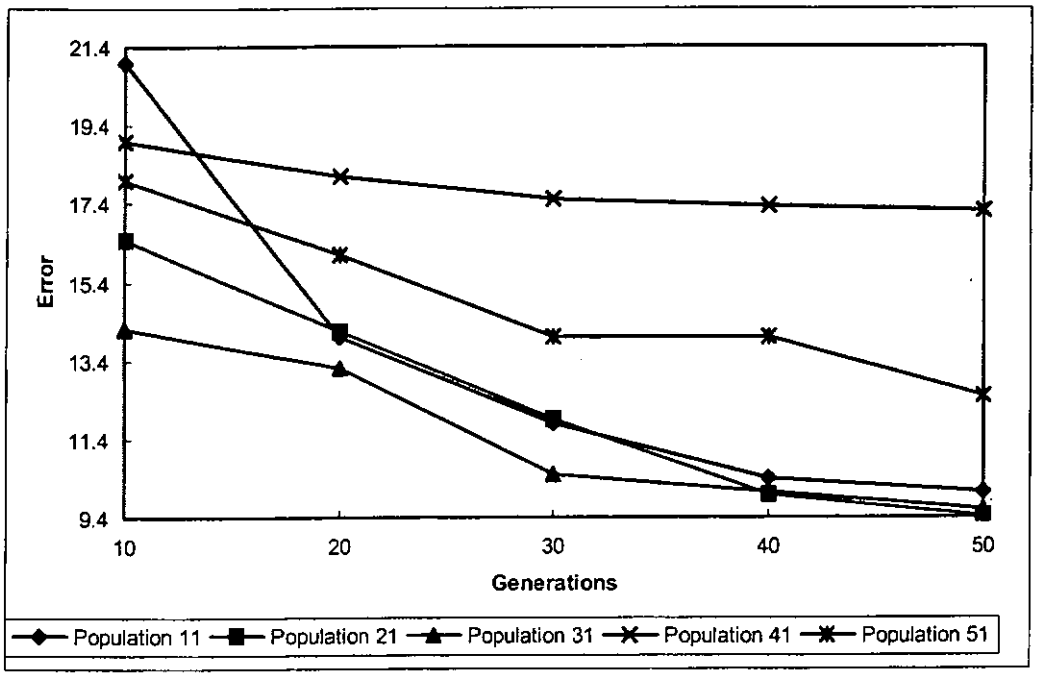


Figure 8.13. Connection option with Tournament Selection, Pc = 0.7, Pm = 0.01

This time the size of population and the probability of crossover are kept constant (pop size =21, Pc=0.7) whilst different probability of mutation are considered.

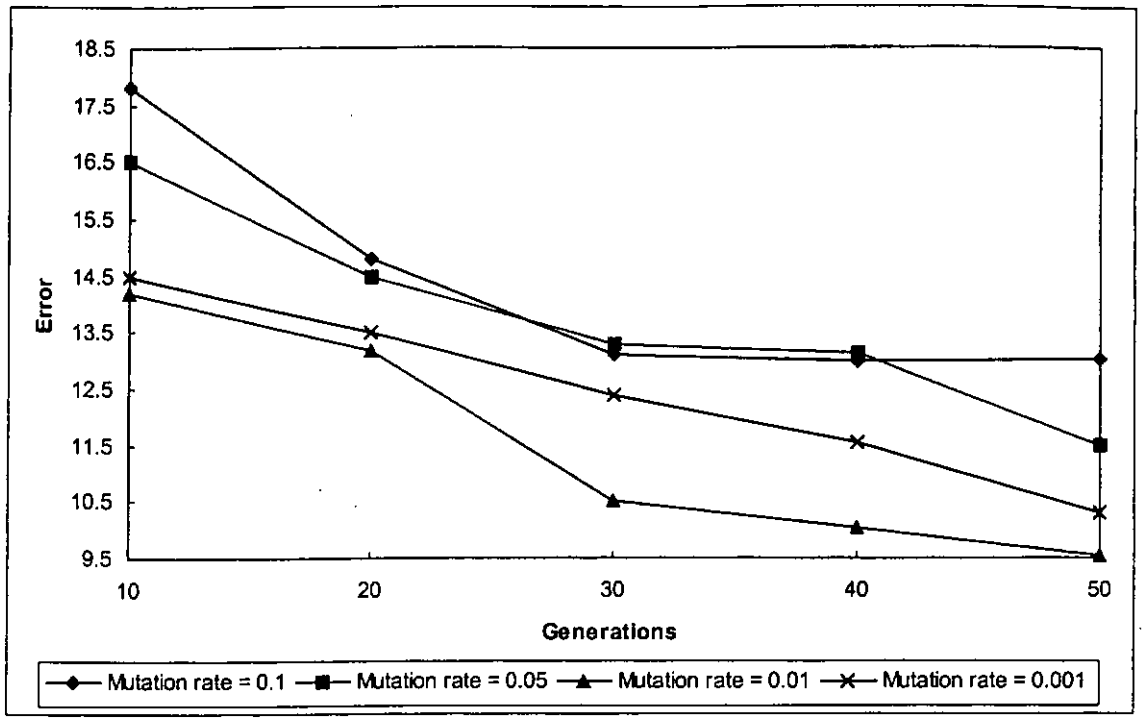


Figure 8.14. Connection option with Tournament Selection, pop size =21, Pc=0.7

In this time the size of population and the probability of mutation are kept constant (pop size =21, Pm=0.01) whilst different probability of crossover are considered.

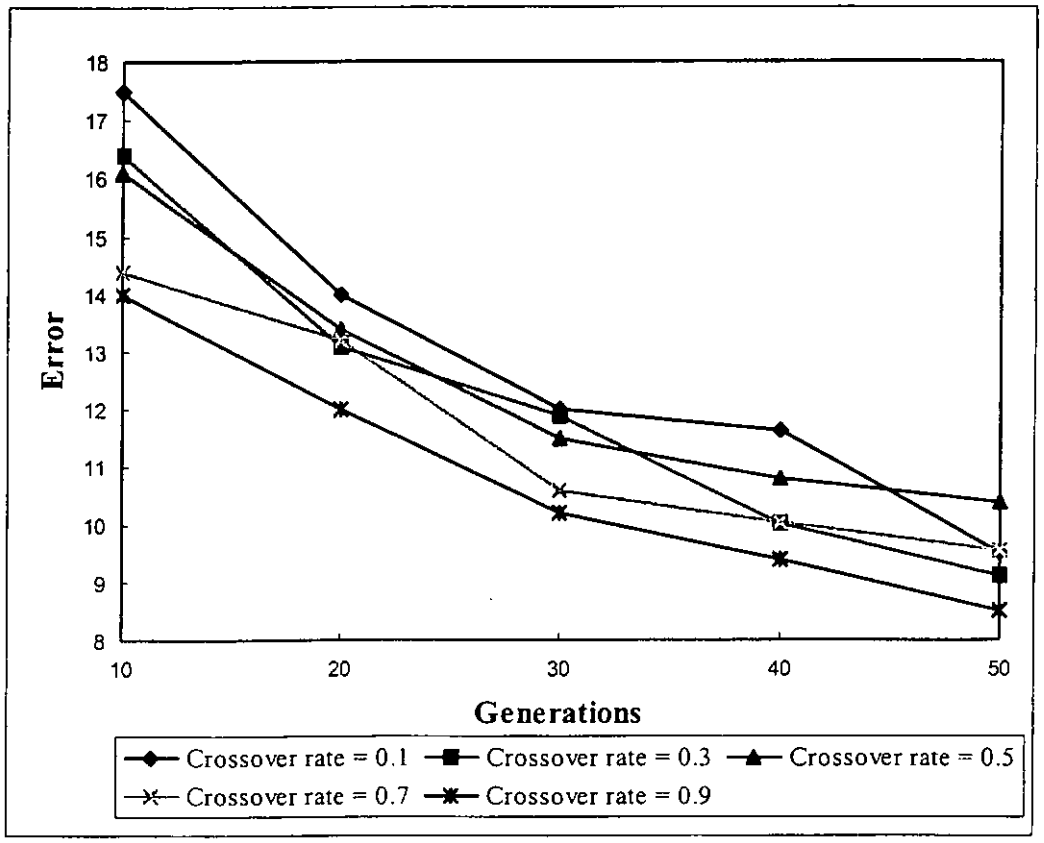


Figure 8.15. Connection option with Tournament Selection, pop size =21, Pm=0.01

8.1.2.3 Ranking Selection: Figure 8.16 to figure 8.18, shows the behavior of an ENN with ranking selection of connection option. The result shown in figure 8.16 is performed by different population sizes, a probability of cross over Pc = 0.7 and probability of mutation Pm = 0.01. It can be known that the population size of 41 is the best result.

A variation for the same approach is presented in figure 8.18.

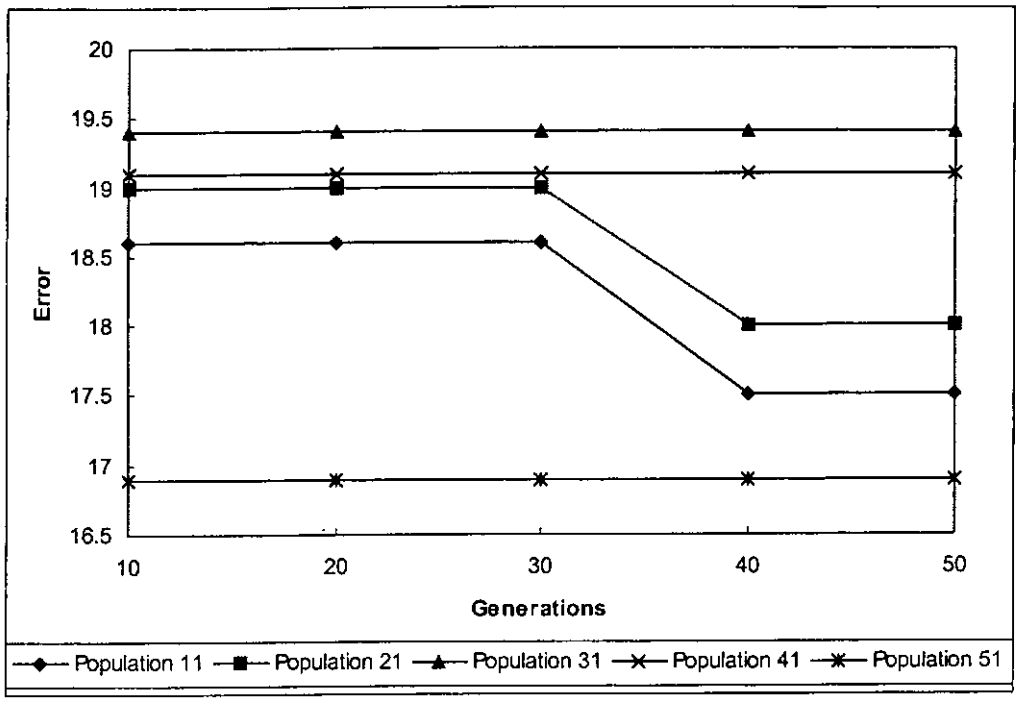


Figure 8.16. Connection option with Ranking Selection, Pc = 0.7, Pm = 0.01

This time the size of population and the probability of crossover are kept constant (pop size =41, Pc=0.7) whilst different probability of mutation are considered.

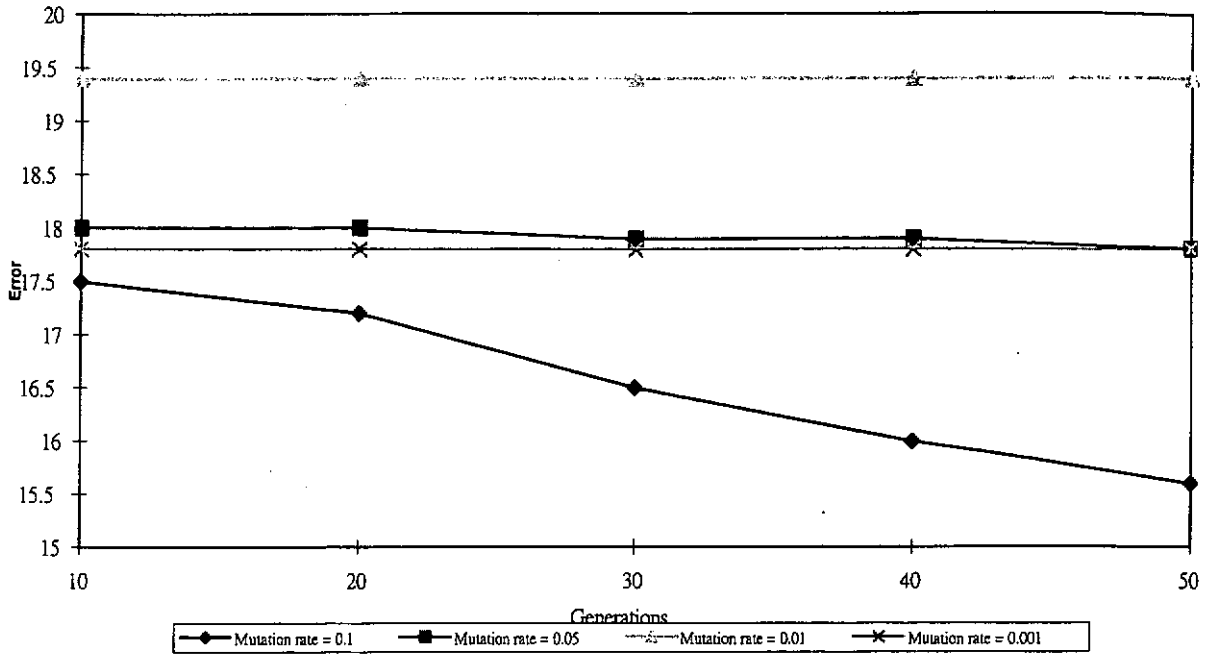


Figure 8.17 Connection option with Ranking Selection, pop size =41, Pc=0.7

In this time the size of population and the probability of mutation are kept constant (pop size =41, Pm=0.1) whilst different probability of crossover are considered.

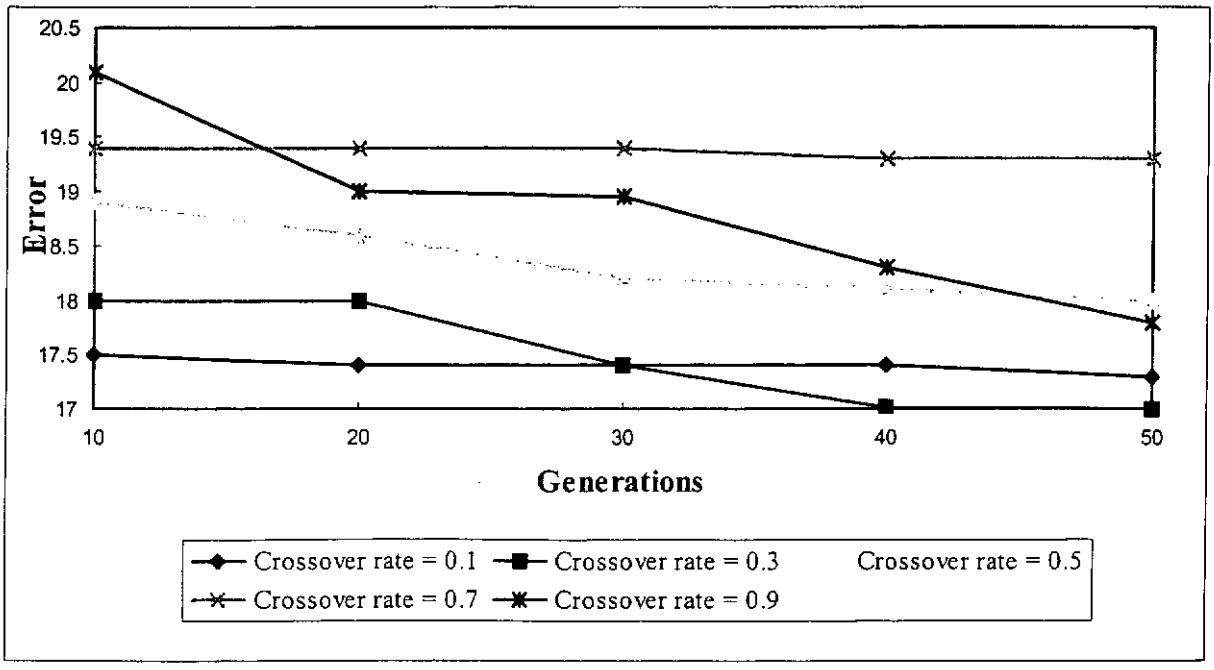


Figure 8.18 Connection option with Ranking Selection, pop size =41, Pm=0.1

8.2 Chapter Summary (Best Parameters)

According to the above results we obtained, we can concluded that the best combination of the probabilities of mutation (M) and crossover (X) as well as the number of population (P) of the three generation selection methods of weight and connection option are tabulated as follows.

	R-wheel (approach 1)	Tournament (approach 2)	Ranking (approach 3)
P	31	21	41
M	0.01	0.01	0.1
X	0.7	0.9	0.3

Table 8.1. Best combination of parameters for connection option

	R-wheel (approach 4)	Tournament (approach 5)	Ranking (approach 6)
P	31	31	41
M	0.01	0.01	0.01
X	0.1	0.5	0.3

Table 8.2 Best combination of parameters for weight option

Chapter 9

Results of test

9.1 Further Work

Based on the best parameters of the six approaches shown in Chapter 8.2. the proposed method is tested on a power system based on a local utility data. Table 9.1 lists the data of the study system. It shows the unit no., minimum generation capacity, maximum generation capacity, and the parameters of the cost curve of each generator. In this project, the application of the proposed method is concentrated on the thermal unit commitment only.

The solution process consists of two steps as follows:

1. The ENN is being trained with the load demand profile (figure 9.1) as the input and the corresponding commitment schedule as the target output.
2. Use DP to post-process those uncertain states.

Unit	Min- Gen (MW)	Max-Gen (MW)	a ($/MW^2 - h)	B ($/MW-h)	c ($/h)
1	200	350	0.0004	8.4	288.4
2	200	350	0.0004	8.4	288.4
3	200	350	0.0004	8.4	288.4
4	200	350	0.0004	8.4	288.4
5	375	680	0.000144	8.43	491.6
6	375	680	0.000144	8.43	491.6
7	375	680	0.000144	8.43	491.6
8	375	680	0.000144	8.43	491.6

Table 9.1 Problem data for the 8-thermal unit system

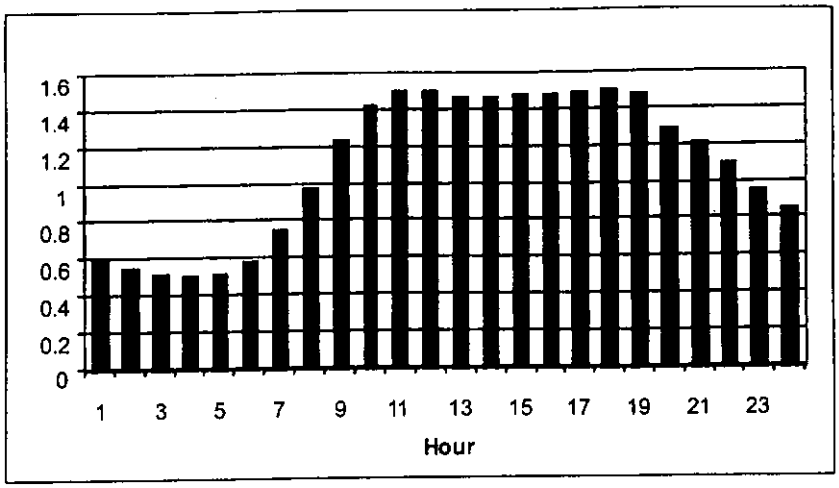


Figure 9.1 Typical Load Profile for the tested system

For the neural network structure, the numbers of input neurons are 24 (24 hours load demand input) and the numbers of output neurons are 196 (24 hours x 8 generators).

For the genetic algorithm structure, three kinds of generation selection method (R-wheel, Tournament and Ranking) for the two evolving option (weight and connection) of the ENN are being used.

In addition, the following data shows the other parameters of the Evolving NN:

Genetic Algorithm

Number of Generation = 100

Neural Network

Input Neurons = 24

Hidden Neurons = 60

Output Neurons = 192

Learning Rate = 0.8

Training Pattern = 30

The comparison tests are performed on an IBM Pentium 150MHz computer. Two case studies are given to illustrate the effectiveness of the ENN-DP. In case 1, the load profile of figure 8.19 was included in the training data. In case 2 the input load demands were randomly deviated at two different hours individually in order to test the generalization of the proposed method.

9.2 Error value according to the best parameter

9.2.1 R-wheel selection for connection option (Approach 1)

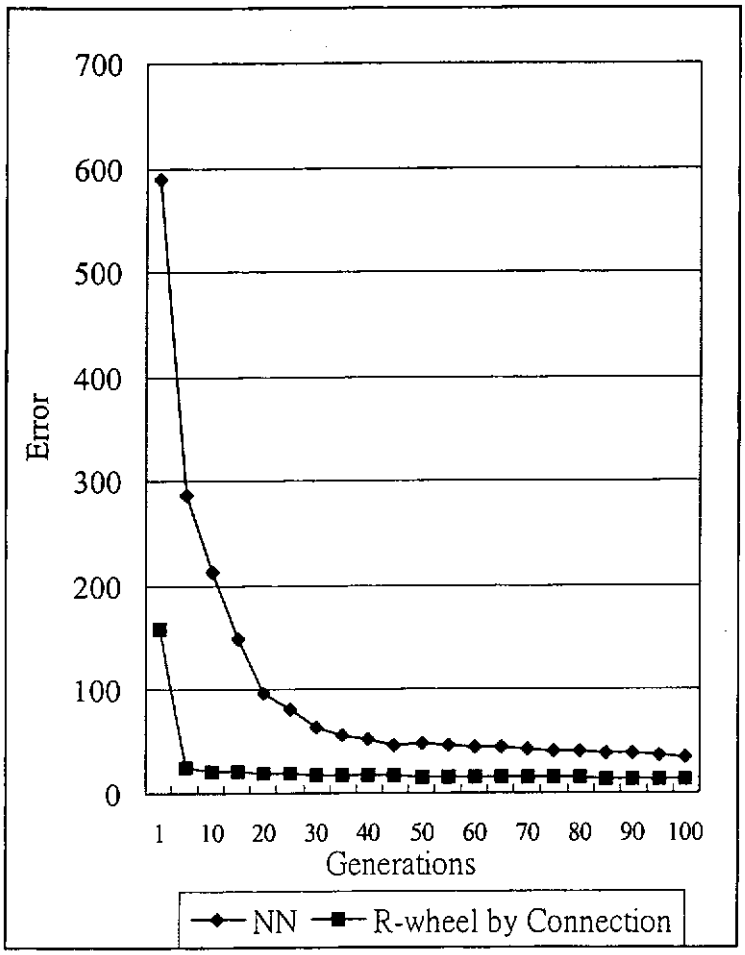


Figure 9.2Comparison of learning Curve between Approach 1 and neural network of error value from 0 to 600

9.2.2 Tournament selection for connection option (Approach 2)

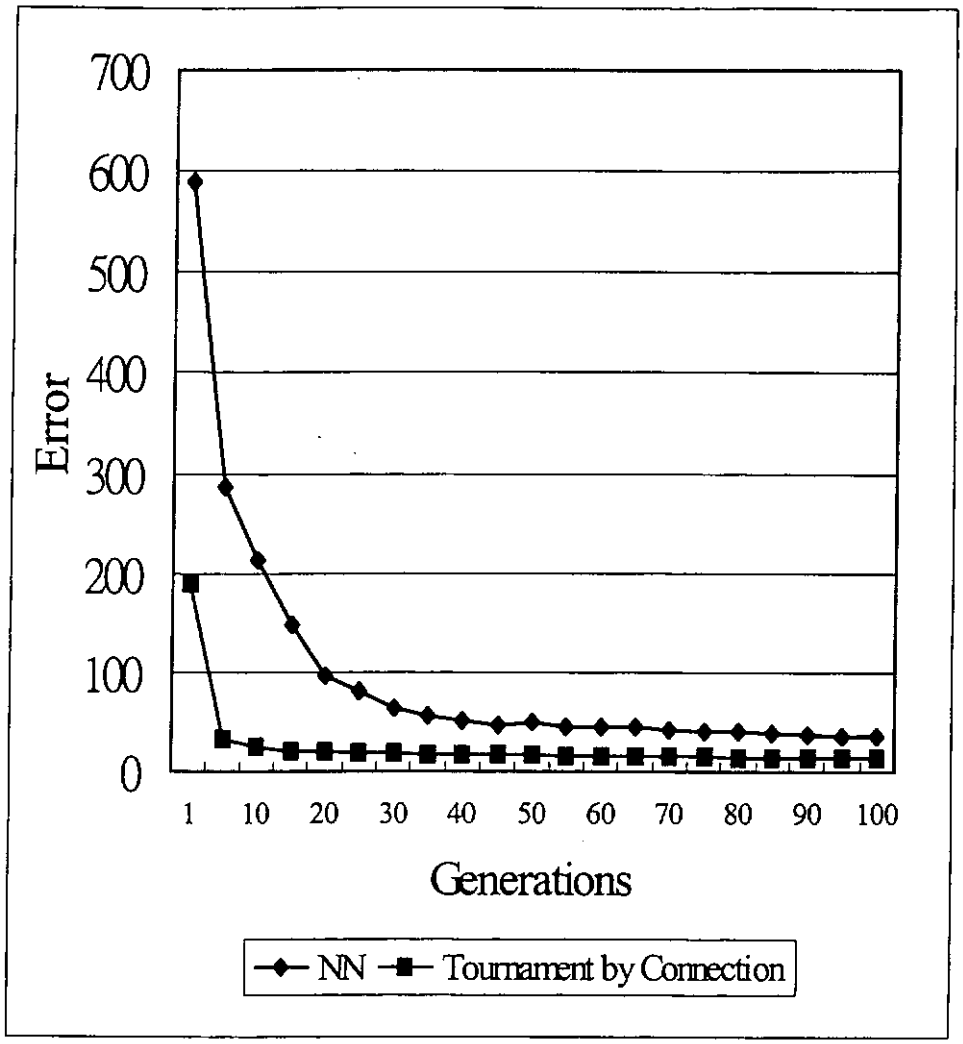


Figure 9.3

9.2.3 Ranking selection for connection option (Approach 3)

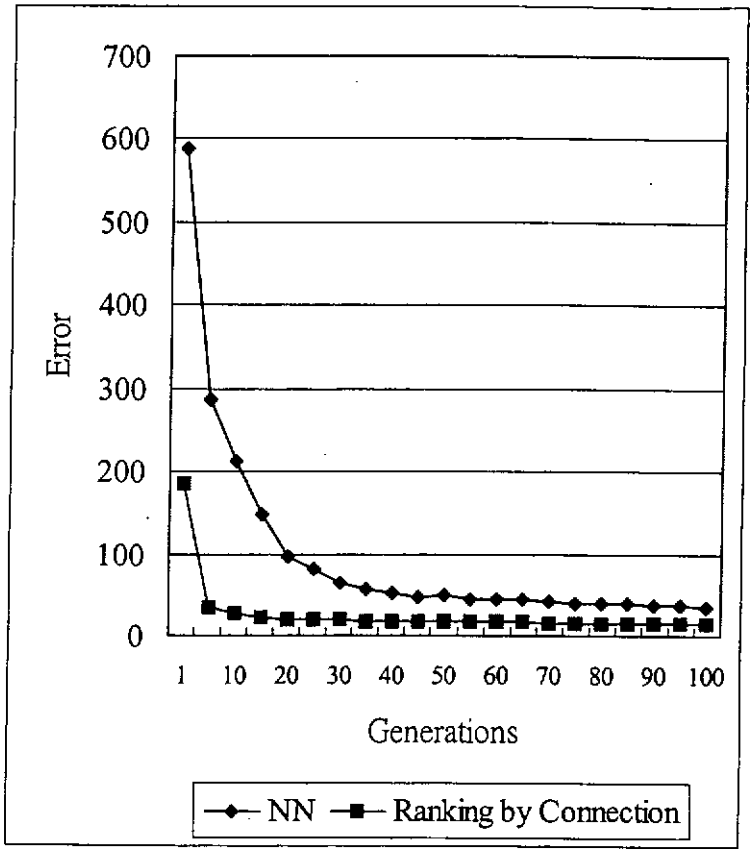


Figure 9.4

9.2.4 R-wheel selection for weight option (Approach 4)

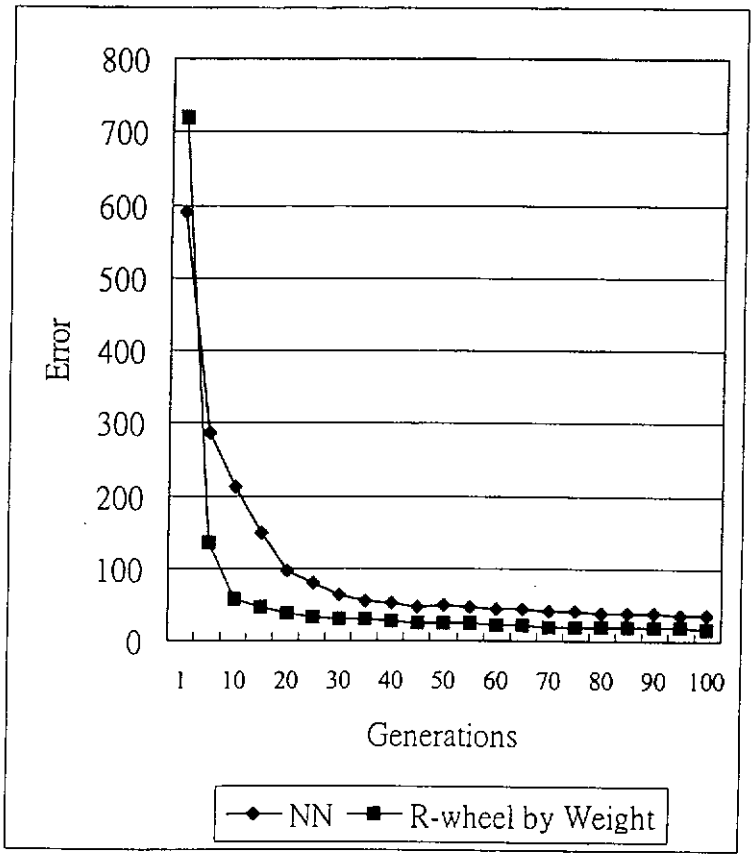


Figure 9.5

9.2.5 Tournament selection for weight option (Approach 5)

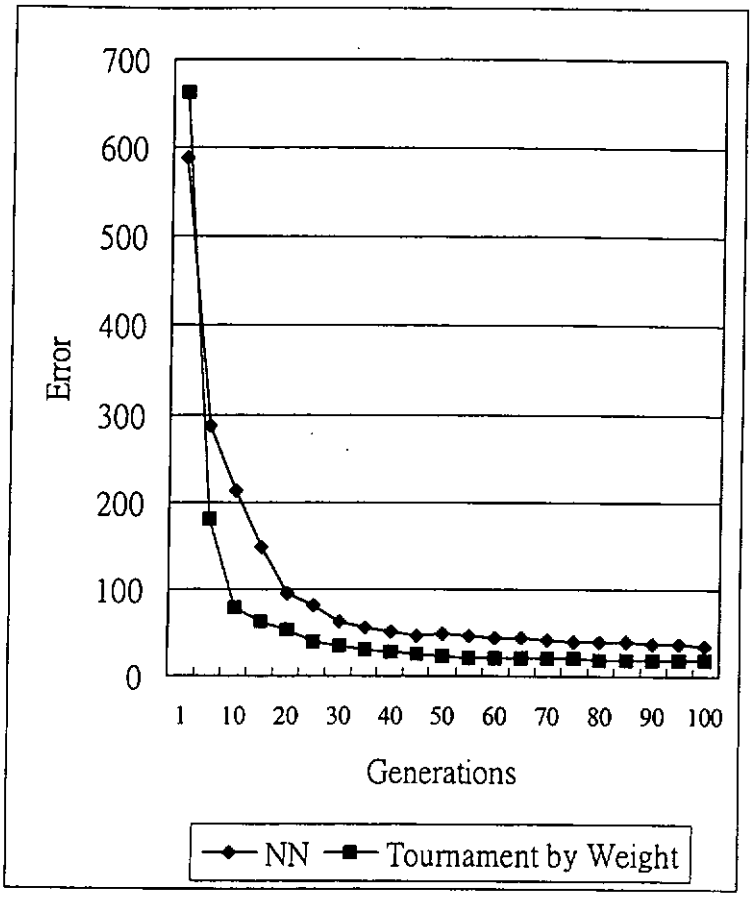


Figure 9.6

9.2.6 Ranking selection for weight option (Approach 6)

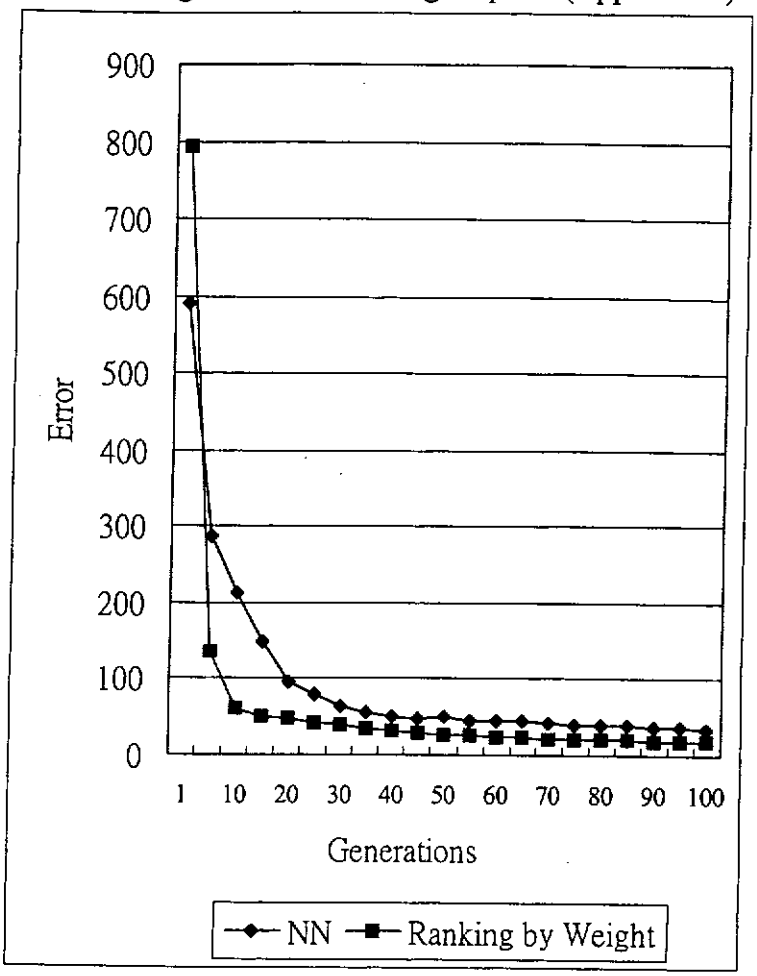


Figure 9.7

Table 4 shows the percentage saving of the execution time and operating cost of various proposed methods when compared with DP.

	Execution Time Saving (%)		Operating Cost Saving (%)	
	Case 1	Case 2	Case 1	Case 2
ENN-DP (Approach 1)	92	83	0.03	-0.91
ENN-DP (Approach 2)	85	82	0.01	-0.92
ENN-DP (Approach 3)	81	82	0.02	-0.95
ENN-DP (Approach 4)	91	78	0.02	-0.92
ENN-DP (Approach 5)	83	79	0.01	-0.94
ENN-DP (Approach 6)	80	76	0	-0.99
NN-DP	80	75	0	-1

Table 4 Percentage execution time saving of various methods when compared with DP

9.3 Chapter Summary

From the results shown in section 9.1 & 9.2, we may concluded that ENN-DP has the following features:

1. Better performance than NN-DP
2. Shorter operating time than NN-DP
3. Connection option & Roulette Wheel selection make the ENN algorithm perform better in solving the UC problem.

Chapter 10

Conclusions

Different approaches of using ENN-DP have been proposed to solve the thermal unit commitment. From the results shown above, ENN-DP has a greater operating time saving than NN-DP. Among the two training options of GA, connection shows a better performance. Among the three generation's selection of GA, Roulette Wheel has the best result.

Generally speaking, ENN-DP has a shorter operating time than ANN-DP and DP due to better initiation of weights or optimized number of connections.

However, the generalization problem of NN still exists on ENN. Since its trained network is only for one system and not readily transferable to another system with different configurations, there should be different training patterns developed for different cases, which may result in longer operating time.

For the trend of deregulation of power generation and privatization of power utilities, an effective and fast UC algorithm is a must for the future use to choose a low cost combination of generators among utility and IPP. The proposed ENN-DP method has promising potential to serve this purpose.

References

[1] H.H Happ, R.C. Johnson and W.J. Wright, large scale hydrothermal unit commitment method and results," IEEE Trans. Power Appar Syst., PAS-90 (1971) pp 1373-1383.

[2] C.K. Pang, G.B. Sheble and F. Albuyeh, "Evaluation of dynamic programming based methods and multiple area representation for thermal unit commitment," IEEE Trans Power Appar Syst PAS-100 (1981) pp1212-1218.

[3] P.P.J. Van den Bosch and G. Honderd, "A solution of the unit commitment problem via decomposition and dynamic programming," IEEE Trans Power Appar Syst, PAS-104(1985) pp1684-1690.

[4] A.I. Cohen and M. Yoshimura, "A branch and bound algorithm for unit commitment," IEEE Trans Power Appar Syst, PAS-102(1983) pp444-451.

[5] S.K. Tong and S.M. Shahidehpour, "Combination of Lagrangian relaxation and linear-programming approaches for fuel constrained unit commitment problems," IEE Proc. C, 136 (1989) pp162-174.

[6] S. Virmani, E.C. Anderian, K. Imhof and S. Mukherjee, "Implementation of a Lagrangian relaxation based unit commitment problem," IEEE Trans Power Syst, 4 (4) (1989) pp1373-1379.

[7] M. Shahidehpour, "Multi stage generation scheduling by neural networks," Proc of AANNM, pp66-70, Clemson, SC, Apr. 1990.

[8] M.H. Sendaula, "Application of artificial neural networks to unit commitment," Proc of ANNPS '91, pp. 256-260, Seattle, WA, Jul. 1991.

[9] P. Ronne-Hansen and J. Ronne-Hansen, "Neural Networks as a tool for unit commitment," Proc of ANNPS '91, pp. 266-270, Seattle, WA, Jul. 1991.

[10] J.H. Holland, "Outline for a logical theory of adaptive systems," Journal of ACM, vol, 3, July 1962, pp. 297-314.

[11] X.Ma,A.A. El-Keib, R.E. Smith, H.Ma, "A Genetic Algorithm based approach to thermal unit commitment of electric power systems," Electric Power Research 34 (1995) pp. 29-36.

[12] Hong-Tzer Yang, Pai-Chuan Yang, Ching-Lien Huang; "Applications of the Genetic Algorithm to the Unit Commitment Problem in Power Generation Industry," IEEE(1995) pp267-274.

[13] S.A Kazarlis, A.G. Bakirtzis, V. Petridis, "A Genetic Algorithm Solution to the Unit Commitment Problem," IEEE Trans on Power Systems, Vol 11, No.1, Feb 1996 pp83-92.

[14] G.B. Sheblé, T.T. Maifeld, K. Brittig, G. Fahd and S Fukurozaki-Coppinger, "Unit Commitment by genetic algorithm with penalty methods and a comparison of Lagrangian search and genetic algorithm-economic dispatch example," Electrical Power & Energy SystemsVol 18, No. 6 (1996) pp. 339-346.

[15] Gerald B. Sheble, George N. Fahd, "Unit Commitment Literature Synopsis," IEEE Transactions on Power Systems, Vol 9, No 1 Feb 1994 pp128-135.

[16] Maurun Candill and Charles Butler, "Understanding Neural Networks : Computer Explorations," The MIT Press 1994

[17] Stephen T. Welstead, "Neural Network and Fuzzy Logic Application in C/C++," John Wiley & Sons, Inc

[18] Laurene Fausert, " Fundamentals of Neural Networks : Architectures, Algorithms and Applications," Prentice Hall International Inc.

[19] M.M. Gupta and D.H. Rao," On the Principles of Fuzzy Neural Networks," Fuzzy Sets and Systems", Vol. 61, p.p.1-18,1994

[20] M.A El-Sharkawi, S.J. Huang, "Application of Genetic-Based Neural Networks to Power System Static Security Assessment"

[21] M.H. Wong, T.S. Chung and Y.K. Wong, "An Evolving Neural Network Approach in Unit Commitment Solution," International Journal Microprocessors and Microsystems. (Paper accepted for publication in June 2000 and to appear soon)

[22] M.H.Wong, Y.K.Wong and T.S. Chung, " A Hybrid Artificial Neural Network-Dynamic Programming Approach to Unit Commitment," Transactions of The Hong Kong Institution of Engineers, vol. 5, no.2, Aug 1998, pp. 49-54.

[23] M.H. Wong, T.S. Chung and Y.K.Wong, " Application of Evolving Neural Network to Unit Commitment," Proceedings of The 1998 International Conference on Energy Management and Power Delivery (Singapore), pp.154-159.

[24] M.H Wong, Y.K. Wong, T.S.Chung, " Parameter Determination of An Evolving Neural Network Approach in Unit Commitment Solution," Proceedings of the 1998 IEEE International Conference on Systems, Man and Cybernetics (USA), SMC'98, pp.1631-1636.

[25] M.H. Wong, T.S. Chung and Y.K. Wong, "An Evolving Neural Network Approach in Unit Commitment Solution: Theory and Performance Evaluation," Proceedings of 34th Universities Power Engineering Conference of Leicester 99 (England).

Appendix 1

Training File

```
%=============================================
%function training
%=============================================
function [nn_in, nn_desired]=training
nn_in = [0.5447 0.5413 0.5393 0.4995;
         0.1920 0.1867 0.1803 0.1699;
         0.6549 0.6221 0.6226 0.6444;
         0.2595 0.2221 0.1801 0.1694;
         0.3044 0.3148 0.3590 0.4112;
         0.6519 0.6209 0.6192 0.5699;
         0.4709 0.5527 0.5777 0.5917;
         0.3951 0.4063 0.4148 0.4112;
         0.5709 0.5238 0.4459 0.3400;
         0.3859 0.4735 0.5097 0.4794;
         0.4243 0.4107 0.4012 0.3961;
         0.4364 0.4126 0.3869 0.3388;
         0.3684 0.3718 0.3976 0.3859;
         0.4218 0.3641 0.2874 0.1842;
         0.2432 0.2512 0.2284 0.3359;
         0.2638 0.2711 0.2735 0.2760;
         0.4481 0.4570 0.4648 0.3951;
         0.5930 0.6189 0.6192 0.6085;
         0.4879 0.4546 0.4308 0.4260;
         0.2784 0.2464 0.2143 0.1505;
         0.3922 0.3633 0.2490 0.2046;
         0.1583 0.1556 0.2323 0.2857;
         0.4364 0.5493 0.5867 0.5762;
         0.2942 0.3408 0.3867 0.3976;
         0.1109 0.1277 0.1369 0.2245;
         0.1711 0.1913 0.2146 0.3058;
         0.5680 0.5677 0.5570 0.5701;
         0.3653 0.3609 0.3643 0.3633;
         0.5473 0.5316 0.5483 0.4825;
         0.4595 0.5721 0.6425 0.6583;
         0.1036 0.1294 0.1342 0.2437;
         0.5908 0.6243 0.6019 0.6320;
         0.2925 0.2044 0.1767 0.1646;
         0.5553 0.5413 0.5396 0.4714;
```

0.1126 0.0903 0.1502 0.2325;
0.4153 0.4063 0.4386 0.4109;
0.3583 0.4636 0.5051 0.5104;
0.2968 0.2592 0.2410 0.1556;
0.4347 0.5170 0.5641 0.5830;
0.1529 0.2163 0.2558 0.2388;
0.4444 0.5636 0.5750 0.5886;
0.3816 0.5160 0.5568 0.5485;
0.5320 0.5410 0.5386 0.5131;
0.2687 0.3648 0.4274 0.3862;
0.2612 0.2367 0.2638 0.2289;
0.3995 0.4051 0.3995 0.3910;
0.1624 0.1726 0.1704 0.2898;
0.2053 0.2189 0.2626 0.2430;
0.0769 0.0968 0.0985 0.0840;
0.1755 0.1966 0.1893 0.2993;
0.5852 0.5803 0.5670 0.5512;
0.2381 0.1978 0.1925 0.3204;
0.2592 0.2886 0.2566 0.2449;
0.5706 0.6017 0.5845 0.6010;
0.3017 0.2383 0.3083 0.2964;
0.1612 0.1859 0.2243 0.2699;
0.2510 0.2726 0.2483 0.2556;
0.5199 0.5158 0.5085 0.4942;
0.3585 0.2648 0.2240 0.1495;
0.4187 0.3612 0.2949 0.1760;
0.5058 0.6252 0.6718 0.6813;
0.3881 0.3971 0.4070 0.3988;
0.5029 0.4590 0.3692 0.2993;
0.3927 0.3306 0.2617 0.1750;
0.5036 0.6476 0.6638 0.6597;
0.4782 0.4733 0.5381 0.4789;
0.4799 0.4172 0.3524 0.2641;
0.5379 0.4966 0.4871 0.5167;
0.4143 0.4022 0.4257 0.4051;
0.4158 0.5257 0.6143 0.6090;
0.4990 0.5066 0.5126 0.5408;
0.4165 0.4129 0.4539 0.4184;
0.6109 0.5978 0.6000 0.5566;
0.4823 0.4257 0.3818 0.3044;
0.6080 0.5869 0.5869 0.5405;
0.6214 0.5905 0.5180 0.4160;
0.3879 0.3561 0.3927 0.4316;
0.1308 0.0760 0.1250 0.1141;
0.2383 0.2590 0.2682 0.2845;
0.1587 0.1583 0.1752 0.2617;
0.2922 0.2291 0.2704 0.2626;
0.5823 0.5944 0.6180 0.5820;
0.4825 0.4126 0.3262 0.2262;
0.0748 0.0774 0.1269 0.1949;

0.1706 0.1847 0.2039 0.1891;
0.1121 0.0801 0.1320 0.1187;
0.4323 0.3726 0.2956 0.1976;
0.6553 0.6420 0.6663 0.6427;
0.4905 0.5920 0.6413 0.6519;
0.4864 0.4172 0.4085 0.2772;
0.5155 0.5172 0.5388 0.5364;
0.2908 0.2879 0.2847 0.2823;
0.1825 0.1646 0.2039 0.1915;
0.6568 0.6629 0.6549 0.6539;
0.1413 0.0874 0.1107 0.1017;
0.2485 0.2388 0.2709 0.2551;
0.1636 0.1813 0.1735 0.2808;
0.0743 0.0743 0.0738 0.1556;
0.3959 0.3337 0.2951 0.1937;
0.1272 0.0777 0.1005 0.0920;
0.4801 0.4546 0.4296 0.4468;
0.6226 0.5595 0.4956 0.3864;
0.3978 0.3852 0.4163 0.3883;
0.5621 0.5049 0.4405 0.3199;
0.6459 0.6726 0.6600 0.6095;
0.4947 0.4447 0.3697 0.2864;
0.1549 0.1536 0.1553 0.1541;
0.0772 0.0796 0.0801 0.1124;
0.4184 0.5075 0.5197 0.5063;
0.4393 0.4374 0.4391 0.3981;
0.3320 0.4126 0.4160 0.4262;
0.1833 0.1539 0.1650 0.1748;
0.0813 0.1010 0.1138 0.0925;
0.2085 0.2466 0.2277 0.2090;
0.5709 0.5927 0.5568 0.5532;
0.3133 0.3325 0.3888 0.4277;
0.3073 0.5182 0.5206 0.5218;
0.2760 0.4126 0.4223 0.3871;
0.1204 0.0930 0.1073 0.0934;
0.5709 0.5345 0.5801 0.5663;
0.4097 0.4100 0.4075 0.3119;
0.0733 0.0728 0.0731 0.0731;
0.4468 0.4393 0.4308 0.4701;
0.3500 0.4626 0.5119 0.5146;
0.1791 0.2049 0.1806 0.1723;
0.0968 0.0925 0.1182 0.1019;
0.3672 0.3609 0.3379 0.3155;
0.5966 0.5813 0.6056 0.6012;
0.5430 0.6789 0.6833 0.6859;
0.4522 0.5519 0.5769 0.5845;
0.6869 0.6716 0.6714 0.6629;
0.3694 0.2549 0.2333 0.1905;
0.2228 0.1825 0.1631 0.1583;
0.6488 0.6432 0.6653 0.6507;

0.6587 0.6752 0.6617 0.6575;
0.4432 0.5791 0.5823 0.5896;
0.2029 0.2238 0.2121 0.2187;
0.2527 0.2774 0.3600 0.3595;
0.2403 0.2629 0.2879 0.2978;
0.5638 0.5760 0.5813 0.5434;
0.1187 0.1396 0.1621 0.2141;
0.3092 0.3097 0.3167 0.3160;
0.2294 0.1818 0.1580 0.2245;
0.1981 0.1553 0.1772 0.2471;
0.3706 0.3449 0.3291 0.2357;
0.2345 0.2735 0.2532 0.2340;
0.1544 0.1650 0.1534 0.2456;
0.2316 0.2476 0.2561 0.3544;
0.4425 0.5636 0.6041 0.6075;
0.6612 0.6650 0.6466 0.6294;
0.2995 0.2422 0.2891 0.2672;
0.5201 0.4468 0.3566 0.2150;
0.2653 0.2427 0.2282 0.1544;
0.4551 0.5112 0.5216 0.5583;
0.5551 0.5379 0.5573 0.5502;
0.4248 0.4051 0.3362 0.2415;
0.3638 0.4575 0.5279 0.5024;
0.2619 0.2558 0.2323 0.2299;
0.1541 0.1532 0.1587 0.1534;
0.5160 0.4964 0.5150 0.4998;
0.4716 0.5002 0.5308 0.4917;
0.6221 0.6100 0.5782 0.6078;
0.5434 0.4871 0.4131 0.3221;
0.4131 0.5189 0.5541 0.5699;
0.5930 0.6061 0.5575 0.5818;
0.2580 0.2816 0.2779 0.2876;
0.0978 0.1163 0.1291 0.2080;
0.1075 0.1381 0.1430 0.2502;
0.1837 0.1711 0.2019 0.1786;
0.4044 0.4000 0.3789 0.3898;
0.5716 0.5524 0.5532 0.5345;
0.0978 0.0998 0.0782 0.1522;
0.2590 0.2437 0.2498 0.2420;
0.4794 0.4357 0.3888 0.3015;
0.4930 0.5063 0.5121 0.5126;
0.4883 0.4163 0.3527 0.2633;
0.2172 0.2114 0.2170 0.3352;
0.6133 0.5859 0.5376 0.4740;
0.0808 0.0808 0.1044 0.0990;
0.6837 0.6823 0.6791 0.6718;
0.6464 0.6286 0.6267 0.5784;
0.2680 0.2998 0.3022 0.3058;
0.2299 0.2308 0.2566 0.2320;
0.3932 0.5012 0.5214 0.5320;

0.1704 0.2282 0.2250 0.2417;
0.1563 0.1602 0.1694 0.1558];

nn_desired = [0.0000 0.0000 0.0000 0.8086 0.0000 0.9691 0.9750 0.9397 0.0000
0.0000 0.0000 0.8200 0.0000 0.9809 0.9735 0.9029 0.0000 0.0000
0.0000 0.8114 0.0000 0.9691 0.9691 0.9118 0.0000 0.0000 0.0000
0.8171 0.0000 0.9265 0.9191 0.7603;
0.0000 0.0000 0.0000 0.0000 0.0000 0.5838 0.5794 0.0000 0.0000
0.8735 0.8706;
0.0000 0.0000 0.0000 0.0000 0.0000 0.7853 0.7868 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.6750 0.6706 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.5471 0.5441 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.5132 0.5132 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.6926 0.6882 0.4632 0.0000
0.0000 0.0000 0.0000 0.0000 0.7206 0.7206 0.4662 0.0000 0.0000
0.0000 0.0000 0.0000 0.8485 0.8338 0.4926 0.0000 0.0000 0.0000
0.0000 0.2088 0.8765 0.8794 0.5265;
0.0000 0.0000 0.8057 0.0000 0.8265 0.9397 0.9426 0.8265 0.0000
0.0000 0.8029 0.0000 0.7309 0.9515 0.9368 0.7294 0.0000 0.0000
0.6286 0.0000 0.8294 0.9485 0.8206 0.8294 0.0000 0.0000 0.0000
0.0000 0.9162 0.7426 0.8794 0.9147;
0.0000 0.0000 0.0000 0.0000 0.7853 0.5750 0.7853 0.7074 0.0000
0.0000 0.0000 0.0000 0.9338 0.6118 0.8956 0.9074 0.0000 0.0000
0.0000 0.0000 0.9441 0.6838 0.9426 0.9294 0.0000 0.0000 0.0000
0.0000 0.9662 0.6941 0.9706 0.9544;
0.0000 0.0000 0.0000 0.0000 0.0000 0.9647 0.9647 0.4647 0.0000
0.0000 0.0000 0.0000 0.0000 0.9529 0.9765 0.5324 0.0000 0.0000
0.0000 0.0000 0.0000 0.9324 0.9779 0.6029 0.0000 0.0000 0.0000
0.0000 0.0000 0.9765 0.9794 0.5353;
0.0000 0.0000 0.0000 0.8057 0.6441 0.8029 0.7971 0.8000 0.0000
0.0000 0.0000 0.8057 0.2485 0.8397 0.8353 0.8353 0.0000 0.0000
0.0000 0.3857 0.0000 0.8426 0.8309 0.8294 0.0000 0.0000 0.0000
0.0000 0.0000 0.7618 0.7574 0.5412;
0.0000 0.0000 0.0000 0.0000 0.8235 0.6853 0.8294 0.0000 0.0000
0.0000 0.0000 0.0000 0.8044 0.8088 0.8074 0.4485 0.0000 0.0000
0.0000 0.0000 0.7750 0.7824 0.7779 0.7529 0.0000 0.0000 0.0000
0.0000 0.7515 0.7191 0.7103 0.7235;
0.0000 0.0000 0.0000 0.0000 0.0000 0.9897 0.9706 0.6103 0.0000
0.0000 0.0000 0.0000 0.0000 0.9868 0.9706 0.5309 0.0000 0.0000
0.0000 0.0000 0.0000 0.9868 0.9721 0.4721 0.0000 0.0000 0.0000
0.0000 0.0000 0.9691 0.9647 0.4662;
0.0000 0.0000 0.0000 0.0000 0.8882 0.8926 0.8632 0.0000 0.0000
0.0000 0.0000 0.0000 0.8309 0.8382 0.8309 0.0000 0.0000 0.0000
0.0000 0.0000 0.7809 0.7824 0.7809 0.0000 0.0000 0.0000 0.0000
0.0000 0.6588 0.7397 0.6544 0.0000;
0.0000 0.0000 0.0000 0.0000 0.7426 0.7485 0.0000 0.7412 0.0000
0.0000 0.0000 0.0000 0.7485 0.7544 0.0000 0.7500 0.0000 0.0000
0.0000 0.0000 0.7941 0.8088 0.0000 0.8059 0.0000 0.0000 0.0000
0.0000 0.5368 0.9015 0.0000 0.9000;
0.0000 0.0000 0.0000 0.0000 0.6809 0.9397 0.9353 0.0000 0.0000
0.0000 0.0000 0.0000 0.4618 0.8735 0.8706 0.0000 0.0000 0.0000
0.0000 0.0000 0.3176 0.7132 0.7103 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.5574 0.5588 0.0000;
0.0000 0.0000 0.0000 0.0000 0.6765 0.7971 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.7529 0.7691 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.5059 0.8779 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.6147 0.8471 0.5735 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.9426 0.6559 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.9632 0.6794 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.9632 0.6941 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.9632 0.7088;
0.0000 0.0000 0.0000 0.0000 0.9059 0.0000 0.9074 0.9015 0.0000
0.0000 0.0000 0.0000 0.9221 0.0000 0.9265 0.9206 0.0000 0.0000
0.0000 0.0000 0.9368 0.0000 0.9397 0.9397 0.0000 0.0000 0.0000
0.0000 0.7235 0.0000 0.8353 0.8353;
0.0000 0.0000 0.0000 0.0000 0.8132 0.9912 0.9735 0.8147 0.0000
0.0000 0.0000 0.0000 0.8897 0.9956 0.9750 0.8897 0.0000 0.0000
0.0000 0.0000 0.8926 0.9971 0.9765 0.8853 0.0000 0.0000 0.0000
0.0000 0.8574 0.9985 0.9750 0.8559;
0.0000 0.0000 0.0000 0.6257 0.0000 0.9559 0.9412 0.7368 0.0000
0.0000 0.0000 0.4714 0.0000 0.9206 0.9250 0.6662 0.0000 0.0000

```
        0.0000    0.0000    0.0000    0.9544    0.9515    0.7044    0.0000    0.0000
   0.0000    0.0000    0.0000    0.9515    0.9456    0.6838;
   0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.9544    0.7324    0.0000
   0.0000    0.0000    0.0000    0.0000    0.0000    0.9206    0.5721    0.0000    0.0000
   0.0000    0.0000    0.0000    0.0000    0.8191    0.4794    0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000    0.4779    0.4338;
   0.0000    0.0000    0.0000    0.0000    0.0000    0.8294    0.7324    0.8147    0.0000
   0.0000    0.0000    0.0000    0.0000    0.8750    0.7500    0.5765    0.0000    0.0000
   0.0000    0.0000    0.0000    0.7868    0.7221    0.0000    0.0000    0.0000    0.0000
   0.0000    0.0000    0.4632    0.7765    0.0000;
   0.0000    0.0000    0.0000    0.0000    0.0000    0.4794    0.4794    0.0000    0.0000
   0.0000    0.0000    0.0000    0.0000    0.4735    0.4691    0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000    0.7088    0.6985    0.0000    0.0000    0.0000    0.0000
   0.0000    0.2191    0.7603    0.7515    0.0000;
   0.0000    0.0000    0.0000    0.0000    0.6147    0.9015    0.8941    0.2338    0.0000
   0.0000    0.0000    0.0000    0.7029    0.9691    0.9500    0.7059    0.0000    0.0000
   0.0000    0.0000    0.7985    0.9882    0.9750    0.7926    0.0000    0.0000    0.0000
   0.0000    0.7588    0.9941    0.9794    0.7588;
   0.0000    0.0000    0.0000    0.0000    0.0000    0.7294    0.7309    0.3221    0.0000
   0.0000    0.0000    0.0000    0.0000    0.8015    0.7956    0.4676    0.0000    0.0000
   0.0000    0.0000    0.0000    0.9382    0.9397    0.4647    0.0000    0.0000    0.0000
   0.0000    0.0000    0.9676    0.9544    0.4868;
   0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.6721    0.0000    0.0000
   0.0000    0.0000    0.0000    0.0000    0.0000    0.7735    0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000    0.3618    0.4676    0.0000    0.0000    0.0000    0.0000
   0.0000    0.0000    0.5868    0.5838    0.1897;
   0.0000    0.0000    0.0000    0.0000    0.0000    0.5176    0.5191    0.0000    0.0000
   0.0000    0.0000    0.0000    0.0000    0.5809    0.5779    0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000    0.6515    0.6485    0.0000    0.0000    0.0000    0.0000
   0.0000    0.1500    0.8529    0.8500    0.0000;
   0.0000    0.0000    0.0000    0.0000    0.9015    0.7368    0.9029    0.9000    0.0000
   0.0000    0.0000    0.0000    0.9015    0.7324    0.9059    0.9000    0.0000    0.0000
   0.0000    0.0000    0.8897    0.7059    0.8897    0.8897    0.0000    0.0000    0.0000
   0.0000    0.9074    0.7324    0.9118    0.9029;
   0.0000    0.0000    0.0000    0.0000    0.7338    0.7412    0.0000    0.7382    0.0000
   0.0000    0.0000    0.0000    0.7206    0.7353    0.0000    0.7309    0.0000    0.0000
   0.0000    0.0000    0.7353    0.7397    0.0000    0.7324    0.0000    0.0000    0.0000
   0.0000    0.7338    0.7338    0.0000    0.7338;
   0.0000    0.0000    0.0000    0.0000    0.7206    0.9412    0.9368    0.7176    0.0000
   0.0000    0.0000    0.0000    0.6750    0.9338    0.9368    0.6750    0.0000    0.0000
   0.0000    0.0000    0.7206    0.9412    0.9397    0.7206    0.0000    0.0000    0.0000
   0.0000    0.3456    0.8412    0.9191    0.8176;
   0.0000    0.0000    0.0000    0.0000    0.6706    0.8809    0.9529    0.2794    0.0000
   0.0000    0.0000    0.0000    0.8368    0.9544    0.9559    0.7191    0.0000    0.0000
   0.6314    0.0000    0.8441    0.9500    0.9500    0.8235    0.0000    0.0000    0.8057
   0.0000    0.8353    0.9559    0.9471    0.8353;
   0.0000    0.0000    0.0000    0.0000    0.0000    0.6279    0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000    0.0000    0.7838    0.0000    0.0000    0.0000    0.0000
   0.0000    0.1371    0.0000    0.5103    0.2324    0.0000    0.0000    0.0000    0.0000
   0.5543    0.0000    0.7015    0.4897    0.0000;
   0.0000    0.0000    0.0000    0.0000    0.7941    0.9912    0.9809    0.8132    0.0000
   0.0000    0.0000    0.0000    0.9397    0.9882    0.9838    0.8706    0.0000    0.0000
   0.0000    0.0000    0.8471    0.9897    0.9824    0.8279    0.0000    0.0000    0.0000
   0.0000    0.9544    0.9868    0.9824    0.9059;
   0.0000    0.0000    0.0000    0.0000    0.7706    0.0000    0.7809    0.2206    0.0000
   0.0000    0.0000    0.0000    0.6176    0.0000    0.6206    0.0000    0.0000    0.0000
   0.0000    0.0000    0.5353    0.0000    0.5353    0.0000    0.0000    0.0000    0.0000
   0.0000    0.4985    0.0000    0.4985    0.0000;
   0.0000    0.0000    0.0000    0.9686    0.0000    0.9912    0.9794    0.8956    0.0000
   0.0000    0.0000    0.8486    0.0000    0.9897    0.9809    0.8721    0.0000    0.0000
   0.0000    0.8971    0.0000    0.9309    0.9824    0.8941    0.0000    0.0000    0.0000
   0.5829    0.0000    0.9353    0.9426    0.6779;
   0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.6824    0.0000    0.0000
   0.0000    0.0000    0.0000    0.0000    0.0000    0.5471    0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000    0.4441    0.4662    0.0000    0.0000    0.0000    0.0000
   0.0000    0.1279    0.6059    0.6750    0.0000;
   0.0000    0.0000    0.0000    0.0000    0.0000    0.9279    0.9265    0.6618    0.0000
   0.0000    0.0000    0.0000    0.0000    0.9118    0.9074    0.6426    0.0000    0.0000
   0.0000    0.0000    0.0000    0.9559    0.9515    0.7500    0.0000    0.0000    0.0000
   0.0000    0.0000    0.9441    0.8426    0.7029;
   0.0000    0.0000    0.0000    0.0000    0.4868    0.8250    0.8588    0.0000    0.0000
   0.0000    0.0000    0.0000    0.4750    0.8956    0.9456    0.4926    0.0000    0.0000
   0.0000    0.0000    0.5559    0.9324    0.9676    0.6044    0.0000    0.0000    0.0000
   0.0000    0.5662    0.9559    0.9662    0.6044;
```

0.0000 0.0000 0.0000 0.0000 0.0000 0.9015 0.8971 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.8221 0.7485 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.9632 0.4971 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.7074 0.2353 0.0000;
0.0000 0.0000 0.0000 0.0000 0.5662 0.9176 0.9132 0.2368 0.0000
0.0000 0.0000 0.0000 0.5912 0.9750 0.9706 0.5956 0.0000 0.0000
0.0000 0.0000 0.7294 0.9750 0.9765 0.7368 0.0000 0.0000 0.0000
0.0000 0.7838 0.9853 0.9765 0.7868;
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.5721 0.3544 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.7574 0.5529 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.9235 0.6265 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.8544 0.5926;
0.0000 0.0000 0.0000 0.0000 0.2044 0.8956 0.9441 0.6485 0.0000
0.0000 0.0000 0.0000 0.6691 0.9824 0.9676 0.7956 0.0000 0.0000
0.0000 0.0000 0.7647 0.9853 0.9706 0.7632 0.0000 0.0000 0.0000
0.0000 0.8750 0.9162 0.9132 0.8618;
0.0000 0.0000 0.0000 0.0000 0.0000 0.8191 0.8147 0.6779 0.0000
0.0000 0.0000 0.6771 0.0000 0.9309 0.9250 0.9221 0.0000 0.0000
0.0000 0.8800 0.0000 0.9574 0.9824 0.9809 0.0000 0.0000 0.0000
0.8171 0.0000 0.9397 0.9824 0.9809;
0.0000 0.8029 0.0000 0.0000 0.9324 0.9397 0.9382 0.0000 0.0000
0.8229 0.0000 0.0000 0.9971 0.9294 0.9279 0.0000 0.0000 0.9171
0.0000 0.0000 1.0000 0.8956 0.8956 0.0000 0.0000 0.8000 0.0000
0.0000 0.9088 0.8956 0.8926 0.0000;
0.0000 0.0000 0.0000 0.0000 0.4882 0.2603 0.8794 0.0000 0.0000
0.0000 0.0000 0.0000 0.5676 0.7382 0.9044 0.0000 0.0000 0.0000
0.0000 0.0000 0.7721 0.9088 0.9088 0.0000 0.0000 0.0000 0.0000
0.0000 0.5265 0.9074 0.9059 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.7941 0.0000 0.7882 0.0000
0.0000 0.0000 0.0000 0.0000 0.7191 0.0000 0.7147 0.0000 0.0000
0.0000 0.0000 0.0000 0.8029 0.0000 0.7956 0.0000 0.0000 0.0000
0.0000 0.0000 0.6941 0.0000 0.6926;
0.0000 0.0000 0.0000 0.0000 0.0000 0.9750 0.9691 0.4765 0.0000
0.0000 0.0000 0.0000 0.0000 0.9706 0.9779 0.5059 0.0000 0.0000
0.0000 0.0000 0.0000 0.9765 0.9735 0.4706 0.0000 0.0000 0.0000
0.0000 0.0000 0.9559 0.9500 0.4632;
0.0000 0.0000 0.0000 0.0000 0.0000 0.4912 0.4926 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.5221 0.5235 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.5191 0.5132 0.0000 0.0000 0.0000 0.0000
0.0000 0.3353 0.7132 0.7074 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.6250 0.6191 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.6662 0.6603 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.8015 0.7897 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.7368 0.7353 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.4662 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.5868 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.5971 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.5088 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.5324 0.5309 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.5956 0.5956 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.5779 0.5691 0.0000 0.0000 0.0000 0.0000
0.0000 0.2985 0.7588 0.7559 0.0000;
0.0000 0.0000 0.0000 0.0000 0.7926 0.9853 0.9735 0.7941 0.0000
0.0000 0.0000 0.0000 0.7750 0.9912 0.9750 0.7750 0.0000 0.0000
0.0000 0.0000 0.6912 0.9926 0.9735 0.7779 0.0000 0.0000 0.0000
0.0000 0.6706 0.9750 0.9735 0.7206;
0.0000 0.0000 0.0000 0.0000 0.7191 0.0000 0.7235 0.0000 0.0000
0.0000 0.0000 0.0000 0.5971 0.0000 0.6015 0.0000 0.0000 0.0000
0.0000 0.0000 0.5824 0.0000 0.5838 0.0000 0.0000 0.0000 0.0000
0.0000 0.7559 0.2265 0.7603 0.1985;
0.0000 0.0000 0.0000 0.0000 0.0000 0.7721 0.7985 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.8779 0.8706 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.7794 0.7750 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.7426 0.7412 0.0000;
0.0000 0.0000 0.0000 0.0000 0.7456 0.9882 0.9794 0.7441 0.0000
0.0000 0.0000 0.0000 0.8368 0.9956 0.9779 0.8353 0.0000 0.0000
0.0000 0.0000 0.7956 0.9735 0.9765 0.7956 0.0000 0.0000 0.0000
0.0000 0.9279 0.9176 0.8912 0.9044;
0.0000 0.0000 0.0000 0.0000 0.9235 0.0000 0.0000 0.9044 0.0000
0.0000 0.0000 0.0000 0.7279 0.0000 0.0000 0.7162 0.1143 0.000
0.0000 0.0000 0.8529 0.0000 0.1338 0.8221 0.3229 0.0000 0.0000
0.0000 0.5824 0.0000 0.4691 0.5779;
0.0000 0.0000 0.0000 0.0000 0.0000 0.4882 0.4882 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.5662 0.5603 0.0000 0.0000 0.0000

```
        0.0000    0.0000    0.0000    0.6838    0.6750    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.8250    0.8103    0.0000;
  0.0000    0.0000    0.0000    0.0000    0.7588    0.0000    0.0000    0.7618    0.0000
  0.0000    0.0000    0.0000    0.8250    0.0000    0.0000    0.8265    0.0000    0.0000
  0.0000    0.0000    0.7500    0.0000    0.0000    0.7544    0.0000    0.0000    0.0000
  0.0000    0.7706    0.0000    0.0000    0.7779;
  0.6571    0.0000    0.0000    0.0000    0.9235    0.0000    0.9735    0.9147    0.5771
  0.0000    0.0000    0.0000    0.9235    0.0000    0.9809    0.9235    0.5743    0.0000
  0.0000    0.0000    0.8956    0.0000    0.9971    0.8926    0.5771    0.0000    0.0000
  0.0000    0.8691    0.0000    0.9588    0.8691;
  0.0000    0.0000    0.0000    0.0000    0.3324    0.9162    0.9235    0.0000    0.0000
  0.0000    0.0000    0.0000    0.0000    0.8015    0.8029    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.5574    0.8000    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.9059    0.0000;
  0.0000    0.0000    0.0000    0.0000    0.0000    0.9338    0.9353    0.6676    0.0000
  0.0000    0.0000    0.0000    0.0000    0.8206    0.8176    0.5500    0.0000    0.0000
  0.0000    0.0000    0.0000    0.7544    0.7735    0.2588    0.0000    0.0000    0.0000
  0.0000    0.0000    0.4750    0.5912    0.0000;
  0.0000    0.0000    0.0000    0.0000    0.7588    0.9941    0.9691    0.3426    0.0000
  0.0000    0.7429    0.0000    0.7191    0.9853    0.9706    0.7309    0.0000    0.0000
  0.8257    0.0000    0.8485    0.9779    0.9574    0.8618    0.0000    0.0000    0.8171
  0.0000    0.8603    0.9956    0.9706    0.8809;
  0.0000    0.0000    0.0000    0.0000    0.7824    0.7868    0.0000    0.7824    0.0000
  0.0000    0.0000    0.0000    0.7235    0.9779    0.0000    0.7044    0.0000    0.0000
  0.0000    0.0000    0.9515    0.7691    0.0000    0.7456    0.0000    0.0000    0.0000
  0.0000    0.8603    0.7882    0.0000    0.7676;
  0.0000    0.0000    0.0000    0.0000    0.7794    0.6559    0.8059    0.8059    0.0000
  0.0000    0.0000    0.0000    0.3338    0.6985    0.8691    0.8794    0.0000    0.0000
  0.0000    0.0000    0.0000    0.9721    0.7309    0.5338    0.0000    0.0000    0.0000
  0.0000    0.0000    0.8647    0.6882    0.2603;
  0.0000    0.0000    0.0000    0.0000    0.0000    0.9529    0.9603    0.4662    0.0000
  0.0000    0.0000    0.0000    0.0000    0.7706    0.7691    0.4632    0.0000    0.0000
  0.0000    0.0000    0.0000    0.5735    0.6882    0.3235    0.0000    0.0000    0.0000
  0.0000    0.0000    0.5235    0.5368    0.0000;
  0.0000    0.0000    0.0000    0.0000    0.8324    0.8824    0.5000    0.8368    0.0000
  0.0000    0.0000    0.5200    0.9294    0.9368    0.8309    0.9588    0.0000    0.0000
  0.0000    0.7914    0.8926    0.8971    0.9118    0.9132    0.0000    0.0000    0.0000
  0.8057    0.8765    0.9044    0.9029    0.8985;
  0.0000    0.0000    0.0000    0.0000    0.5294    0.8824    0.9662    0.5191    0.0000
  0.0000    0.0000    0.0000    0.4676    0.9721    0.9544    0.4735    0.0000    0.0000
  0.0000    0.0000    0.6191    0.9794    0.9706    0.6912    0.0000    0.0000    0.0000
  0.0000    0.2559    0.9500    0.9721    0.7235;
  0.0000    0.0000    0.0000    0.0000    0.6456    0.6868    0.9721    0.6029    0.0000
  0.0000    0.0000    0.0000    0.6176    0.3191    0.9735    0.6176    0.0000    0.0000
  0.0000    0.0000    0.5985    0.0000    0.9397    0.5971    0.0000    0.0000    0.0000
  0.0000    0.4779    0.0000    0.8632    0.2588;
  0.8343    0.0000    0.0000    0.0000    0.9500    0.0000    0.9676    0.9118    0.7457
  0.0000    0.0000    0.0000    0.8412    0.0000    0.9500    0.8338    0.5943    0.0000
  0.0000    0.0000    0.8559    0.0000    0.9485    0.8412    0.7029    0.0000    0.0000
  0.0000    0.9074    0.0000    0.9838    0.8779;
  0.0000    0.0000    0.0000    0.0000    0.6956    0.9088    0.9059    0.0000    0.0000
  0.0000    0.0000    0.0000    0.6206    0.9088    0.9074    0.0000    0.0000    0.0000
  0.0000    0.0000    0.7779    0.9029    0.8985    0.0000    0.0000    0.0000    0.0000
  0.0000    0.6368    0.9103    0.9074    0.0000;
  0.0000    0.0000    0.0000    0.0000    0.5794    0.8853    0.8985    0.1559    0.0000
  0.0000    0.0000    0.0000    0.6838    0.9382    0.9324    0.6309    0.0000    0.0000
  0.0000    0.0000    0.9132    0.9750    0.9676    0.8662    0.0000    0.0000    0.0000
  0.0000    0.8706    0.9529    0.9794    0.8868;
  0.0000    0.0000    0.0000    0.8171    0.0000    0.9574    0.9529    0.6926    0.0000
  0.0000    0.0000    0.8743    0.0000    0.7750    0.9779    0.8662    0.0000    0.0000
  0.0000    0.8400    0.0000    0.8926    0.9588    0.8221    0.0000    0.0000    0.0000
  0.9314    0.0000    0.9956    0.9809    0.8206;
  0.0000    0.0000    0.0000    0.0000    0.7809    0.8368    0.9059    0.0000    0.0000
  0.0000    0.0000    0.0000    0.8324    0.8412    0.8279    0.0000    0.0000    0.0000
  0.0000    0.0000    0.9162    0.9206    0.9132    0.0000    0.0000    0.0000    0.0000
  0.0000    0.8426    0.8485    0.8441    0.0000;
  0.0000    0.0000    0.0000    0.0000    0.8632    0.9912    0.9853    0.8618    0.0000
  0.0000    0.0000    0.0000    0.8221    0.9897    0.9882    0.8221    0.0000    0.0000
  0.0000    0.0000    0.8309    0.9853    0.9912    0.8279    0.0000    0.0000    0.0000
  0.0000    0.7132    0.9706    0.9721    0.7162;
  0.5771    0.0000    0.0000    0.0000    0.8206    0.0000    0.9853    0.8191    0.5743
  0.0000    0.0000    0.0000    0.6471    0.0000    0.9838    0.6529    0.5800    0.0000
  0.0000    0.0000    0.3397    0.0000    0.9838    0.6912    0.4686    0.0000    0.0000
  0.0000    0.0000    0.0000    0.9853    0.6176;
```

```
0.0000  0.0000  0.0000  0.0000  0.8515  1.0015  0.9779  0.8529  0.0000
0.0000  0.0000  0.0000  0.7926  0.9926  0.9779  0.7926  0.0000  0.0000
0.0000  0.0000  0.7912  0.9956  0.9794  0.7897  0.0000  0.0000  0.0000
0.0000  0.6485  0.9956  0.9794  0.6515;
0.0000  0.7971  0.0000  0.0000  0.8471  0.8294  0.8559  0.8221  0.0000
0.0000  0.0000  0.0000  0.9206  0.8588  0.8868  0.9118  0.0000  0.0000
0.0000  0.0000  0.8456  0.8529  0.8471  0.5926  0.0000  0.0000  0.0000
0.0000  0.7897  0.8676  0.8632  0.0000;
0.0000  0.0000  0.0000  0.0000  0.6279  0.7206  0.6294  0.3721  0.0000
0.0000  0.0000  0.0000  0.7147  0.7265  0.7162  0.0000  0.0000  0.0000
0.0000  0.0000  0.7912  0.7985  0.7897  0.0000  0.0000  0.0000  0.0000
0.0000  0.8824  0.8779  0.8544  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.3338  0.4588  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.0000  0.4603  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.7574  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.6912  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.7235  0.7206  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.7853  0.7838  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.8132  0.8118  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.8632  0.8603  0.0000;
0.0000  0.0000  0.0000  0.0000  0.4662  0.0000  0.4956  0.0000  0.0000
0.0000  0.0000  0.0000  0.4662  0.0000  0.4926  0.0000  0.0000  0.0000
0.0000  0.0000  0.4662  0.1279  0.4676  0.0000  0.0000  0.0000  0.0000
0.0000  0.4794  0.3485  0.7574  0.0000;
0.0000  0.0000  0.0000  0.0000  0.5500  0.8868  0.3338  0.0000  0.0000
0.0000  0.0000  0.0000  0.6603  0.7279  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.8176  0.8206  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.7926  0.7985  0.0000  0.0000;
0.0000  0.0000  0.0000  0.0000  0.9838  0.7294  0.9132  0.9015  0.0000
0.0000  0.0000  0.0000  0.9603  0.8574  0.8897  0.8941  0.0000  0.0000
0.0000  0.0000  0.9618  0.8853  0.9588  0.9382  0.0000  0.0000  0.0000
0.0000  0.8824  0.8706  0.8868  0.8868;
0.0000  0.0000  0.0000  0.6771  0.0000  0.8824  0.8794  0.8132  0.0000
0.0000  0.0000  0.0000  0.0000  0.8971  0.7353  0.8676  0.0000  0.0000
0.0000  0.0000  0.0000  0.7647  0.7309  0.4809  0.0000  0.0000  0.0000
0.0000  0.0000  0.7632  0.6074  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.4529  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.0000  0.4691  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.3000  0.4691  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.4985  0.6824  0.0000;
0.0000  0.0000  0.0000  0.0000  0.5147  0.0000  0.0000  0.5191  0.0000
0.0000  0.0000  0.0000  0.5588  0.0000  0.0000  0.5603  0.0000  0.0000
0.0000  0.0000  0.6191  0.0000  0.0000  0.6162  0.0000  0.0000  0.0000
0.0000  0.5735  0.0000  0.0000  0.5721;
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.6794  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.0000  0.4853  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.8000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.7191  0.0000;
0.0000  0.0000  0.0000  0.8057  0.0000  0.6059  0.9294  0.6691  0.0000
0.0000  0.0000  0.8114  0.0000  0.3309  0.8691  0.6397  0.0000  0.0000
0.0000  0.6086  0.0000  0.0000  0.8706  0.6074  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.7221  0.4750;
0.0000  0.0000  0.8086  0.0000  0.7853  1.0029  0.9735  0.7926  0.0000
0.0000  0.8114  0.0000  0.7456  1.0015  0.9735  0.7515  0.0000  0.0000
0.8086  0.0000  0.8147  1.0059  0.9750  0.8250  0.0000  0.0000  0.8086
0.0000  0.7426  1.0000  0.9779  0.7574;
0.0000  0.0000  0.0000  0.0000  0.8206  0.8426  0.8235  0.4853  0.0000
0.1200  0.0000  0.0000  0.8721  0.8912  0.8824  0.8794  0.0000  0.8000
0.0000  0.0000  0.8706  0.8750  0.8588  0.8691  0.0000  0.8200  0.0000
0.0000  0.8838  0.8897  0.8735  0.8809;
0.0000  0.0000  0.0000  0.0000  0.5162  0.7338  0.9074  0.7897  0.0000
0.0000  0.0000  0.0000  0.0000  0.8368  0.9103  0.7809  0.0000  0.0000
0.0000  0.0000  0.0000  0.9647  0.9471  0.5632  0.0000  0.0000  0.0000
0.0000  0.0000  0.7162  0.7162  0.2471;
0.0000  0.0000  0.0000  0.8314  0.0000  0.9897  0.9765  0.7294  0.0000
0.0000  0.0000  0.9114  0.0000  0.9103  0.9324  0.8221  0.0000  0.0000
0.0000  0.9086  0.0000  0.9588  0.9721  0.8662  0.0000  0.0000  0.0000
0.8057  0.0000  0.9559  0.9515  0.9279;
0.0000  0.0000  0.0000  0.0000  0.0000  0.8794  0.8824  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.8750  0.8691  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.8632  0.8618  0.0000  0.0000  0.0000  0.0000
0.0000  0.2000  0.7559  0.7544  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.5588  0.5471  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.5000  0.4971  0.0000  0.0000  0.0000
```

```
        0.0000  0.0000  0.0000  0.6191  0.6162  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.5809  0.5794  0.0000;
0.0000  0.8257  0.0000  0.0000  0.8882  0.8941  0.8912  0.8809  0.0000
0.9400  0.0000  0.0000  0.8824  0.8897  0.8868  0.8735  0.0000  0.9371
0.0000  0.0000  0.8735  0.8779  0.8765  0.8574  0.0000  0.9314  0.0000
0.0000  0.8662  0.8750  0.8706  0.8706;
0.0000  0.0000  0.0000  0.0000  0.0000  0.5971  0.0000  0.2588  0.0000
0.0000  0.0000  0.0000  0.0000  0.5294  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.6706  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.6162  0.0000  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.7544  0.7515  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.7265  0.7206  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.8206  0.8206  0.0000  0.0000  0.0000  0.0000
0.0000  0.2103  0.6706  0.6647  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.4956  0.4956  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.5515  0.5471  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.5250  0.5265  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.6485  0.6456  0.4074;
0.0000  0.0000  0.0000  0.0000  0.0000  0.4500  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.4500  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.4471  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.2735  0.4485  0.0000  0.2206;
0.0000  0.0000  0.0000  0.0000  0.7956  0.8029  0.8000  0.0000  0.0000
0.0000  0.0000  0.0000  0.6632  0.6809  0.6779  0.0000  0.0000  0.0000
0.0000  0.0000  0.2603  0.7676  0.7603  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.5868  0.5868  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.5118  0.2588  0.0000
0.0000  0.0000  0.0000  0.0000  0.0000  0.4706  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.6088  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.5574  0.0000;
0.0000  0.0000  0.0000  0.0000  0.7691  0.5985  0.7706  0.7706  0.0000
0.0000  0.0000  0.0000  0.8279  0.2647  0.8324  0.8294  0.0000  0.0000
0.0000  0.0000  0.8662  0.0000  0.8676  0.8691  0.0000  0.0000  0.0000
0.0000  0.9191  0.0000  0.8985  0.8897;
0.0000  0.0000  0.0000  0.0000  0.9118  0.9956  0.9735  0.8912  0.0000
0.0000  0.0000  0.0000  0.8324  0.8632  0.8412  0.8529  0.0000  0.0000
0.0000  0.0000  0.2515  0.9353  0.9000  0.9162  0.0000  0.0000  0.0000
0.0000  0.0000  0.7985  0.6191  0.9235;
0.0000  0.0000  0.0000  0.0000  0.0000  0.9750  0.9706  0.4647  0.0000
0.0000  0.0000  0.0000  0.0000  0.9397  0.9176  0.4765  0.0000  0.0000
0.0000  0.0000  0.0000  0.9897  0.9676  0.5647  0.0000  0.0000  0.0000
0.0000  0.0000  0.9456  0.9397  0.4676;
0.0000  0.0000  0.0000  0.0000  0.8485  0.8544  0.8529  0.8500  0.0000
0.0000  0.0000  0.0000  0.6279  0.8132  0.8103  0.8074  0.0000  0.0000
0.0000  0.0000  0.2588  0.8059  0.8029  0.8015  0.0000  0.0000  0.0000
0.0000  0.0000  0.5132  0.7132  0.7118;
0.0000  0.0000  0.8086  0.0000  0.7529  1.0000  0.9765  0.7676  0.0000
0.0000  0.8171  0.0000  0.8368  0.9926  0.9838  0.8412  0.0000  0.0000
0.8057  0.0000  0.7985  0.9971  0.9868  0.8015  0.0000  0.0000  0.5514
0.0000  0.6985  0.9853  0.9750  0.7500;
0.0000  0.0000  0.0000  0.0000  0.6147  0.8853  0.8809  0.6162  0.0000
0.0000  0.0000  0.0000  0.3279  0.8162  0.8044  0.7456  0.0000  0.0000
0.0000  0.0000  0.0000  0.7515  0.7559  0.7324  0.0000  0.0000  0.0000
0.0000  0.0000  0.7029  0.7794  0.2529;
0.0000  0.0000  0.0000  0.0000  0.4691  0.0000  0.4691  0.0000  0.0000
0.0000  0.0000  0.0000  0.4632  0.0000  0.4676  0.0000  0.0000  0.0000
0.0000  0.0000  0.4676  0.0000  0.4735  0.0000  0.0000  0.0000  0.0000
0.0000  0.4662  0.0000  0.4676  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.4676  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.0000  0.4824  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.4853  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.6809  0.0000;
0.0000  0.6371  0.0000  0.0000  0.4706  0.8632  0.8735  0.0000  0.0000
0.7686  0.0000  0.0000  0.8235  0.9294  0.9265  0.0000  0.0000  0.8171
0.0000  0.0000  0.8588  0.9353  0.9338  0.0000  0.0000  0.8629  0.0000
0.0000  0.7515  0.9368  0.9353  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.9691  0.9735  0.7191  0.0000
0.0000  0.0000  0.0000  0.0000  0.9691  0.9735  0.7074  0.0000  0.0000
0.0000  0.0000  0.0000  0.9706  0.9735  0.7162  0.0000  0.0000  0.0000
0.0000  0.0000  0.9176  0.9544  0.5397;
0.0000  0.0000  0.0000  0.0000  0.0000  0.7485  0.7544  0.5088  0.0000
0.0000  0.0000  0.0000  0.0000  0.9632  0.9559  0.5809  0.0000  0.0000
0.0000  0.0000  0.0000  0.9882  0.9632  0.5691  0.0000  0.0000  0.0000
0.0000  0.0000  0.9868  0.9676  0.6279;
```

0.0000 0.0000 0.0000 0.0000 0.5544 0.0000 0.0000 0.5559 0.0000
0.0000 0.0000 0.0000 0.4647 0.0000 0.0000 0.4676 0.0000 0.0000
0.0000 0.0000 0.5588 0.0000 0.0000 0.4412 0.0000 0.0000 0.0000
0.0000 0.5279 0.0000 0.0000 0.5309;
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.4926 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.6118 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.6897 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.5603 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.6324 0.6309 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.7485 0.7456 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.6912 0.6882 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.6353 0.6309 0.0000;
0.0000 0.0000 0.0000 0.0000 0.8309 0.8971 0.8971 0.8338 0.0000
0.0000 0.0000 0.0000 0.9485 0.8853 0.8676 0.8897 0.0000 0.0000
0.0000 0.0000 0.9044 0.8088 0.8088 0.8515 0.0000 0.0000 0.0000
0.0000 0.7956 0.9647 0.8500 0.7412;
0.0000 0.0000 0.0000 0.0000 0.0000 0.9500 0.9485 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.9618 0.9529 0.1000 0.0000 0.0000
0.0000 0.0000 0.0000 0.8956 0.8897 0.5706 0.0000 0.0000 0.0000
0.0000 0.0000 0.9882 0.9794 0.6235;
0.0000 0.0000 0.0000 0.0000 0.0000 0.6235 0.6206 0.6176 0.0000
0.0000 0.0000 0.0000 0.8706 0.7588 0.7544 0.7559 0.0000 0.0000
0.0000 0.0000 0.8044 0.7441 0.8059 0.8000 0.0000 0.0000 0.0000
0.0000 0.8059 0.7426 0.8074 0.8059;
0.0000 0.0000 0.0000 0.0000 0.5397 0.5735 0.0000 0.5588 0.0000
0.0000 0.0000 0.0000 0.8235 0.8324 0.0000 0.8441 0.0000 0.0000
0.0000 0.0000 0.8176 0.8721 0.0000 0.8691 0.0000 0.0000 0.0000
0.0000 0.7868 0.7809 0.0000 0.7779;
0.0000 0.0000 0.0000 0.0000 0.0000 0.2515 0.4779 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.5632 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.6500 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.5662 0.0000;
0.0000 0.0000 0.0000 0.0000 0.7515 0.9750 0.9809 0.7515 0.0000
0.0000 0.0000 0.0000 0.7515 0.8956 0.8338 0.7574 0.0000 0.0000
0.0000 0.0000 0.8265 0.9132 0.9574 0.8176 0.0000 0.0000 0.0000
0.0000 0.7353 0.9735 0.9838 0.7382;
0.0000 0.0000 0.0000 0.0000 0.0000 0.9853 0.9794 0.5176 0.0000
0.0000 0.0000 0.0000 0.0000 0.9926 0.9779 0.5132 0.0000 0.0000
0.0000 0.0000 0.0000 0.9809 0.9691 0.5191 0.0000 0.0000 0.0000
0.0000 0.0000 0.7132 0.7118 0.4647;
0.0000 0.0000 0.0000 0.0000 0.0000 0.4441 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.4412 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.4426 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.4426 0.0000 0.0000;
0.0000 0.0000 0.0000 0.0000 0.9044 0.0000 0.9044 0.8985 0.0000
0.0000 0.0000 0.0000 0.8868 0.0000 0.8882 0.8868 0.0000 0.0000
0.0000 0.0000 0.8809 0.0000 0.8838 0.8456 0.0000 0.0000 0.0000
0.0000 0.9691 0.0000 0.9574 0.9221;
0.0000 0.0000 0.0000 0.0000 0.0000 0.7956 0.7941 0.5309 0.0000
0.0000 0.0000 0.8057 0.0000 0.8868 0.8824 0.6191 0.0000 0.0000
0.0000 0.8114 0.0000 0.9456 0.9456 0.7926 0.0000 0.0000 0.0000
0.8086 0.0000 0.9500 0.9471 0.8044;
0.0000 0.0000 0.0000 0.0000 0.0000 0.5426 0.5426 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.6176 0.6235 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.5471 0.5471 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.5235 0.5206 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.5868 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.5603 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.7162 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.6176 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.8294 0.8338 0.5618 0.0000
0.0000 0.0000 0.0000 0.0000 0.8324 0.8500 0.5044 0.0000 0.0000
0.0000 0.0000 0.0000 0.7926 0.7897 0.4647 0.0000 0.0000 0.0000
0.0000 0.0000 0.7265 0.7235 0.4618;
0.0000 0.0000 0.0000 0.0000 0.8382 0.9735 0.9632 0.8397 0.0000
0.0000 0.0000 0.0000 0.7956 0.9691 0.9632 0.7941 0.0000 0.0000
0.0000 0.0000 0.8618 0.9735 0.9691 0.8647 0.0000 0.0000 0.0000
0.0000 0.8382 0.9868 0.9794 0.8382;
0.0000 0.0000 0.0000 0.0000 0.6544 0.9603 0.9647 0.7103 0.8143
0.0000 0.0000 0.0000 0.8706 0.9809 0.9721 0.8706 0.8143 0.0000
0.0000 0.0000 0.8838 0.9824 0.9706 0.8838 0.8457 0.0000 0.0000
0.0000 0.8838 0.9824 0.9706 0.8838;
0.0000 0.0000 0.0000 0.0000 0.5353 0.8691 0.8426 0.4926 0.0000
0.0000 0.0000 0.0000 0.8456 0.8676 0.7426 0.8882 0.0000 0.00
0.0000 0.0000 0.9015 0.8706. 0.8221 0.9015 0.0000 0.0000 0.0000

0.0000 0.9118 0.8618 0.8647 0.9029;

0.0000 0.0000 0.0000 0.8086 0.9456 0.9485 0.9441 0.9074 0.0000
0.0000 0.0000 0.8229 0.9191 0.9309 0.9118 0.8838 0.0000 0.0000
0.0000 0.8114 0.9118 0.9221 0.9162 0.9000 0.0000 0.0000 0.0000
0.8057 0.9015 0.8985 0.8985 0.9029;
0.0000 0.0000 0.0000 0.0000 0.0000 0.7574 0.6544 0.8265 0.0000
0.0000 0.0000 0.0000 0.0000 0.5985 0.0000 0.9456 0.0000 0.0000
0.0000 0.0000 0.0000 0.5471 0.0000 0.8662 0.0000 0.0000 0.0000
0.0000 0.0000 0.5191 0.0000 0.6353;
0.0000 0.0000 0.0000 0.0000 0.0000 0.6574 0.6926 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.5544 0.5515 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.4926 0.4956 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.4824 0.4765 0.0000;
0.0000 0.0000 0.8000 0.0000 0.8074 0.9544 0.9515 0.8059 0.0000
0.0000 0.8029 0.0000 0.8015 0.9294 0.9515 0.8015 0.0000 0.0000
0.8029 0.0000 0.8603 0.9426 0.9544 0.8603 0.0000 0.0000 0.8000
0.0000 0.8265 0.9412 0.9382 0.8250;
0.0000 0.8229 0.0000 0.0000 0.8897 0.8941 0.8912 0.8926 0.0000
0.8171 0.0000 0.0000 0.9471 0.9426 0.8794 0.9015 0.0000 0.8171
0.0000 0.0000 0.8941 0.9015 0.8971 0.8956 0.0000 0.8086 0.0000
0.0000 0.8956 0.9015 0.9000 0.8706;
0.0000 0.0000 0.0000 0.0000 0.7985 0.5941 0.9691 0.3235 0.0000
0.0000 0.0000 0.0000 0.8471 0.8662 0.9750 0.8206 0.0000 0.0000
0.0000 0.0000 0.7985 0.9632 0.9662 0.8000 0.0000 0.0000 0.0000
0.0000 0.8206 0.9647 0.9647 0.8221;
0.0000 0.0000 0.0000 0.0000 0.0000 0.6147 0.6147 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.6809 0.6750 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.6441 0.6412 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.6662 0.6588 0.0000;
0.0000 0.0000 0.0000 0.0000 0.7647 0.0000 0.0000 0.7662 0.0000
0.0000 0.0000 0.0000 0.6912 0.2971 0.0000 0.6926 0.0000 0.0000
0.0000 0.0000 0.7529 0.6750 0.0000 0.7529 0.0000 0.0000 0.0000
0.0000 0.7250 0.7294 0.0000 0.7235;
0.0000 0.0000 0.0000 0.0000 0.0000 0.7294 0.7265 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.7985 0.7941 0.0000 0.0000 0.0000
0.0000 0.0000 0.1015 0.8294 0.8132 0.0000 0.0000 0.0000 0.0000
0.0000 0.1382 0.8412 0.8250 0.0000;
0.0000 0.0000 0.0000 0.0000 0.8662 0.9206 0.8456 0.7838 0.0000
0.0000 0.0000 0.0000 0.9279 0.7338 0.9103 0.9176 0.0000 0.0000
0.0000 0.0000 0.9559 0.7338 0.9118 0.9206 0.0000 0.0000 0.0000
0.0000 0.8294 0.7309 0.9044 0.8279;
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.7191 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.8456 0.0000 0.0000 0.0000
0.0000 0.0000 0.4088 0.0000 0.5735 0.0000 0.0000 0.0000 0.0000
0.0000 0.4691 0.0000 0.8279 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.9382 0.9353 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.9397 0.9368 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.9647 0.9544 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.9603 0.9544 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.6971 0.0000 0.6926 0.0000
0.0000 0.0000 0.0000 0.0000 0.5500 0.0000 0.5515 0.0000 0.0000
0.0000 0.0000 0.0000 0.4809 0.0000 0.4765 0.0000 0.0000 0.0000
0.0000 0.0000 0.6838 0.0000 0.6765;
0.0000 0.0000 0.0000 0.0000 0.0000 0.6000 0.6000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.4721 0.4691 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.5382 0.5353 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.6412 0.6353 0.2206;
0.0000 0.0000 0.0000 0.0000 0.7456 0.7515 0.0000 0.7485 0.0000
0.0000 0.0000 0.0000 0.6471 0.7221 0.0000 0.7206 0.0000 0.0000
0.0000 0.0000 0.2559 0.8779 0.0000 0.8603 0.0000 0.0000 0.0000
0.0000 0.0000 0.7132 0.0000 0.7147;
0.0000 0.0000 0.0000 0.0000 0.0000 0.7103 0.7103 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.8294 0.8279 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.7691 0.7647 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.7103 0.7074 0.0000;
0.0000 0.0000 0.0000 0.0000 0.4676 0.0000 0.4676 0.0000 0.0000
0.0000 0.0000 0.0000 0.5000 0.0000 0.5000 0.0000 0.0000 0.0000
0.0000 0.0000 0.4647 0.0000 0.4647 0.0000 0.0000 0.0000 0.0000
0.0000 0.6412 0.2029 0.6441 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.7029 0.7000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.7515 0.7485 0.0000 0.0000 0.0000
0.0000 0.0000 0.1382 0.7088 0.7044 0.0000 0.0000 0.0000 0.0000
0.0000 0.5794 0.7882 0.7794 0.0000;

0.0000 0.0000 0.0000 0.0000 0.6956 0.8162 0.8103 0.3588 0.0000
0.0000 0.0000 0.0000 0.7676 0.9515 0.9559 0.7397 0.0000 0.0000
0.0000 0.0000 0.8676 0.9824 0.9574 0.8529 0.0000 0.0000 0.0000
0.0000 0.8632 0.9926 0.9588 0.8662;
0.0000 0.0000 0.0000 0.8057 0.9044 0.9059 0.8985 0.8824 0.0000
0.0000 0.0000 0.8143 0.9103 0.9176 0.9029 0.8794 0.0000 0.0000
0.0000 0.8029 0.8750 0.8809 0.8735 0.8750 0.0000 0.0000 0.0000
0.8029 0.8471 0.8529 0.8500 0.8500;
0.0000 0.0000 0.0000 0.0000 0.0000 0.7456 0.7426 0.3265 0.0000
0.0000 0.0000 0.0000 0.0000 0.7353 0.7324 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.9044 0.8471 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.8118 0.8074 0.0000;
0.0000 0.0000 0.0000 0.0000 0.5794 0.9956 0.9824 0.5941 0.0000
0.0000 0.0000 0.0000 0.4941 0.9221 0.9559 0.3353 0.0000 0.0000
0.0000 0.0000 0.4603 0.8515 0.8485 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.6500 0.6529 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.8059 0.8015 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.7382 0.7324 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.8059 0.5765 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.4721 0.4632 0.0000;
0.5714 0.0000 0.0000 0.0000 0.9279 0.0000 0.6721 0.8632 0.9257
0.0000 0.0000 0.0000 0.9794 0.0000 0.6824 0.9588 0.8829 0.0000
0.0000 0.0000 0.9824 0.0000 0.7397 0.9838 0.8657 0.0000 0.0000
0.0000 0.9838 0.0000 0.9676 0.9853;
0.0000 0.0000 0.0000 0.0000 0.8221 0.9397 0.9338 0.6676 0.0000
0.0000 0.0000 0.0000 0.8250 0.9397 0.9382 0.5559 0.0000 0.0000
0.0000 0.0000 0.7544 0.9412 0.9397 0.7412 0.0000 0.0000 0.0000
0.0000 0.7279 0.9397 0.9368 0.7294;
0.0000 0.0000 0.0000 0.0000 0.0000 0.9882 0.9824 0.6029 0.0000
0.0000 0.0000 0.0000 0.0000 0.9897 0.9838 0.4809 0.0000 0.0000
0.0000 0.0000 0.0000 0.8632 0.8603 0.3132 0.0000 0.0000 0.0000
0.0000 0.0000 0.7338 0.7294 0.0000;
0.0000 0.0000 0.0000 0.0000 0.0000 0.8294 0.8250 0.5500 0.0000
0.0000 0.0000 0.7143 0.0000 0.8971 0.8926 0.6147 0.0000 0.0000
0.0000 0.8486 0.0000 1.0029 0.9794 0.7794 0.0000 0.0000 0.0000
0.8057 0.0000 0.9662 0.9588 0.7044;
0.0000 0.0000 0.0000 0.0000 0.0000 0.8029 0.7838 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.7765 0.7735 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.7059 0.7015 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.6985 0.6941 0.0000;
0.0000 0.0000 0.0000 0.0000 0.4618 0.0000 0.4721 0.0000 0.0000
0.0000 0.0000 0.0000 0.4632 0.0000 0.4647 0.0000 0.0000 0.0000
0.0000 0.0000 0.4588 0.0000 0.5029 0.0000 0.0000 0.0000 0.0000
0.0000 0.4647 0.0000 0.4647 0.0000;
0.0000 0.0000 0.0000 0.8143 0.0000 0.9882 0.9765 0.7426 0.0000
0.0000 0.0000 0.8057 0.0000 0.9824 0.9382 0.6721 0.0000 0.0000
0.0000 0.8343 0.0000 0.9426 0.9706 0.7779 0.0000 0.0000 0.0000
0.8057 0.0000 0.9162 0.9544 0.7426;
0.0000 0.8029 0.0000 0.0000 0.6088 0.9176 0.9176 0.0000 0.0000
0.8143 0.0000 0.0000 0.7794 0.9324 0.9000 0.0000 0.0000 0.8686
0.0000 0.0000 0.9500 0.9397 0.8794 0.0000 0.0000 0.7400 0.0000
0.0000 0.7868 0.9088 0.9029 0.0000;
0.0000 0.0000 0.0000 0.0000 0.9412 0.9956 0.9574 0.8750 0.0000
0.0000 0.0000 0.0000 0.8574 0.9941 0.9809 0.8632 0.0000 0.0000
0.0000 0.0000 0.7647 0.9985 0.9750 0.7647 0.0000 0.0000 0.0000
0.0000 0.8662 0.9971 0.9735 0.8456;
0.0000 0.0000 0.0000 0.0000 0.6559 0.9926 0.9868 0.6574 0.0000
0.0000 0.0000 0.0000 0.4353 0.9971 0.9882 0.5309 0.0000 0.0000
0.0000 0.0000 0.0000 0.9926 0.9897 0.5206 0.0000 0.0000 0.0000
0.0000 0.0000 0.7603 0.7559 0.4353;
0.0000 0.0000 0.0000 0.0000 0.3382 0.8059 0.8015 0.5574 0.0000
0.0000 0.0000 0.0000 0.5956 0.9265 0.9235 0.6985 0.0000 0.0000
0.0000 0.0000 0.7426 0.9412 0.9397 0.7338 0.0000 0.0000 0.0000
0.0000 0.7941 0.9324 0.9309 0.7956;
0.0000 0.0000 0.0000 0.0000 0.8500 0.9294 0.9250 0.8882 0.0000
0.0000 0.0000 0.0000 0.8765 0.9721 0.9706 0.8529 0.0000 0.0000
0.0000 0.0000 0.9324 0.5691 0.9691 0.9074 0.0000 0.0000 0.0000
0.0000 0.8662 0.8235 0.9691 0.8662;
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.9265 0.6368 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.9515 0.7544 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.9500 0.7338 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.9544 0.7882;
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.5926 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.7044 0.0000 0.0000 0.1286

```
        0.0000  0.0000  0.0000  0.2059  0.5103  0.0000  0.0000  0.4429
0.0000  0.0000  0.0000  0.5147  0.5176  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.6515  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.0000  0.8368  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.2588  0.6074  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.6088  0.7324  0.1750;
0.0000  0.0000  0.0000  0.0000  0.0000  0.5574  0.5559  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.5235  0.5132  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.6132  0.6103  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.5397  0.5426  0.0000;
0.0000  0.0000  0.0000  0.0000  0.8118  0.8397  0.0000  0.7985  0.0000
0.0000  0.0000  0.0000  0.8074  0.8103  0.0000  0.8059  0.0000  0.0000
0.0000  0.0000  0.7647  0.7676  0.0000  0.7632  0.0000  0.0000  0.0000
0.0000  0.7868  0.7912  0.0000  0.7838;
0.0000  0.0000  0.0000  0.0000  0.7897  0.9412  0.9294  0.8029  0.0000
0.0000  0.0000  0.0000  0.6971  0.9809  0.9721  0.6971  0.0000  0.0000
0.0000  0.0000  0.6912  0.9824  0.9838  0.6941  0.0000  0.0000  0.0000
0.0000  0.6471  0.9882  0.9529  0.6500;
0.0000  0.0000  0.0000  0.0000  0.0000  0.5926  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.6044  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.4735  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.4691  0.4529  0.0000;
0.0000  0.0000  0.0000  0.0000  0.0000  0.5279  0.5221  0.5191  0.0000
0.0000  0.0000  0.0000  0.0000  0.4941  0.4882  0.4941  0.0000  0.0000
0.0000  0.0000  0.0000  0.5074  0.5015  0.5044  0.0000  0.0000  0.0000
0.0000  0.0000  0.4897  0.4882  0.4882;
0.0000  0.0000  0.0000  0.0000  0.5029  0.9603  0.9368  0.5044  0.0000
0.0000  0.0000  0.0000  0.5191  0.8912  0.8912  0.3382  0.0000  0.0000
0.0000  0.0000  0.5485  0.9059  0.9015  0.0000  0.0000  0.0000  0.0000
0.0000  0.3147  0.7574  0.7544  0.0000;
0.0000  0.0000  0.0000  0.8086  0.0000  0.8794  0.9515  0.7397  0.0000
0.0000  0.0000  0.8086  0.0000  0.9706  0.9647  0.7162  0.0000  0.0000
0.0000  0.8086  0.0000  0.9897  0.9721  0.7250  0.0000  0.0000  0.0000
0.8114  0.0000  0.9838  0.9721  0.7324;
0.0000  0.0000  0.0000  0.0000  0.4574  0.9926  0.9559  0.5529  0.0000
0.0000  0.0000  0.0000  0.0000  0.9897  0.9750  0.5574  0.0000  0.0000
0.0000  0.0000  0.0000  0.8382  0.8353  0.4632  0.0000  0.0000  0.0000
0.0000  0.0000  0.6353  0.6309  0.3294;
0.0000  0.0000  0.0000  0.0000  0.0000  0.6588  0.0000  0.6574  0.0000
0.0000  0.0000  0.0000  0.0000  0.6382  0.0000  0.6426  0.0000  0.0000
0.0000  0.0000  0.0000  0.6588  0.0000  0.6559  0.0000  0.0000  0.0000
0.0000  0.5603  0.7485  0.0000  0.7221;
0.5629  0.0000  0.0000  0.0000  0.8603  0.8574  0.8603  0.8485  0.0000
0.0000  0.0000  0.0000  0.8853  0.8912  0.8868  0.8868  0.0000  0.0000
0.0000  0.0000  0.8118  0.8176  0.8147  0.8132  0.0000  0.0000  0.0000
0.0000  0.4559  0.8059  0.8029  0.8074;
0.0000  0.0000  0.0000  0.0000  0.0000  0.4897  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.4897  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.6324  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.6000  0.0000  0.0000;
0.8200  0.0000  0.0000  0.0000  0.8838  0.9824  0.9706  0.8838  0.8029
0.0000  0.0000  0.0000  0.8838  0.9824  0.9706  0.8838  0.8057  0.0000
0.0000  0.0000  0.9265  0.8985  0.9397  0.9353  0.8029  0.0000  0.0000
0.0000  0.9147  0.9191  0.9147  0.9088;
0.0000  0.0000  0.0000  0.0000  0.9750  0.9912  0.9897  0.9603  0.0000
0.0000  0.0000  0.0000  0.9176  0.9926  0.9882  0.9103  0.0000  0.0000
0.0000  0.0000  0.9162  0.9882  0.9882  0.9044  0.0000  0.0000  0.0000
0.0000  0.7191  0.9941  0.9882  0.8029;
0.0000  0.0000  0.0000  0.0000  0.0000  0.8059  0.8176  0.0000  0.0000

0.0000  0.0000  0.0000  0.0000  0.9029  0.9132  0.0000  0.0000  0.0000

0.0000  0.0000  0.0000  0.9044  0.9265  0.0000  0.0000  0.0000  0.0000

0.0000  0.0000  0.9294  0.9235  0.0000;

0.0000  0.0000  0.0000  0.0000  0.0000  0.4706  0.4618  0.4603  0.0000
0.0000  0.0000  0.0000  0.0000  0.4676  0.4662  0.4647  0.0000  0.0000
0.0000  0.0000  0.0000  0.5191  0.5191  0.5162  0.0000  0.0000  0.0000
0.0000  0.0000  0.4691  0.4676  0.4691;
0.0000  0.0000  0.0000  0.7571  0.0000  0.9382  0.6500  0.4044  0.0000
0.0000  0.0000  0.9514  0.0000  0.9676  0.8838  0.6956  0.0000  0.0000
0.0000  0.8829  0.0000  0.9809  0.9750  0.7485  0.0000  0.0000  0.0000
0.8429  0.0000  0.9912  0.9794  0.8191;
```

```
0.0000   0.0000   0.0000   0.0000   0.0000   0.5647   0.4676   0.0000   0.0000
0.0000   0.0000   0.0000   0.0000   0.7426   0.6397   0.0000   0.0000   0.0000
0.0000   0.0000   0.0000   0.6838   0.6794   0.0000   0.0000   0.0000   0.0000
0.0000   0.0000   0.7338   0.7309   0.0000;

0.0000   0.0000   0.0000   0.0000   0.0000   0.4750   0.4721   0.0000   0.0000
0.0000   0.0000   0.0000   0.0000   0.4882   0.4824   0.0000   0.0000   0.0000
0.0000   0.0000   0.0000   0.5162   0.5103   0.0000   0.0000   0.0000   0.0000
0.0000   0.0000   0.4721   0.4721   0.0000];
```