INTER-DOMAIN ROUTING – PRICING POLICY AND ROUTE SELECTION USING NEURAL NETWORKS

By

Wong Leung-chung Chris

A Thesis

submitted in partial fulfillment of the requirements

for the Degree of Master of Philosophy

Division of Information Engineering

The Chinese University of Hong Kong

June 1997

Acknowledgement

I would like to use this acknowledgement column to express my gratitude to those who supported me during this project. Without their helps, suggestions, friendships and entertainment, I would not be able to bring this thesis into its final form.

First of all, thank God, the creator of heavens and the earth, for life, hope and health. Thank Lord Jesus, who never left me behind and His words are always my inspiration, courage and faith.

I would also like to thank Dr. K.W.Cheung, my supervisor, who gave me abundant freedom of research so that I can carry on with my interest on artificial intelligence and also his kindly advice and supervision. I would also want to thank Dr. C.P.Kwong, my undergraduate supervisor, who taught me the essence of being a researcher and inspires me on the field of artificial intelligence research. And also various neural researchers, who have lay the foundations of neural networks. Their contributions has definitely shaped the agenda of neural networks research.

Besides, thank Eunice, my best and dearest friend, who always cares for me, supports me and encourages me; my mother who reserve the dinners whenever I come home late when working on this project; my sisters and her husband, for

their concerns and their support when I made up my decision to pursue this M. Phil. degree.

Furthermore, thank Albert Sung for many thankful discussions and joys on sharing our interests; Thank the members of Lightwave Communication Laboratory – Arion Ko, Patrick Lam, C.K.Chan, Simon Lau, Samatha Chan, Y.H. Wang and Y. Gao, and also the members of Performance Evaluation Laboratory and Wireless Communication Laborary – Andy Yuen, Qin An, Li Xia, Y.K. Lee, Derek Chan, Y.K.Ma, Freeman Hui and K.Y. Chan for joy and memories during these two years of M.Phil studies.

Abstract

This thesis addresses the commercialization aspect of the Internet and the consequence of a new inter-domain routing paradigm, namely, pricing of bandwidth that depends on traditional performance metrics such as delay and throughput, during route computations.

We start with a general overview of the subject on routing and the related issues on the Internet, especially on the inter-domain routing aspect. We will state our vision and objective of this thesis in chapter 2. First, we explain the needs for the pricing strategies on inter-domain routing protocols. Then, we propose a pricing scheme. Thirdly, define our problem statement of price constrained routing such that each individual's utility is optimized. Since, the problem is NP-complete, we propose to use neural networks to tackle the routing problem. Thus, in chapter 3, we first have a review on the applications of neural networks on packet routing. Then following some of the major branches of previous neural-routing research, we discuss two neural network based routing strategy on chapter 4 and chapter 5 respectively.

In chapter 4, we use a Multi-Layer Feedforward Network (MLFN) to solve the stated problem. We begin with some theoretical studies on the MFLN and then we verify the model with simulation results. Chapter 5 deals with another

type of neural network known as the Hopfield Neural Network and in chapter 5 we have shown theoretically that the Hopfield NN is also capable of tackling this routing problem. Finally, we conclude in chapter 6.

Contents

1 Introduction **1**

1.1 Routing Overview 2

1.2 Routing in the Internet 5

1.2.1 Inter-Domain Routing 6

1.2.2 Intra-Domain Routing 7

1.2.3 The Future Trend 7

2 Inter-Domain Routing **9**

2.1 Inter-Domain Routing Protocols 9

2.1.1 Exterior Gateway Protocol (EGP) 10

2.1.2 Border Gateway Protocol (BGP) 11

2.1.3 Inter-Domain Policy Routing (IDPR) 12

2.1.4 Other Protocols 13

2.2 The Need for Pricing on Inter-Domain Routing Protocols 13

2.3 Pricing Scheme on the Inter-Domain level 15

2.4 Routing Protocols to Support Pricing on the Internet 16

2.4.1 Routing Towards Multiple-Additive Metrics 16

2.4.2 Network Model, Notations and Assumptions 16

2.4.3 The Problem Statement 18

3 Application of Neural Nets in Route Selection **20**

3.1 Neural Network (NN) Overview 20

3.1.1 Brief History on Neural Network Research 20

3.1.2 Definition of Neural Network 21

3.1.3 Neural Network Architectures 22

3.1.4 Transfer Fucntion of a Neuron 24

3.1.5 Learning Methods 25

3.1.6 Applications in Telecommunications 26

3.2 Review on the Applications of Neural Networks in Packet Routing 27

3.2.1 The JEB Branch 27

3.2.2 The Hopfield/Energy Minimization Branch (HEM) . . . 29

3.2.3 Supervised Learning (SL) 34

3.3 Discussions 35

4 Route Selection as "Link-state" Classification **36**

4.1 Multi-Layer Feedforward Network (MLFN) 37

4.1.1 Function Approximation Power of MLFN 38

4.1.2 Choosing MLFN parameters 40

4.1.3 Traning a MLFN 41

4.2 The Utility Function 43

4.3 The Neural Network Architecture 46

4.3.1 Routing Graph Representation with Successor Sequence Table (SST) 46

4.3.2 The Neural Network Layout 52

4.3.3 How the Neural Network Controller Works 55
4.3.4 Training 56
4.4 Simulation 56
4.4.1 Performance Parameters 56
4.4.2 Simulation Results 57
4.5 Conclusions and Discussions 70

5 Route Selection as Energy Minimization - A Theoretical Study 73
5.1 The Hopfield/Tank NN Model 73
5.2 Boltzman's Machine 76
5.3 Boltzman's Machine Model for Multiple-Metrices Routing 79
5.4 Conclusions 82

6 Conclusions 84

Bibliography 86

Chapter 1

Introduction

Routing in communication networks has been an area of extensive research over many years. It is one of the most complex problems within the domain of communication networks. Firstly, the goal of routing is not always well-defined, as there are too many factors, constraints to be considered, and the worse of all - many of them are contradicting with one another. On describing such a complex goal, Streenstrup define it as [42] : The goal of routing in a communications network is to direct user traffic from source to destination in accordance with the traffic's service requirements and the network's service restrictions. Its objective is to maximize network performance while minimize costs. Its constraints come from the underlying network technology and the traffic dynamics. The network performance includes delay, bandwidth, reliability, security and etc. The costs includes both the equipment cost and the operation cost. However, depending on the type of communication networks and the services provided on the network, these objectives and constraints may have different meanings and weights.

In this thesis we shall confine the scope of the problem to Internet routing. Furthermore, we shall concentrate on the inter-domain routing. We begin with a general overview of the routing issues and then focus on the inter-domain routing problem - Due to the commercialization of today's Internet, a pricing mechanism will be needed, but currently no routing algorithm that can support both price and performance had been proposed yet, such that each individual's utility is optimized. In this thesis two different neural network based routing algorithms are proposed to tackle this pricing/performance routing problem.

1.1 Routing Overview

There are many different angles of view on the subject of routing - depending on the type of communication networks: (1) how to distributing basic information for routing decision; (2) where routing decision are being made; (3) on the routing constraints and objectives; (4) on the algorithms and others. If these topics were discussed one by one, it requires several books to cover all of them! Instead, a list of some common terms on this subject will be discussed below. Hopefully, through these descriptions, the reader could get an general overview on the subject of routing. (Generally, this thesis deals with routing in packet switching networks thus more terms related to this topic would be discussed)

1. In **Centralized Routing**, a centralized node is responsible for making all the routing decisions for the whole network.

2. In **Distributed Routing**, each communication node is responsible for making routing decisions.

3. In **Source Specified Routing**, the originating node is responsible for specifying the entire path from it to the destination node.

4. In **Hop by Hop Routing**, each node will only choose a next hop node as its successor for a specific destination. It must be a kind of distributed routing.

5. In **Packet Switching Network (PSN)**, data is divided into smaller unit known as "packets", and routing is done by the communication node which first stores the entire packet and then forwards it.

6. In **Circuit Switching Network (CSN)**, a physical circuit is first setup and all the data is routed though this circuit to the destination.

7. **Virtual Circuit (VC)** routing or **Connection-Oriented** routing on a PSN means that a virtual circuit was first setup and then all information are routed through this VC path.

8. In **Datagram** routing or **Connectionless** routing on a PSN, each packet is treated independently and may route through different paths. No connection setup phase is required as compared with the VC approach, but packet may arrive out of sequence.

9. In **Hierarchical Routing**, big networks are divided into smaller networks known as domains. These domains may be further decomposed into smaller networks known as sub-domains. Routing in such hierarchical network is also divided into different levels. Each node in a domain knows only its paths to other nodes in the same domain. For communication

with other domains, border gateways are present on each domain which are responsible for these inter-domain communications.

10. The term **Routing Algorithm** is usually used for describing the algorithm that performs the route decisions for a communication network.

11. In **Adaptive** routing, the routing algorithm will react to topology or performance variation of the network, so that the network's objective can still be matched.

12. In **Static** routing, the routes are pre-computed. These pre-defined routes would never change until they are intervened by the network administrator or by the proper authorities.

13. **Shortest Path (SP) Routing** is a routing algorithm which chooses the paths with the least total cost from the source nodes to their destinations. The cost may be composed of different metrics, like delay, reliability, etc..

14. **Vector-Distance Algorithms** are a class of distributed/hop-by-hop routing algorithm, such that each node tells its neighbours the distances from it to other nodes. Then a node makes the routing decisions according to the information obtained. A well-known example of this is the Distributed Bellman-Ford (DBF) algorithm.

15. **Link-State Algorithm** are a class of distributed/source specific routing algorithm, such that each node obtains the whole network information (link states) by a flooding or broadcasting algorithm and then computes the optimal paths to different destinations.

16. **Congestion Control** by a routing algorithm means that the routing algorithm tries to select the routes, such that some congested region can be avoided.

17. A routing algorithm is **stable** if a small change in network topology would not affects the routing decision to be changed greatly.

18. A routing algorithm is **fair** if every node in a network receive similar performance/sharing on the network resources.

In describing the above terms, we are referencing to [44], [42], [47] and various papers. In the next section, we would concentrate on the routing issues in the global Internet.

1.2 Routing in the Internet

Internet is the world's largest PSN, recently its growing speed is probably beyond the imaginations of its former ARAPANET developers in the 1960s. Today, the Internet connects multiple-administrative domains, including universities, government bodies, commercial bodies and etc..

Since the size of the Internet is so tremendous, it is not possible for each routing node to know the routes to every other node. Moreover, a connection on the Internet may span across different administrative domains (ADs), and each AD would like to set up its own routing policies. In order to meet with these two characteristics, hierarchical routing was employed in the Internet. The Internet is divided into different regions known as administrative domains (ADs). Each AD is an independent authority that control over a set of networks

and gateways. A routing node within an AD knows all the details about the routes to each destination inside this AD, but knows nothing about the nodes of other ADs. We classify these connections within an AD is concerned with intra-domain routing. On the other hand, whenever a connection is required to contact the nodes of other ADs, the source node would choose a route to a Border Gateway (BG) of its AD, then the BG would make a connection to the BG of the destination domain. Finally, that foreign BG would pass the messages to the destination node. The BG-to-BG connection belongs to inter-domain routing, while the BG-to-node traffics are considered to be intra-domain traffic and would be handled by the corresponding intra-domain routing protocols.

1.2.1 Inter-Domain Routing

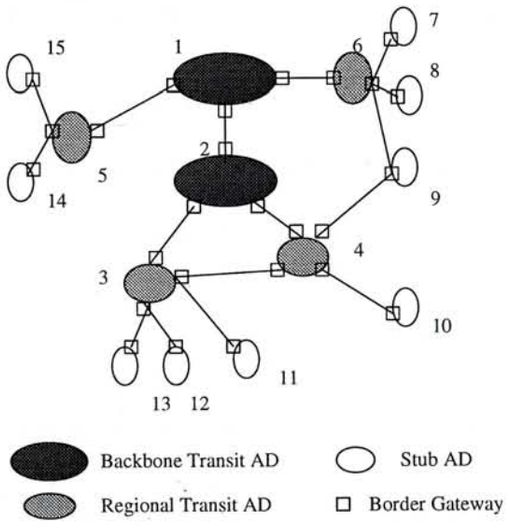


Figure 1.1: An Example Domain Topology

Inter-domain routing protocols are for routing at the domain level. Generally, a router known as the Border Gateway (BG) is responsible for inter domain routing. The BG knows the domain level network topology and makes the

routing decision to the destination domain. An example of domain level topology is shown in Figure 1.1. There may be multiple BGs within a domain. Examples of inter-domain routing protocols are EGP (Exterior Gateway Protocol), BGP (Border Gateway Protocol) and IDPR (Inter-Domain Policy Routing). We will discuss each of them in details in the next chapter.

1.2.2 Intra-Domain Routing

On the contrary, intra-domain routing protocols deals with the traffics within an administrative domain. The problem of intra-domain routing is generally less complicated then its inter-domain counterpart. Examples of intra-domain routing protocols are : OSPF and IS-IS

1.2.3 The Future Trend

Within a few years of time the Internet is shifting from its academic oriented nature into a multi-polarity and multi-functionality global wide information exchange backbone. The commercialization of the Internet has made many researchers to ponder on new ideas on the Internet. A popular topic of research is on the pricing policies on the Internet [35]. Routing is directly concerned with the sharing of network resources as the path generated by the routing algorithm crosses many different components in a network. The control and sharing of resources have long been an economic research topic [10]. It would not be surprised that routing and pricing on the Internet would be considered as the same problem very soon. Consider this price constrained routing, a particular important level of routing is on the inter-domain level. In the next chapter we would

begin with some reviews on existing inter-domain routing protocols and then we would establish our problem of price constrained routing on the Internet.

Chapter 2

Inter-Domain Routing

2.1 Inter-Domain Routing Protocols

The Internet is a heterogeneous network, which consists of different interconnected component networks operated and owned by different organizations. Each organization has its own policies and network configurations. The component networks belong to and administrated by the same organization is called an *Administrative Domain* (AD). Inter-domain routing protocols is responsible for the traffic between these domains. A major difference between the Inter-Domain routing and Intra-Domain routing is that the former is constrained by the policies of the interconnected ADs. In this chapter, we will go through some existing or proposed inter-domain routing protocols, like EGP, BGP, IDPR and etcs.. Finally, we will establish the need for pricing in today's heavily commercialized Internet and also the needs for an Inter-Domain routing protocol that allows a pricing policy to be incorporated. The problem statement of price constrained routing is also defined in this chapter.

2.1.1 Exterior Gateway Protocol (EGP)

The Exterior Gateway Protocol (EGP) [38], was the first inter-domain routing protocol introduced in the Internet in the early 80s. The EGP was designed for strictly two-level hierarchical network architecture with the top-level domain called the *core*. The ADs are not allowed to overlap. EGP relies on a distance-vector algorithm to construct the routing table. Due to its strictly two level architecture, the loop formation or count-to-infinity [44] [9] problem which is a common problem with most distance-vector algorithm would not be encountered in EGP. Periodic full updates are employed to exchange the routing information. Besides, EGP messages are carried directly over IP with no transport layer protocols. Furthermore, policy-base routing is very limited in the EGP – Individual ADs are allowed to hide portions of their routing database that they are not willing to share. Also, ADs are free to manipulate route metrics that they assign to other ADs in order to favour or preclude certain transit AD hops.

Limitations of EGP

The EGP was designed in the early 80s, although it could meet the demand at that time, it is no longer suitable for today's Internet. First, today's Internet is becoming more complex and a strictly two-level hierarchical model is no longer suitable for the Internet. EGP on today's Internet will bring back the count-to-infinity problem. Second, periodic full updates of routing information use a lot of bandwidth and this was exacerbated by the tremendous size of today's Internet. Thirdly, EGP messages use IP directly, but the huge size of today's Internet implies very long EGP messages such that a single IP packet may not hold the whole message. Since IP does not have fragmentation and defragmentation

capability, this may cause the EGP message to be corrupted. Besides, policy routing is vital in today's Internet, but EGP's notion of policy is very limited, for instance it does not support any Type-Of-Service (TOS) based policy and etc. .

2.1.2 Border Gateway Protocol (BGP)

BGP was first developed in the late 80s and had undergone various modifications. The most recent version of the protocol is BGP-4 [31]. BGP places no restriction on the inter-domain topology as opposes to the EGP's two level hierarchy. It is today's standard inter-domain routing protocol on the Internet. The best-effort transport layer protocol TCP is employed to transmit the BGP messages. Although BGP still relies on a vector-distance algorithm for routing, a list of domains that the routing information traversed so far was kept so as to eliminate any loop formations. This properly causes BGP to be labelled as a path-vector routing protocol. Instead of periodic full updates, BGP exchanges the routing information via incremental updates, i.e., only the changes are exchanged. Besides, route aggregation is employed to reduce the volume of routing information that needs to be handled. In addition to route metrics, BGP allows a *Degree of Preference* to be associated with the route. A route is chosen by firstly consider the degree of preference and tie-breaking with the metrics. BGP offers policy based routing in ways such as: resource sharing, selection among multiple entry/exit points of a AD and etc.. A policy language in BGP is express as: *[Network-list, AD-path] = preference* , which means that a routing update for a network in *Network-list* is received via *AD-path* (i.e. the sequence of ADs traversed so far) and if its preference metric is better than that of a path

currently in use, then this update must be used for subsequent routing.

Limitations of BGP

Although BGP is far better than EGP, it still has the following shortcomings – BGP provides at most one route to a destination. This lacks the flexibility of supporting multiple routes with different performance characteristics or administrative preferences. Furthermore, BGP does not support source-specific policies. Finally, "the degree of preference first and then the metrics" routing rule does not reflects the users' needs, thus limits the flexibility of routing policies.

2.1.3 Inter-Domain Policy Routing (IDPR)

IDPR [11] was designed as a policy-routing architecture that accommodates transit service provider policies as well as source-specific polices. The major difference of IDPR to the previous protocols is that link-state source routing algorithm is applied instead of vector-distance hop-by-hop routing algorithm. The traditional hop-by-hop routing protocols do not efficiently support the wide range of policies especially policies concerning types of service (TOS) that is anticipated in the global Internet. On the other hand, source routing combined with policy advertised in routing updates allows transit ADs to apply source-specific polices while permitting stub ADs to select routes based on local criteria. Besides, source routing place a relaxed consistency requirement for routing databases than the hop-by-hop case.

Limitation of IDPR

In [11], the proposed algorithm for route computation is the SPF with some aspects of policy are encoded in metrics before computation. As SPF only supports a single metric, combination of policy metrics and performance metrics to a single metric in a link-by-link basis is not a good idea. The policy metrics may lose its meaning at all. In the later sections of this chapter, we will discuss a proper way to combine this two kinds of metrics via utility functions.

2.1.4 Other Protocols

There is a even newer protocol known as the Inter-Domain Routing Protocol (IDRP) which was based on most of the BGP features but with improvements like – multi-protocol support and multiple routes to a destination with different performance characteristics. Nevertheless, it is still a path-vector algorithm and the support for source-specific policy is still limited. In this thesis, we are following the IDPR trend of link-state routing algorithm. In addition, there are yet some less significant proposals for inter domain routing protocols like IDPR II [2] and PBIR [22].

2.2 The Need for Pricing on Inter-Domain Routing Protocols

The beginning of the Internet traces back to the ARPANET in the 1970s. Nowadays, its development is reaching its golden era. Advances in hardware and software technologies have made many new services which was not feasible in

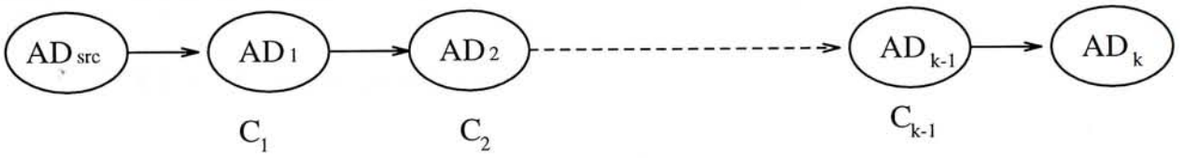
the past now becomes popular to the public. For instance, voice phone, video phone, multi-media web pages, web-shopping and etc.. In general, this batch of services requires high bandwidth, low delay and security ensured.

In the routing aspect, this increasing variety of services not only implies tougher performance requirements, but also implies a more complex scenario of resource sharing on the Internet. In the early days of the Internet, the traffics were relatively smaller and the services were mainly narrow band traffics, like gopher, email, telnet, and etc. At that time, the Internet users generally possessed a minimum anticipation on this "free" information medium. We may say that at the early stage of Internet, the supply was greater than the demand and the Pareto optimal was not reached. However, the rapidly growing Internet is becoming more attractive and people place a greater expectation on it. Now the paradigm is shifting, the demand is much greater than the supply. People are scrambling for network resources without regard that their acts would degrade services to other people and jams the whole network.

In economic term this scenario is concerned with externalities : effects which are "external" to the decision makers [19]. The best way to solve this problem is to let market mechanism to work - i.e. charging a price to the end-users or the administrative domains. Generally, the control of resources have long been an economic research and there are many ways we could borrow from their results [10]. On the other side, those who have taken part on the inter-domain routing researches have recently gone into this pricing issue [35] [32]. However, according to [35], the problem of "How to price" is still a controversial issue. In this thesis, we adapt a very simple pricing scheme, i.e. each AD setup a self-estimated cost for packets that use its resources. A more formal definition of our pricing scheme

would be discussed in the next section.

2.3 Pricing Scheme on the Inter-Domain level



$$\text{Total Cost Charged} = C_1 + C_2 + \ldots + C_{k-1}$$

Figure 2.1: Pricing scheme on the Inter-Domain level

Our idea of pricing is very simple, since each AD shares some of its resources (e.g. single border gateway or some parts of its network) for traffics that was not originated within the AD itself, the AD should impose a price on those traffics to cover its operation cost and compensate for the performance degradation incurred to its own users. Figure 2.1 shows this idea – an inter-domain traffic across several domains would be charged a cumulative price $P = \sum_{i=1}^{n} c_i$. Where c_i is the cost incurred to AD_i to handle this traffic. Concerning how the c_i are determined, it solely depends on the policy of AD_i. For instance they may charge according to the TOS, AD_PATH and etc. .

If all the traffics seek their minimum cost routes with the above pricing policy, according to [32], there would be a global minimum resource consumption. However, what if a user requires an urgent connection, he/she is willing to pay a higher price to trade with the performance? In such case, the above pricing scheme lacks the flexibility. In fact, a good market is a market with competitions, where each users' needs could be reflected. This is just what our thesis

concentrated. In the following sections, we will define our problem statements clearly.

2.4 Routing Protocols to Support Pricing on the Internet

2.4.1 Routing Towards Multiple-Additive Metrics

Regarding the performance aspect, traditional Internet routing protocols consider only one performance metrics, i.e. delay. Examples of such protocols are : Bellmen-Ford Algorithm that relies on a Vector-Distance Algorithm; Dijkstra's shortest path algorithm that rests on a Link-State Algorithm [42] and others [37]. Routing subject to multiple performance metrics, in most cases is a NP-complete problem. In the past, the services carried by the Internet is monotonous, thus a single metric routing was sufficient to most users. However, this scenario is likely to be changed and both price and performance would need to be considered.

2.4.2 Network Model, Notations and Assumptions

The Internet G is modelled as a connected graph in which the ADs are modelled as the edges of the graph and any pair of border gateways between those ADs as the nodes of the graph. (Refer to Figure 2 for an example of our model)

Each edge is associated with a link cost vector $\underline{V}$ with each element of $\underline{V}$ corresponds to a link metrics. Generally, there are two kinds of link metric : i) non-additive link attributes e.g. bandwidth; ii) additive link metrics e.g. delay,

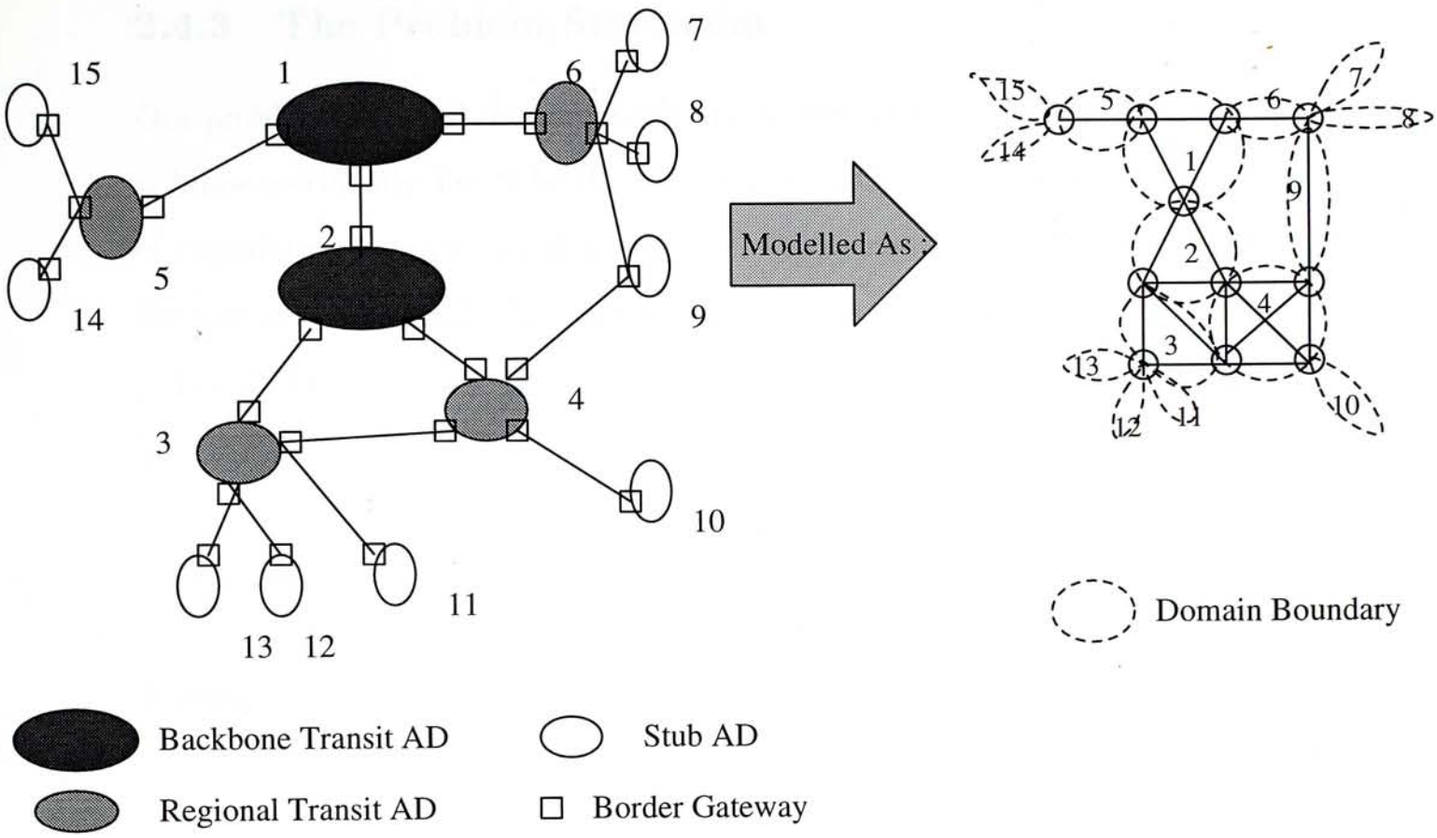


Figure 2.2: Network Model and Nodes and Edges

reliability (loss probability), price, hop-count and etc.. Furthermore we assume the followings :

1. All packets transmitted over an operational link are received correctly and in the proper sequence within a finite time.

2. The node learns the network topology and link cost vectors by a reliable network layer protocol that distributes those information.(eg. link-state flooding algorithm)

3. The routing information distribution protocol distributes all the routing information completely within every periodic intervals.

2.4.3 The Problem Statement

Our problem is to find the maximum utility path from a source i to destination j. More specifically, Let N be the set of nodes in G. Let vector $\underline{P_{ij}}$ be the vector of cumulative link metrics of acyclic paths from node i to j. Let vector $\underline{P^*_{ij}}$ be the vector of cumulative link metrics of the optimal acyclic path from node i to j. Let U_{ij} be the utility ofP_{ij}. $U_{ij} = U(P_{ij})$ and $U^*_{ij} = U(P*_{ij})$ where $U^*_{ij} \geq U_{ij}$ for all P_{ij}.

Our problem is to determine this P^*_{ij}.

Example

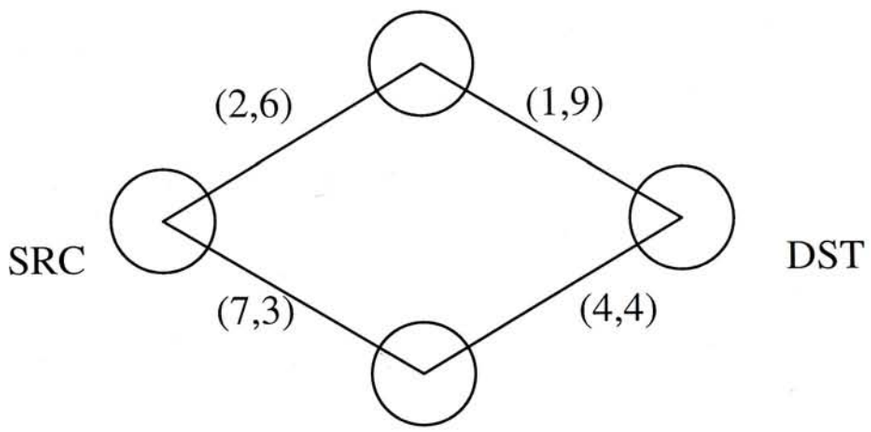


Figure 2.3: An Example Illustrates our Problem Statement

Refering to Figure 2.3 for a simple example, there are two routes from SRC to DST. Each edge is associated with two metrics and an arbitrary utility function is defined by $U(m1, m2) = \frac{1}{e^{-m1}+e^{-m2}}$. The utility value for the two paths are $U(3, 15) = 20.09$ and $U(11, 7) = 1076.90$ respectively, therefore the upper path would be selected as the optimal path.

Finding a path subject to more than one additive metrics for non-linear utility function is an NP-complete problem [49] [51]. In the future Internet, routing will tend to be a multiple metrics problem. Since neural network has the potential to provide "good" solutions to NP-complete problems, it is therefore a good candidate for tackling this NP-complete problem. In the next chapter, we will discuss the application of Neural Networks on routing.

Chapter 3

Application of Neural Nets in Route Selection

3.1 Neural Network (NN) Overview

3.1.1 Brief History on Neural Network Research

Artificial Neural Networks (NN) were inspired by the divine creation of the biological brain which was first applied to the computing field 40 years ago (1957) when Frank Rosenblatt, Charles Wightman, and others developed the first successful neurocomputer Mark I Perceptron. However, long before that time, the 1943 paper of Warren McCullock and Walter Pitts had shown that even simple types of neural networks could, in principle, compute any arithmetic or logical function. With the early successes of NN by many researchers during the late 1950s and early 1960s, people were getting proud and prophesied that artificial brains were just a few years away from development and other incredible

statements. However, most of the early neural researcher, generally, did not follow an analytical point of view but based on a qualitative and experimental aspect, which bothered many established scientists and engineers who observed the field of research. The prophecy turned out to be false, when Minsky and Papert's 1969 book *Perceptrons* proved mathematically that a perceptron could not implement the simple XOR logical function, nor many other such predicate functions. These sound facts had brought neural research into a dark age. At that time, as described by Hecht-Nielsen [17], NN research had to go "underground". After a decade (1967-1982) of quietness, the application of NN was shifted to the engineering field. The quiet works of many excellent researchers like Stephen Grossberg, Teuvo Kohonen, John Hopfield and many others had put the field of NN on a firm footing and prepared the way for renaissance of the field. In 1987, the first open conference on neural networks, the IEEE International Conference on Neural Networks, was held in San Diego. And the *IEEE Transactions on Neural Networks* was first published in 1990. The applications of NN had shown success on many engineering fields like, signal processing, pattern recognition, statistical analysis, robust control, combinatorial optimization, and artificial intelligence.

3.1.2 Definition of Neural Network

In this thesis, we define Neural Network (NN) as a network of many simple processors (units), each unit can be described as a multi-input single-output (MISO) dynamic system which is characterized by its transfer function and dynamic equations (learning equations). Besides, the units are connected by

uni-directional communication channels ("connections") which usually carry numeric data, encoded by any of various means and each unit operates only according to its local data and on its inputs received via the connections.

3.1.3 Neural Network Architectures

After decades of development on the NN, generally there are hundreds or even thousands of NN architectures developed. A conservative classification is according to their connection topologies. Accordingly, the 3 major classes are

1. **Non-Recurrent Neural Networks** - This class of NN consists of different "layers". A layer is a group of neuron units that do not have any inter-connection within the units. Connections are only allowed form layers to other layers. There is an *input layer* and an *output layer*, all other layers are regarded as *hidden layers.* The connection within layers could be either fully connected (i.e. each unit is connected to all units in the next layer) or partially connected. A special class of non-recurrent neural networks that does not have any hidden layers is known as Perceptron. Examples of non-recurrent NN are : Multi-Layer Feedforward Networks (MLFN), The Self-Organizing Maps (SOM) and etc. MLFN had shown success on the pattern recognition and classification fields.

2. **Recurrent Neural Netwroks** - Each unit of a recurrent NN are connected to all other units including itself. Generally, the concept of "layers" is not-applicable to this kind of NN. Examples on this kind of NN are : Hopfield NN, Boltzman's machine. This kind of NN had been used for constrained optimization and content addressible memory.

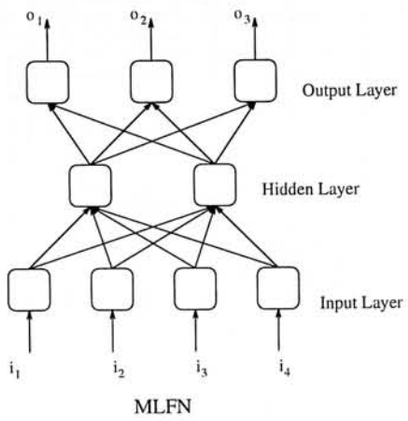


Figure 3.1: A Non-recurrent Neural Network

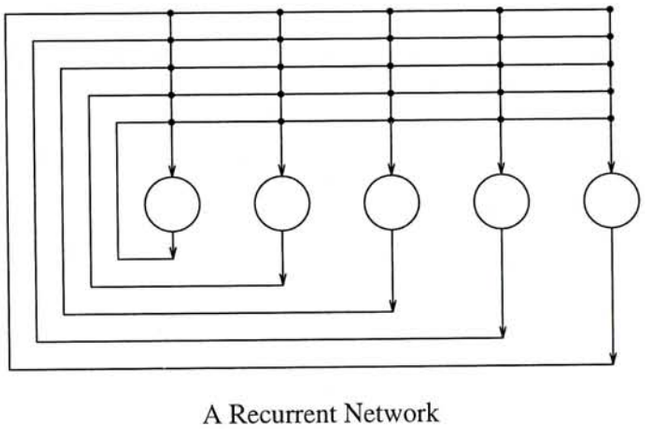


Figure 3.2: A Recurrent Neural Network

3. **Partial Recurrent Neural Networks** - This kind of NN is a hybird of (1) and (2). The concept of layer still applicable, but this time interconnections within the units of the same layers are allowed. There are many NN architecture falls into this categories, like the ART(Adaptive Resonance Theory) family, Jordan NN, ELman NN, Autoassociative Memory Network. This kind of NN was used for classification and pattern recognition.

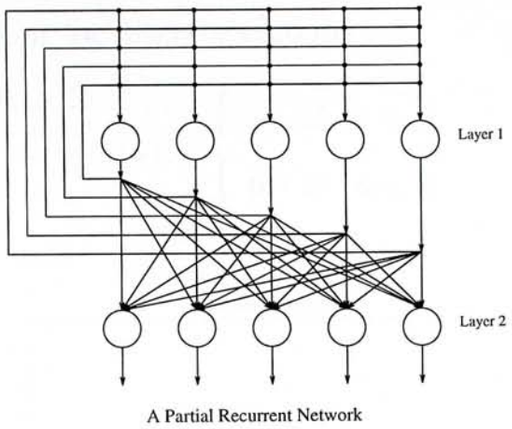


Figure 3.3: A Partial Recurrent Neural Network

3.1.4 Transfer Fucntion of a Neuron

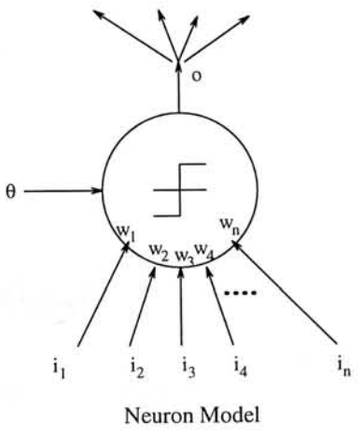


Figure 3.4: A Neuron

Generally, the *sigmoid* function is used as the transfer function of a neuron :

$$f(u) = \frac{1}{1+e^{-u}} = \frac{1}{2(1+tanh(u/2))} \tag{3.1}$$

where,

$$u = \sum_{j}^{n} w_j i_j - \theta \tag{3.2}$$

and i_j is the jth input, w_j is defined as the weight of the jth input and θ is known as the *bias* or *threshold* of the neuron. Sometimes the *sign* function is

also used for simplicity or for discrete systems :

$$sign(u) = \begin{cases} 1, & \text{if } u \geq 0, \\ 0, & \text{if } u < 0. \end{cases} \tag{3.3}$$

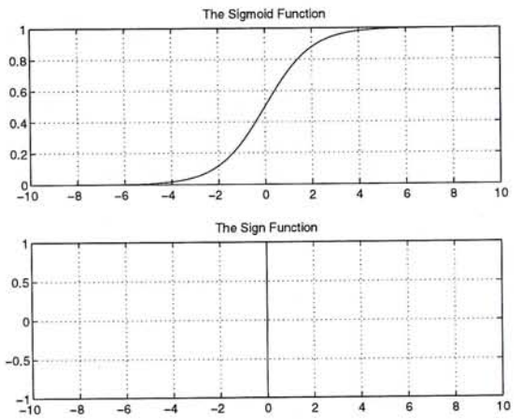


Figure 3.5: The Neuron Transfer Functions

3.1.5 Learning Methods

Again, there are many learning or training methods (Generally, each type of NN would have its own kind of learning method). Despite this large variety of learning methods, we could still classify them according to two learning philosophies : 1) Supervised Learning and 2) Unsupervised Learning.

1. **Supervised Learning** - Supervised learning means learning with teachers, i.e. we have a teacher who provide some solved examples to the NN. Then the NN learns form these examples and hopefully the NN can capture the "intelligence" within those example and produce the right response to other samples that had not been encountered before. Examples of learning algorithm belongs to this class are : BackPropagation

(BP), Counter-Propagation (CP), Learning Vector Quantization (LVQ), Probabilistic Neural Network (PNN), General Regression Neural Network (GRNN) and etc. .

2. **Unsupervised Learning** - Contrary to supervised learning, unsupervised learning requires the NN dynamics to be capable of, by just observing the given samples, converging to some stable states where the objective of the NN is achieved. Some well-known examples are : Hopfield Network, Kohonen Self-Organizing Map (SOM), Adaptive Resonance Theory (ART), Bidirectional Associative Memory (BAM) and etc..

According to Masters [24], there is a third type of learning that lies between supervised and unsupervised learning methods known as the *Reinforcement Learning*. In reinforcement learning, the neural network is allowed to react to each training sample. It is then told whether its reaction was good or bad. However, very few practical applications are known.

3.1.6 Applications in Telecommunications

There had been already a whole book devoted to this topic now [3]. According to it and various works, applications of neural networks on the telecommunications fields include:

1. Traffic Control, Admission Control, Congestion Control and Cell Scheduling in ATM. [3]

2. Channel Equalization [3, 4, 5]

3. Spread Spectrum Communication Systems [3]

4. Dynamic Channel Assignment and Mobile Communication Design [3]

5. Packet Radio and Satellite Communication [3, 20]

6. Switching, Routing and Multicast Routing [1, 13]

In the next section, we will focus on the applications of NN packet switching network routing. We begin with a review of previous research on this topic.

3.2 Review on the Applications of Neural Networks in Packet Routing

There had been already many works on the application of neural network in packet routing based on the shortest-path (SP) approach. Generally they could be classified into three major families. The JEB approach [6, 50, 21], the Hopfield/Energy minimization approach [20, 1, 27, 12, 7, 26, 39, 23, 46, 43, 14] and the Supervised Learning approach : [33, 8, 15]. In the following paragraphs, we would discuss each branch in detail and conclude about its suitability in solving our problem.

3.2.1 The JEB Branch

The JEB approach was first proposed by J.E.Jensen, M.A.Eshera and S.C.Barash in 1990 [21]. Wang [6] used the 3 initial letters of the authors and called it the JEB network. It was improved by Wang on [50]. The original JEB approach is a distributed routing algorithm for packet switching network which uses a 2 layer feed-forward neural network as a routing decision maker for next hop at

each communication node. It uses a Hebbian learing method for training the net and hopefully to find the shortest path for the packets. Training and operation are carried out at the same time. In order for training to be achieved, each packet are required to include fields like : origin of the packet, destination of the packet, the neighboring node from which the packet was received and the cost of the path that messsage was picked up. A single training update occurs whenever a packet arrives and using the above information as guidance for training. Whenever a link fails or changes its cost, it adapts the changes as the training information in the packets is changed. In this way, the JEB net would forget about its previous learned path.

Wang [6] summarized the problems of the original JEB network as follows :

1. Uneven training of different communication nodes.

2. Poor response to link change .

3. Poor response when a link is down.

Wang then proposed some modifications to tackle the above problems, namely, message echoing mechanism and local link awareness. However, problem (3) is only partially solved [6]. Wang applied his modified JEB net to solve a 8 nodes and a 27 nodes communication network. In the simulations, the JEB nets were trained first before actual simulation measurements were taken. Under dynamic network condition, the maximum percentage of optimal and near optimal paths formed is 77% for the 8 node network and 71% for the 27 node network (but 20,000 actions required for the training phase). Under extreme condition, these percentages drop to 58% and 63% respectively. The paper did not include any sub-optimal path statistics.

3.2.2 The Hopfield/Energy Minimization Branch (HEM)

This branch seems to be the most common approach to the neural-routing problem. The idea is to formulate an energy function $E(\underline{V})$ such that the minimization of E over $\underline{V}$ corresponds to solving the shortest path problem. Most of the approaches in this branch are based on the Hopfield and Tank's neural network model [45] of solving discrete combinatorial optimization problems. Hopfield and Tank demonstrated the computational power of their network by applying their model to the Traveling Salesman Problem (TSP). Since then many researchers have attempted to apply the Hopfield model to solve other types of combinatorial optimization problem. Not surprisingly the Hopfield model was becoming the main stream in this shortest path problem. Many previous researches are based on this method : [12, 23, 46, 14, 43] and with a few appearances on the IEEE Transactions : [20] and [1]. In [7] it was further developed to solve the Constrained Steiner Tree (CST) multicast routing problem. The original Hopfield NN is a fully connected recurrent neural network. A typical sigmoid function is used for the transfer function of a single neuron :

$$V_i = g_i(U_i) = \frac{1}{1 + e^{-\lambda_i U_i}} \tag{3.4}$$

where U_i and V_i are the input and output to the ith neuron respectively.

The energy function is defined as :

$$E(\vec{V}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} I_iV_i. \tag{3.5}$$

where T_{ij} is known as the connection matrix of the network, i.e. the link weght from neuron i to neuron j. I_i is the bias or threshold of the ith neuron. The

dynamics of the ith neuron are described by :

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i} \tag{3.6}$$

where τ is the circuit's time constant.

Model 1

When mapping this model to the shortest path problem for a N-communication node network, a commonly used approach was using a NxN Hopfield network to find a single source-destination path. Generally, there are two ways of knowledge representation of a 'path'. The first is based on Hopfield & Tank's work on the TSP problem : A path is denoted by a matrix V_{xi}, where : $V_{xi} = 1$ if communication node x is the ith node to be visited in this source-destination path. $V_{xi} = 0$ otherwise. An example of this representation is as follows :

Suppose we find a path from source node 1 to destination node 2 on a 4-node communication network, the path found is 1-4-2. We could use the following matrix to represent this path :

node	1	2	3	4
1	1	0	0	0
2	0	0	1	0
3	0	0	0	0
4	0	1	0	0

Note that the matrix representation of a path by this approach is not unique, since we are allowed to use 1-3-3-3-3, 1-1-1-3-3, 1-1-1-1-3, or etc. to represent the path 1-3. Thus a hop is choosen whenever there is a transition between V_{yi} and $V_{y,i+1}$, Let C_{ij} be the cost for the communication link from node i to j.

For non-exist links, a very large cost is assigned. Thompopoules et. al. [43] formulated the energy function as :

$$E = \frac{A}{2}\sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n} V_{ik}C_{ij}V_{j,k+1} + \frac{B}{2}\sum_{k=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} V_{ik}V_{jk} + \frac{C}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} V_{ij} - n\right)^2 \quad (3.7)$$

In this energy function, this first A term corresponds to the total cost of the path. The B and C terms are introduced to force the energy function to converge to a valid path. The B term is to force each column contains at most a single 1, while the C term enforces that there will be exactly n 1's in the final solution. However, Ali and Kamoun [1] criticize this model to have the following weakness :

1. It requires a prior knowledge of the number of nodes forming the shortest path. A conservative approach is to set this number equals to the total number of the nodes, which is the maximum number of nodes that the shortest path may consist.

2. Since each neuron belonging to the first and last column has a fixed output voltage, the neural network is then designed to find the shortest path between only a given source-destination pair. The neural configuration has to be changed in order to find the shortest between other pairs.

3. When mapping this model into the Hopfield circuit, the link weight would correspond to the resistance of the synaptic connections. In a dynamic communication network, these resistance are to be changed continuously in order to adapt to the changes in link cost.

Model 2

On the other side, Ali and Kamoun formulates another model to solve the shortest path problem : They modify the path representation to : $V_{xi} = 1$ if the link from node x to i is in the path, and $V_{xi} = 0$ otherwise. Their energy function is given by :

$$\begin{aligned} E &= \frac{\mu_1}{2}\sum_{x=1}^{n}\sum_{\substack{i=1\\ i\neq x\\ (x,i)\neq(d,s)}}^{n} C_{xi}V_{xi} + \frac{\mu_2}{2}\sum_{x=1}^{n}\sum_{\substack{i=1\\ i\neq x\\ (x,i)\neq(d,s)}}^{n} \rho_{xi}V_{xi} \\ &+\frac{\mu_3}{2}\sum_{x=1}^{n}\left(\sum_{i=1,i\neq x}^{n} V_{xi} - \sum_{i=1,i\neq x}^{n} V_{ix}\right)^2 + \frac{\mu_4}{2}\sum_{i=1}^{n}\sum_{x=1,x\neq 1}^{n} V_{xi}(1-V_{xi}) + \frac{\mu_5}{2}(1-V_{ds}) \end{aligned} \tag{3.8}$$

where s and d denote the source and destination node respectively.

In the above energy function, the μ_1 term minimizes the total cost of a path by taking into account the cost of existing links. The μ_2 term is for removing the non-existent links from the solution. The μ_3 term ensures that the number of incoming links to a node is equal to the number of outgoing links. The μ_4 term pushes the state of the neural network to converge to one of the 2^{n^2-n} corners of the solution hypercube, defined by $V_{xi} \in \{0, 1\}$. The μ_5 term makes the final solution to contain the link (virtual or real) from d to s and therefore both the source and destination would be in the solution path. Note that the final solution would always be a loop, starting from s, goes through some intermediate nodes if any, then reach destination d and a single (virtual or real) link back to s. The dynamic equations are given by :

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - \frac{\partial E}{\partial V_{xi}}$$

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - \frac{\mu_2}{2}C_{xi}(1-\delta_{xd}\delta_{is}) - \frac{\mu_2}{2}\rho_{xi}(1-\delta_{xd}\delta_{is}) - \mu_3 \sum_{y=1,y\neq x}^{n}(V_{xy} - Vyx)$$
$$+\mu_3 \sum_{y=1,y\neq i}^{n}(V_{iy} - V_{yi}) - \frac{\mu_4}{2}(1-2V_{xi}) + \frac{\mu_5}{2}\delta_{xd}\delta_{is}$$

$$\forall(x,i) \in \bar{N} \times \bar{N} \; and \; x \neq i \tag{3.9}$$

Simulation Results For The Two Models

The first model claims to converge to either optimal or valid paths when applying to a 9 and 16 node networks. While the second model claims to find all optimal paths after 3000 to 8000 iterations and 100 runs with different link weights for a 5 node network. Besides, two individual examples were included : i) For a 5 node network it takes 38 iterations to converge to the optimal. ii) For a 15 node network it takes 91 iterations to converge to the optimal.

Other Approaches

There had been many attempts worked on this branch most of them are either modifying the above energy functions or the dynamical equations. In [27] the mean-field annealing method was applied to minimize the energy function. They said that the mean-field annealing method could prevent local minima trapping, which is a common problem with the Hopfield model. Furthermore in [39, 26], a VLSI implementation of the Hopfiled based routing model was proposed.

Remarks

1. Based on simulation results, Tsai [46] found that the convergent complexity of his/her model is independent of the number of nodes in the communication network.

2. A Note on the above approach is that the setting of the coefficient parameters A, B, C in model 1 or the μ's in model 2 are very critical for the algorithm to converge. Ali and Kamoun had analyse this issue and given some guidance on choosing those parameters in their model [1]. Furthermore, Aiyer et. al. [41] applied the eigenvector analysis on the original Hopfield Network and gave a more analytical way of choosing those parameters.

3. There are some critique on the Hopfield approach by Wang [6] that the approach does not guarantee valid solutions especially when applied to large systems.

3.2.3 Supervised Learning (SL)

The motivation under this branch is probably due to the many success of applying neural networks in the field of pattern recognition. Supervised learning means to provide a set of solved samples to the neural net for training. Then after this training phase, due to the interpolation and prediction property of neural networks, the NN should give a reasonable and hopefully intelligent response to samples that was not encountered before. Generally, the multi-layer feed-forward network (MLFN) is most commonly used in supervised training. Collett and Pedrycz [8] used an 8 layer MLFN trained with BP algorithm to

tackle this problem and a 99 % accuracy was found for a 16 node network. However, it seemed that they were just considering the static routing problem. On the other hand, Cavalieri et al [33] applied the CP algorithm in solving a 10 node network that could adapt to different routing topologies. Their results were very close to the optimal one. Their way of tackling this problem is very similar to the pattern recognition approach, in which each communication node chooses the next hop successor according to the link states of the whole communication networks in a distributed fashion. One may consider the "link states" as the patterns and the next hop successor as the pattern identifier, then it is exactly the pattern recognition / classification problem.

Apart from the above SP mainstream, Frisiani et al [15] combined team theory and BP on getting the optimal routing strategies.

3.3 Discussions

Obviously, among the three major streams, only the Hopfield/Energy Minimization stream was based on strong analytic foundations. For the JEB and most of the SL streams, a rather qualitative and experimental approach was adapted. However, the idea of the SL is very similar to the pattern recognition or classification problem which had already shown many successes with neural networks. Therefore, at this point we would continue with the HEM and the SL stream. We will continue to explore these streams, as our problem of multiple metrics routing has not yet been tackled. Thus, in the next two chapters, two algorithm based on these two streams would be proposed.

Chapter 4

Route Selection as "Link-state" Classification

A wide range of problems in signal processing belongs to the class of classification or recognition. In this thesis, we consider "recognition" and "classification" as the same word, so does "signal" and "pattern". Neural networks had been applied to this field for quite a long period and many architectural and training strategies had been developed. PNN and MLFN are the popular choices that had widely been applied to this problem. The two books by Timothy Masters [25, 24] are devoted to them. The advantage of MLFN over PNN is that, its execution speed is among the fastest of all known NN models. Thus MLFN is a strong candidate for solving the problem, because the speed of making routing decisions greatly affects the router's throughput. In this chapter we propose a MLFN based neural route selector. We begin with an overview of the MLFN, then we define the fuzzy utility function we used. Finally we discuss about the proposed NN architecture and its performance is verified with simulation results

and compared with previous works.

4.1 Multi-Layer Feedforward Network (MLFN)

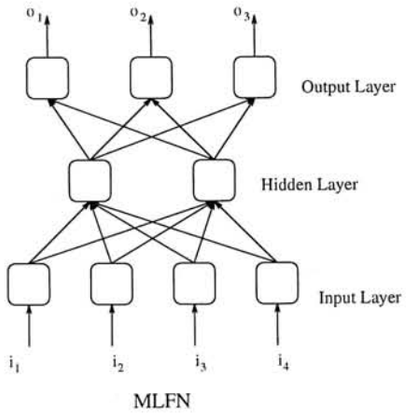


Figure 4.1: MLFN

MLFN is a typical non-recurrent neural network with a clear layered structure. The connections between the layers are fully connected. The layer that accepts the external inputs is known as the input layer, while the layer that produces the outputs is the output layer. All intermediate layers are known as hidden layers. Figure 4.1 shows a typical MLFN with 1 input layer, 1 output layer and 1 hidden layer. The number of neuron units could be different for different layers. The whole NN can then be viewed as an MIMO (Multi-input, Multi-output) system.

Considering a M layer MLFN, where layer 0 is the input layer and layer M is the output layer. Denote the system input vector space $\underline{u} = [u_1; u_2; \ldots; u_N]$ system output vector space $\underline{o} = [o_1; o_2; \ldots; o_3]$, weight matrices $w_{ij}^{(l)}$ where it connects from unit i of layer $l-1$ to unit j of layer l. Furthermore we denote the output matrix $v_i^{(l)}$ as the output of the unit i in layer l (Note : $v_i^{(M)} =$

o_i, $i = 1, 2, \ldots, N$), and the bias matrix $\theta_i^{(l)}$ denoting the bias of unit i in layer l. We have the following system equations :

$$o_j = f\left(\sum_i w_{ij}^{(M-1)} v_i^{(M-1)} - \theta_j^{(M)}\right) \tag{4.1}$$

$$v_j^{(l)} = f\left(\textstyle\sum_i w_{ij}^{(l-1)} v_i^{(l-1)} - \theta_j^{(l)}\right), \quad l = 1, 2, \ldots, M-1. \tag{4.2}$$

$$v_j^{(0)} = u_i \tag{4.3}$$

where f is the sigmoid function or sign function given by Eq. 3.1 and Eq. 3.3 respectively. From the above 3 equations, we could formulate the system equation as :

$$\underline{o} = g(\underline{i}, \underline{w}, \underline{\theta}) \tag{4.4}$$

Thus, the whole system is characterized by the input vector $\underline{i}$, the weight matrices $\underline{w}$ and the bias matrix $\underline{\theta}$. If we denote the desire output vector by $\underline{d}$, then training means to applying dynamic equations to $\underline{w}$ and $\underline{\theta}$, such that the mean-square error (MSE) of $\underline{o}$ and $\underline{d}$ would converge to the minimum. The detail of the training algorithm would be discussed in the next section. But before going into that, let us discuss the approximation properties of the MLFN.

4.1.1 Function Approximation Power of MLFN

Typically, signal classification by MLFN is usually done in the following manner :

1. The signal to be classified is encoded to an n-dimensional vector $\underline{x}$ and fed into the inputs of the MLFN

2. Each class of the signal would be encoded to a binary m-dimensional vector $\underline{y}$ and the number of outputs of the MLFN is m. Thus, only one output port is allowed to be turned on at the same time.

3. A successful classification of an input signal is represented by turning on the corresponding output port.

From the above, we know that signal classification is actually a function mapping problem such that we have to determine $f : \underline{x} \longrightarrow \underline{y}, \underline{x} \in R^n, \underline{y} \in R^m$. Thus, the approximation power of a MLFN is very important factor affecting its suitability for the signal classification problem.

Theorem 4.1: *Kolmogorov's Mapping Neural Network Existence Theorem* : Given any continous function $f : [0,1]^n \longrightarrow R^m, f(\underline{x}) = \underline{y}, f$ can be implemented exactly by a three-layer feedforward neural network having n fanout processing units in the input layer, $(2n+1)$ processing units in the hidden layer and m processing units in the output layer.

Although this is a very important theorem, it is strictly an existence theorem. No clue on the form of the neuron transfer function is given. But thanks to Hecht-Nielsen [17], who established another theorem with the sigmoid transfer function MLFN:

Theorem 4.2 : Given any $\epsilon > 0$ and any square-integrable function $f : [0,1]^n \longrightarrow R^m$ (i.e. $\int_{[o,1]^n} |g(\underline{x})|^2 d\underline{x}$ exists) , there exists a three-layer MLFN with the sigmoid transfer function (Eq.3.1) that can approximate f to within ϵ mean squared error accuracy.

Ensured by the above theorems, we can established that the MLFN with sigmoid theorem can closely approximate any arbitrary linear or non-linear decision surface provided the number of hidden layers and hidden units are sufficient.

4.1.2 Choosing MLFN parameters

The major parameters affecting the performance of a MLFN are the number of hidden layers, the number of hidden units and the training method. Winston [53] consider that choosing these parameters and training the MLFN is an art. Even until recently, analytical suggestion on these topics is still unavailable, thus we could only trust the experts' advice – Masters, in [24] gives some basic guidelines and they are summarized as follows :

1. **Number of Hidden Layers** : Generally, only one hidden Layer is enough. There are only a few rare situations in which two hidden layers may be preferable to one. More than two hidden layers are never theoretically needed.(Masters said that he had never seen a real-life problem in which more than two are needed.) This fact is also implied by Weiss and Kulikowski's book [40] that additional layers (more than 2) do not add any representational power to the discrimination.

2. **Number of Hidden Units Per Layer** : Use as few hidden neurons as possible. Start out with just two then train and test the network. When required, add only one more neuron at a time. Accord to Masters' experience, he had never saw practical networks with more than 10 hidden units and usually about 3 to 6 are optimal. The problem of too many hidden units was further discussed in Winston's book [53] that too much

hidden units lead to "overfitting" and the NN loses its predictive power.

4.1.3 Traning a MLFN

Problem of Overtraining

Now we come to another difficult problem about training a MLFN. In the human sense, training makes perfect, however, this is not the case of training a MFLN. Overtraining an MLFN would make it fits well with the training samples supplied but lose its predictive power. To detect overtraining, a usual practice is to construct a training test set, which is used as an indicator for training termination. Ideally, the training test set should be sufficiently comprehensive so that if the network performs well on it then the ultimate problem could be considered solved. A typical training curve is given below. Finally, a validation set is also needed to test the final performance of the trained network.

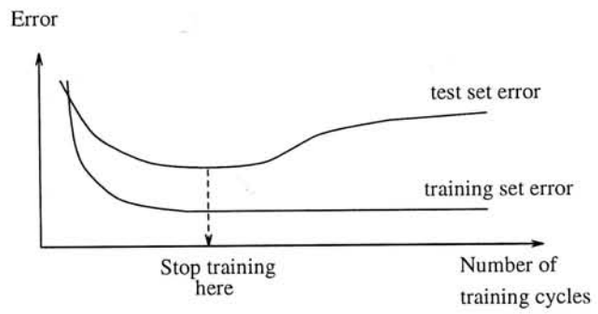


Figure 4.2: Training set error vs. training test set error

Deterministic or Stochastic Methods

Another problem concerns with training a MLFN is on choosing the training method. Deterministic Optimization methods have faster convergence speed

but possibly could be trapped by local minima. The only way to avoid a local minimum is to start with another initial state. On the other side, Stochastic methods are more likely to find the global minimum, but at a cost of slower convergence speed. Examples of deterministic optimization algorithms are, the classic BackPropagation and its derivatives, Conjugate Gradient Methods and Levenberg-Marquardt Learning. Examples of Stochastic methods are Simulated Annealing Methods and Monte-Carlo algorithm. There are some hybrid approaches given in [25] which should take up the advantages of the above two methods.

4.2 The Utility Function

The utility function used is a fuzzy utility function as shown in Fig. 4.6. This fuzzy mapping function is produced by the following fuzzy rule-base : (The reader may refer to [48] & [34] for information on fuzzy logic)

Delay \ *Price*	**Very Small**	**Small**	**Medium**	**Large**	**Very Large**
Very Small	Superb	Superb	Very Good	Good	Average
Small	Superb	Very Good	Good	Above Ave.	Poor
Medium	Very Good	Good	Average	Poor	Bad
Large	Good	Above Ave.	Poor	Very Bad	Terrible
Very Large	Above Ave.	Average	Bad	Terrible	Terrible

The normalized membership functions for the linguistic variables are shown in the figures 4.3 to 4.5.

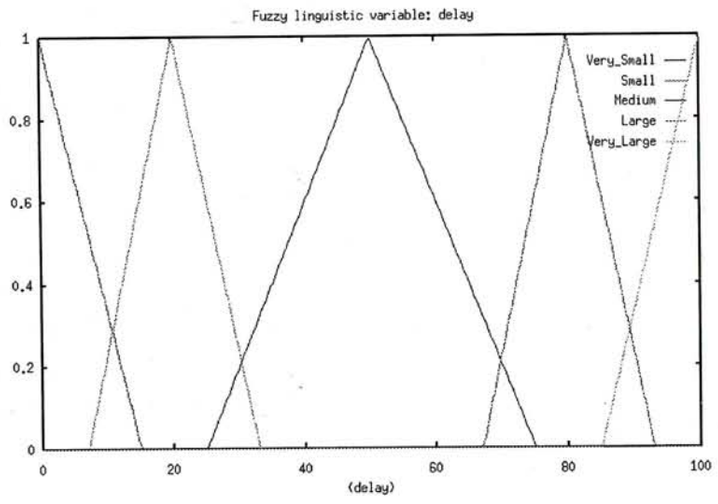


Figure 4.3: Membership functions for "Delay"

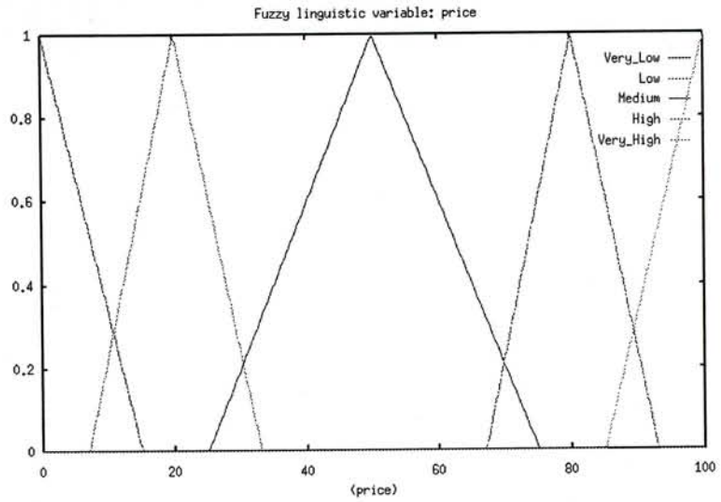


Figure 4.4: Membership functions for "Price"

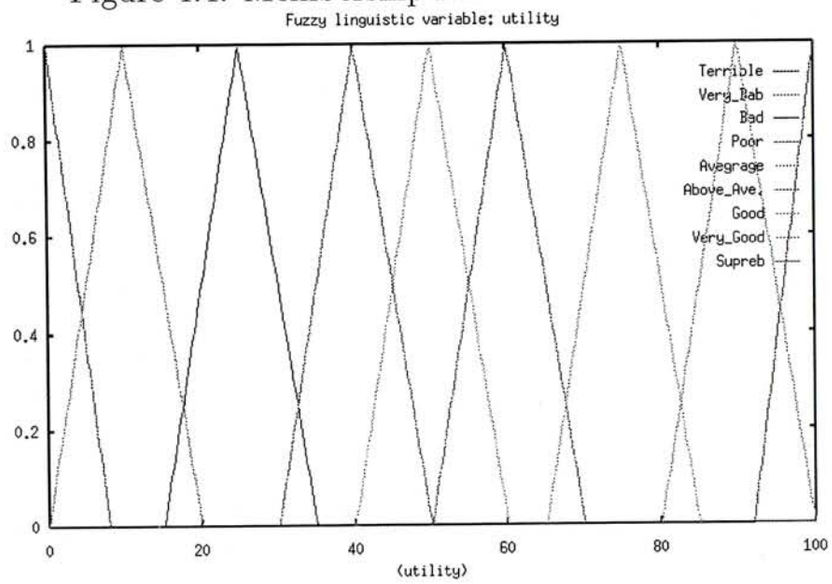


Figure 4.5: Membership functions for "Utility"

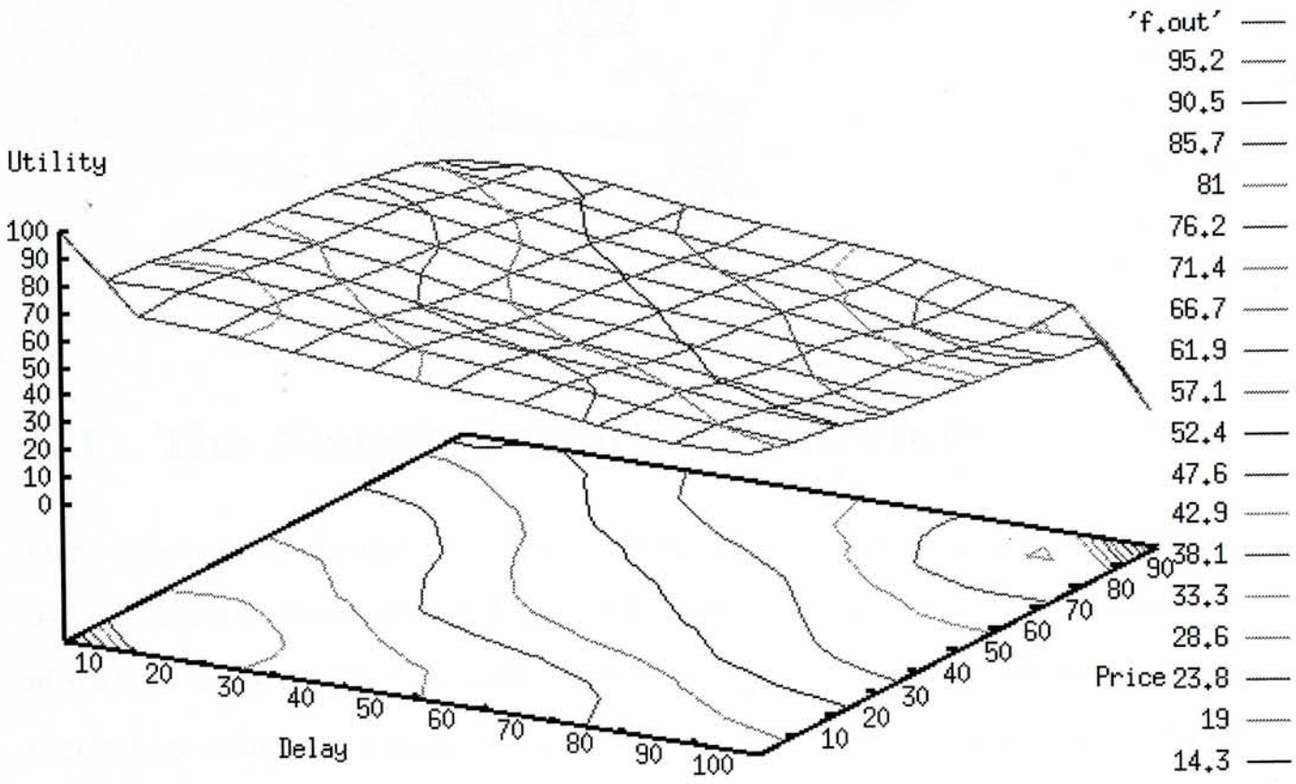


Figure 4.6: A utility function surface with two metrics : Delay & Price

Using mini-inference rule, center average defuzzifier and the final output surface was given Fig 4.6. As shown in the figure, this function is really complex. There are local minia, basins as well as uneven slopes. We have choose this special function in order to demonstrate our NN controller's ability to tackle arbitrary functions.

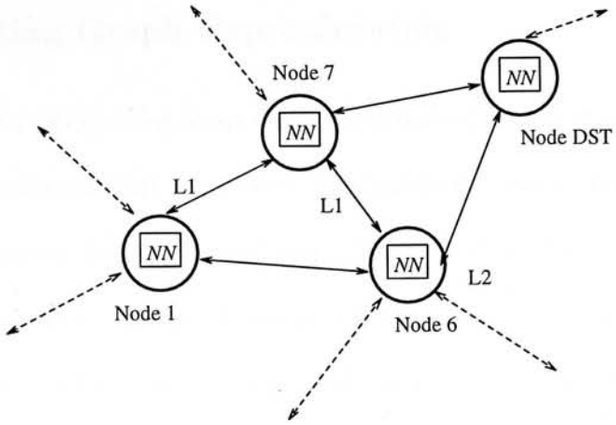


Figure 4.7: The Architecture

4.3 The Neural Network Architecture

Our proposed routing algorithm is a distributed , source specific, link-state routing algorithm as illustrated in Figure 4.3. Each node specifies the whole path of packets originated at that node. The source routing feature could be implemented by either the virtual circuits (VC) method or putting the entire path information on the packets. Our method of distributed routing is superior to [33] in that, loops or degenerated paths can be detected without any increase in the bandwidth for link-state distributions.

4.3.1 Routing Graph Representation with Successor Sequence Table (SST)

Our goal is to generate a routing graph rooted at the source node and terminates at every other node. We call it a graph rather than a tree. It is because, the paths to the destinations may cross one another and it is better to use the more general term - graph.

Traditional Routing Graph Representation

In a series of papers concerning loop-free distributed routing algorithms [30],[29],[16], the entire paths information to every destinations were encoded so that the source node can detect loops formation. They use a very clever method such that, for each destination node, instead of keeping a sequence of nodes to be visited, they only record the successor node and the second-to-last node (or the predecessor to the specify destination). This is equivalent of keeping a routing table as shown below :

Destination	Successor	Second-to-last
1	S_1	P_1
2	S_2	P_2
⋮	⋮	⋮
N	S_N	P_N

The idea of this representation is that if the path a to i is optimal, then the path a to j, where j is a one hop neighbor of i, is among the optimal set of paths. Which is just the essential assumption of the Dijkstra and Bellman-Ford routing algorithm. More specifically, let us now review the Bellman-Ford's equation :

Let D_{ij} represent the distance of the minimum-distance route form source node i to destination j, and d_{pq} be the distance of a direct link between node p and q. Assuming the link distance are additive, the shortest path between i and j can be obtained by solving Bellman's Equation :

$$D_{ii} = 0, \forall i \tag{4.5}$$

$$D_{ij} = \min_k (d_{ik} + D_{kj}), \forall i \neq j \tag{4.6}$$

When mapping to the multi-metrics utilization optimization routing, a critical question is whether this equation is still valid. Generalizing Eq 4.6 to the problem by denoting $U(.)$ as the arbitrary utility function and representing D and d by vectors, we get the following proposition :

Proposition 4.1 :

$$U(\underline{D}_{ij}) = \max_k [U(\underline{d}_{ik} + \underline{D}_{kj})], \forall i \neq j$$

Assume (4.7) is true. Then consider the following scenario :

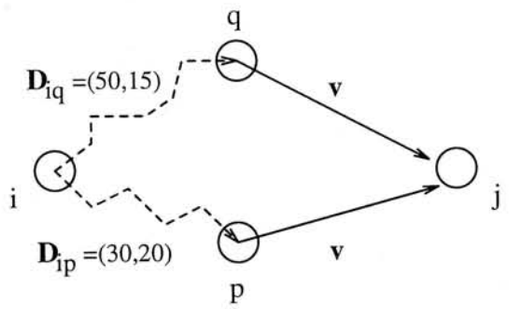


Figure 4.8:

Suppose $U(\underline{D}_{ip}) > U(\underline{D}_{iq})$, by proposition 4.1, we will find that $U(\underline{D}_{ij}) = U(\underline{v} + \underline{D}_{ip})$ and the optimal route should be $i - p - j$. However, consider the utility function given in Fig. 4.6 :

Suppose $\underline{D}_{ip} = (30, 20)$ and $\underline{D}_{iq} = (50, 15)$ such that $U(30, 20) > U(50, 15)$. Further, let $\underline{v} = (10, 10)$. From figure 4.6 $U(\underline{D}_{ip} + \underline{v}) = U(50, 30) < U(60, 25) = U(\underline{D}_{iq} + \underline{v})$ which implies that $U(\underline{D}_{ij}) = U(\underline{D}_{iq} + \underline{v}) = U(60, 25)$ which is a contradiction of Proposition 4.1. Therefore the above successor & second-to-last-hop representation is not sufficient for the model.

In fact, this representation is only good for monotonous utility functions, or more specifically, if $U(\underline{u}_1) > U(\underline{u}_2)$ then $U(\underline{u}_1 + \underline{v}) > U(\underline{u}_1 + \underline{v})$. Moreover, if

the problem can be mapped by this approach, we could use the Bellman-Ford, Dijkstra or other shortest path based algorithm for the problem and it is not NP-Complete at all. In this approach the outcome is restricted to a tree not a graph.

Solution Representation Using the SST

As the above representation fails, another solution representation is adopted. The method is by specifying a successor sequence table (SST). The size of the SST is N-by-N with N being the number of nodes on the communication network. Denote each SST entry by $SST_{ij}, i = 0, 1, \cdots, (N-1); j = 0, 1, \cdots, (N-1)$. The entry $SST_{pq} = m$ means that the next hop successor for node p to destination q is m. We will show in the following example, how the SST is sufficient to represent the routing graph.

Example

Consider the following 6 node network :

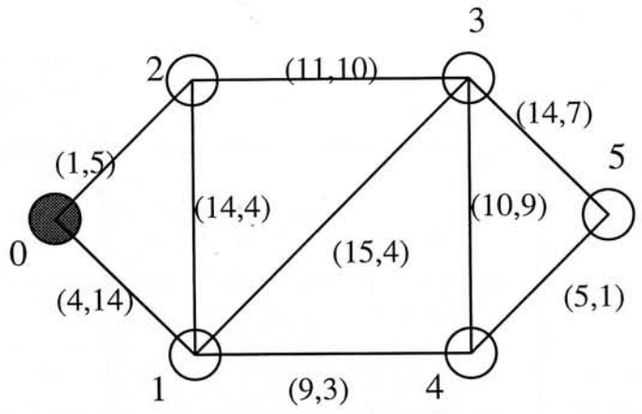


Figure 4.9: Example network

We use the same utility function as shown in Figure 4.7 and find out the following optimal paths :

0 - 1	0 - 2	0 - 2 - 3
0 - 2 - 1 - 4	0 - 1 - 4 - 5	

Note that the route 0 - 2 - 1 - 4 is a violation of the basic Bellmam-Ford/Dijkstra assumption. If we draw the routing graph, we have the following :

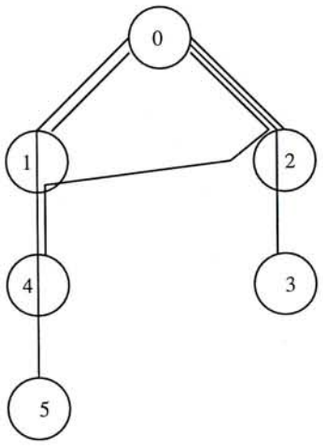


Figure 4.10: The Routing Graph

The above routing graph could be represented by the following SST :

Starting Node \ Destination Node	0	1	2	3	4	5
0		1	2	2	2	1
1					4	4
2				3	1	
3						
4						5
5						

Intermediate Successor Nodes

The way to understand the above SST is as follows :

1. Our source node is 0, thus we first refer to the first row ($SST_{0x}, x = 0, 1, 2, \cdots, 5$).

2. Suppose we want to know the path from 0 to 1, we then look at the second column ($SST_{x1}, x = 0, 1, 2, \cdots, 5$) and find SST_{01} to be 1, which means that the succesor from 0 to destination 1 for node 0 is 1. Thus we can tell that this path is 0 -1.

3. Similarly, we can tell the path from 0 to 2 is 0 - 2.

4. For the path 0 to 3, we refer to the column $SST_{x3}, x = 0, 1, 2, \cdots, 5$. We find the successor from 0 to destination 3 for node 0 is 2. Now our starting node is 2, so we look at SST_{23} and find out that the successor form 2 to destination 3 is 3. Finally, we have the whole path 0 - 2 - 3.

5. For the path 0 to 4, again we refer to the column $SST_{x4}, x = 0, 1, 2, \cdots, 5$. The successor for 0 to 4 starting on 0 is found to be 2. Then we look at SST_{24} and find out that the successor for 0 to 4 starting on 2 is 1. So, look at SST_{14}, we have successor for 0 to 4 starting on 1 is 4. Finally we have to path 0 - 2 - 1 - 4.

6. The way we find out the final 0 to 4 path is similar to the above.

Through the above example we have observe that the routing graph may not look like a tree at all. It is a strange tree with some branches coming across others. Furthermore, we have explained how the routing graph is represented by the SST. In the following sections, we would discuss how this SST is represented

with the corresponding neural network architecture. Through out the following sections, we are using figure 4.9 as the example to illustrate the idea.

4.3.2 The Neural Network Layout

Let us start with the following example NN layout :

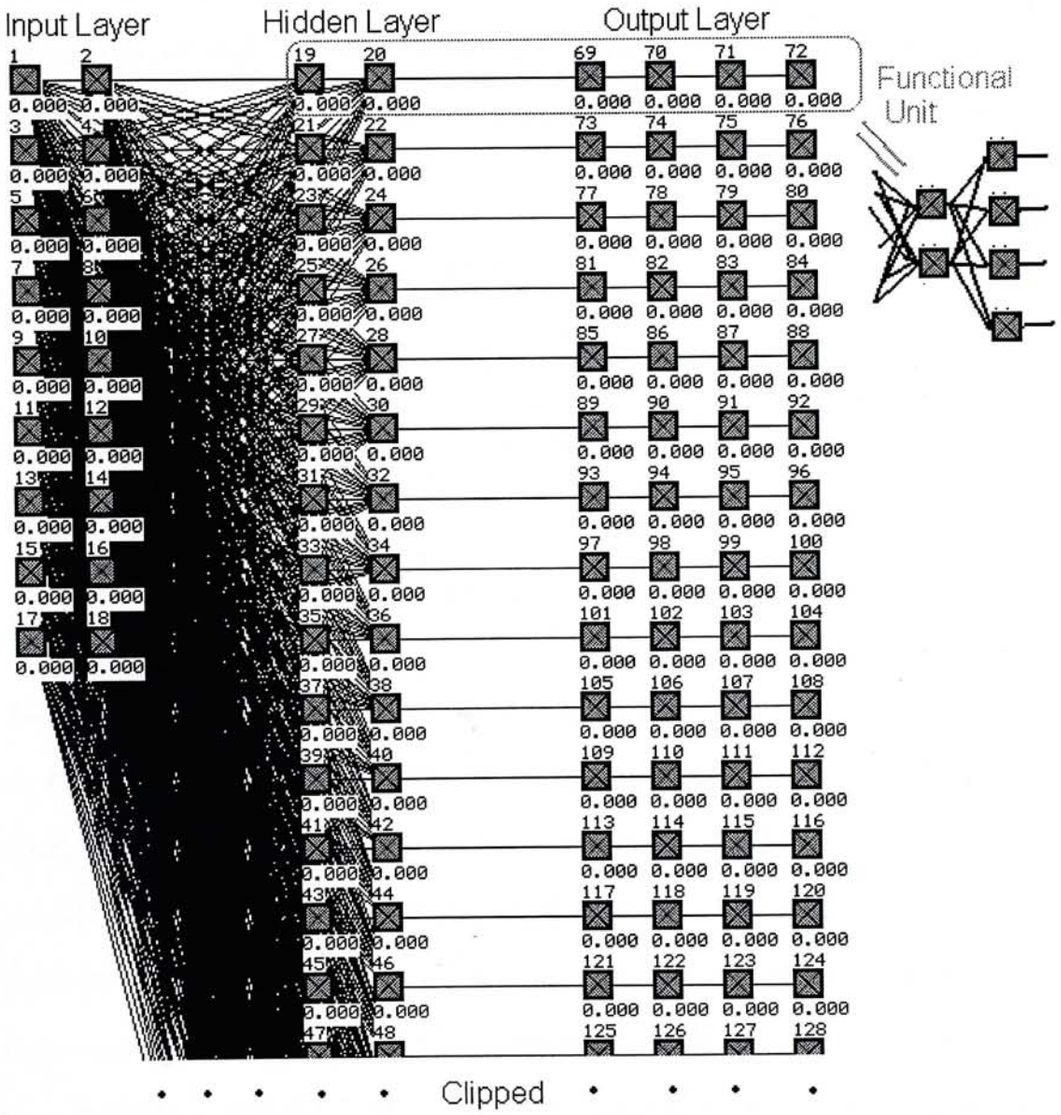


Figure 4.11: An Example Neural Layout

Referring to the above figure, there are three layers. Form the left to the right, they are the input layer, hidden layer and output layer respectively. The inputs are the link state information. In this example, there are a total of 9 links and each link has two states, namely, price and delay. Therefore, the number of input units is 18. The input layer and the hidden layer are fully connected, which means that each unit in the input layer is connected to every unit on the hidden layer and vice versa. For the connections between the hidden layer and the output layer, it can be decomposed into smaller units known as the functional units as shown below :

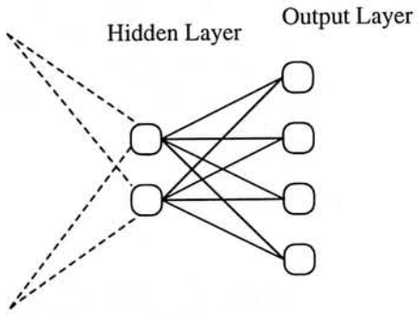


Figure 4.12: A Functional Unit

Each functional unit is made up of 2 hidden units and 4 output units which are fully connected to each other. The funtional unit encodes a particular intermediate successor node which corespond to an entry in the SST. The way to encode this information is illustrated in the figure 4.13. The width of the codewords is equal to the maximun number of neighbor of the nodes of the communication network. The possible codewords are in the following forms $:100\cdots0; 0100\cdots0; 0010\cdots0; \cdots; 000\cdots1$ and also $000\cdots0$ which means null successor. Each codeword specifies a node that is directly attached to a starting node.

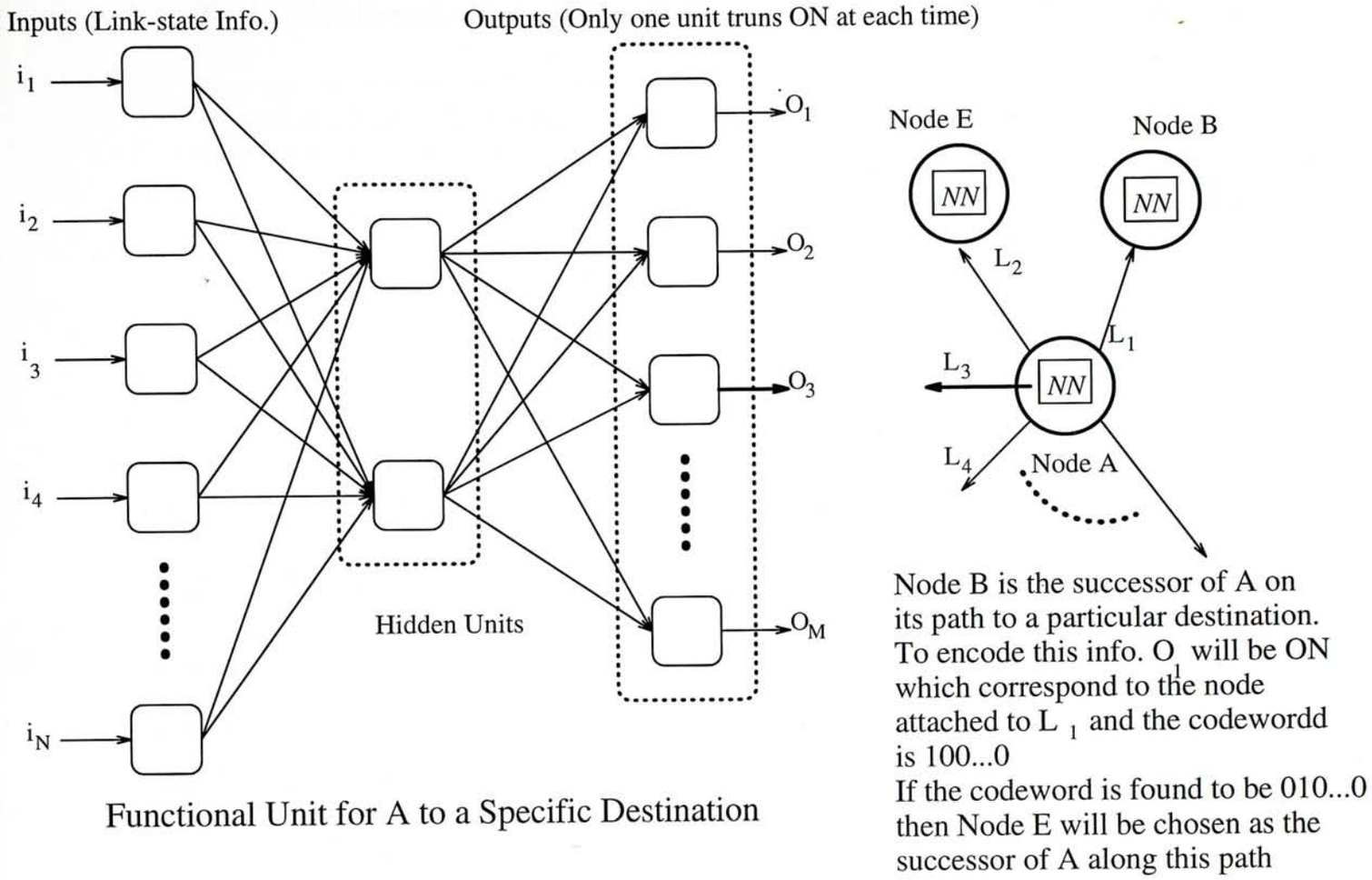


Figure 4.13: To Encode a SST entry by the Functional Unit

Generally, there will be at most of N^2 functional units (N is the number of communication node). Furthermore, the width of the codewords is at most $N-1$. Thus the total number of output neurons is about N^3. But, usually the number of neurons needed is far less than this number. Consider our example on figure 4.9, a 4-bit codeword width is enough and the corresponding codewords are 1000, 0100, 0010, 0001 and 0000 (means null successor).

To decode the codewords, there will be a lookup table like the one below:

Starting Node \ Codeword	1000	0100	0010	0001	0000
0	1	2	X	X	X
1	0	2	3	4	X
2	0	1	3	X	X
3	1	2	4	5	X
4	1	3	5	X	X
5	3	4	X	X	X

4.3.3 How the Neural Network Controller Works

Each functional unit is a MLFN which takes all the link state (LS) information into account and then decides the intermediate successor which correpond to an entry on the SST. The SST entry $SST_{ij} = y$ means to go form the source node to destination j starting form i is y. During training, different combinations of link-state information are the inputs, while the solved and encoded SST are the taught outputs. And through this training process, each individual functional unit will learn its way to react to different link-state inputs. During the actual operation, each functional unit will independently, produce the proper output according to their "experience" for a particular input link-state pattern. (This is why we call this approach the pattern/link-state classfication approach) Finally, grouping the functional units together, we have the whole picture of the SST. With this SST, we have the routing graph.

4.3.4 Training

The training algorithm chosen is the scaled-conjugated gradient (SCG) method [54] developed by Moller [28] in 1993. It is one of the fastest deterministic training algorithms for MLFN and there is no critical parameters affecting the convergence speed to be set. SCG had been shown to be considerably faster than the standard backpropagation and other conjugate gradient methods. The reader should refer to [28] for details of it. Besides, we use a route search engine written in C to generated solved examples (i.e. take the link state as the input, produce the SST as the teaching output) as a reference for training. Furthermore, we have adopted the training method in 4.1.3 to avoid over-training.

4.4 Simulation

The simulations were performed using a neural network simulation package - SNNS (Stuttgart Neural Network Simulator). It was developed by the Institute for Parallel and Distributed High Performance Systems, University of Stuttgart. It supports over 30 learning algorithms including the SCG algorithm.

4.4.1 Performance Parameters

We use the following parameters as measurement of the performance of the NN routing algorithms :

Percentage of Valid Paths : The NN controller sometimes may produce invalid paths. Invalid paths falls under the following categories :

1. The path does not end up with the desire destination.

2. Loops presence in the path.

3. The path includes a failed link.

Mean-Square-Error (MSE) & Root-MSE (RMSE) : This is the measurement of the MSE & RMSE between the output path and the teaching path utility values. Only classified paths would be counted in the calculation.

Mean-Absolute-Error (MAE) : This is the average deflection of the output paths and the teaching paths.

Variance for MAE : The variance of MAE is given by :

$$(MSE - MAE^2)^{\frac{1}{2}} \tag{4.7}$$

Number of 100% paths & the corresponding percentage : This represents the total number of optimal paths. For the calculation of the percentage, it is the fraction of optimal paths out of the total number of classified paths.

Number of sub-optimal (-/+5) paths & the corresponding percentage : This represents the number and fraction of sub-optimal paths with absolute difference < 5.0 (The utility is a value between 0 and 100). In calculating the fraction, again only the classified paths will be counted.

4.4.2 Simulation Results

For simplicity, we assume the link metrics are the same in both directions. Furthermore, each metric is encoded into 16 levels, with 16 meaning the link is dead and 1 meaning the link is at excellent condition. A breath-first-search

algorithm written in C was used to generate examples for training the NN. Three network models were used for the simulations, they are shown in Figure 4.12.

The 16 levels were normalized into the range $(0, 1]$ by the equation below :

$$y = \begin{cases} x/20 & \text{if} \quad x \neq 16, \\ 1 & \text{if} \quad x = 16 \end{cases} \tag{4.8}$$

It is through this way that the effect of a dead link is magnified by the NN controller.

The simulations were divided into 2 phases, the training phase and the operation phase. During the training phase, randomly generated network patterns with solutions were provided and the NN is trained with the SCG algorithm. In the operation phase, randomly generated network patterns were used for testing the actual performance of the NN controller. We also perform simulations for cases of i) without any link failures, ii) with link failures. The Hop Unit parameters for the NN controller are 1 Hidden Layer with 2 Hidden Units. The network models used for the simulations are shown in Fig 4.12. Model (a) is 6-node-network, model (b) is a 10-node-network, while model (c) is a 19-node-network. The simulation details and results for each of the network models were as shown below. Graphical summary of some major performance parameters of the operational sets were presented on Fig 4.13 to 4.17.

Model (a) - (Without link failures)

Traning

1. Hop Unit parameters : 1 Hidden Layer with 2 Hidden Unit.

2. Patterns used for training = 200 (i.e. 1000 paths)

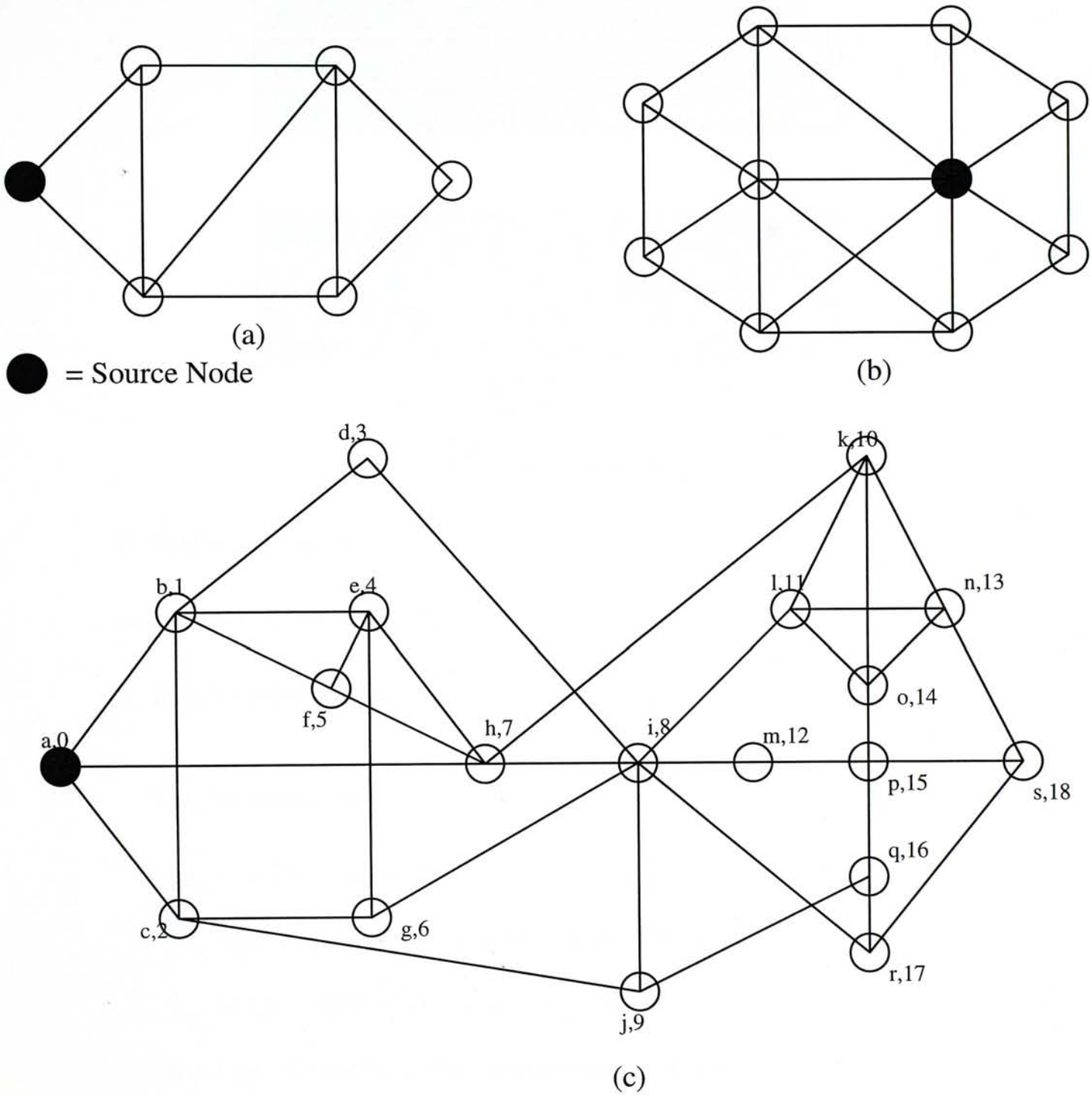


Figure 4.14: Network Models for Simulations

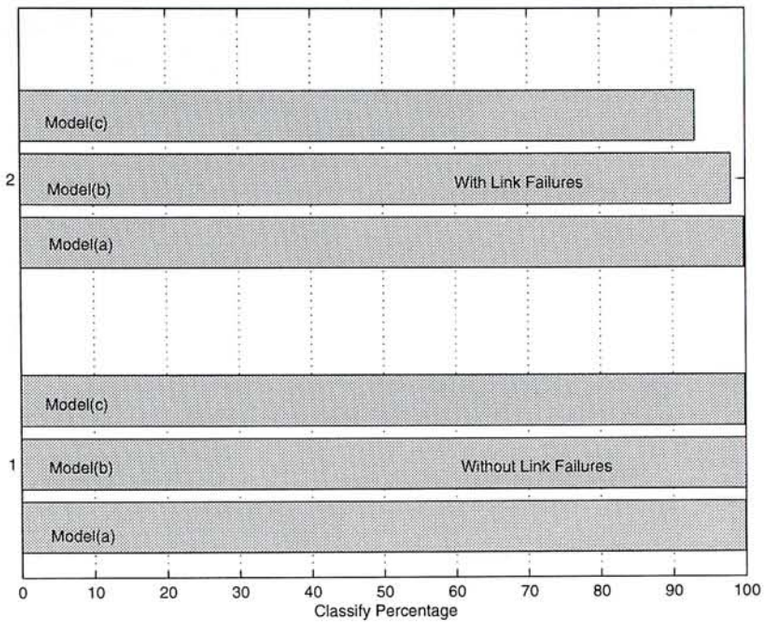


Figure 4.15: Classify Percentages

3. Patterns used for training test = 200

4. Number of iterations determined = 70

5. Results after training

For Training Set

(a) Classify = 1000, Unclassify = 0, Success = 100 %

(b) MSE for classified samples = 5.11 (RMSE = 2.26)

(c) MAE = 0.53 with variance = 2.20

(d) Optimal paths = 861 , percentage = 86.1%

(e) Sub-optimal paths = 959, percentage = 95.9%

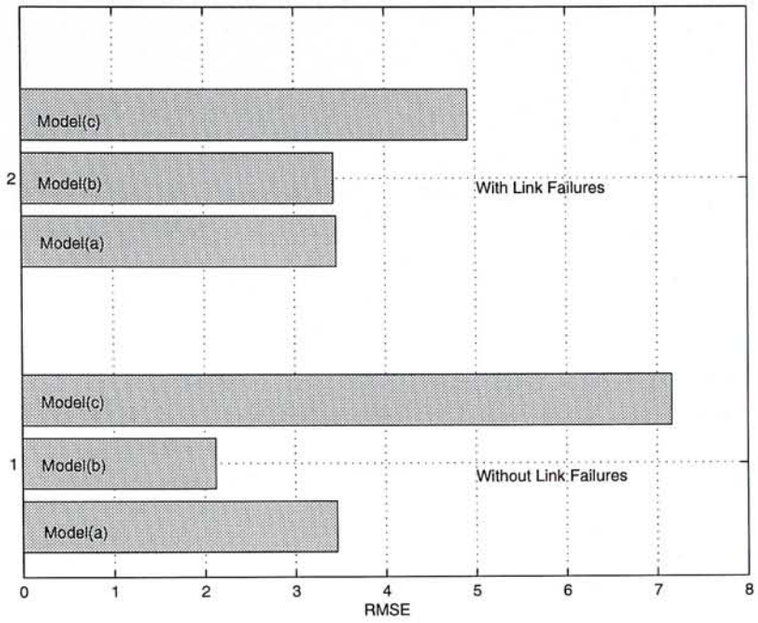


Figure 4.16: RMSE

For Training Test Set

(a) Classify = 1000, Unclassify = 0, Success = 100 %

(b) MSE for classified samples = 6.57 (RMSE = 2.56)

(c) MAE = 0.67 with variance = 2.47

(d) Optimal paths = 821 , percentage = 82.1%

(e) Sub-optimal paths = 953, percentage = 95.3%

Operation

1. Patterns used = 500 (i.e. 2500 paths)

2. Classify = 2498, Unclassify = 2, Success = 99.92 %

3. MSE for classified samples = 11.95 (RMSE = 3.46)

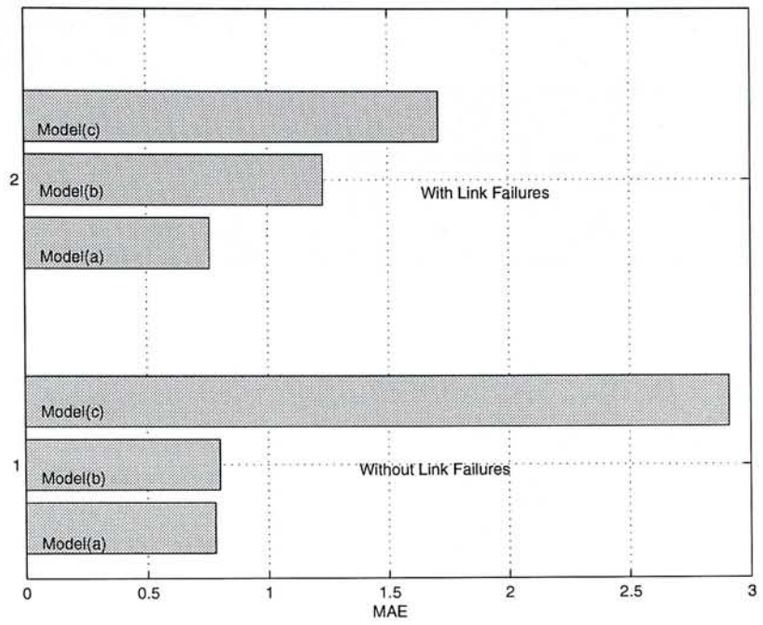


Figure 4.17: MAE

4. MAE = 0.78 with variance = 3.37

5. Optimal paths = 1999 , percentage = 80.02%

6. Sub-optimal paths = 2372, percentage = 94.96%

Model (a) - (With link failures)

Training

1. Hop Unit parameters : 1 Hidden Layer with 2 Hidden Unit.

2. Patterns used for training = 200 (i.e. 1000 paths)

3. Patterns used for training test = 200

4. Number of iterations determined = 80

5. Results after training

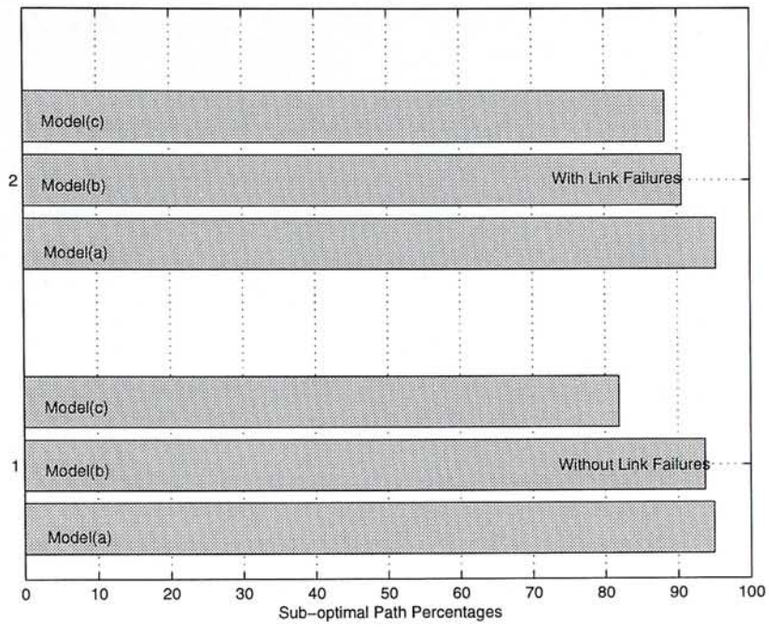


Figure 4.18: Sub-optimal Paths Percentages

For Training Set

(a) Classify = 1000, Unclassify = 0, Success = 100 %

(b) MSE for classified samples = 10.82 (RMSE = 3.29)

(c) MAE = 0.50 with variance = 3.25

(d) Optimal paths = 868 , percentage = 86.8%

(e) Sub-optimal paths = 974, percentage = 97.4%

For Training Test Set

(a) Classify = 999, Unclassify = 1, Success = 99.9 %

(b) MSE for classified samples = 6.808 (RMSE = 2.61)

(c) MAE = 0.72 with variance = 2.51

(d) Optimal paths = 819 , percentage = 81.98%

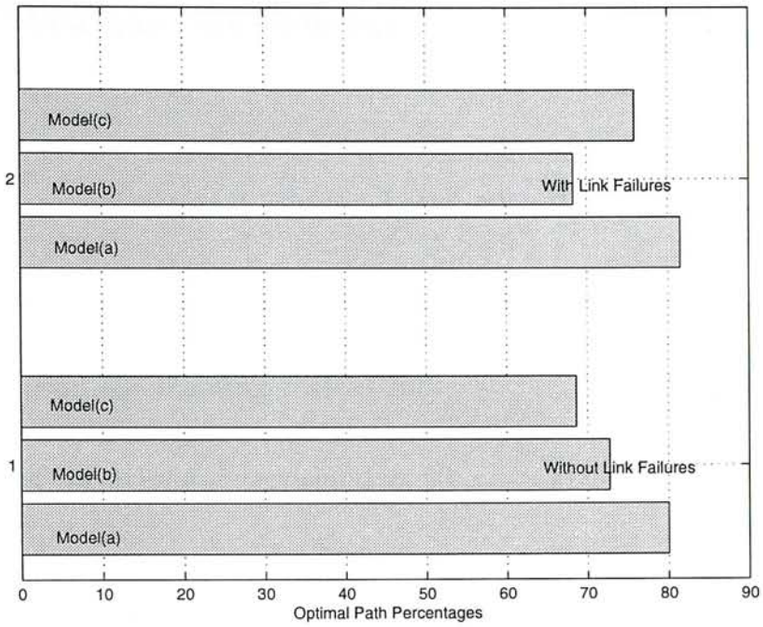


Figure 4.19: Optimal Paths Percentages

(e) Sub-optimal paths = 954, percentage = 95.5%

Operation

1. Patterns used = 500 (i.e. 2500 paths)
2. Classify = 2495, Unclassify = 5, Success = 99.80 %
3. MSE for classified samples = 11.99 (RMSE = 3.46)
4. MAE = 0.76 with variance = 3.38
5. Optimal paths = 2034 , percentage = 81.52%
6. Sub-optimal paths = 2377, percentage = 95.27%

Model (b) - (Without link failures)

Traning

1. Hop Unit parameters : 1 Hidden Layer with 2 Hidden Unit.
2. Patterns used for training = 200 (i.e. 1800 paths)
3. Patterns used for training test = 200
4. Number of iterations determined = 50
5. Results after training

   For Training Set

   (a) Classify = 1800, Unclassify = 0, Success = 100 %

   (b) MSE for classified samples = 2.52 (RMSE = 1.59)

   (c) MAE = 0.44 with variance = 1.53

   (d) Optimal paths = 1483 , percentage = 82.39%

   (e) Sub-optimal paths = 1753, percentage = 97.39%

   For Training Test Set

   (a) Classify = 1800, Unclassify = 0, Success = 100 %

   (b) MSE for classified samples = 4.50 (RMSE = 2.12)

   (c) MAE = 0.79 with variance = 1.97

   (d) Optimal paths = 1326 , percentage = 73.67%

   (e) Sub-optimal paths = 1629, percentage = 94.00%

Operation

1. Patterns used = 500 (i.e. 4500 paths)
2. Classify = 4500, Unclassify = 0, Success = 100.00 %
3. MSE for classified samples = 4.49 (RMSE = 2.12)
4. MAE = 0.80 with variance = 1.96
5. Optimal paths = 3273 , percentage = 72.73%
6. Sub-optimal paths = 4217, percentage = 93.71%

Model (b) - (With link failures)

Traning

1. Hop Unit parameters : 1 Hidden Layer with 2 Hidden Unit.
2. Patterns used for training = 300 (i.e. 2700 paths)
3. Patterns used for training test = 300
4. Number of iterations determined = 100
5. Results after training

   For Training Set

   (a) Classify = 2653, Unclassify = 47, Success = 98.26 %

   (b) MSE for classified samples = 12.90 (RMSE = 3.59)

   (c) MAE = 1.22 with variance = 3.38

(d) Optimal paths = 1909 , percentage = 71.96%

(e) Sub-optimal paths = 2404, percentage = 90.61%

For Training Test Set

(a) Classify = 2650, Unclassify = 50, Success = 98.15 %

(b) MSE for classified samples = 12.46 (RMSE = 3.53)

(c) MAE = 1.21 with variance = 3.32

(d) Optimal paths = 1857 , percentage = 70.08%

(e) Sub-optimal paths = 2410, percentage = 90.94%

Operation

1. Patterns used = 500 (i.e. 4500 paths)
2. Classify = 4410, Unclassify = 90, Success = 98.00 %
3. MSE for classified samples = 11.80 (RMSE = 3.43)
4. MAE = 1.23 with variance = 3.21
5. Optimal paths = 3013 , percentage = 68.32%
6. Sub-optimal paths = 3995, percentage = 90.59%

Model (c) - (Without link failures)

Traning

1. Hop Unit parameters : 1 Hidden Layer with 2 Hidden Unit.

2. Patterns used for training = 500 (i.e. 9000 paths)

3. Patterns used for training test = 500

4. Number of iterations determined = 90

5. Results after training

For Training Set

(a) Classify = 8994, Unclassify = 6, Success = 99.93 %

(b) MSE for classified samples = 55.68 (RMSE = 7.46)

(c) MAE = 3.02 with variance = 6.82

(d) Optimal paths = 6205 , percentage = 68.99%

(e) Sub-optimal paths = 7341, percentage = 81.62%

For Training Test Set

(a) Classify = 8986, Unclassify = 14, Success = 99.84 %

(b) MSE for classified samples = 55.51 (RMSE = 7.45)

(c) MAE = 3.03 with variance = 6.80

(d) Optimal paths = 6173 , percentage = 68.70%

(e) Sub-optimal paths = 7327, percentage = 81.54%

Operation

1. Patterns used = 700 (i.e. 12600 paths)

2. Classify = 12586, Unclassify = 14, Success = 99.89 %

3. MSE for classified samples = 51.32 (RMSE = 7.16)

4. MAE = 2.91 with variance = 6.55

5. Optimal paths = 8635 , percentage = 68.61%

6. Sub-optimal paths = 10306, percentage = 81.88%

Model (c) - (With link failures)

Traning

1. Hop Unit parameters : 1 Hidden Layer with 2 Hidden Unit.

2. Patterns used for training = 500 (i.e. 9000 paths)

3. Patterns used for training test = 500

4. Number of iterations determined = 150

5. Results after training

   For Training Set

   (a) Classify = 8401, Unclassify = 499, Success = 93.34 %

   (b) MSE for classified samples = 21.30 (RMSE = 4.62)

   (c) MAE = 1.52 with variance = 4.36

   (d) Optimal paths = 6499 , percentage = 77.36%

   (e) Sub-optimal paths = 7514, percentage = 89.44%

For Training Test Set

(a) Classify = 8421, Unclassify = 579, Success = 93.57 %

(b) MSE for classified samples = 23.36 (RMSE = 4.83)

(c) MAE = 1.62 with variance = 4.55

(d) Optimal paths = 6473 , percentage = 76.86%

(e) Sub-optimal paths = 7522, percentage = 89.32%

Operation

1. Patterns used = 700 (i.e. 12600 paths)
2. Classify = 11724, Unclassify = 876, Success = 93.05 %
3. MSE for classified samples = 24.22 (RMSE = 4.92)
4. MAE = 1.71 with variance = 4.61
5. Optimal paths = 8897 , percentage = 75.89%
6. Sub-optimal paths = 10349, percentage = 88.27%

4.5 Conclusions and Discussions

Form the simulation results, we observe much better percentages than any other works from the JEB branch. Besides, the JEB branch does not adapt well with link failures, the percentage of sub-optimal paths drop to around sixty. But the model still maintains an overall sub-optimal percentage of over eighty. Furthermore, we are solving the multiple metrics utility optimization routing,

but the JEB branch is only designed for single metric shortest path routing. Thus the result is significantly better than the JEB approach.

When compared to Cavalieri et al. [33]'s Counter-Propagation NN, the model has the following advantages: Firstly, the model is a source routing algorithm such that loops or invalid paths can be detected first, while [33] uses a distributed hop-by-hop routing strategy, thus some packets may get lost or trapped by loops. Secondly, the model produces a routing graph to every destination at a time, but in [33] the desired destination is also one of the inputs to the NN controller, which means that it only produces a next hop successor at a time. Consider the training aspect, one training sample in the model could provide information to a routing graph to every destinations, but in [33], one training sample could only provide the next-hop successor to a specific destination. Therefore, the training samples are much more "condensed" and the model requires much less samples for training the NN controller. Thirdly, again the problem we are solving is much more complex than in [33], where only the shortest path routing is considered.

Although we cannot achieve a 100 % results, which means that in actual practices, the algorithm had to be hybrid with a fallback routing strategy [51]. This fallback algorithm could be some heuristic algorithms like the one proposed in [18]. As only 10 % to 20 % of the workload requires the fallback algorithm to work, the algorithm is still efficient and reduces a lot of CPU resources. Furthermore, the NN controller once trained does not need any routing table at all – due to its 3 layered parallel structure, obtaining a route through direct computation is as fast as looking through a routing table from the memory. Finally, we observe that the iterations needed for training seemed to be linearly related, which means that algorithm is suitable for even larger networks.

However, one major weakness of supervised learning is that if the network is too large, it would be very difficult to generate the teaching samples. Thus the pattern recognition approach is superb for small or medium networks but not suitable for large networks (number of nodes > 100). For larger networks, the Hopfield approach should be more applicable, the author had developed the theories of applying the Hopfield NN to tackle this problem.

Chapter 5

Route Selection as Energy Minimization - A Theoretical Study

The idea of this energy minimization approach is to derive an enegry function, and then model a NN controller such that the dynamics of the NN controller would converge towards lower energy level and at last obtain the minimum at which the routing objective is achieved. The Hopfield/Tank NN model [45] is the most popular choice for this purpose. We have reviewed the history in Chapter 3, now let us take a more detailed look at the Hopfield/Tank NN model.

5.1 The Hopfield/Tank NN Model

The Hopfield NN is a kind of typical recurrent NN, it is a fully connected recurrent network with "self-loops" eliminated as shown below :

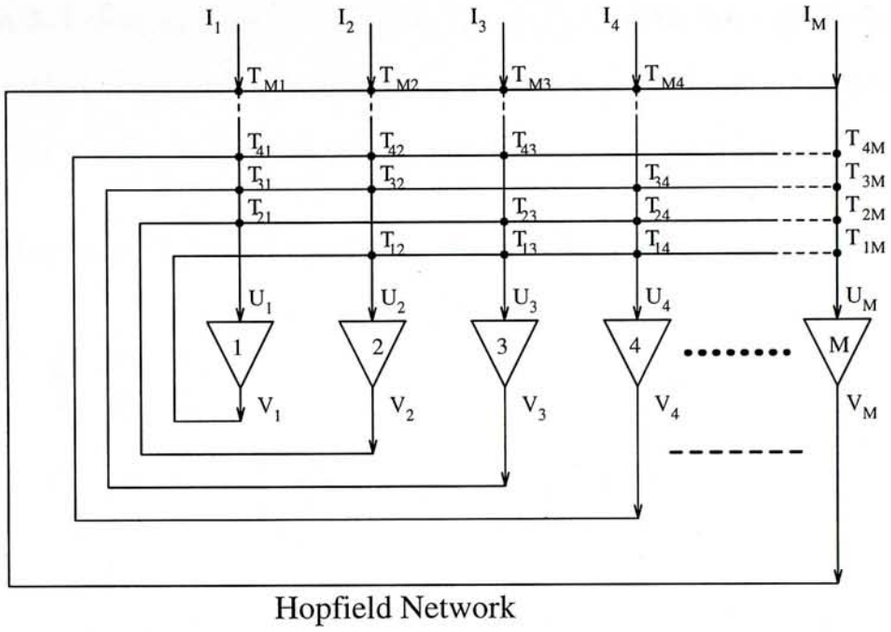


Figure 5.1: A Hopfield Network

The transfer function of neuron unit i is given by :

$$V_i = g_i(U_i) = \frac{1}{1 + e^{-\lambda_i U_i}} \tag{5.1}$$

where U_i and V_i is the input and output to the ith neuron respectively.

The energy funcion is defined as :

$$E(\underline{V}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} T_{ij}V_iV_j - \sum_{i=1}^{M} I_iV_i. \tag{5.2}$$

Where T_{ij} is known as the connection matrix of the network, i.e. the link weght from neuron i to neuron j. I_i is the bias of the ith neuron. The dynamics of the ith neuron are described by :

$$\frac{dU_i}{dt} = -\frac{\partial E}{\partial V_i} \tag{5.3}$$

$$= \sum_{j=1}^{M} T_{ij}V_j + I_i \tag{5.4}$$

τ is the circuit's time constant.

Theorem 5.1 For symmetric T_{ij} (i.e $T_{ij} = T_{ji}$) with zero diagonal, one can show that the system described by Eq 5.2 is a Lyapunov function (refer to [36] for a introduction of that), which decreases in times.

Proof : Our proof is based similar to the proof on [52]

$$\begin{aligned}
\frac{dE}{dt} &= \sum_{k=1}^{M} \frac{\partial E}{\partial V_k} \frac{dV_k}{dt} \\
&= \sum_{k=1}^{M} \frac{\partial}{\partial V_k} \left(-\frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} T_{ij} V_i V_j - \sum_{i=1}^{M} I_i V_i \right) \frac{dV_k}{dt} \\
&= \sum_{k=1}^{M} \left[\frac{\partial}{\partial V_k} \left(-\frac{1}{2} \sum_{i \neq k}^{M} T_{ik} V_i V_k - \frac{1}{2} \sum_{j \neq k}^{M} T_{kj} V_k V_j - \frac{1}{2} T_{kk} V_k^2 \right) - I_k \right] \frac{dV_k}{dt} \\
&= \sum_{k=1}^{M} \left[\left(-\frac{1}{2} \sum_{i \neq k}^{M} T_{ik} V_i - \frac{1}{2} \sum_{j \neq k}^{M} T_{kj} V_j - T_{kk} V_k \right) - I_k \right] \frac{dV_k}{dt}
\end{aligned}$$

Since $T_{ij} = T_{ji}$ and $T_{kk} = 0$, applying Eq. 5.4. we have :

$$\begin{aligned}
\frac{dE}{dt} &= \sum_{k=1}^{M} \left(-\sum_{i=1}^{M} T_{ki} V_i - I_k \right) \frac{dV_k}{dt} \\
&= \sum_{k=1}^{M} \left(-\frac{dU_k}{dt} \right) \frac{dg_i(U_k)}{dt} \\
&= -\sum_{k=1}^{M} \left(\frac{dU_k}{dt} \right) \frac{dg_k}{dU_k} \frac{dU_k}{dt} \\
&= -\sum_{k=1}^{M} \left(\frac{dU_k}{dt} \right)^2 \frac{dg_k}{dU_k}
\end{aligned}$$

As $\left(\frac{dU_k}{dt}\right)^2 \geq 0$ and from Eq 5.1, $\frac{dg_k}{dU_k} > 0$, thus $\frac{dE}{dt} \leq 0$ and thus E decreases in times with stationary points when $\frac{dU_i}{dt} = 0$. **QED**

This theorem guarantees that the Energy function can be minimized by this network.

Although generalizing this Hopfield NN model to cope with the utility function is not difficult (which would be disscussed in the last section of this Chapter), a major weakness of the Hopfield NN is that it is likely to be trapped by local minima. Furthermore, the above equations are for continuous system which means that it is difficult to simulate them. Although there are many algorithms for simulating continous system, like Gear, Rk45, and etc. , some studies show that the convergence is strongly affected by the parameters of these algorithms. There is another Hopfield based discrete NN model known as the Boltzman's Machine, which is less likely to be trapped by local minima and also easier to be simulated as it is a discrete system. Therefore, in the following studies, we would use this kind of Boltzman's Machine NN.

5.2 Boltzman's Machine

De Wilde [52] defined the Boltzman's Machine as the neural network operating with noise. The neurons in a Boltzman's Machine do not have a deterministic transfer functions, instead they have probabilistic transfer functions : Let $X_i \in \{1, -1\}$, U_i and θ_i be the output, input and threshold of neuron unit i respectively. Applying the Hopfield NN configuration and with some modifications to the transfer function, we have :

$$U_i = \sum_{j=1}^{M} T_{ij} \frac{X_j + 1}{2} - \theta_i \tag{5.5}$$

M is the total number of neuron units and T_{ij} is both symmetric and zero-diagonal.

$$Pr[X_i = 1] = \frac{1}{1 + e^{-\beta_i U_i}} \tag{5.6}$$

$$Pr[X_i = -1] = \frac{1}{1 + e^{\beta_i U_i}} \tag{5.7}$$

One may checked that :

$$Pr[X_i = 1] + Pr[X_i = -1] = \frac{2 + e^{\beta_i U_i} + e^{-\beta_i U_i}}{(1 + e^{-\beta_i U_I})(1 + e^{\beta_i U_i})} = 1. \tag{5.8}$$

$\beta_i \geq 0$, is the inverse "Temperature" of the system. Let $\beta_i = 1/T$ where T denotes the "Temperature" of the system. From (5.6), (5.7), we may observed that for very low Temperatures (or $\beta_i \to \infty$)

$$Pr[X_i = 1] = \begin{cases} 1 & \text{if } U_i > 0 \\ 0 & \text{if } U_i < 0 \\ 0.5 & \text{if } U_i = 0 \end{cases} \tag{5.9}$$

$$Pr[X_i = -1] = \begin{cases} 0 & \text{if } U_i > 0 \\ 1 & \text{if } U_i < 0 \\ 0.5 & \text{if } U_i = 0 \end{cases} \tag{5.10}$$

When the Temperature is very high (or $\beta_i \to 0$)

$$Pr[X_i = 1] = Pr[X_i = -1] = 0.5 \tag{5.11}$$

Intutively, we may say that at high temperatures, the neuron transfer functions behaves less predictable and at low temperatures, the neuron transfer functions cools down to a more deterministic way. A deterministic equivalent of (5.6), (5.7) could be formulate as :

$$X_i(t + \tau) = sign(U_i(t)) \tag{5.12}$$

$$X_i(t + \tau) = sign\left(\sum_{j=1}^{M} T_{ij} \frac{X_j(t) + 1}{2} - \theta_i\right) \tag{5.13}$$

$$X_i(t + \tau) = sign\left(-\frac{\partial E}{\partial V_i}\right) \tag{5.14}$$

The corresponding energy function is :

$$E(\underline{X(t)}) = -\sum_{i=1}^{N}\sum_{j=1}^{M} T_{ij}\frac{X_i(t)+1}{2}\frac{X_j(t)+1}{2} + \sum_{i=1}^{M} X_i(t)\theta_i. \qquad (5.15)$$

Theorem 5.2 The above energy function (5.15) decreases during the operation of the network when (5.14) is used as the transfer function if $T_{ij} = T_{ji}$

Proof: Consider the partial derivative

$$\frac{\partial E}{\partial X_i} = -\frac{1}{2}\sum_{k=1}^{M} T_{ik}\left(\frac{X_k+1}{2}\right) - \frac{1}{2}\sum_{j=1}^{M} T_{ji}\left(\frac{X_j+1}{2}\right) + \theta_i \qquad (5.16)$$

$$= \left[\sum_{j=1}^{M} T_{ik}\left(\frac{X_k+1}{2}\right) - \theta_i\right]. \qquad (5.17)$$

T_{ij} is symmetric with zero diagonal. Switching to finite differences, this formula becomes:

$$\Delta E = -\Delta X_i\left(\sum_{j=1}^{n} T_{ij}\frac{X_j+1}{2} - \theta_i\right). \qquad (5.18)$$

When the term inside the $\sum_{j=1}^{n} T_{ij}\frac{X_j+1}{2} - \theta_i \geq 0$, then according to (5.14) $\Delta X_i \geq 0$ and $\Delta E \leq 0$. Similarly, if this $\sum_{j=1}^{n} T_{ij}\frac{X_j+1}{2} - \theta_i < 0$, $\Delta X_i \leq 0$ and $\Delta E \leq 0$. Thus, the energy E decreases during the operation of the network. **QED**

In the actual simulations, we start with a high temperature and then allows it to cool down gradually. This is actually the idea of *Simulated Annealing*. It is by this mechanism, the local minima may be avoided and the global minimum is more likely to be found.

5.3 Boltzman's Machine Model for Multiple-Metrices Routing

Here, back to the multiple-metrices routing problem. The following is a generalized energy function for multiple metrics routing, which has not been discussed else where.

$$\begin{aligned} E &= -\sum_{j=1}^{M}\sum_{k=1}^{M} T_{jk}\left(\frac{X_j(t)+1}{2}\right)\left(\frac{X_k(t)+1}{2}\right) \\ &\quad -Util\left[\sum_{j=1}^{M}\underline{I}_j\left(\frac{X_j(t)+1}{2}\right)\right]-\sum_{l=1}^{M} J_l\left(\frac{X_l(t)+1}{2}\right) \end{aligned} \tag{5.19}$$

Where $Util(\underline{X}) : \underline{X} \in \Re^C \rightarrow Y \in \Re$ is the utility function. The following neuron transfer functions was choosen :

$$X_i(t+\tau) = sign\left(-\frac{\partial E}{\partial X_i}\right) \tag{5.20}$$

Theorem 5.3 The energy function (5.19) decreases during the operation of the network when (5.20) is used as the transfer function if $T_{ij} = T_{ji}$ and $Util(\cdot)$ is differentiable.

Proof: Consider the partial derivative

$$\begin{aligned} \frac{\partial E}{\partial X_i} &= -\frac{1}{2}\sum_{k=1}^{M} T_{ik}\left(\frac{X_k+1}{2}\right)-\frac{1}{2}\sum_{j=1}^{M} T_{ji}\left(\frac{X_j+1}{2}\right). \\ &\quad -\frac{1}{2}\sum_{n=1}^{C} I_i^{(n)} \cdot \frac{\partial\{Util(\sum_{m=1}^{M}\underline{I}_m\frac{X_m+1}{2})\}}{\partial\left(\sum_{m=1}^{M} I_m^{(n)}\frac{X_m+1}{2}\right)}-\frac{1}{2}J_i \end{aligned} \tag{5.21}$$

Since T_{ij} is symmetric with zero diagonal, thus we have

$$\frac{\partial E}{\partial X_i} = -\left[\sum_{j=1}^{M} T_{ij}\frac{X_j+1}{2} + \frac{1}{2}\sum_{n=1}^{C} I_i^{(n)} \cdot \frac{\partial\{Util(\sum_{m=1}^{M} \underline{I}_m \frac{X_m+1}{2})\}}{\partial\left(\sum_{m=1}^{M} I_m^{(n)} \frac{X_m+1}{2}\right)} + \frac{1}{2}J_i\right] \quad (5.22)$$

where C is the number of metrics on a communication link, $Util$ is the utility function, and $I_i^{(n)}$ is the n-th element of the vector $\underline{I}_i$. Note that the second summation term and the J_i term inside $sign[\cdot]$ is correspond to the θ_i in (5.6), (5.7) In discrete form :

$$\Delta E = -\Delta X_i \left[\sum_{j=1}^{n} T_{ij}\frac{X_j+1}{2} + \frac{1}{2}\sum_{n=1}^{C} I_i^{(n)} \cdot \frac{\partial\{Util(\sum_{m=1}^{M} \underline{I}_m \frac{X_m+1}{2})\}}{\partial\left(\sum_{m=1}^{M} I_m^{(n)} \frac{X_m+1}{2}\right)} + \frac{1}{2}J_i\right]. \quad (5.23)$$

When the term inside the $[\cdot] \geq 0$, then according to (5.20) $\Delta X_i \geq 0$ and $\Delta E \leq 0$. Similarly, if this $[\cdot] < 0$, $\Delta X_i \leq 0$ and $\Delta E \leq 0$. Thus, the energy E decreases during the operation of the network. **QED**

In the probabilistic situation, we would have a probabilistic version of (5.20), and ΔX_i may sometimes (especially at high temperatures) in the opposite way and causing ΔE to increase. It is by this way the local minima may be avoided and the global minimum is more likely to be found.

Base on Ali's [1] energy function, we derive a similar function as below :

$$\begin{aligned} E \;=\; & -\mu_1 \cdot Util\left[\sum_{x=1}^{N}\sum_{\substack{i=1\\ i\neq x}}^{N}\sum_{(x,i)\neq(d,s)} \underline{C}_{xi} V_{xi}\right] + \mu_2 \sum_{x=1}^{N}\sum_{\substack{i=1\\ i\neq x}}^{N}\sum_{(x,i)\neq(d,s)} \rho_{xi} V_{xi} \\ & + \mu_3 \sum_{x=1}^{N}\left(\sum_{i=1, i\neq x}^{N} V_{xi} - \sum_{i=1, i\neq x}^{N} V_{ix}\right)^2 + \mu_4(1 - V_{ds}) \end{aligned} \quad (5.24)$$

where

$$V_{xi} = \frac{X_{xi} + 1}{2} \tag{5.25}$$

with $V_{xi} = 1$, if the link form node x to i is in the path and $V_{xi} = 0$ otherwise.

The index xi is an uni-dimensional index, the actual index should be given by $j = x * N + i$, where N is the number of communication nodes considered. This way of indexing had been widely used since JJ Hopfield's TSP formulation [45]. The physical meaning of this equation is similar to (3.8) - μ_1 term maximizes the total utility of a path by taking into account the metrics of existing links. The μ_2 term is for removing the nonexistent links from the solution. The μ_3 term is to ensure that the number of incoming links to a node is equal to the number of outgoing links. The original μ_4 term is used to push the state of the neural network to converge to one of the 2^{n^2-n} corners of the solution hypercube, defined by $V_{xi} \in \{0, 1\}$. In the Boltzman's machine the results are always in $\{0, 1\}$, thus it is omitted in here. Instead, the μ_4 term here is equivalent to the original μ_5 term which makes the final solution to contain the link (virtual or real) from d to s and therefore both the source and destination would be in the solution path. Note that the final is always a loop, starting from s, goes through some intermediate nodes if any, then reach destination d and a single (virtual or real) link back to s. Implied from (5.20), the neuron dymanic equations are given by :

$$U_{xi} = -\frac{\partial E}{\partial X_{Xi}} \tag{5.26}$$

$$\begin{aligned} U_{xi} &= \frac{\mu_1}{2} \sum_{n=1}^{C} C_{xi}^{(n)} (1 - \delta_{xd}\delta_{is}) \left(\frac{\partial (Util)}{\sum\sum C_{xi}^{(n)} \frac{X_{xi}+1}{2}} \right) \\ &\quad -\frac{\mu_2}{2} \rho_{xi} (1 - \delta_{xd}\delta_{is}) \end{aligned}$$

$$-\frac{\mu_3}{2}\sum_{\substack{y=1\\y\neq x}}^{N}\left(\frac{X_{xy}+1}{2}-\frac{X_{yx}+1}{2}\right)+\frac{\mu_3}{2}\sum_{\substack{y=1\\y\neq x}}^{N}\left(\frac{X_{iy}+1}{2}-\frac{X_{yi}+1}{2}\right)$$
$$+\frac{\mu_4}{2}\delta_{xd}\delta_{is}. \tag{5.27}$$

where,

$$\delta_{ij}=\begin{cases}1 & if \quad i=j\\ 0 & if \quad otherwise\end{cases} \tag{5.28}$$

Comparing the coefficients of (5.27) to the 2-D version of (5.22) as shown below:

$$\frac{\partial E}{\partial X_{xi}}=-\left[\sum_{y=1}^{N}\sum_{j=1,j\neq y}^{N}T_{xi,yj}\frac{X_{yj}+1}{2}+\frac{1}{2}\sum_{n=1}^{C}I_{xi}^{(n)}\cdot\frac{\partial\{Util(\sum_{m=1}^{N}\sum_{j=1,j\neq m}^{N}\underline{I_m}\frac{X_{mj}+1}{2})\}}{\partial\left(\sum_{m=1}^{N}\sum_{j=1,j\neq m}^{N}I_{mj}^{(n)}\frac{X_{mj}+1}{2}\right)}+\frac{1}{2}J_{xi}\right]. \tag{5.29}$$

we have the following relations:

$$T_{xi,yi}=-\mu_3\delta_{xy}-\mu_3\delta_{ij}+\mu_3\delta_{jx}+\mu_3\delta_{iy} \tag{5.30}$$

$$I_{xi}^{(n)}=\mu_1 C_{xi}^{(n)}(1-\delta_{xd}\delta_{is}) \tag{5.31}$$

$$J_{xi}=\mu_2\rho_{xi}(1-\delta_{xd}\delta_{is})+\mu_4\delta_{xd}\delta_{is} \tag{5.32}$$

5.4 Conclusions

So far we have discuss the theoretical feasibility of solving the multiple merics utility optimization routing problem with the Hopfield NN. From the above analytical results, it is clear that the problem can be modelled within the Hopfield NN framework. Theoretical studies show that the modified Hopfield NN converges during the operation of the network. And the concept of the energy function is a clone of Ali's work [1] which had been proved to be successful both

analytically and via simulations. It is likely that the model will also produce such results, provided that the μ parameters are chosen properly. Again, we may refer to [1] about the ways of choosing those parameters.

Chapter 6

Conclusions

In this thesis, we have discussed the trend towards a commercialized global Internet. As such commercialization does not favour an administrative domain to carry transit domain traffics without placing a proper charge for them. This is because the volume of such kind of transit traffics may cause a great degradation to the domain's intra-domain and inter-domain services. The emerging high bandwidth services on the Internet would also make this problem even worse. This is concerned with the well-known economic issue known as externalities, where market mechanism fails. In order to restore the market, price strategies should be employed. Our thesis has proposed a simple yet efficient and practical pricing scheme, namely, each domain imposes a price for every transit domain traffics. If every route uses its least-cost route, a global minimum of resource consumption would be resulted. But in this thesis, we have done more than that. We adapted the concept of utility function, by which each person's needs about the Internet services can be reflected. For instance, some people want high quality services with higher prices, while some others may just want the

least costv services.

In the technology aspect, although some inter-domain routing protocols like BGP-4 allows multiple link attributes to be considered in route decision, the current practices with multiple link attributes are either combining them to a single metric, then applies the shortest path algorithms, or consider one of metrics as the dominant factor applies the shortest path algorithms, and then tie-breaking by the secondary metrics. However, both approaches do not satisfy our needs. Thus, our thesis has proposed two neural network framework to solve our unique routing problem. In chapter 4, we have used a MLFN based neural network to solve this problem and simulation results have been presented to support the algorithm. This MLFN is potentially VLSI implementable. In chapter 5, we have shown theoretically that the Hopfield NN can also be applied to this problem.

Bibliography

[1] Mustafa K. Mehmet Ali and Faouzi Kamoun. Neural networks for shortest path computation and routing in computer networks. *IEEE Transaction on Neural Networks Vol 4, No 6 (Nov)*, 1993.

[2] Zorica Avramovic. Policy based routing in the defense information system network. In *MILCOM'92.*

[3] Nirwan Ben Yuhas et al. *Neural Networks in Telecommunications.* Kluwer Academic Publishers, 1994.

[4] G.J. Gibson C.F.N. Cowan Chen, S. and P.M.Grant. Adaptive equalization of finite non-linear channels using multilayer perceptrons. In *Signal Processing*, volume 20, pages 107–119, 1990.

[5] G.J. Gibson C.F.N. Cowan Chen, S. and P.M.Grant. Reconstruction of binary signals using an adaptive radial-basis-function equalizer. In *Signal Processing*, volume 22, pages 77–93, 1991.

[6] Paul N.Weissler Chia-Jiu Wang. The use of aritifcial neural networks for optimal message routing. *IEEE Network*, pages 16–24, March/April 1995.

[7] Goutam Chakraborty Chotipat Pornavalai and Norio Shiratori. A neural network approach to multicast routing in real-time communication networks. *IEEE Inter. Conf. on Network Protocols*, 1995.

[8] Mark Collett and Witold Pedrycz. Application of neural networks for routing in telecommunications networks. *IEEE GLOBECOM' 93.*

[9] Douglas Comer. *Internetworking with TCP/IP - Principles, Protocols and Architecture.* Prentice Hall Internationals, Inc., 1988.

[10] Partha Dasgupta. *The Control of Resources.* Harvard University Press, Cambridge, Massachusetts, 11982.

[11] Martha Steenstrup Deborah Estrin and Gene Tsudik. A protocol for route establishment and packet forwarding across multidomain internets. *IEEE Transactions on Networking*, 1(1):56–70, February 1993.

[12] Mike Dixon. An optimal neural routing algorithm. *Proc. of IEEE Singapore Inter . Conf. on Networks/Inter. Conf. on Information Engineering' 93.*

[13] Annop Ghanwani Erol Gelenbe and Vijay Srinivasan. Improved neural heuristics for multicast routing. *IEEE Journal on Selected Areas in Communications*, 15(2), February 1997.

[14] Thomas Fritsch and Wolfgang Mandel. Communication network routing using neural nets - numerical aspects and alternative approaches. *IEEE IJCNN' 91.*

[15] L. Siccardi G. Frisiani, T. Parisini and R. Zoppoli. Team theory and back-propagation for dynamic routing in communication networks. *IEEE IJCNN' 91.*

[16] J.J. Garcia-Luna-Aceves and Shree Murthy. A loop-free path-finding algorithm : Specification, verification and complexity. *IEEE INFOCOM'95.*

[17] Robert Hecht-Nielsen. *Neurocomputing.* Addison-Wesley Pub. Co., 1990.

[18] M. I. Henig. The shortest path problem with two objective functions. *European Journal of Operational Research*, 25:281 – 291, 1985.

[19] Michael Jacobs. *The Green Economy, Environment, Susstainable Development and The Politics of the Future.* Pluto Press, 1991.

[20] Craig M. Barnhart Jeffrey E. Wieselthier and Anthony Ephremides. A neural network approach to routing without interference in multihop radio networks. *IEEE Transaction on Communications Vol 42, No 1 (Jan)*, 1994.

[21] M.A. Eshera J.E.Jensen and S.C. Barash. Neural network controller for adaptive routing in survivable communication networks. *IEEE IJCNN' 90.*

[22] Brijesh Kumar. Models, implementations and design options for inter-domain policy routing protocols. In *Proc. of Computer System and Software Engineering*, 1992.

[23] Liu Ze-min Liu Rong and Zhou Zheng. Neural network approach for communication network routing problem. *IEEE TENCON' 93.*

[24] Timothy Masters. *Signal and Image Processing with Neural Networks : A C++ Sourcebook.* Wiley, 1993.

[25] Timothy Masters. *Advanced Algorithms for Neural Networks : A C++ Sourcebook.* Wiley, 1995.

[26] Jack L. Meador. Spaciotemporal neural networks for shortest path optimization. *Proc. of ISCAS' 95.*

[27] Graeme R. Cole Michael W. dixon and Matthew I. Bellgard. A neural network shortest path algorithm for routing in packet-switched communication networks. *Porceeding of ICC' 95.*

[28] Martin F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[29] Shree Murthy and J.J. Garcia-Luna-Aceves. Dynamics of a loop-free path-finding algorithm. *IEEE GLOBECOM'95.*

[30] Shree N. Murthy. Design and analysis of distributed routing algorithms. Master's thesis, University of California, Santa Cruz, June 1994.

[31] Y. Rekhter and T. Li. A border gateway protocol 4 (bgp-4). *Internet rfc 1771*, March 1995.

[32] Vlad Rutenburg. Fair charging policies and minimum-expected-cost routing in internets with packet loss. In *INFOCOM'91.*

[33] A. Di Stefano S. Cavalieri and O. Mirabella. A neural-network-based approach for routing in a packet switching network. *IEEE IJCNN' 92.*

[34] C. Labinsky S. Nordhoff, B. Landorff and R. Hartwig. *FOOL & FOX Fuzzy System Development Tools Users Manual (Rel 1.2).* University of

Oldenburg (Germany), April 1995. This document can be obtained via : http://condor.informatik.uni-oldenburg.de/FOOL.html.

[35] D.Estrin S. Shenker, D. Clark and S.Herzog. Pricing in computer networks : Reshaping the research agenda. *Telecommunications Policy*, 20(3):183–201, 1996.

[36] Edward R. Scheinerman. *Invitation to Dynamic Systems.* Prentice Hall Inc., 1996.

[37] Mischa Schwartz. *Telecommunication Networks : Protocols, Modeling and Analysis.* Addison-Wesley Pub. Co., 1987.

[38] Linda J. Seamonson and Eric C. Rosen. 'stub' exterior gateway protocol. *Internet RFC 888*, January 1984.

[39] Nasir Shaikh-Husin and Jack L. Meador. Spatiotemporal neural networks for link-state routing protocols. *Proceeding of ISCAS'96.*

[40] Casimir A. Kulikowski Sholom M. Weiss. *Computer Systems that Learn - Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann Publishers, Inc., 1992.

[41] Mahesan Niranjan Shreeram V.B. Aiyer and Frank Falliside. A theoretical investigaion into the performance of the hopfield model. *IEEE Transactions on Neural Networks*, 1(2), June 1990.

[42] Martha E. Steenstrup. *Routing in Communications Networks.* Prentice Hall International, Inc, 1995.

[43] Lei Zhang Stelios C.A. Thomopoulos and Chin Der Wann. Neural network implementation of the shortest path algorithm for traffic routing in communication networks. *IEEE IJCNN' 91.*

[44] A. S. Tanenbaum. *Computer Networks.* Prentice Hall International, Inc, 1989.

[45] D.W. Tank and J.J. Hopfield. Simple 'neural' optimization networks : An a/d converter, signal decision circuit and a linear programming circuit. *IEEE Trans. Circuits. Syst.*, CAS-33(5):533–541, 1986.

[46] Kuang Tsai and Richard P. Ma. Artificial neural networks for distributed adaptive routing on dynamic topology networks. *IEEE IJCNN' 92.*

[47] Jean Walrand. *Communication Networks : A First Course.* Aksen Associates Inc. Publishers, 1991.

[48] Li-Xin Wang. *Adaptive Fuzzy Systems and Control : Design and Stability Analysis.* Prentice Hall International, Inc., 1994.

[49] Zheng Wang and Jon Crowcroft. Bandwidth-delay based routing algorithm. In *IEEE GLOBCOM'95.*

[50] Paul N. Weissler and Chia-Jiu Wang. The use of artificial neural networks for optimal message routing. *IEEE MILCOM' 92.*

[51] Michael G. Hluchyi Whay C. Lee and Pierre A. Humblet. Routing subject to quality of service constraints in integrated communication networks. *IEEE Network*, 1995.

[52] Philippe De Wilde. *Neural Network Models : An Analysis - (Lecture Notes in Control & Information Sciences; Vol.210.* Springer, 1996.

[53] Patrick Henry Winston. *Artificial Intelligence 3rd Edition.* Addison Wesley, 1992.

[54] Andreas Zell et al. *Stuttgart Neural Network Simulator User Manual, Version 4.1.* University of Stuttgart, Institute for Parallel and Distributed High Performance Systems, 1995. This manual can be obtained via : ftp.informatik.uni-stuttgart.de.