INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Optimal Traffic Signal Control System Using an Artificial Neural Network

By

JIANZHONG FAN

A dissertation submitted to the Graduate Faculty in Engineering
In partial fulfillment of the requirements for the degree of
Doctor of Philosophy, The City University of New York

1998

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

7/7/98
Date

Chair of Examining Committee

7/7/98
Date

Executive Officer

Professor Mitsuru Saito

Professor Neville A. Parker

Professor Robert E. Paaswell

Professor Norbert Oppenheim

Professor Anthony Tzes

The City University of New York

Abstract

Optimal Traffic Signal Control System Using an Artificial Neural Network

By
Jianzhong Fan

Adviser: Professor Mitsuru Saito

This dissertation research aims to develop an artificial intelligence based signal timing system. The system was intended to dynamically control a traffic signal by finding an optimal signal timing to minimize delay at signalized intersections within a few seconds. This optimal traffic signal control system (OTSCS) consists of two major components: an artificial neural network designed for analyzing level of service at a signalized intersection and a heuristic search.

Level of service analysis neural network (LOSANN) is a flat connection network with an error back-propagation algorithm. In order to sufficiently represent the traffic environment, LOSANN has 132 neurons at its input layer. Such a large number of input neurons tend to make the learning process relatively difficult and time consuming. Therefore, a variety of network architectures, learning modes and learning rate types are studied in this dissertation. Two learning rate types, linear and weighted exponential learning rate, are created to enhance the learning ability of LOSANN. The output of LOSANN is the average stopped delay per vehicle at an intersection.

An optimum traffic signal timing model (OTSTM) integrates LOSANN to obtain a signal timing which minimizes the average stopped delay at an intersection. A new heuristic search strategy named "direction search" is created in this study. It has an order of growth of $O(n^2)$ in the worst case. The n is the number of traffic signal cycle lengths that could be used at the specific intersection. The numeral examples tested in the research show that the search speed of direction search is more the 10 times faster than that of the conventional depth-first search and the solutions of these two searches are very close.

OTCSC is programmed by Visual Basic Release 5.0 as comprehensive research tool. It provides options for selecting artificial neural network models and optimizing traffic signal models. OTSCS can be used to analyze level of service at an intersection, optimize signal timing, and understand the learning behavior of neural networks and the features of heuristic searches.

To
My wife and son

Acknowledgments

The author whishes to extend his greatest appreciation to his adviser, Dr. Mitsuru Saito of Brigham Young University, who has provided him with great support and encouragement in finishing this difficult task.

Special thanks must be given to Dr. Claire McKnight of The City University of New York, who has mentored the author after his adviser left the university.

Also, the author would like to acknowledge the assistance received from the other members of his doctoral committee, Dr. Neville A. Parker, Dr. Robert E. Paaswell, Dr. Norbert Oppenheim of the City University of New York, and Dr. Anthony Tzes of Polytechnic University.

In addition, the author thanks the Traffic Planning Department of the New York City Department of Transportation which provided necessary data to train and test the model developed in this research."

CONTENTS

List of Tables

List of Figures

CHAPTER 1: INTRODUCTION

1.1 Background and Motivation

Today, many large cities in the world are confronted with inveterate traffic congestion. Traffic congestion has resulted in great monetary loss. United States General Accounting Office (1985) reported that in the United States congestion-caused delays alone result in productivity losses of up to $ 100 billion annually. This evaluation did not include the loss of life, air pollution and wasted fuel. Each year, about 2 billion gallons of fuel are wasted due to traffic congestion, according to the Texas Transportation Institute (1990). Such congestion is caused by two factors. The first is chronic excess traffic demand on the facilities during peak periods. The second is capacity reduction by various kinds of incidents. To alleviate traffic congestion in urban areas, use of Advanced Traffic Management System (ATMS) has steadily grown since the early 1990s. The primary function of ATMS is to provide traffic engineers with the capability of monitoring traffic conditions dynamically, to adjust traffic operations accordingly, and to respond to incidents quickly. Ever-advancing communications and computer technologies are the backbone of ATMS.

In urban areas, traffic control relies primarily on traffic signals. An ATMS in urban areas should consist of all or part of the following components: traffic detectors, computerized signal controllers, variable message signs. In this context, the efficient operation or a computerized signal control system is required to effectively control traffic movement with minimal delay, is a challenge to the traffic engineer. One of the major

difficulty is that the Level of Service (LOS) at a signalized intersection, which is the foundation of developing an optimal signal control system, can not be easily and precisely estimated since:

a) A large number of factors, such as traffic flow, signal timing, geometric conditions, driver behavior, and weather condition, not only affect the quality of traffic operation, but also each other.
b) The relationships among these factors and the relations between them, and the quality of traffic operation are often nonlinear and at times not clearly understood.
c) Dynamic traffic flow characteristics make the relationships among the factors even more complex.

In order to satisfy the requirement of ATMS, that is, quick response to temporal anomalies in traffic conditions, we need to develop an adaptive traffic control procedure to replace traditional evaluation methodologies, such as the operational analysis of the 1994 Highway Capacity Manual. The new procedure must be able to determine appropriate signal timings quickly and accurately under the complex relations between the quality of traffic operation and the traffic environment. In other words, the new traffic control system must have properties similar to the human decision making process. It needs to be able to represent and solve the problem through self-learning, rather than relying on given equations or reaching a reasonable decision from data intensive statistical analyses. From a computational point of view, artificial neural networks (ANNs) have been shown to offer these two characteristics. The primary motivation of this study is to introduce artificial

neural network and heuristic optimization techniques to Advanced Transportation Management Systems.

1.2 Study Goals

The ultimate goal of this study is to develop an optimal traffic signal control system (OTSCS) that can be used to quickly design signal timing for isolated intersections, using artificial intelligence. Given a range of signal cycle length, OTSCS is capable of determining an optimal green time split that would result in the lowest average stopped delay for the intersection as a whole. OTSCS consists of two major components:

1. An artificial neural network designed for level of service analysis. This neural network model was named LOS analysis neural network (LOSANN). LOSANN analyzes the level of service at a signalized intersection by its experience space rather than by a traditional analytical approach or a knowledge based expert system. The experience space is constructed by data from filed surveys or from previous studies. OTSCS also has the ability to quickly update its knowledge by self-learning.
2. A heuristic search-based optimum traffic signal timing model (OTSTM). This model is able to quickly find an optimum traffic signal timing by merged with LOSANN.

1.3 Scope of the Study

This study focuses on the optimal signal control problem of isolated signalized intersections. The typical intersection has four approaches. As an extension, 3-leg intersections are also included in the study. Each approach of the intersection has many

factors affecting traffic operation, such as the number of lanes, parking condition, presence of heavy vehicles, the number of bus stops per hour, and so forth. The signal timing selections requires the traffic engineer to find two basic elements: cycle length and green time split that will result in the minimum average stopped delay at signalized intersections. The scope of OTSCS is to analyze LOS of signalized intersections with a two-phase sequence plan. OTSCS that consists of an ANN model and an optimization heuristic will be capable of providing an optimal signal timing for isolated intersections based on real time traffic situations.

In order to accumulate knowledge and experience in the application of ANNs to transportation engineering, it is necessary to understand how the activation function, the propagation algorithm, and the network architecture will affect the behaviors of ANN models. This study focus on the evaluation of the effects of the following factors on ANN models

1. Number of layers and neurons in hidden layers
2. Mode of learning process: *pattern mode* or *batch mode*
3. Learning rate: a constant, a momentum constant, or a changing parameter
4. Weight initiation method: random, or not random

1.4 Benefit of the Study

This study developed an optimal traffic signal control system (OTSCS) comprising of artificial neural network and heuristic search techniques. OTSCS was meant for dynamic traffic control. Traditional signal timing analyses such as use of the Highway Capacity Software (HCS) and similar products are not capable of performing the timing routine

quickly based on real time traffic conditions; they require evaluation of outputs by experienced traffic engineers. OTSCS combines these two steps of signal timing into one decision making system; it can accomplish both analysis and design. Therefore, it can optimize traffic signal timing of an isolated intersection within a few seconds after the dynamic traffic condition is input into OTSCS.

OTSCS can not only make a decision much more quickly but also is more accurate than the traditional techniques. Capacity analysis done by traditional techniques can sometimes be significantly different from the specific traffic environment under study, since the parameters used to estimate LOS are generally derived from statistical studies that reflect average conditions nationwide. OTSCS can overcome this weakness because it is a self-learning system. OTSCS can be trained by the data which describes specific traffic environment to "understand" the relationships that may be too complex or even not clear to solve analytically.

In addition to developing OTSCS, this study examined feasibility of an ANN-based local traffic controller and the level of accuracy it can provide; it also examined how unique properties of ANNs can be exploited for local signal control at isolated intersections. Development of OTCSC used static data as an input, but the concept can be expanded to allow the use of dynamic data because the method of passing data do not change the relationships exited between level of service at an intersection and traffic environment. In this respect, this study contributes to the advancement of research on local dynamic traffic control which are a significant component of Advanced Traffic

Management Systems (ATMSs), as well as to the research in practical use of ANNs to transportation engineering.

OTSCS is programmed to become a comprehensive research tool to understand ANNs and heuristic searches as applied to signal timing. It provides options that pertain to the architecture, learning mode and learning rate. Besides this capability, OTSCS has two optimal solution models: state space search and direction search.

1.5 Methodology

This study consisted of the following six major tasks:

1. Conduct a literature review of applications of ANNs to transportation engineering, especially to traffic signal control;
2. Study and review theories, concepts and methods related to ANNs and optimization algorithm, and set the direction for realizing the study goal as stated in the "Study Goals" section;
3. Prepare a pattern set (experience space) to train and test ANN models using the results of level of service analyses done by the Highway Capacity Manual method for signalized intersections.
4. Develop artificial neural networks to evaluate the level of service of isolated signalized intersections;
5. Design heuristic algorithms, identify optimal solution and integrate the algorithms with the ANN models to build OTSCS;
6. Develop a PC-based software.

1.6 Organization of the Dissertation

This dissertation consists of seven chapters, a bibliography and two appendices. The first chapter covers the motivation and the objective of the research. The second chapter presents the findings from a literature survey. The third chapter discusses the methodology used to develop an optimal traffic signal control system (OTSCS) and its two components: a) ANN based traffic level of service analysis model named LOS analysis neural network (LOSANN); b) the optimal traffic signal timing model (OTSTM). The fourth chapter describes in detail the development of LOSANN model. The fifth chapter reports the results of training and testing LOSANN. The sixth chapter analyzes the optimal strategies which are used in OTSTM. And finally, the seventh chapter summarizes the research. Appendix A discusses the derivation of a general form of the error back-propagation algorithm used to perform weight adjustments in an ANN. An OTSCS software was developed to test various LOSANN and OTSM applications. Appendix B provides an introduction for using the OTSCS software.

CHAPTER 2: LITERATURE REVIEW

2.1 Artificial Neural Networks

2.1.1 Definition, Catalog and Function

Artificial neural networks (ANNs) have been motivated from their inception by the recognition that the brain computes in an entirely different way from the conventional digital computer. Since Hebb (1949) published one of the first rules for learning by neurons, many artificial neural network models have been developed. Figure 2-1 displays a variety of ANN types, which are discussed in this section. Although ANNs

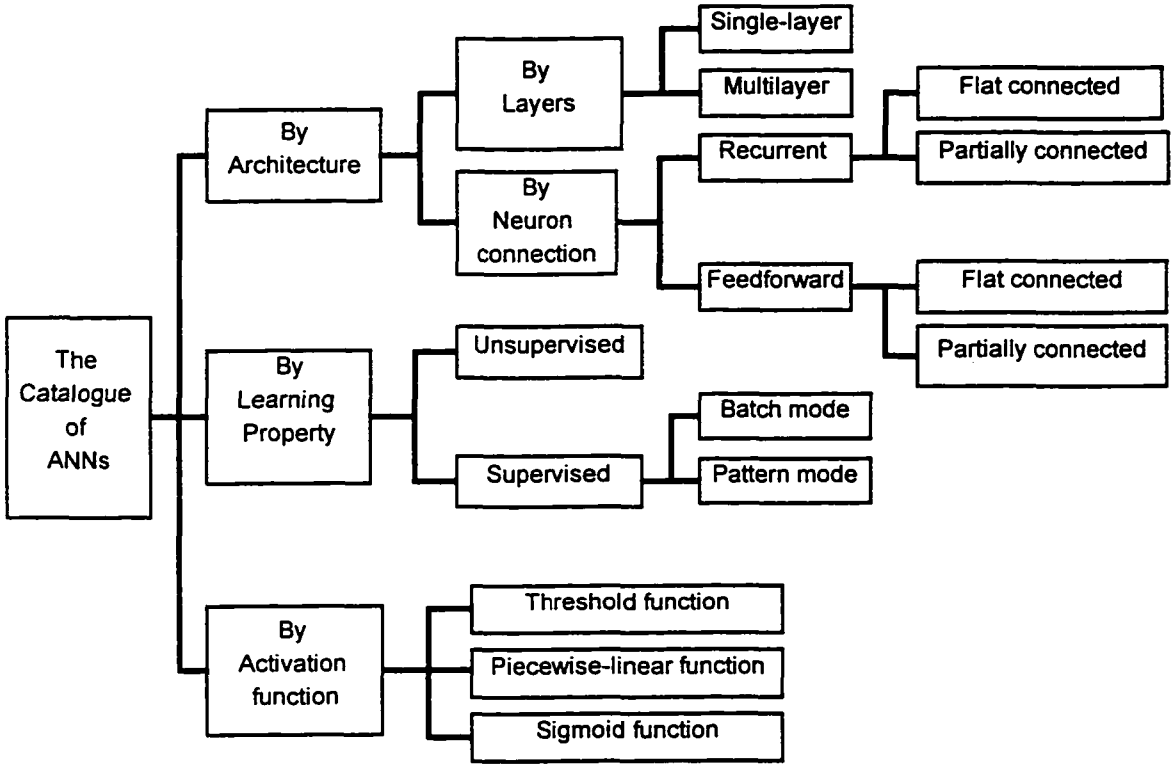


Figure 2-1 Catalogue of ANNS

may have different architectures, learning properties and activation functions, they have some common characteristics. In 1990, Aleksander and Morton (1990) defined ANNs as follows:

"A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

- Knowledge is acquired by the network through a learning process.
- Interneuron connection strengths known as synaptic weights are used to store the knowledge."

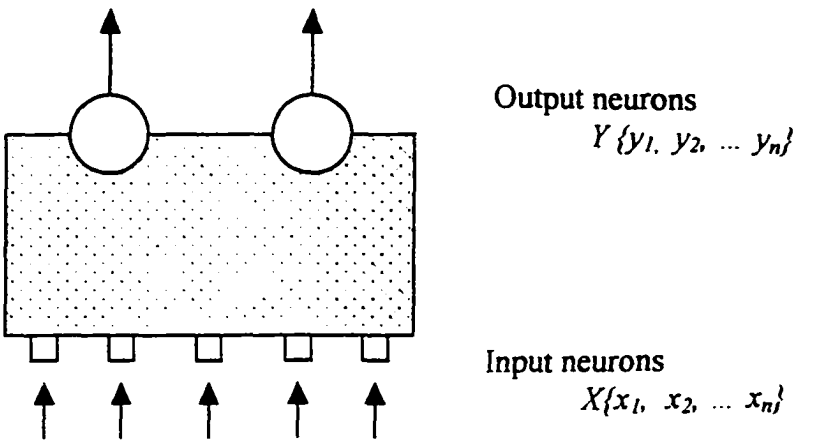


Figure 2-2 ANNs functions as a black box

On the whole, a neural network functions like a black box (see Figure 2-2). A subset of neurons $X_0\{x_1, \ldots x_p\}$, is considered as network inputs. This input set of neurons is always on the first layer of an ANN. The neurons on the last layer are output neurons $Y_k\{y_1, \ldots, y_q\}$. If a network N is considered as a black box function and X_0 is a vector of inputs to network N, then the output vector Y_k can be written as

$$Y_k = N(X_0) \qquad (2\text{-}1)$$

where X_0 and Y_k are vectors. The black box function 2-1 appears very simple but it requires effort to fully understand it. The following discussions are largely based on *Neural Networks* by Haykin (1994) and *Neural Network Learning and Expert Systems* by Gallant (1994). The notations used by Haykin and Gallant have been modified to make the notations in this dissertation internally consistent.

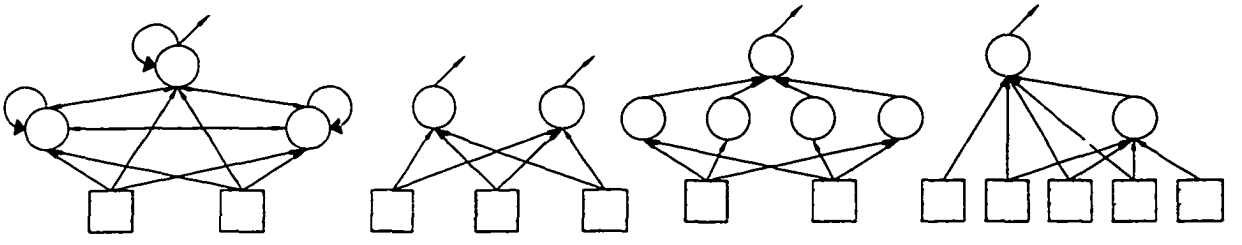
2.1.2 Network Architectures

The structures of neural network models are shown in Figure 2-3. Though their structures are different, each of them consists of a network of neurons that are joined by directed arcs. The neurons and arcs comprise the network topology. Each arc has a numerical weight, w_{ji}, that specifies the influence of neuron y_i on y_j. It is important to make a note of the manner in which the subscripts of the neuron's weight w_{ji} are written. The first subscript refers to the neuron in question and the second subscript refers to the input end of the neuron to which the weight refers. The weights determine the behavior of the network, playing somewhat the same role as in a conventional program.

It is often convenient to organize the neurons of a network into layers. A k-layer network is defined as a network where neurons are grouped into $k+1$ subsets, that is, $X_0\{x_1, \ldots, x_p\}, Y_1\{y_{0,\ldots} y_q\}, \ldots, Y_k\{y_k \ldots, y_r\}$. Here, X_0 is the input layer and Y_k is the output layer. The neuron y_i refers to an element in the set $Y_i\{y_{0,\ldots} y_q\}$, that is, $y_i \in Y_j\{y_{0,\ldots} y_q\}$. If we define C_i as the number of elements in the set Y_i, then C_i is equal to $q+1$.

A layer that is neither the input layer nor the output layer is called a hidden layer. The neurons on the hidden layers are called hidden neurons. By adding one or more hidden layers, the network is able to extract higher-order statistics (Haykin ,1994). This ability of hidden neurons is particularly valuable when the size of the input layer is large.

By their architectures, ANNs can be defined as single-layer networks or multilayer networks. Single-layer networks such as the one shown in Figure 2-3(b) do not have any hidden layer. Other networks shown in Figure 2-3 are multilayer networks.



(a) (b) (c) (d)

Figure 2-3 Architectures of neural networks

Networks can be classified by their connection type as well. Before describing this kind of classification in detail, *feed-forward* networks and *recurrent* networks need to be defined first. The networks are *recurrent networks* if they contain directed cycles; otherwise, the networks are *feed-forward networks*. Figure 2-3a shows a totally connected, recurrent network since the neurons, except those on the input layer, in a totally connected network are linked to each other, as well as to themselves. All others in Figure 2-3 are *feed-forward* networks. Among *feed-forward networks*, there are networks defined as flat networks, in which each neuron y_j in layer j is connected to every neuron y_{j+1} in the next layer $j+1$ for $j \in [0, k]$. Figure 2-3b shows a single layer flat network while Figure 2-3c shows a multilayer flat network. Networks whose neurons in neighbor layers are not completely connected are called partially connected networks. Figure

2-3d shows a partially connected *feed-forward* network.

2.1.3 Properties of Neuron

A neural network consists of a set of neurons which are fundamental to the operation of the neural network. Neurons in a network function as information-processing units. Figure 2-4 shows a general neuron model. Each neuron sums the weighted input signals y_i which constitute a linear combination:

$$u_j = \sum_{i=1}^{p} w_{ji} y_i \qquad (2\text{-}2)$$

Then the amplitude of the output of a neuron is controlled by an activation function expressed as

$$y_j = \varphi(u_j - \theta_j) \qquad (2\text{-}3)$$

where φ() is an activation function. The activation function must satisfy some conditions which will be discussed later. A *threshold* θ_j is included in the activation (2-3). The

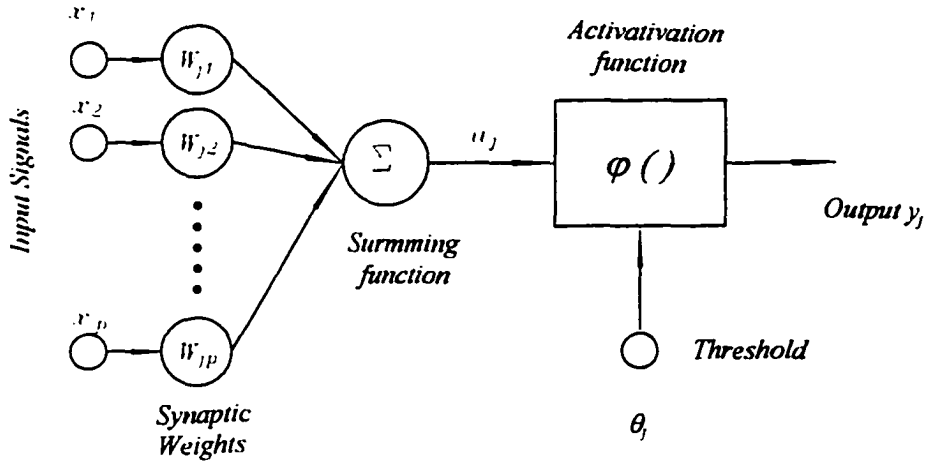


Figure 2-4 Activation computed for a neuron

threshold θ_j has the effect of lowering the net input of the activation function. The value of *thresholds* can be positive or negative.

ANNs have the following three important characteristics:

1. Neuron inputs and activations may be discrete, taking on values {0,1} or {-1,0,1} or they may be continuous, assuming the input values are inside the interval [0, 1] or [- 1,1]. Here, the symbol {} defines a set. For example, The set {0, 1} defines consists of 0 and 1. The another symbol [ ] defines a domain. For example, The domain [0, 1] includes any real number from 0 to 1. In the case where inputs and activations are {0,1} or [0,1], the value "0" usually means "false" and the value "1" means "true". The value between "0" and "1" indicates the degree of proximity to these two extreme points. In the case where inputs and activations are {-1, 0, 1} or [-1, 1], the value "-1" means "false", the value "1" means "true" and the value "0" means "unknown".
2. A *threshold* θ_j should always be available on each layer, except on the output layer. A *threshold* θ_j has the effect of applying a bias to the output of the linear combiner as shown below:

   $$v_j = u_j - \theta_j$$

   As a result of the affine transformation, the graph of v_j versus u_j might no longer pass through the origin.
3. The activation function $\varphi(\ )$ shown in Equation (2-3) defines the output of a neuron in terms of the activity level at its input. Haykin (1994) listed three activation functions applied to ANNs: *Threshold function, Piecewise-Linear function* and *Sigmoid Function. Threshold function* is used for describing the all-or-none properties. *Piecewise-Linear Function* is used for an approximation to a nonlinear amplifier. *Sigmoid Function* is by far the most common form of

activation function used in the construction of ANNs. If a nonlinear function, such as a sigmoid function, is used as an activation function, it must be bounded, and piecewise differentiable.

Since the threshold θ_j is an external parameter of artificial neuron, Equation (2-2) and (2-3) can be combined as follows for convenience:

$$v_k = \sum_{j=0}^{p} w_{kj} y_j \qquad (2\text{-}4)$$

$$y_k = \varphi(v_k) \qquad (2\text{-}5)$$

where $y_0 = -1$ and whose weight $w_{k0} = \theta_j$.

2.1.4 Learning Process

Mendel and McClaren (1970) defined *learning* in the context of neural networks as follows:

> *Learning is a process by which the free parameters of a neural network are adapted through a continuing process of simulation by the manner in which the parameter changes take place.*

This definition of the learning process implies the following sequence of events:

1. The neural network is stimulated by its environment.
2. The neural network undergoes changes as a result of this stimulation.
3. The neural network responds in a new way to the environment, because of the changes that have occurred in its internal structure.

Following the above sequence, different learning machines were constructed to solve different learning problems. Learning problems can be catalogued into unsupervised and supervised learning problems (see Figure 2-5) according to their properties (Stephen, 1994).

For unsupervised learning problems, there are no specified correct responses for the network. A self-organizing machine is often used to solve unsupervised learning problems.

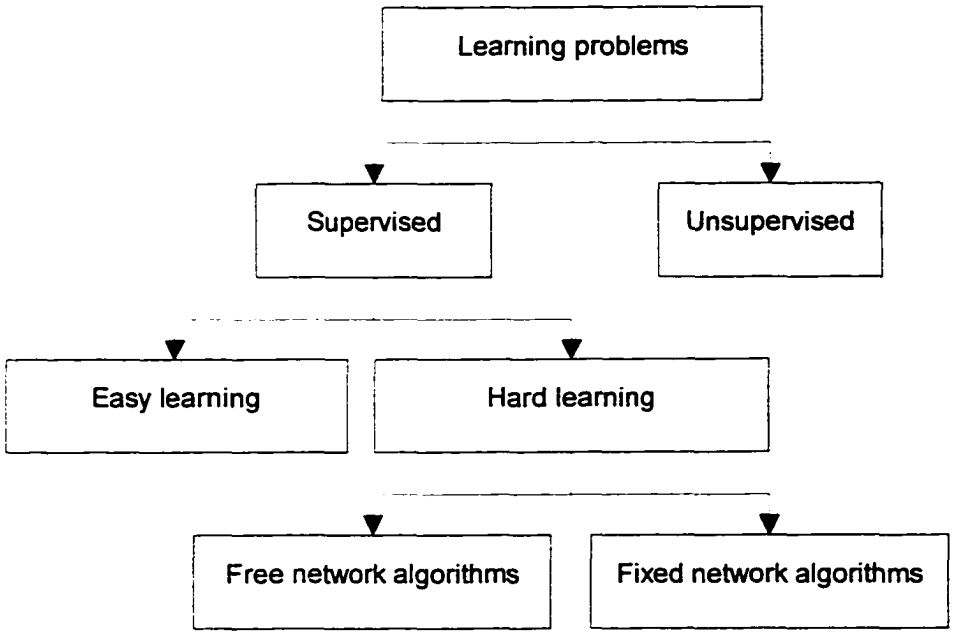


Figure 2.5 Various types of learning problems

Learning machines commonly applied to pattern recognition belong to the supervised learning problem category. Supervised learning problems are further subdivided into *easy learning* and *hard learning* problems. An *easy learning* problem has training examples that specify input values and desired activations for all intermediate and output neurons. By contrast, *a hard learning* problem gives desired activations only for the final output neurons but it does not provide any information for the intermediate neurons. *Hard learning* problems can be further divided into *free-network* problems and *fixed-network* problems. The difference between the two networks is that change to the network topology is allowed only in

free-network problems, while the weights are the only parameters that can be adjusted in fixed-network problems.

2.1.5 Fundamental Network Models

Multilayer perceptrons (MLP's) and back-propagation networks (BPN's) are the two most common neural network models. Multilayer perceptrons use discrete values for both neuron inputs and activations, and every neuron in this model computes its activation as a linear discriminate (or threshold logic function):

$$S_j = \sum_{i=0}^{p} w_{ji} y_i$$

$$u_j = \begin{cases} +1 & (true) & if\ S_j > 0 \\ -1 & (false) & if\ S_j < 0 \\ 0 & (unknown) & if\ S_j = 0 \end{cases}$$

where u_j is a new activation at neuron j and S_j is a linear combination of neurons' input signals.

On the contrary, back-propagation networks need a continuous value range [0,1] or [-1,1] for neuron inputs and activations. In addition, back-propagation networks have three distinct characteristics (Simon 1994):

1. Each neuron in the network calculates its output by a nonlinear activation function. The activation function should be differentiable everywhere. Usually, the *logistic function*

$$y_j = \frac{1}{1 + e^{-v_j}}$$

is used as the activation function. To have the activation function range from -1 to +1, an antisymmetric function, such as hyperbolic tangent, is used (Haykin, 1994).

2. The network contains one or more layers of *hidden neurons* that are not the input or output layer of the network. These neurons in hidden layers enable the network to learn complex tasks by extracting progressively more meaningful features from the input patterns (vectors).
3. The network exhibits a high degree of *connectivity*, determined by the neurons of the network. A change in the connectivity of the network requires a change in the population of neuron connections, their weights or both.

Back-propagation networks are very popular, since with this kind of network an *error back-propagation* algorithm can be applied to hard learning problems. The basic technique of *error back-propagation* algorithms is discussed in the next subsections.

2.1.6 Error Back-Propagation Algorithms

In a practical application of a back-propagation algorithm, learning results from the experience space consisting of many presentations of training examples. During the learning process, one complete presentation of a set of training patterns in the experience space is called an *epoch*. The learning process is maintained on an epoch by epoch basis until the neuron weights and threshold levels of the network stabilize and the average squared error over the entire training set converges to a minimum value. There are two basic ways to train ANNs: *pattern mode* and *batch mode* (Haykin, 1994). In the *pattern mode* of back-propagation learning, weight updating is performed after the presentation of each training example. Generally, this type of learning mode runs quickly. By contrast, in the *batch mode* of back-propagation learning, weight updating is performed after the presentation of

all the training examples. It has been reported that the *batch mode* is more powerful than the *pattern mode* since this mode considers the patterns in the set as a whole rather than as individuals. A disadvantage of the *batch mode* is that it naturally runs slower.

The weights are updated during the learning process. *Gradient descent* plays an important role in most connectionist learning algorithms as a basic technique for adjusting weights. In order to train an artificial neural network, we need a set of training patterns. Each pattern is associated with an input vector $X\{x_1,...,x_p\}$ and a desired response vector $D\{d_1,...,d_m\}$. Each input vector X entering a neural network N produces a functional output vector $Y_k\{y_1,...,y_m\}$. If an error signal vector is defined as

$$E = D - Y \qquad (2\text{-}6)$$

This error signal vector can then be used to adjust the weights vector $W_{ji}\{w_1,...w_l\}$.

Suppose there is a differentiable function E that takes a set of weights $W_{ji}\{w_1,...,w_j\}$, then the training example may produce an error vector $E(W_{ji})$ for these weights. The revised weight vector W^* can be computed by

$$W^*_{ji} = W_{ji} - \eta \Delta E(W_{ji}) \qquad (2\text{-}7)$$

where $\Delta E(W_{ji})$ is the gradient direction in weight space that would result in a maximum increase of the error when an infinitesimally small weight change is made in that direction. η is a small positive number that governs the descent step size.

The training process continues using *gradient descent* repeatedly until weight adjustments vector $\Delta E(W_{ji})$ approaches zero.

2.1.7 Least Mean-Squares and Momentum Least Mean-Squares Algorithms

The Least Mean-Squares (LMS) Algorithm has attracted a great deal of research interest as a popular technique used for the error back-propagation process. Convergence conditions are critical for LMSA. Feuer and Weinstein (1985) presented a statistical analysis of the least mean-square adaptive algorithm with uncorrected Gaussian data. Gaussian data refers to the input set $X_0\{x_1, x_2, \ldots x_l\}$ which has zero mean. They derived the necessary and sufficient conditions for the convergence of the algorithm within a finite variance as shown in the following equation:

$$0 < \mu < \frac{1}{\lambda_j}, \qquad j = 1,2,3,\ldots n \qquad (2\text{-}8)$$

$$\sum_{j=1}^{n} \frac{\mu\lambda_j}{1-2\mu\lambda_j} < 1 \qquad (2\text{-}9)$$

where μ is a constant gain which controls the rate of convergence of the algorithm, and λ_j are the eigenvalues of R, where R is defined as:

$$R = E\{X(k)X(k)^T\} \qquad (2\text{-}10)$$

where $X(k)$ is the data vector at the k^{th} time instant. The resulting algorithm is characterized by:

$$W(k+1) = (I-2\mu R)W(k) + 2\mu E\{X(k)d(k)\}$$

where $W(k)$ is the weight vector of the k^{th} iteration cycle and $d(k)$ is the desired signal.

In practical applications, it may be very difficult to use these functions continuously although their forms appear very simple. The problem is that it is not easy to identify whether all the vector data are independent in n dimensional space.

The Momentum Least Mean-Squares algorithm (MLMS) is a second-order weight data expression. The MLMS algorithm is a general form of LMS algorithm and is useful in applications where error bursting is a problem because of its smoothing effect. Roy and Shynk (1990) found that the convergence analysis of MLMS yields novel behavior which leads to complex eigenvalues of the transition matrix for the mean weight vector. They indicated that the MLMS algorithm becomes unstable when $|\alpha| \rightarrow 1$, where α is a coefficient of the first-order partial derivatives of the error signal function in the MLMS equation of the MLMS (see Equation (2-11)). Several computer simulation examples supported their conclusion, that is, while the MLMS algorithm has smoother convergence, no significant gain can be expected in convergence speed over the conventional LMS algorithm.

In order to apply MLMS to the error back propagation algorithm, a general formula for calculating weight adjustments was derived by Hagiwara (1992) based on the following two assumptions:.

1. $E_n = \sum_{n=1}^{N} E(n)$ where $E(n)$ is the sum of squared error of the n^{th} training pattern at the output layer and N is the total number of training patterns.
2. The most recent weights are assumed in calculating E^n, that is

$$\Delta w_{ji}(n) \propto -\frac{\partial E(n)}{\partial w_{ji}(n)}$$

Using a logistic activation function, he expressed the learning rule as

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1) \qquad (2\text{-}11)$$

where n represents the iteration number, $\delta_j(n)$ a local gradient, η a learning rate, and α a constant which determines the effect of the past weight changes upon the current direction of movement in weight space (see Equation (3-9)). For consistency in this proposal, the notations in Equation (2-11) have been changed from Hagiwara's notations.

2.2 Application of ANNs to Transportation Engineering

2.2.1 Overview

ANNs offer the following useful properties and capabilities:

- Nonlinearity
- Input-Output mapping
- Adaptability
- Contextual information
- Fault tolerance
- Very Large Scale Integration (VLSI) implementation ability
- Uniformity of analysis and design
- Neural biological analogy

These properties of ANNs can be exploited to efficiently solve transportation problems. Transportation problems are often very complicated and simple analytical solutions may not be able to fully represent reality. Faghri (1992) reported that ANN models can be applied to all transportation problems, that is, planning, operation and control, administration and finance, construction and maintenance, and design. However, there has been relatively little research or practical application of ANNs to the field of transportation engineering, although this situation is changing. Since 1989, a few

macroscopic traffic simulation programs using the characteristics of ANNs models have been developed. They have been applied to traffic control problems such as short-term prediction of traffic variables, traffic-responsive selection of timing plans, and traffic assignment. Examples of applications of ANN models to transportation engineering are briefly summarized in the following subsections.

2.2.2 Planning

An ANN model with the ADaptive Linear Element (ADALINE) model was used by Faghri and Hua (1992) to calibrate parameters for trip generation forecasting. This is a very simple neural network with a linear activation function, four input neurons, and one output neuron. The inputs neurons reflect four socioeconomic indeces which are believed to be significant for trip production:

1. the number of single family houses
2. the number of permanent single family residents
3. the number of multi-family dwelling units
4. the number of families with single automobile ownership

The trip rate is the output neuron.

Another ANN model that has a hidden layer with two output neurons was also discussed in the same paper in which a linear transfer function was used. The error back-propagation algorithm was applied in the model training process. Twenty patterns were used to train these two networks and seven patterns were used to test the models.

Their results indicated that ANN models worked better than linear regression models in this particular application. The results of the ADALINE based model were slightly closer to the observed values than those of the ANN model with back-propagation. The authors thought that the small difference between these two models might have resulted from the stacked errors during iterations since the floating data type was used for computations. Another noteworthy finding was that the training of the *error back-propagation* model is much more efficient than the ADALINE based model. The latter took 10,000 iterations whereas the former took only 2,500 iterations to reach the minimum error during the training of the testing pattern set.

2.2.3 Operation and Control

In 1991, a multilayer neural network model was introduced by Nakatsuji and Kaku (1991a) to develop a self-organizing traffic control system which would optimize the signal phase splits. The system operation had two parts: the training process and optimization process. The back-propagation method was used in the training process, and the model produced a steady input-output relationship between the splits and the measures of effectiveness.

Their ANN model consisted of three layers of neurons. Control variables (splits of signal phases) were inputs and objective variables (traffic variables such as queue lengths or other performance indexes) were outputs. They reported that it would be possible to optimize phase splits that vary with time when the time sequence of splits and traffic variables are used as control variables. Assuming an isolated intersection with four legs controlled by two signal phases and a number of cycle periods of N, the population

of neurons of the input layer is 2N and that of the output layer becomes 4N when phase splits which vary each cycle are estimated. The population of neurons in hidden layers is modified to test its effect on the outcome (Nakatsuji and Kaku, 1991b).

In the optimization process, a stepwise method combining the *Cauchy machine* with a *feed-back* method was proposed. The Cauchy machine is a Monte Carlo method and gives the weight adjustments using a statistical method. It was introduced to accelerate the convergence and avoid the entrapment into local minima. The *feed-back* method is based on the steepest descent method and makes the weight adjustments in a deterministic way. Both training and testing patterns that include inputs and outputs were produced by the dynamic equation which is used in the TRANSYT program.

They used a typical 4-leg intersection controlled by a 3-phase traffic signal. Each approach had 2 lanes - 2 inflow links. Hence, there were 3 neurons in the input layer and 8 neurons in the output layer. The 3 input neurons represented 3 phases whereas the 8 output neurons represented the queue length of each inflow link. The error was calculated by the root mean square (RMS) method. The neural network was trained with a conventional digital computer.

Nakatsuji and Kaku (1991b) conducted a complementary study which focused on finding the relationship between the architecture of ANN and training abilities, and on performing the initial training effectively. Effects of the number of neurons in a hidden layer were discussed, and a multiple split model was described in detail. Split, offset and

inflow traffic values were used as input. Measures of effectiveness, such as queue lengths or performance indexes, were defined as output signals. Major findings in that paper were summarized by the author as follows:

1. Numerous intermediate layers and neurons in the layers did not always improve the training abilities of neural networks.
2. The multiple split model significantly improved both the estimation error and the computation time.
3. The multi-channel model, in which neural networks for both offsets and traffic volumes were installed at every intersection and joined together at an intermediate layer, performed best.
4. Increasing the number of training patterns while allowing some patterns to slightly exceed the threshold was very effective.

Another traffic-control related neural network model called the Neural Signal Control System (NSCS) was developed by Hua et. al (1995). A mechanism of human visual nerves was incorporated in the model. The principle considered in the model was that the phase pattern currently displayed was always, among all available patterns, the one which carried the highest traffic demand measured in the link group or phase basis.

The NSCS is a partial-connect multilayer network. This model contains a total of 5 layers. The first layer is divided into several slabs. Each of these slabs included some neurons that were dedicated to a specific group of links and contain traffic information of different movements. In the second layer, a competition takes place on a "*winner take*

all" basis. The third layer is used to determine which group should be currently considered based on the output of the second layer. The weights of the connections change in an incremental or decremental manner as time is measured incrementally. The step size of adjustment for weights is governed by user-specified maximum green time interval and the minimum green time interval.

A case study was conducted to compare the performance of this model with that of SOAP. A hypothetical isolated intersection with four approaches was used as a test case. All approaches had a left-turn pocket and a through lane. No right-turn traffic was considered because its effect would be insignificant.

A simulation-neural network model has been developed to evaluate dilemma zone problems at high-speed signalized intersections (1993). The levels of effectiveness of traffic control at a high-speed intersection approach were expressed as the probability of being caught in a dilemma zone, the speed of a vehicle in different segments of the intersection approach, and the vehicle conflict rate. The ANN developed model employed the back-propagation method. It had 27 neurons in the input layer and one neuron in the output layer. There were two hidden layers. The root mean square error varied between 1 and 5 percent. 8900 observations at the four high-speed signalized intersects were used to train and test the ANN model.

In addition to the logit function, fuzzy logic was introduced to create activation functions in an ANN model (Hsiao et al., 1994). A Fuzzy Logic Incident Patrol System

(FLIPS) was developed to solve many of the problems inherent in traditional incident-detection algorithms. The system can be constructed automatically from training examples to find the optimal input and output membership functions. An empirical database, which includes 5177 observations collected in Toronto, was used to evaluate the potential effectiveness of the FLIPS.

2.2.4 Construction and Maintenance

A general-purpose microcomputer-based neural network software called Neural Works was used in the study of priority-rating problems (1988). In 1994, Fwa and Chan separately tested the ability of this simple back-propagation neural network with a linear function, with a nonlinear function, and with subjective priority assessments obtained from engineers to relate priority ratings to pavement conditions. They reported that the test results were positive, and that neural network based priority rating methods could be used for the maintenance planning at the network level.

2.2.5 Design

In 1994, Pant and Balakrishnan developed an ANN with a back-propagation algorithm to study the behavior of gap acceptance by drivers at intersections with stop signs that involved complex interactions of numerous geometric, traffic, and environmental factors. This ANN model contained one hidden layer with a linear activation function.

In total, 5230 patterns were used to train and test the ANN model. The test data set accounted for 24 percent of the total patterns. The results produced from the neural network model and from the binary-logit model were compared with the observations

recorded in the field. The comparative analysis revealed that the neural network model predicted the percentage of accepted or rejected gaps more accurately than the binary-logit model.

2.3 Optimization Algorithm

Finding an optimal solution of signal timing for an isolated signalized intersection could be a typical operations research problem if the relationships between measurement of effectiveness (MOE) and traffic, geometric, and signalization conditions could be mathematically described. Although there are many methods and algorithms available, not all operation research problems can be solved effectively by traditional mathematical methods. On the other hand, if these relationships are not very clear or if they cannot be described mathematically, it is necessary to find other approaches. The literature review on optimization techniques focus on the following two topics:

- State Space Search
- Heuristic Search

The discussions on these two topics are largely based on the study of Luger and Stubblefield (1993).

2.3.1 State Space Search

State space search was defined as below (Luger and Stubblefield, 1993):

> "N is the set of nodes or states of the graph, These correspond to the states in a problem-solving process.

A is the set of arcs (or links) between nodes. These correspond to the steps in the problem-solving process. S, a nonempty subset of N, contains the start state(s) of the problem. GD, a nonempty subset of N, contains the goal state(s) of the problem. The states in GD are described using either:

1. A measurable property of the state encountered in the search.
2. A property of the path developed in the search.

A solution path is a path through this graph from a node in S to a node in GD. "

For the optimization of traffic signal timing, the state of GD can be defined by a measurable property such as the minimum average stopped delay per vehicle of signalized intersection. The optimization of traffic signal timing is obtained from the path to reach the state GD. The significant advantage of state space search strategy is that it guarantees finding the global solution except in combination problems. An interesting question is how to find the state of GD. In other words, it is necessary to review the strategies for state space search. Briefly speaking, the strategies of state space search include the search direction and the search order.

Data-Driven and Goal-Driven Search

Search direction is the first consideration when applying the state space search. A state space may be searched in two directions: data-driven and goal-driven. In data-driven search, or sometimes called a forward chaining, the problem solver starts from the given facts of the problem. The search proceeds by applying rules to facts to produce new facts, which are in turn used by the rules to generate more new facts. The process

terminates when it generates a path that satisfies the goal condition. In goal-driven search, the problem solver begins with the goal, finds the rules that can produce the goal, and chains backward through successive rules and subgoals to the given facts of the problem.

Whether the data-driven or goal- driven approach is used depends on the problem's characteristics. Luger and Stubblefield (1993) pointed that a data-driven search is appropriate for the problem if

1. All or most of the data are given in the initial problem statement, and / or
2. There are a large number of potential goals, but only a few approaches to use the facts and given information of a particular problem.

Otherwise, a goal-driven search is suggested if

1. A goal or hypothesis is given in the problem statement or can easily be formulated; and / or
2. There are a large number of rules that match the facts of the problem and thus produce an increasing number of conclusions or goals; and / or
3. Problem data are not given but must be acquired by the problem solver.

Generally, an optimization problem should use data-driven search since the goal state is a potential solution of the problem and it is unknown at the beginning of the search.

Depth-First and Breadth-First Search

In addition to specifying a search direction (data-driven or goal-driven), a search order or priority must be determined when designing a search algorithm. There are two main possibilities: depth-first search and breadth-first search.

In depth-first search, when a state is examined, all of its children and their descendants are examined before any of its siblings. Since depth-first search always goes the deepest in the state space, it is not guaranteed to find the shortest path to a state the first time that state is encountered.

Breadth-first search, in contrast, explores the space in a generation-by-generation fashion. Only when there are no more states to be explored at a given generation does the algorithm move on to the next generation. Because breadth-first search considers every node at each generation before going deeper into the space, all states are first reached along the shortest path from the start state. Thus, it is guaranteed to find the shortest path from the start state to the goal.

According to the above discussion, it is obvious that when the search problem is not related to a search path, either the depth-first or breadth-first search strategy should be considered.

2.3.2 Heuristic Search

The design heuristic searches has long been a core concern of artificial intelligence research. Heuristics here is defined as rules for choosing those branches in a state space that are most likely to lead to an acceptable problem solution. Heuristics can be

combined with an expert system so that they can effectively proceed search in most cases. Like all rules of discovery and invention, heuristics can lead a search algorithm to a suboptimal solution or fail to find any solution since heuristics use limited information and they are seldom able to predict the exact behavior of the state space farther along in the search. This is the weakness of heuristic search. In order to overcome the inherent limitation of heuristic search, understanding the behaviors of the state space is most important.

Artificial intelligent (AI) problem solvers employ heuristics in two basic situations (Luger and Stubblefield, 1993):

1. A problem may not have an exact solution because of inherent ambiguities in the problem statement or available data.
2. A problem may have an exact solution, but the computational cost of finding it may be prohibitive.

The second situation is what this dissertation research addresses. The response time is a critical issue in dynamic traffic signal control. The optimal algorithm design for signal timing must not only consider the optimal solution of signal timing that results in the minimum traffic delay, but also the amount of time spent to obtain the solution. Therefore it is necessary to study the strategy of heuristic search.

Best-First Search

Best-first search is a general algorithm for heuristically searching any state space. It is equally applicable to data- and goal-driven searches and supports a variety of heuristic evaluation functions. The goal of best-first search is to find a goal state by looking at as few states as possible. Logically, the more informed the heuristic, the fewer states are processed in finding the goal.

The process of implementing best-first search is similar to "*hill climbing*" (Pearl 1984). Best-first strategies expand the current node in the search and evaluate its children. The best child is always selected for further examination. The search terminates when it reaches a state that is better than any of its children.

The problem with best-first search is that the algorithm cannot recover from getting into infinite loops since no history is kept, and does not guarantee a global solution.

2.4 Summary of Literature Review

Based on the literature review of this chapter, it was found that:

1. Currently the Level of Service of signalized intersections is often analyzed by the methodology introduced by the Highway Capacity Manual (HCM). Since the parameters discussed in HCM were statistically estimated based on the nation wide survey and research, the result of this method might not reflect the traffic operation prevailing at a specific area.

2. Relationships between the average stopped delay and the traffic environment are often too complex to be described mathematically. Little effort has been made to analyze level of service of signalized intersection by using ANNs.
3. Previous ANN models used in signal timing were developed by greatly simplifying the traffic environment. More parameters need to be added to the ANN models before applying them to practical traffic control.
4. Few ANN models have been tailored for traffic signal control. Previous studies usually just applied commercial ANN models to traffic engineering.
5. Further research is needed to study the learning behavior of ANNs. A great effort must be made to find the best mix of architecture, learning process, and the type of learning rate for an ANN model.
6. For a Level of Service analysis or optimal signal timing to be done dynamically, the running time of the algorithm is crucial. Combining a neural network and a heuristic search seems to meet this requirement.

CHAPTER 3: METHODOLOGY OF DEVELOPING OTSCS

3.1 Overview of Methodology

As indicated in the literature review, much effort has been made to apply ANNs to transportation engineering but continuous improvements are needed to make them ready for practical use. The goal of this research was to develop a computer software called Optimal Traffic Signal Control System (OTSCS) which integrated a LOS Analysis Neural Network and an Optimal Traffic Signal Timing Model (OTSTM). OTSCS as a whole is an artificial intelligence system. This system not only applies to ATMSs but also allows the user to study characteristics of neural networks and optimal solution search methods. OTSCS' framework is shown in Figure 3-1. The main features of the system are described below:

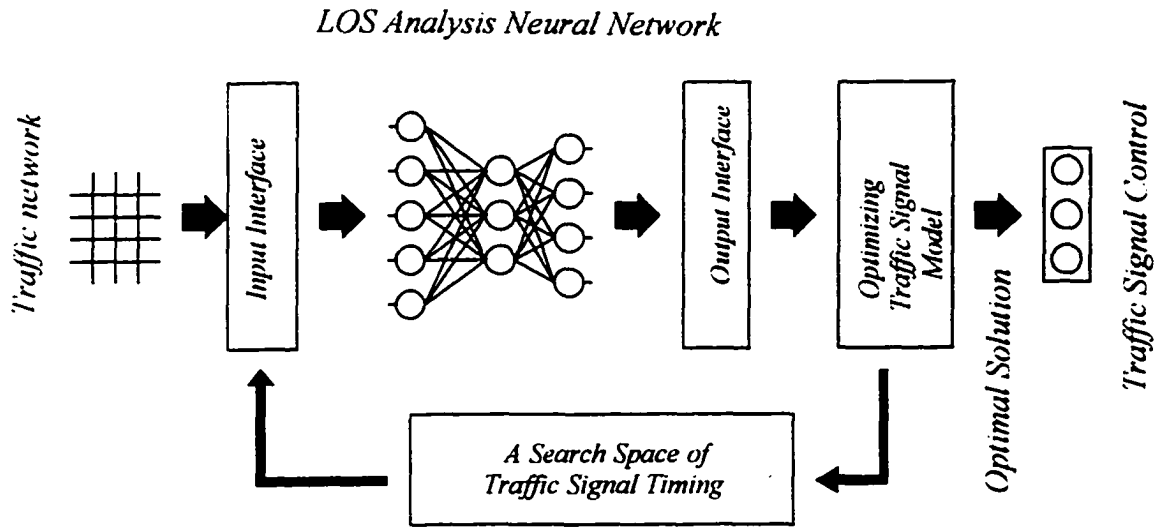


Figure 3-1 Framework of OTSCS

1. The input interface of the trained ANN receives two kinds of information: the traffic environment data and the range of traffic signal cycle length. The traffic environment information includes geometric and traffic conditions of each

intersection in the street network. The range of traffic signal cycle length bound by the user and traffic management requirements provides a scope for the OTSCS to search for the optimal signal timing that minimizes the vehicle delay at intersections in the traffic network. The search space might include thousands of different selections of signal timing.

2. Network data, such as traffic conditions, geometric conditions, and assumed signal timing conditions, are entered into the LOSANN through the input interface. The LOSANN evaluates the Measure of Effectiveness (MOE), and the result(s) is sent to its output interface. The MOE could be defined by the average stopped delay per vehicle or the queue length behind the stop lines. (See Section 3.2 and 3.3)
3. An optimum traffic signal timing model (OTSTM) that uses one of the available heuristic search methods is connected to the input and the output interface of LOSANN. Following the rules determined by the selected search model, the program creates a solution search loop. Each time only one possible traffic signal timing is sent to the input layer of LOSANN. By comparing the current activation signal with the previous alternative at the output of LOSANN, the optimization program replaces the previous signal timing by the current one if the MOE of the current signal timing is better than that of the previous one. Otherwise, the previous option holds. OTSTM is able to find an optimal traffic signal timing, or to find a suboptimal timing within a very short time period that is acceptable for real time traffic signal control (See Chapter 6).

3.2 Patterns and Pattern Collection

Level of Service analysis neural network (LOSANN) is the "brain" of OTSCS. OTSCS works in term of knowledge that is learned but not innate. Before integrating it into OTSCS, LOSANN must be trained and tested carefully. In this section, the types of pattern used to train and test LOSANN are described.

3.2.1 Input and Output of LOSANN

The pattern set for training and testing LOSANN consists of two parts: input signal(s) and output signal(s). Input signals are used to represent a specific traffic environment, such as traffic, geometric, and signalization conditions of a signalized intersection. The input factors related to traffic operation at an intersection are those used by Chapter 9 of the 1994 HCM and are listed in Table 3-1:

Table 3-1 Inputs of LOSANN

Type of Condition	Index	Parameter	
Geometric conditions	1	Number of lanes	
	2	Lane width	ft
	3	Grade	%
	4	Parking	ft
Traffic conditions	5	Volumes by movement	vph
	6	PHF	%
	7	Heavy vehicle	%
	8	Bus Stops	
	9	Arrive type	
	10	Conflict pedestrian	
Signalization conditions	11	Green times	sec
	12	Yellow times	sec
	13	Phase plan	
	14	Lost time	sec

The output signal(s) is a Measure of Effectiveness (MOE) which corresponds to a given set of input information describing the intersection's environment under study. The MOE should be able to reflect the level of service at an intersection. There are a few MOEs used to describe the service quality of an intersection, such as queue length, fuel consumption due to delay, and the average stopped delay per vehicle. In this study, the latter is used as LOSANN's output.

3.2.2 Pattern Collection

The pattern set (experiential knowledge space) plays a significant role in the development of LOSANN. The ideal approach is to collect the necessary data directly in the field. This first hand data set is always the best to train and test any kind of ANN model. Collecting data on traffic, geometric and signalization conditions are relatively simple. Usually, signal timing is given and all necessary geometric conditions can be measured in the field. To count traffic and turning movements, a variety of detectors, such as inductive loop detectors, radar detectors, photocell detector, and video camera, can be used.

On the other hand, collecting MOEs in the field, such as average per vehicle delay or queue length, is difficult. According to the recommendation of the 1994 HCM, the stopped delay per vehicle at a signalized intersection can be estimated by the stopped vehicle counts taken in the field. It can be calculated by the following equation:

$$D = (V_s I) / V \qquad (3\text{-}1)$$

where : D = stopped delay per vehicle (sec/veh)

V_s = sum of stopped - vehicle counts

I = count interval

V = volume during study period

Although the equation is crude, it provides an adequate measurement if it can reflect the level of service at that intersection. But, this is a time consuming method unless it is done automatically by a sensor.

When MOE data are not available from field surveys, their values are estimated by traffic analysis software packages, such as HCS, SOAP, PASSERII, or TRANSIT-7F, can be used as surrogate values to train and test LOSANN. When the analysis results of these packages are used, we have to accept the fact that the relationships developed in the LOSANN might not necessarily reflect the reality in the field. However, the capability of the LOSANN model remains valid. In this study, the MOE values were collected from the reports of HCS (version 2.4).

3.3 Tool for Developing OTSCS

An IBM compatible PC computer is the only equipment needed to develop and use OTSCS. OTSCS was developed with Microsoft Visual Basic 5.0® on a Microsoft Windows 95® platform. A PC with 133MHz Pentium processor or higher is strongly suggested to train LOSANN since the training usually consumes a considerable amount of time.

OTSCS consists of the LOS Analysis Neural Network and the Optimal Traffic Signal Timing Model. It also has modules to perform data conversion and reporting.

The Graphic User Interface (GUI) which accompanies OTSCS facilitates all these applications. Appendix B contains a user manual for the OTSCS software.

CHAPTER 4: DEVELOPMENT OF LEVEL OF SERVICE ANALYSIS NEURAL NETWORK

4.1 Overview of LOSANN

LOSANN is a multilayer flat connection network with an *error back-propagation algorithm*. Though there is no guidelines navigating what kind of ANN model is most suitable for traffic signal control, the following four condition statements tell us what might be an appropriate option.

1. The evaluation of Measures of Effectiveness (MOE) is an extremely complex task. The number of factors necessary to represent the environment is so large that it may require the LOSANN to add one or more layers of hidden neurons to learn this task by extracting more meaningful features from input vectors.
2. It is impossible to make all the input and output values, such as peak hour factors, turning movements, the number of lanes, and average stopped delay, discrete into either {0, 1} or {-1,1}. All these values must be normalized within the continuous interval [0,1].
3. This is a supervised learning problem since an MOE is used to check whether the activation in the output set is correct or not.
4. It is a "hard learning problem". This means that the patterns enable us to correct only the activation for the output neuron, rather than to correct all activations for the intermediate neurons.

As an independent model, LOSANN consisted of three modules:

1. Data conversion module. This program is used to filter the information in the Highway Capacity Software (HCS) output reports, convert them to the designed pattern format, and send them to the input neurons of LOSANN.
2. Neural network selection module. This module provides the user various types of ANNs. The best selection chosen through training and testing is kept as LOS Analysis Neural Network.
3. Report generation module. This module gives the user an opportunity to review the training and testing results.

In the following three sections, these modules are described in detail.

4.2 Data Conversion Module

For training and testing LOSANN, patterns consisting of data collected from field surveys and HCS reports must be fed into the ANN model. To prepare patterns for the ANN model, the information was filtered and normalized into [0, 1].

4.2.1 Information Filter

To develop an ANN model that can be utilized in traffic signal control systems, it is necessary to paste as much of the major factors affecting MOE as possible to the input layer of the ANN model. HCM requires 14 factors to determine average stopped delay per vehicle at a signalized intersection and evaluate its traffic level of service. All of these factors could be input data to LOSANN (see Table 4-1). Five factors (the number of lanes, the volumes by movement, green times, yellow times and phase plane) are absolutely necessary. The others are optional. A conversion module was programmed to

gather all or part of these factor values and the intersection delay from the detailed report of HCS based on user's requirement.

Figure 4-1 shows a typical HCS detailed report for signalized intersections.

Figure 4-1 HCS Report

```
HCM: SIGNALIZED INTERSECTION SUMMARY   Version 2.4        04-09-1997

================================================================================
Streets: (E-W) 149th STREET            (N-S) MELROSE AVENUE
Analyst: J. Fan                        File Name: 149_3MSE.HC9
Area Type: CBD                         12-2-95 ST PEAK
Comment: EXISTING CONDITIONS (Part 2)
================================================================================
           |   Eastbound  |  Westbound   |  Northbound  |  Southbound
           | L    T    R | L    T    R | L    T    R | L    T    R
           |---- ---- ----|---- ---- ----|---- ---- ----|---- ---- ----
No. Lanes  |      2  <    |      2  <    |    > 2  <    |      1  <
Volumes    |      557   37|      397   20|  25  327    7|       30   50
PHF or PK15|     0.96 0.96|     0.90 0.90|0.94 0.94 0.94|     0.96 0.96
Lane Width |     11.0     |     11.0     |     15.0     |     12.0
Grade      |        0     |        0     |        0     |        0
% Heavy Veh|        8    8|        6    6|  25   25   25|       13   13
Parking    |(Y/N) Y    20 |(Y/N) N       |(Y/N) N       |(Y/N) N
Bus Stops  |            10|            10|            24|           19
Con. Peds  |          1000|          1000|          1000|          10
Ped Button |(Y/N) N       |(Y/N) N       |(Y/N) N       |(Y/N) N
Arr Type   |        2    2|        2    2|   2    2    2|        2    2
RTOR Vols  |           0  |           0  |           0  |           0
Lost Time  |     3.00 3.00|     3.00 3.00|3.00 3.00 3.00|     3.00 3.00
Prop. Share|  -1        -1|  -1        -1|  -1        -1|  -1        -1
Prop. Prot.|            -2|            -2|            -2|           -2
--------------------------------------------------------------------------------
                          Signal Operations
Phase Combination 1     2     3     4 |           5     6     7     8
EB  Left                              |NB  Left    *
    Thru          *                   |    Thru    *
    Right         *                   |    Right   *
    Peds          *                   |    Peds    *
WB  Left                              |SB  Left
    Thru          *                   |    Thru    *
    Right         *                   |    Right   *
    Peds          *                   |    Peds    *
NB  Right                             |EB  Right
SB  Right                             |WB  Right
Green          32.0P                  |Green    26.0P
Yellow/AR       5.0                   |Yellow/AR 5.0
Cycle Length:  68 secs  Phase combination order: #1 #5
--------------------------------------------------------------------------------
                    Intersection Performance Summary
     Lane  Group:   Adj Sat    v/c     g/C                    Approach:
     Mvmts    Cap     Flow    Ratio   Ratio   Delay   LOS    Delay   LOS
     -----   ----   -------   -----   -----   -----   ---    -----   ---
EB   TR      1296     2593    0.501   0.500    10.9    B      10.9     B
WB   TR      1483     2966    0.328   0.500     9.6    B       9.6     B
NB   LTR     1089     2644    0.368   0.412    12.2    B      12.2     B
SB   TR       298      723    0.279   0.412    11.7    B      11.7     B
              Intersection Delay =  10.9 sec/veh Intersection LOS = B
Lost Time/Cycle, L =    6.0 sec   Critical v/c(x)     = 0.441
--------------------------------------------------------------------------------
```

Table 4-1 Factors Affecting Measurement of Effectiveness at a Signalized Intersection

Index	Factors		Characteristic
1	Number of lanes		Necessary input
2	Lane width	ft	Optional input
3	Grade	%	Optional input
4	Parking	ft	Optional input
5	Volumes by movement	vph	Necessary input
6	PHF	%	Optional input
7	Heavy vehicle	%	Optional input
8	Bus Stops		Optional input
9	Arrival type		Optional input
10	Conflict pedestrian		Optional input
11	Green times	sec	Necessary input
12	Yellow times	sec	Necessary input
13	Phase plan		Necessary input
14	Lost time	sec	Optional input
15	Average stopped delay per vehicle	sec	Output

4.2.2 Data Normalization

All of the data, no matter whether they are input data or output data, must be normalized to a range between 0 and 1. The following equation was used to normalize the data:

$$\hat{x}_i = \frac{x_i - x_{i\min}}{x_{i\max} - x_{i\min}} \qquad (4\text{-}1)$$

where

$\hat{x}_i$ = a normalized measurement with the index i listed in Table 4-1,

x_i = a measurement with the index i listed in Table 4-1,

x_{imax} = the maximum measurement,

x_{imin} = the minimum measurement.

The desired output data, d_i, can also normalized by replacing x in Equation (4-1) by d.

For convenience, x_i and d_i are used as normalized values in the subsequent sections

4.3 Neural Network Selection Module

4.3.1 Architecture of LOSANN

The general architecture of LOSANN is shown in Figure 4-2. It is a flat connection artificial neural network with multiple layers since error back-propagation algorithm will be employed. Neurons and arcs comprise the network topology. Each arc has a numerical weight, $w_{ji,k(k-1)}$, that specifies the influence of neuron *i* in the *k-1* layer on neuron *j* in the *k* layer. The subscript of the weight, $w_{ji,k(k-1)}$, has two parts. The first part of the subscript, *ji*, indicates the positions of relative neurons in layer k and k-1 respectively. The second part is the layer index. The weight, $w_{ji,k(k-1)}$ determines the behavior of the network. Further notation appears in subsequent discussions of the model.

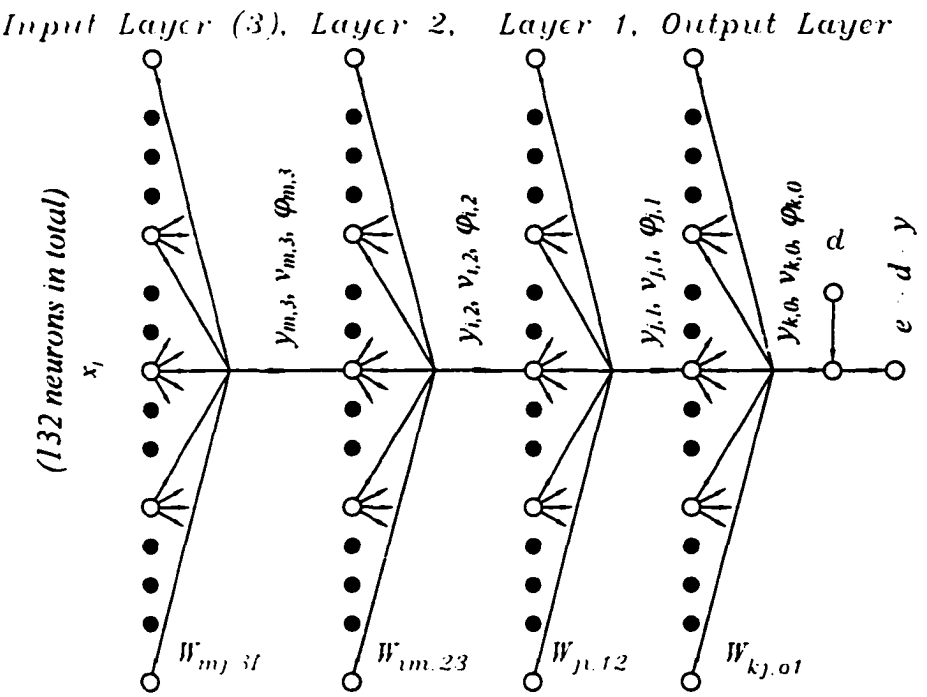


Figure 4-2 Architecture of LOS Analysis Neural Network

The number of input neurons is determined by the factors used to describe the traffic environment. In this study, the number of input neurons is set to132 at the maximum (see Table 4-2). There was only one neuron in the output layer for estimating average stopped delay per vehicle for the entire intersection.

Table 4-2 Input Factors and Neurons

Type of Condition	Parameter	Number of Input Neurons
Geometric conditions	Number of lanes	$4 * 3^a = 12$
	Lane width ft	$4 * 3^a = 12$
	Grade %	4
	Parking maneuvers	4
Traffic conditions	Volumes by movement, vph	$4 * 3^a = 12$
	PHF	$4 * 3^a = 12$
	Heavy vehicle %	$4 * 3^a = 12$
	Bus Stops	4
	Arrive type	$4 * 3^a = 12$
	Conflict pedestrian	4
Signalization conditions	Green times sec	$2 * 2^b = 4$
	Yellow times sec	$2 * 2^b = 4$
	Phase plan	$4*3*2^c = 24$
	Lost time sec	$4 * 3^a = 12$

[a] Generally, there are three movements, such as left turn, through, right turn, in each approach of a 4-leg intersection

[b] Assume that each signal has two phases.

[c] Assume that a phase plan requires two phases and three movements in each approach.

4.3.2 Learning Process

The learning process of LOSANN is shown in Figure 4-3. The learning process includes two basic signal flows, the activated signal flow and the error signal flow, which run in opposite directions. The forward pass of the activated signal flow starts at the input layer of LOSANN. Each pattern is associated with an input signal vector $x\{x_1,...,x_p\}$ and a desired response signal vector $d\{d_1,...,d_m\}$. The elements of the input signal vector x are

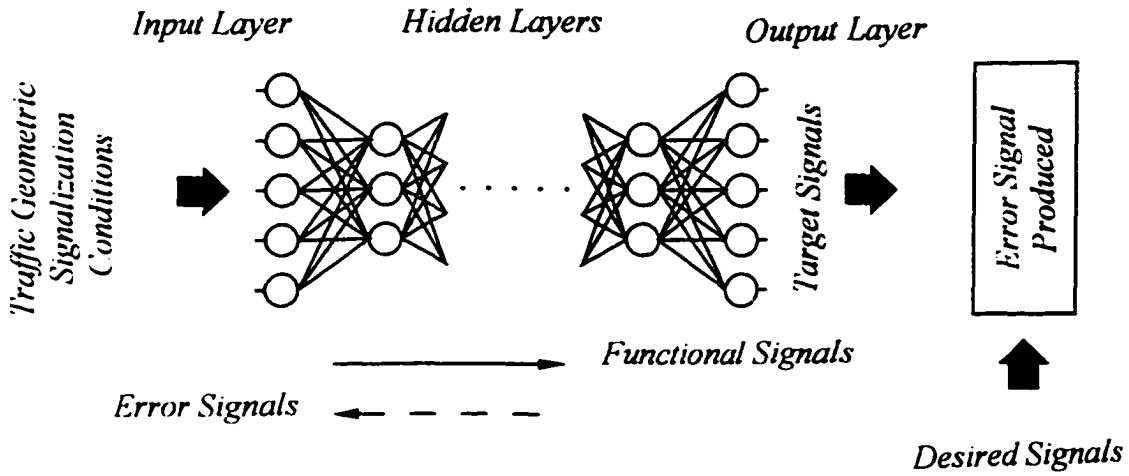


Figure 4-3 Back-propagation learning process in LOSANN

listed in Table 4 -2 and there is only one element, which is the average stopped delay per vehicle for the entire intersection, in the desired response signal vector. When an input signal vector x enters a neural network, LOSANN generates an activated signal at each neuron by

$$y_{j,k}(n) = \varphi(v_{j,k}(n)) \qquad \forall n \in N,\ \forall j \in J,\ \forall k \in K \qquad (4-2)$$

$$\varphi(v_{j,k}(n)) = \frac{1}{1+e^{-v_{j,k}(n)}} \qquad \forall n \in N,\ \forall j \in J,\ \forall k \in K \qquad (4-3)$$

$$v_{j,k}(n) = \sum_{i=0}^{I} w_{ji,k(k-1)}(n) y_{i,k-1} \qquad \forall n \in N,\ \forall j \in J,\ \forall i \in I,\ \forall k \in K \qquad (4-4)$$

where N is the size of the training pattern set and n is the index of the training pattern. K is the number of layers (the input and the output are included). J and I are the number of neurons in the *kth* and the *(k-1)th* layer, respectively. The subscript before the comma is the index of the neuron and second part of the subscript is the layer index of the neural network. For example, $w_{ji,k(k-1)}(n)$ refers to the value of weight on the arc which connects neuron i in the *(k-1)* layer and neuron j in the k layer. $\varphi(*)$ represents an activation function which

must be smooth (i.e., differentiable) everywhere. $v_{j,k}(n)$ is a linear combination function. Regarding Equation (4-4), two things must be explained:

1. $y_{0,k} = -1$ $\quad k \neq$ *Output Layer*

2. $y_{i,1} = x_i$

The first item above indicates that the linear combination function includes the threshold, whose value is -1. The second indicates that the activated signal is replaced by the input signal in the input layer of the neural network.

The activated signal vector, y_k, that consists of $y_{j,k}$, is propagated forward layer by layer. LOSANN generates an output signal vector y at its output layer. After the output signal vector has been compared with the desired response vector, an error signal vector is calculated by

$$e(n) = d(n) - y(n) \qquad n \in N \qquad (4-5)$$

where $e(n)$ = the vector of error signals

$d(n)$ = the vector of desired signals

$y(n)$ = the vector of the output signals

If an error signal vector exists, the weights in the network must be modified. The process of modifying the weights is the process of training the neural network which may be called "the learning process of the neural network".

There are two basic approaches to train ANN models: *pattern mode* and *batch mode* (Haykin, 1994). In this study, both of these learning modes are used. The OTSCS software allows the user to choose either one. Using the *pattern mode* of back-propagation learning,

weight updating is performed after the presentation of each training pattern. Therefore, the error function is defined as the *instantaneous sum of squared errors*

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \qquad (4\text{ -}6)$$

where

$$e_j(n) = d_{j,0}(n) - y_{j,0}(n) \qquad j \in C \qquad (4\text{ -}5a)$$

It is obvious that Equation (4 -5a) is the scalar version of the Equation (4 –5). In Equation (4-5a), $e_j(n)$ is an error signal, $d_{j,0}(n)$ is a desired signal, and $y_{j,0}(n)$ is an activated signal in the output layer for each pattern. C is the set of the neurons in the output layer and the coefficient ½ is added for convenience in subsequent calculations.

By contrast, using the *batch mode* of back-propagation learning, weight updating is performed after the presentation of all the training examples. In this case, the error function $E(n)$ must be replaced by the *average squared error*, E_{av}, which is defined as

$$E_{av} = \sum_{n=1}^{N} E(n) \qquad (4\text{ -}7)$$

where $E(n)$ is the instantaneous sum of squared errors defined as Equation (4-6). N notes the size of the total training patterns and n is the index of the training pattern.

No matter what kind of back-propagation learning mode is used, the backward pass starts at the output layer by passing the error signals backward through the network. All the weights in the network are modified by

$$w_{ji,k(k+1)}(n+1) = w_{ji,k(k+1)}(n) + \Delta w_{ji,k(k+1)}(E(n)) \qquad (4\text{ - }8)$$

In Equation (4-8), $\Delta w_{ji,k(k+1)}(E(n))$ is the weight adjustment, which is discussed in Section 4.3.3.

During the learning process, one complete presentation of the training patterns in the experience space is called an *epoch*. The learning process proceeds on an epoch-by-epoch basis, until the neuron weights and threshold levels of the network stabilize and the average squared error over the entire training set converges to a minimum value. This type of learning process is defined as the *Error Back-Propagation algorithm*. LOSANN is an artificial neural network model with *Error Back-Propagation algorithm*.

4.3.3 Weight Adjustment

This section discusses how the weights in LOSANN are adjusted. Since LOSANN allows both the pattern learning mode and the batch learning mode, weight adjustment equations are derived for both training modes.

Pattern Mode

For the pattern mode of back-propagation learning process, a Least Mean-Squares (LMS) algorithm was used. The LMS used in this study is written as

$$\text{Minimize} \qquad E(n) = \frac{1}{2}\sum_{o \in C} e_j^2(n) \qquad (4 - 9)$$

where $E(n)$ is defined as Equation (4-6). For convenience, the definitions of the error signal, $e_j(n)$ and of the functional signal, $y_{j,k}(n)$, are repeated as follows:

$$e_j(n) = d_{j,o}(n) - y_{j,o}(n) \qquad j \in C \qquad (4 - 5a)$$

$$y_{j,k}(n) = \varphi(v_{j,k}(n)) \qquad j \in J; \quad k \in K \qquad (4 - 2)$$

where J is the neuron set in layer k; K the layer set of LOSANN. As mentioned in Section 4.3.2, $\varphi(*)$ is an activation function; $v_{j,k}(n)$ is the linear combination function of neurons:

$$v_{j,k}(n) = \sum_{i \in C_{k+1}} w_{ji,k(k+1)}(n) y_{i,(k+1)}(n) \quad j \in J;\ i \in C_{k+1}\ \ k \in K \qquad (4-4)$$

where C_{k+1} is the neuron set in layer $k+1$.

The *delta rule* was used to calculate the weight adjustment. The *delta rule* states that the weights must be modified in such a way that guarantees the weight adjustment move along the gradient descent direction in the weight space and reach the minimum sum square error at the end. The *delta rule* is mathematically written as

$$\Delta w_{ji,k(k+1)}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji,k(k+1)}(n)} \qquad (4-10)$$

where $E(n)$ is an error function defined by the learning process mode; η is a parameter. Equation (4-10) has two parts: the gradient direction of the weight space and a parameter, η, called "learning rate". The use of minus sign in the equation accounts for gradient descent. The learning rate governs the step size of descent. Characteristics of various learning rates are discussed in Section 4.3.4.

Now, let us derive the equation of the weight adjustment for LOSANN with a single layer architecture. In this case, there are only the input layer and the output layer. The input layer is indexed as 1 and the output layer as "0". Neuron j located in the output layer is supplied with the desired response for the pattern. Using the chain rule of calculus, Equation (4-10) can be written as

$$\Delta w_{ji,01}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji,01}} = -\eta \frac{\partial E(n)}{\partial e_{j,0}(n)} \frac{\partial e_{j,0}(n)}{\partial y_{j,0}(n)} \frac{\partial y_{j,0}(n)}{\partial v_{j,0}(n)} \frac{\partial v_{j,0}(n)}{\partial w_{ji,01}(n)}$$

$$= -\eta e_j(n) \frac{\partial \varphi(v_{j,0}(n))}{\partial v_{j,0}(n)} x_i(n) \qquad (4\text{-}11)$$

where $e_j(n)$ is defined by Equation (4-5a); $v_{j,0}(n)$ by Equation (4-4). Note that $y_{i,1}(n)$, the result of $\frac{\partial v_{j,0}(n)}{\partial w_{ji,01}(n)}$, in Equation (4-11) is replaced by $x_i(n)$, which is defined as the input signal on the ith input neuron . If the local gradient $\delta_{j,0}(n)$ is defined by

$$\delta_{j,o} = -\frac{\partial E(n)}{\partial v_{j,0}(n)} \qquad (4\text{-}12)$$

Then Equation (4-11) can be rewritten as

$$\Delta w_{ji,21}(n) = \eta \delta_{j,0}(n) x_i(n) \qquad (4\text{-}11a)$$

To apply Equation (4-11), the derivative of the activation function, $\varphi'(v_{j,0})$, must be calculated. When the activation function, $\varphi(v_{j,0})$ is defined as the logistic function:

$$\varphi(v_{j,0}) = \frac{1}{1+e^{-v_{j,0}}}$$

the weight adjustment is calculated by

$$\Delta w_{ji}(n) = \eta e_j(n) y_j(n)[1 - y_j(n)] x_i(n) \qquad (4\text{-}11b)$$

since

$$\delta_{j,0} = -\frac{\partial E(n)}{\partial v_{j,0}} = e_j(n) y_j(n)[1 - y_j(n)]$$

A general formula used to calculate the weight adjustment is derived in Appendix A. The difference between the single layer model and multilayer model is that the error signals transmitted to a hidden neuron should be recursively determined by the error signals of the directly connected neurons (Simon, 1994). The general formula is written as

$$\Delta w_{ji,k(k+1)}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji,k(k+1)}(n)} = \eta \delta_{j,k}(n) y_{i,(k+1)}(n) \qquad (4\text{-}13)$$

where $\delta_{j,k}(n)$, the local gradient for neuron j on layer k, is defined as

$$\delta_{j,k} = -\frac{\partial E}{\partial y_{j,k}} \frac{\partial y_{j,k}}{\partial v_{j,k}} = \frac{\partial \varphi(v_{j,k}(n)}{\partial v_{j,k}(n)} \sum_{m \in C_{k-1}} \delta_{m,(k-1)} w_{mj,(k-1)k} \qquad (4\text{-}14)$$

Equation (4 –13) is similar to Equation (4-11b). Since the logit function (4-3) is used as the activation function in LOSANN, the local gradient for neuron j on layer k can be rewritten as

$$\delta_{j,k}(n) = y_{j,k}(n)(1 - y_{j,k}(n)) \sum_{m \in C_{k-1}} \delta_{m,(k-1)}(n) w_{ji,(k-1)k}(n) \qquad (4\text{-}14a)$$

Formula 4-13 is readily used in the pattern mode of back-propagation learning which examines each training pattern (example) in the course of computing a set of weights for LOSANN.

Batch Model

If the batch mode of back-propagation learning is applied, the control function should be defined as

$$\text{Minimize} \qquad E_{av} = \sum_{n=1}^{N} E(n) \qquad (4\text{-}15)$$

where E_{av} is the *average squared error*. *E(n)* defined as Equation (4-6) is the sum of the squared errors of the pattern with the index *n* at the output layer. *N* is the total number of training patterns in the database. Using the *delta rule*, a general formula for calculating weight adjustments was derived by Hagiwara (1994). Replacing *E(n)* in Equation (4-10) with E_{av} for the output neuron, $y_{j,o}$, the weight adjustment of the single layer LOSANN with batch learn mode is

$$\Delta w_{ji,01} = -\eta \frac{\partial E_{av}}{\partial w_{ji,01}} = -\sum_{n=1}^{N} \eta \frac{\partial E(n)}{\partial y_{j,0}(n)} \frac{\partial y_{j,0}(n)}{\partial v_{j,0}(n)} \frac{\partial v_{j,0}(n)}{\partial w_{ji,01}(n)}$$

where *N* is the total training patterns in the experience space. Introducing the concept of the local gradient defined by Equation (4-11), the formula shown above can be rewritten as

$$\Delta w_{ji,01} = -\eta \frac{\partial E_{av}}{\partial w_{ji,01}} = \eta \sum_{n=1}^{N} \delta_{j,0}(n) x_i \qquad \text{(4-16a)}$$

In a general setting, for the hidden neuron, $y_{j,k}$, the weight adjustment is

$$\Delta w_{ji,k(k+1)} = -\eta \frac{\partial E_{av}}{\partial w_{ji,k(k+1)}} = -\eta \sum_{n=1}^{N} \frac{\partial E(n)}{\partial w_{ji,k(k+1)}(n)} \qquad k \neq \textit{Output Layer}$$

Substituting Equation (4-13) into the above formula, we have

$$\Delta w_{ji,k(k+1)} = -\eta \frac{\partial E_{av}}{\partial w_{ji,k(k+1)}} = \eta \sum_{n=1}^{N} \delta_{j,k}(n) y_{i,(k+1)}(n) \qquad k \neq \textit{Output Layer} \quad \text{(4-16b)}$$

Based on Equations (4-16a) and (4-16b), the weight adjustment is reorganized as

$$\Delta w_{ji,k(k+1)}(n) = -\eta \frac{\partial E_{av}}{\partial w_{ji,k(k+1)}(n)} = \eta \sum_{n=1}^{n} \delta_{j,k}(n) y_{i,(k+1)}(n)$$

$$= \eta \delta_{j,k}(n) y_{i,k+1}(n) + \alpha \sum_{n=1}^{n-1} \delta_{j,k}(p) y_{i,k+1}(p)$$

$$= \eta \delta_{j,k}(n) y_{i,k+1}(n) + \alpha \Delta w_{ji,k(k+1)}(n-1) \qquad (4\text{-}16)$$

where η is a learning rate, and α a constant which determines the effect of the past weight changes upon the current direction of movement in weight space; n the index of training pattern; $\delta_{j,k}$(n) a local gradient which is defined as Equation (4-14).

The new weight in the ANN with the batch learning mode can be calculated by

$$w_{ji,k(k+1)}(n+1) = w_{ji,k(k+1)}(n) + \eta \delta_{j,k}(n) y_{i,k+1}(n) + \alpha \Delta w_{ji,k(k+1)}(n-1) \qquad (4\text{-}17)$$

4.3.4 Learning Rate

It was mentioned that the learning rate in the weight adjustment formulas controls the step size so as to keep the search procedure moving in the direction of the descent gradient. The learning rate is very important but so far there is no efficient method available to determine its magnitude for ANN models. Usually the learning rate is a positive constant whose value is experimentally determined. In this study, a constant learning rate was initially used.

Two other types of learning rates were then derived and used in this study. They were named *linear learning rate* and *weighted exponent learning rate,* respectively. Considering the fact that the error signal becomes smaller and smaller as the algorithm continues, it is reasonable to think that a variable learning rate will be helpful in training LOSANN. The *linear learning rate* was defined in this study as

$$\eta_l = \eta_0 (\lambda - \frac{I}{I_{max}}) \qquad (4\text{-}18)$$

where η_0 and λ are positive constants; I the epoch index; and I_{max} the maximum number of epoch. λ is used to control the minimum value of the linear learning rate. For example, if λ is 1.03 and η_0 is 0.3, the linear learning rate will change from 0.309 to 0.09.

On the other hand, the *weighted exponent learning rate* was defined in this study as

$$\eta_e = (\beta + (0.5 - \beta)\frac{I}{I_{\max}})e^{-\frac{I}{I_{\max}}} \qquad (4\text{-}19)$$

where β is a positive constant parameter. Equation (4-19) is a combination of a linear part and an exponential part. The magnitude of the exponent part decreases from 1 to 0.3679 as the algorithm proceeds. The linear part of the equation functions as a weight and its magnitude changes between β and 0.5 as the number of learning epoch increases. The constant value 0.5 is used to control the learning rate, η_e, not to become too small.

4.3.5 Flow Chart of Training LOSANN

The training procedure of LOSANN is represented by a flow chart shown in Figure 4-4. Sixteen steps are involved in the training:

1. Determine ANN model. Two options are available:
    a. to create a new model, go to the next step.
    b. to load an existing model, go to Step 14.
2. Set the characteristics of the new model. The characteristics of a new model include the model architecture, the number of neurons, the mode of learning, and the type of *error back-propagation* algorithm.

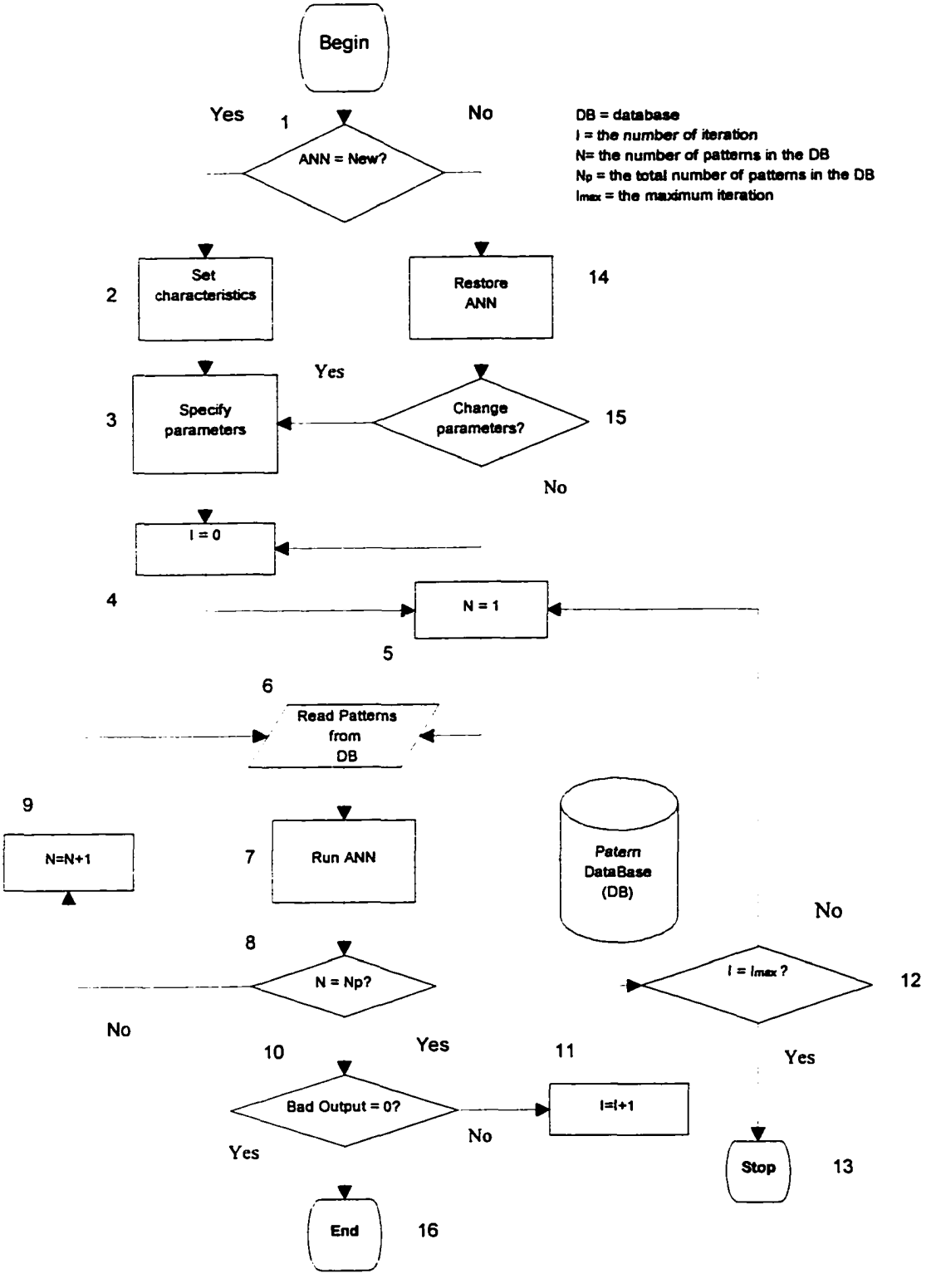


Figure 4-4 Flow chart of training procedure

3. Specify parameter values, such as the initial weights, learning rate η, the tolerance ε, and the maximum iteration I_{max} to run the model.
4. Set an initial iteration number I = 0.
5. Set an initial pattern number N = 1.
6. Read a pattern from the pattern database.
7. Train the ANN model.
   a. Evaluate the activation signal, that is, the Measure of Effectiveness.
   b. Count the number of bad outputs if an error signal is larger than the tolerance.
   c. Adjust weights
8. Check the number of patterns used to train the ANN.
   a. Go to the next step, if the pattern is not the last pattern in the database.
   b. Go to Step 10, if the pattern is the last one in the database.
9. Point to the next pattern in the database and go to Step 6.
10. Determine whether the training procedure needs to continue.
    a. Go to Step 15 if the number of accumulated bad outputs is equal to zero.
    b. Go to the next step if the number of accumulated bad outputs is larger than zero.
11. Point to a new iteration or epoch (I = I + 1).
12. Check whether the iteration has exceeded the maximum iteration.
    a. Go to Step 5 and start a new iterative procedure if $I \leq I_{max}$.
    b. Go to Step 13 Stop training if $I > I_{max}$.
13. Stop training.

14. Restore the existing ANN model

15. Determine whether the parameters need to be renewed.

    a. Go to Step 3, if the answer is "Yes".

    b. Go to Step 4, if the answer is "No".

16. End the training procedure.

4.4 Report Generation Module

As mentioned before, an artificial neural network can be compared to a black box. To select the best ANN model to analyze level of service at a signalized intersection, it is necessary to analyze the learning behavior of ANN models, including the degree and the speed of convergence. For the convenience of analyzing learning behavior, LOSANN contains a report generation module. This module has two sections: the "Log" section and the "Explorer section". The log section saves all the information about training or testing ANN models. The explorer section can display the information by numerical data or graphics.

Once a pattern is passed through LOSANN during the training process, an error signal is generated. If the error signal is greater than the tolerance set by the user (e.g. the user sets the tolerance equal to 15% of the targeted value), the output of the pattern is counted as a bad output. After one learning epoch, the total number of bad outputs and the root mean square error (RMSE) are added to a temporary file. When the training process is completed, the temporary file can be saved as a permanent file. All these subroutines are included in the log section by the report generation module. Based on

the data in the permanent file, the learning behaviors of the specific ANN model can be studied. The log section is also able to keep the testing information.

The Explorer section contains the subroutines to open the data files generated by the log section and display these data through Microsoft Excel®. For details about the report generation module, see Appendix B.

CHAPTER 5: TRAINING AND TESTING LOSANN

5.1 Patterns Preparation

The input layer of LOSANN has 132 neurons in the maximum which represent traffic, geometric and signalization conditions (See Table 4-2). The output layer has one neuron representing the average stopped delay per vehicle at an intersection. Ideally, input and output patterns need to be collected by field observations. In this developmental study, the input data of the patterns were collected in the field from more than 60 intersections located in Bronx Center area and downtown Flushing. The output data of the patterns were obtained from delay analysis performed by Highway Capacity Software (HCS) since delay measurements in the field have not yet been perfected. This approach is justified in this study because its purpose is to test the learning ability of LOSANN. This is the key for applying LOSANN to an actual signal control system.

Determining the number of patterns needed to train and test an ANN model is a challenge which can only be solved by experience. Brainmaker®'s User Guide (1993) suggested estimating the number of training patterns by the following equation:

Training Facts = 2*(inputs + hiddens + outputs)

to 10 * (inputs + hiddens + outputs) (5-1)

The number of testing patterns is about 10% ~ 20 % of the total number of patterns. Following this suggestion, four hundred ninety patterns have been prepared using the data obtained from Bronx Center Transportation Study and Downtown Flushing Transportation Improvement Study. Both of these two studies were prepared by the New York City Department of Transportation. The original data included traffic flow (at

peak hours on weekdays and on Saturdays), geometric and signalization conditions. Among these patterns, 62.2% did not have through traffic in at least one approach of an intersection and 16.3% had more than 2 signal phases. No through traffic in an approach indicates that the street is one-way or that the intersection has no more than three legs. The situations represented by the patterns were diverse. Table 5-1 lists the value ranges of primary factors. Table 5-2 lists the Level of Service of these patterns. These patterns were partitioned into two groups, one for training LOSANN (80% of patterns) and another for testing LOSANN (20% of patterns).

Table 5-1: Maximum Value of Primary Factors in the Pattern Set

	Lane #	Lane width	Parking	# Vehicles	% Heavy Vehicles	Bus Stops	PHF	Conflict Pedestrians
Max.	6	15	20	3388	50	40	0.98	2444

Table 5-2. Level of Service in the Patterns

LOS	A	B	C	D	E	F
No. of Patterns	17	260	76	27	1	109
% in Total	3.47	53.06	15.51	5.52	0.20	22.24

5.2 Measurements of Training and Testing LOSANN

Two measurements were defined to identify the quality of training or testing LOSANN in this study. The number of bad outputs is one such measurement. A "bad output" refers to the activated signal in the output layer of LOSANN that produces an error signal whose value is beyond the pre-defined tolerance set by the user. In this study the tolerance was set at 15%.

The other measurement is a root mean square error (RMSE), that is

$$RMSE = \frac{\sum_{n=1}^{N}\sum_{i=1}^{I}\sqrt{(d_i(n) - y_i(n))^2}}{N * I} \qquad (5\text{-}2)$$

where

$d_i(n)$ = the desired signal loaded on the i^{th} neuron in the output layer, which is associated with the n^{th} pattern in the experience space.

$y_i(n)$ = the activated signal on the i^{th} neuron in the output layer, which is produced by the n^{th} pattern in the experience space.

I = the total number of neurons in the output layer. (in this study, I =1)

N = the total patterns in the experience space.

5.3 Discussion of a Single Layer LOSANN Model

A single layer LOSANN has 132 input neurons. Thus all of the factors listed in Table 4-2 are the input of this type of LOSANN. The pattern learning mode and the batch learning mode were separately used to train the single layer LOSANN. In this study, unless otherwise noted, the maximum running epoch was set to 3,000 and records were made every 60 epochs. Figures 5-1 and 5-2 plot the training results from the pattern learning mode when the learning rate η was set at 0.9, 0.5 and 0.3, respectively. These figures demonstrate that the learning process is convergent but is not stable. Unstable convergence is no surprise for training ANN models, since the learning rate η that controls the step size moving along the gradient descent direction was determined heuristically. In other words, it cannot be calculated by a mathematical method.

Figure 5-1 and Figure 5-2 also indicate that an ANN model with the lowest proportion of bad output does not always guarantee the smallest RMSE. Figure 5-1 shows that there were 69 bad outputs after running 1620 epochs, when the learning rate η was set at 0.3. This value is lower than the rate at the 3000^{th} epoch. But the latter had a

smaller RSME value (10.30%) than the former (RMSE = 10.62%, .See Figure 5-2). This relatively large RMSE resulted in bad outputs which increased by 8% later in the testing process.

Table 5-3. Training a Single Layer LOSANN with Different Learning Rate Types (I)

Rate Type	I_{max}	η	η_0	λ	β	Bad #	Bad %	RMSE
Constant	3000	0.3	-	-	-	76	19.4	0.1030
Linear	3000	-	0.5	1.03	-	67	17.1	0.0990
Exp.	3000	-	-	-	0.3	68	17.3	0.0999

Figures 5-3 and 5-4 show the training results from different learning types. The learning rates used for training and the final results are listed in Table 5-3. Here the constant learning rate was set at 0.3 since that was the best rake among the constant learning rates (see Figures 5-1 and 5-2). From Figures 5-3 and 5-4, we find that using a linear or weighted exponential learning rate, LOSANN produced fewer bad outputs than using a constant learning rate after running 3,000 epochs. Another observation was that a variable learning rate, regardless of whether it was of the linear type or weighted exponent type, was more effective in lowering RMSE than the constant rate type in LOSANN with the single layer architecture.

Figure 5-1: Bad Output vs. Running Epoch
(η = constant)

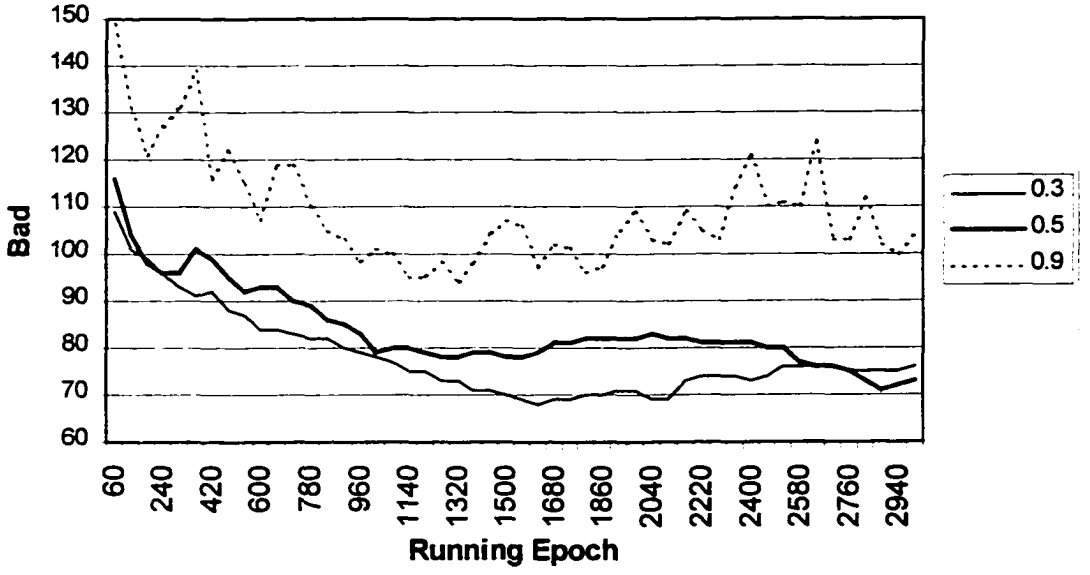


Figure 5-2: RMSE vs Running Epoch
(η=constant)

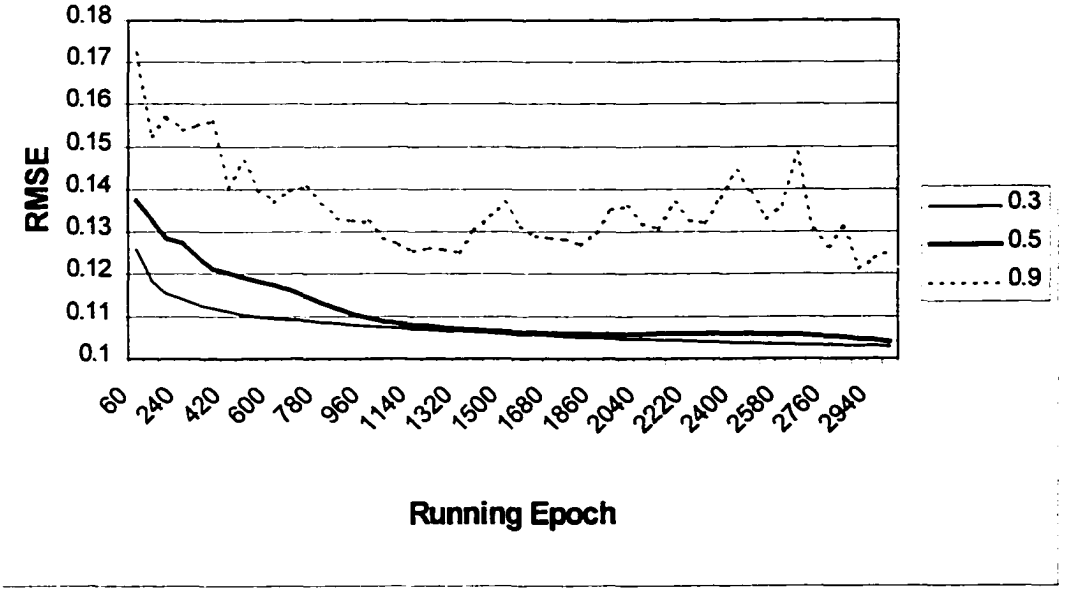


Figure 5-3: Bad output vs. learning rate type and running epoch

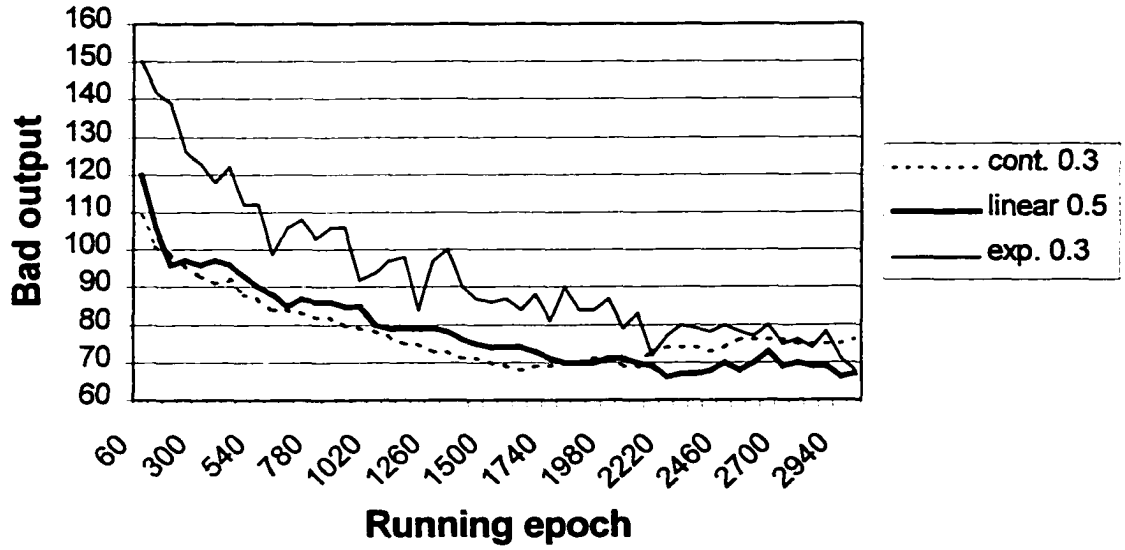


Figure 5-4: RMSE vs. learning rate type and running epoch

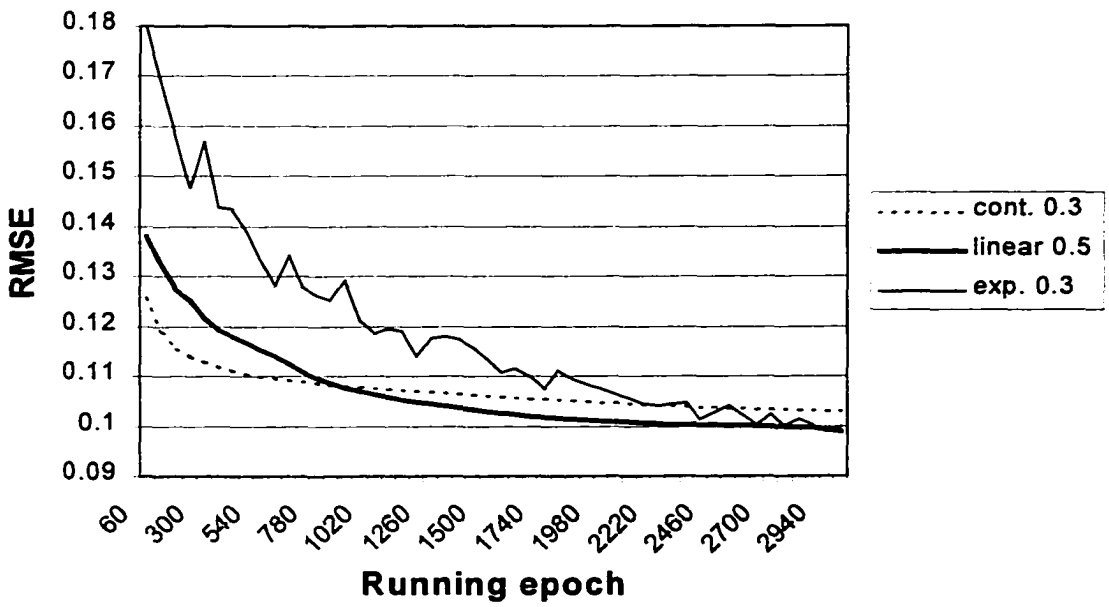
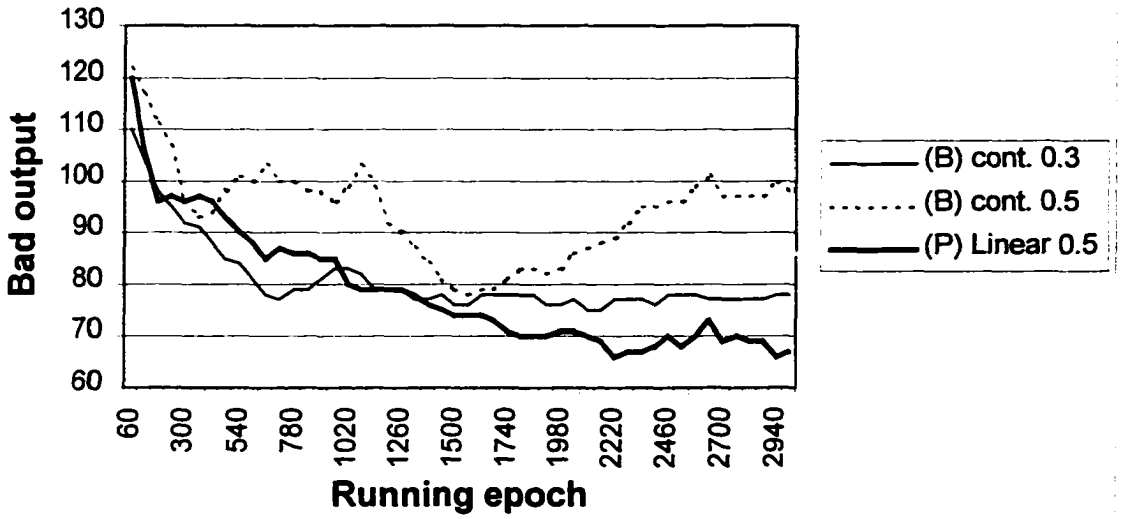
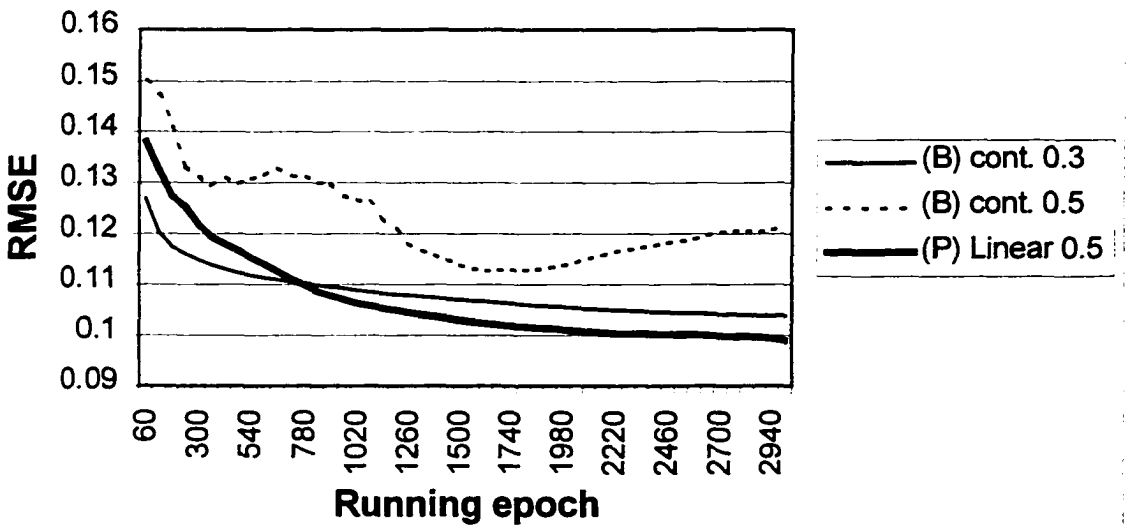


Figure 5-5: Bad output vs. running epoch (using different learning modes)



Note: In the legend of Figure 5-5, (B) indicates "Batch Mode"; (P) refers to "Pattern Mode".

Figure 5-6: RMSE vs. running epoch (Using different learning modes)



Note: In the legend of Figure 5-6, (B) indicates "Batch Mode"; (P) refers to "Pattern Mode".

Table 5-4 lists the training results when the maximum epoch was set at 15,000. We find that the convergence speed becomes very slow after 3,000 running epochs and the RMSE value becomes relatively stable at around 0.09.

Table 5-4 Training a Single Layer LOSANN with Different Learning Rate Types (II)

Rate Type	I_{max}	η	η_0	λ	β	Bad #	Bad %	RMSE
Constant	15000	0.3	-	-	-	64	16.32	0.0989
Linear (Eq. 4-18)	15000	-	0.5	1.03	-	65	16.58	0.0950
Exp. (Eq. 4-19)	15000	-	-	-	0.3	63	16.07	0.0923

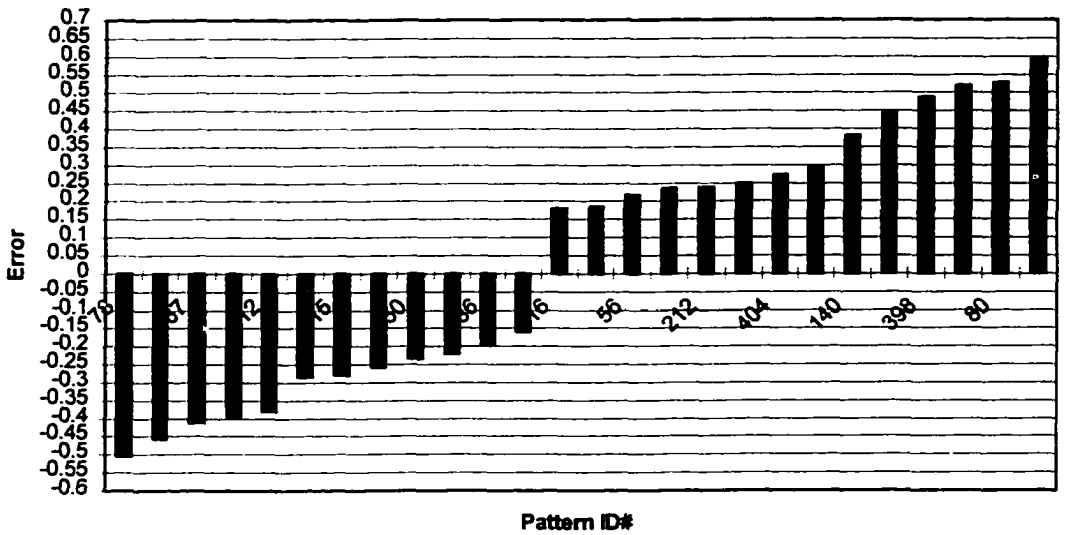
In addition to the pattern learning mode, the batch learning mode was also used to train LOSANN. Figures 5-5 and 5-6 show the results. For the batch mode, the learning rate η was set at 0.3 and 0.5, respectively, as well as α. We found that using pattern mode with a changeable learning rate, such as a linear learning rate, was more effective for this ANN model.

The test result is shown in Table 5-5. According to the data shown in Table 5-5, it is obvious that the best signal layer ANN model used for traffic analysis is the model with a linear learning rate. Figure 5-7 plots more detailed information about the test result using LOSANN with a linear learning rate when $\eta_0=0.5$ and $\lambda=1.03$. When the tolerance was set at 0.20 (the deviation of estimated average stopped delay per vehicle is within the range of ± 16 sec), the number of bad outputs dropped down to 22 (22.48%).

Table 5-5 Results of Testing (Tolerance = 0.15)

	Pattern Mode					Batch Mode
Rate Type	Constant			Linear	Exp.	
Parameter	$\eta=0.3$	$\eta=0.5$	$\eta=0.9$	$\eta_0=0.5$; $\lambda=1.03$	$\beta=0.3$	$\eta=0.3$ $\alpha=0.1$
# Bad	29	39	45	26	30	32
% Bad	29.59	39.80	45.91	26.53	30.06	32.65

Figure 5-7: Testing Result
(Learning Rate is a linear type)



5.4 Discussion of Multilayer LOSANN Models

Training an ANN is an art rather than science. In this section, multilayer LOSANN models are discussed in the following sequence:

1. One hidden layer LOSANN model, using pattern learning mode
2. One hidden layer LOSANN model, using batch learning mode
3. Two hidden layer LOSANN model, using pattern learning mode.

The focus of this analysis was on the learning behavior of LOSANN versus its architecture, learning rate and learning mode.

The patterns described in Section 5.1 were used to train and test multilayer LOSANN models. The neurons on the input layer and the output layer are equal to 124

and 1, respectively: Parking maneuvers and grade factors listed in Table 4-2 were excluded since the most patterns in the pattern set did not provide such information, thus reducing the number from 132 to 124. Tolerance levels of training and testing were set at 15% in either case. The number of bad outputs and the roots of mean square of error (RMSE) were the measures used to identify the learning ability of multilayer LOSANN models.

5.4.1 LOSANN with One Hidden Layer

The training and testing results of LOSANN with one hidden layer using a pattern learning mode are summarized in Table 5-6. The maximum training epoch was set at 3000. The first column of the table lists the neurons on the hidden layer. The last column of Table 5-6 shows the learning type used in the pattern learning mode. The linear learning rate type was defined by Equation (4-18). The parameter λ in Equation (4-18) was set at 1.03. The exponential learning rate type derived by Equation (4-19). The parameter β in Equation (4-19) affects the change of the learning rate. The next-to-last column shows the purpose of running LOSANN. The number of training patterns was 416; the number of testing patterns was 74.

To understand the learning behavior of LOSANN with one hidden layer and pattern back-propagation learning mode, the average bad output and average RMSE were calculated for different numbers of hidden neurons and learning rate types. The results are listed in Table 5-7. Based on these analysis results, we found that:

Table 5-6 Summary Results of Training and Testing One Layer LOSANN Using Pattern Learning Mode

(Maximum training epoch = 3000)

η (η_0, β) =	2		1.5		1		0.5		By	Rate Type
Neurons	Bad	RMSE	Bad	RMSE	Bad	RMSE	Bad	RMSE		
4	11	3.48%	10	3.83%	16	3.48%	10	3.06%	Train	Constant
4	16	12.87%	17	11.37%	14	10.36%	15	12.32%	Test	Constant
4	16	3.58%	16	3.67%	20	4.07%	8	3.15%	Train	Linear
4	17	12.59%	18	15.03%	16	13.54%	17	13.49%	Test	Linear
4	8	3.29%	16	3.99%	4	2.49%	3	2.24%	Train	Exponential
4	16	13.18%	21	12.79%	13	10.61%	15	12.17%	Test	Exponential
10	8	1.83%	7	2.40%	8	1.77%	5	2.29%	Train	Constant
10	18	11.56%	16	13.30%	16	10.81%	17	11.65%	Test	Constant
10	0	1.37%	1	1.67%	2	1.94%	7	2.48%	Train	Linear
10	23	14.59%	25	17.34%	20	16.40%	19	13.97%	Test	Linear
10	2	2.19%	4	1.84%	2	1.52%	2	1.66%	Train	Exponential
10	16	11.04%	20	14.11%	20	14.81%	16	12.87%	Test	Exponential
20	7	1.89%	5	1.18%	2	1.25%	3	1.44%	Train	Constant
20	16	11.98%	19	14.28%	15	10.07%	19	13.43%	Test	Constant
20	0	1.32%	0	1.20%	1	1.47%	3	2.04%	Train	Linear
20	19	15.27%	23	19%	15	11.05%	19	14.46%	Test	Linear
20	1	1.27%	3	1.31%	1	1.27%	1	1.60%	Train	Exponential
20	18	14.52%	21	16.47%	19	13.70%	20	13.28%	Test	Exponential
62							0	1.94%	Train	Constant
62							19	13.56%	Test	Constant

Note: The first row in Table 5-6 listed the values used as η for the constant learning rate type; η_0 for the linear learning rate type; and β for the weighted exponential learning rate type.

Table 5-7: General Learning Behavior of LOSANN with One Hidden Layer

(Pattern Mode, The Max. Epoch = 3000)

Hidden neurons	Constant		Linear		Exponential		By
	Bad	RMSE	Bad	RMSE	Bad	RMSE	
4	11.8	3.46%	15.0	3.62%	7.8	3.00%	Training
10	7.0	2.07%	2.50	1.86%	2.50	1.80%	
20	4.25	1.44%	1.00	1.51%	1.50	1.36%	
4	15.5	11.73%	16.3	12.93%	17.0	12.92%	Testing
10	16.8	11.83%	21.75	15.58%	18.00	13.21%	
20	17.25	12.44%	19.00	14.88%	19.50	14.49%	

1. The training process is convergent. LOSANN with one hidden layer has the ability to reduce RMSE to lower than 2%, and does not produce any bad output (See Table 5-6. There are four modes that have this ability)

2. The training process of LOSANN with one hidden layer is relatively stable compared with the single layer LOSANN (see Figures 5-1 to 5-6 and Figures 5-8 and 5-9).
3. The training quality improves as the number of hidden neurons increases for all three learning rate types (See the summary results in Table 5-7. The three learning rate types are constant, linear and weighted exponential types).
4. In general, the training quality measured by the number of bad outputs and RMSE improves when weighted exponential or linear learning rate type, rather than constant rate, is used. A weighted exponential learning rate results in the highest training quality, followed by a linear learning rate type, and constant type when the number of hidden neurons were fixed (see Table 5-7)
5. The LOSANN that has a better training quality does not always have a better testing result.

Batch learning mode was also used to train LOSANN with one hidden layer. The training and testing results for the batch learning mode are shown in Table 5-8. Comparing these data with the data listed in Table 5-6, we found that using a batch learning mode did not significantly improve the learning ability of LOSANN in this case. However, using batch learning mode spent much more time for training LOSANN than using pattern mode.

Figure 5-8: Bad Output vs. Learning Rate Type (One Hidden Layer with 20 Neurons)

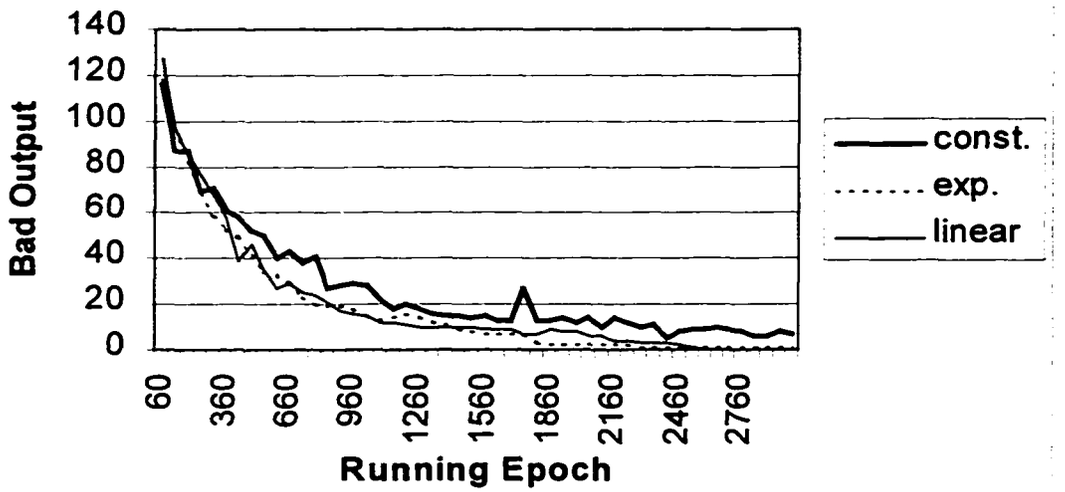


Figure 5-9: RMSE vs. Learning Rate Type (One Hidden Layer with 20 Neurons)

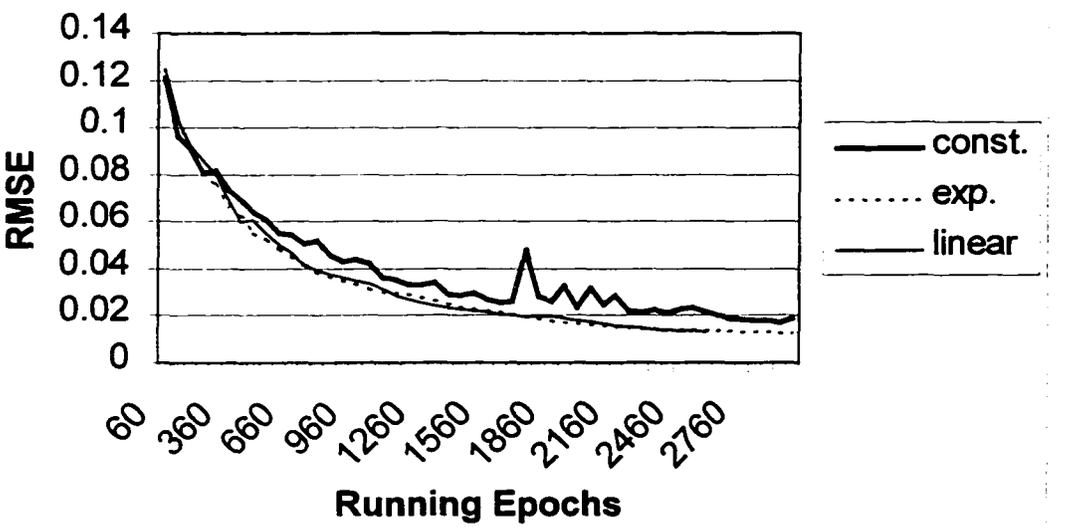


Table 5-8 Results of Training and Testing LOSANN Using Batch Learning Mode

(One Hidden layer)

Hidden neurons	η, η_0, β	2		1.5		1		0.5		By	Rate Type
	α	Bad	RMSE	Bad	RMSE	Bad	RMSE	Bad	RMSE		
4	0.5	7	2.45%	8	2.97%	19	3.74%	23	3.96%	Train	Constant
4	0.5	19	13.87%	18	13.06%	18	12.97%	27	18.11%	Test	Constant
4	0.9	11	3.54%	14	3.92%	10	3.45%	8	3.30%	Train	Linear
4	0.9	16	11.34%	17	11.69%	19	14.72%	19	16.69%	Test	Linear
4	0.5	23	4.01%	12	3.47%	8	3.65%	7	2.46%	Train	Exponential
4	0.5	18	12.93%	23	16.58%	24	16.02%	16	13.82%	Test	Exponential
10	0.9	2	1.4%	6	2.4%	6	1.5%	7	1.9%	Train	Constant
10	0.9	16	11.2%	16	12.5%	15	11.3%	16	13.1%	Test	Constant
10	0.9	2	1.9%	5	2.4%	1	1.9%	4	2.1%	Train	Linear
10	0.9	20	13.6%	18	13.5%	22	14.9%	16	12.1%	Test	Linear
10	0.9	4	2.1%	1	2.0%	2	1.5%	2	1.6%	Train	Exponential
10	0.9	19	13.3%	16	12.5%	17	13.1%	23	15.0%	Test	Exponential
20	0.9	5	1.6%	2	1.0%	1	1.6%	2	1.1%	Train	Constant
20	0.9	15	13.2%	21	17.3%	20	14.5%	22	13.6%	Test	Constant
20	0.9	0[a]	1.3%	2	1.2%	0[b]	1.4%	4	2.1%	Train	Linear
20	0.9	17	13.6%	22	14.8%	20	15.1%	20	15.1%	Test	Linear
20	0.9	0[c]	1.2%	0[d]	1.4%	1	1.5%	3	1.8%	Train	Exponential
20	0.9	20	12.5%	17	14.6%	18	15.5%	19	13.3%	Test	Exponential

Note:

The first row in Table 5-8 listed the values used as η for the constant learning rate type; η_0 for the linear learning rate type; and β for the weighted exponential learning rate type.

a. This is the result after running 2,340 epochs.
b. This is the result after running 2,940 epochs.
c. This is the result after running 2,339 epochs.
d. This is the result after running 2,750 epochs.

5.4.2 LOSANN with Two Hidden Layers

The learning behavior of LOSANN with two hidden layers was studied to evaluate the marginal increase in its training quality. The results of using a pattern learning mode to train and test the two LOSANN are listed in Table 5-9. Although only four cases were studied, the results showed that LOSANN with two hidden layers usually could neither increase the training convergence speed, nor improve the testing quality, as compared with the LOSANN with one hidden layer. There was only one exception which occurred when LOSANN had 4 neurons on the first hidden layer and 10 neurons on the second

hidden layer (the number of bad outputs was equal to 8 (10.88%)). This LOSANN model also has a high training quality. Its training RMSE was equal to 3.23% and the total number of bad outputs was 16 (3.84%).

Table 5-9 Results of Training and Testing LOSANN Using Pattern Learning Mode
(Two Hidden Layers The Max. Training Epoch = 3000)

η (η_0, β) =	2		1.5		1		0.5		By	Rate Type
Neurons	Bad	RMSE	Bad	RMSE	Bad	RMSE	Bad	RMSE		
4-10	5	2.28%	2	1.87%	8	2.01%	16	3.23%	Train	Constant
4-10	18	14.28%	14	9.44%	13	10.85%	8	7.91%	Test	Constant
4-4	15	3.46%	12	2.93%	16	3.27%	8	2.65%	Train	Linear
4-4	16	12.63%	19	13.54%	18	13.19%	17	13.20%	Test	Linear
4-10	3	3.00%	1	1.31%	6	1.65%	6	2.05%	Train	Exponential
4-10	14	13.27%	19	15.96%	17	12.48%	13	9.86%	Test	Exponential
10-10	5	1.34%	6	2.31%	5	1.59%	2	1.52%	Train	Constant
10-10	17	12.49%	18	12%	19	12.82%	23	15.18%	Test	Constant

Note: The first row in Table 5-9 listed the values used as η for the constant learning rate type; η_0 for the linear learning rate type; and β for the weighted exponential learning rate type

Based on the data listed in Table 5-6 and 5-9, we found two types of LOSANN models that have higher learning behavior than the others. One is Type H2C-4-10, which has two hidden layers with constant rates, and 4 neurons in the first hidden layer and 10 neurons in the second layer. Another is Type H1L-20, which has one hidden layer with linear learning rate and 20 neurons in its hidden layer. Table 5-10 displays the architecture, learning mode, learning rate type, learning rate, and the learning behavior of

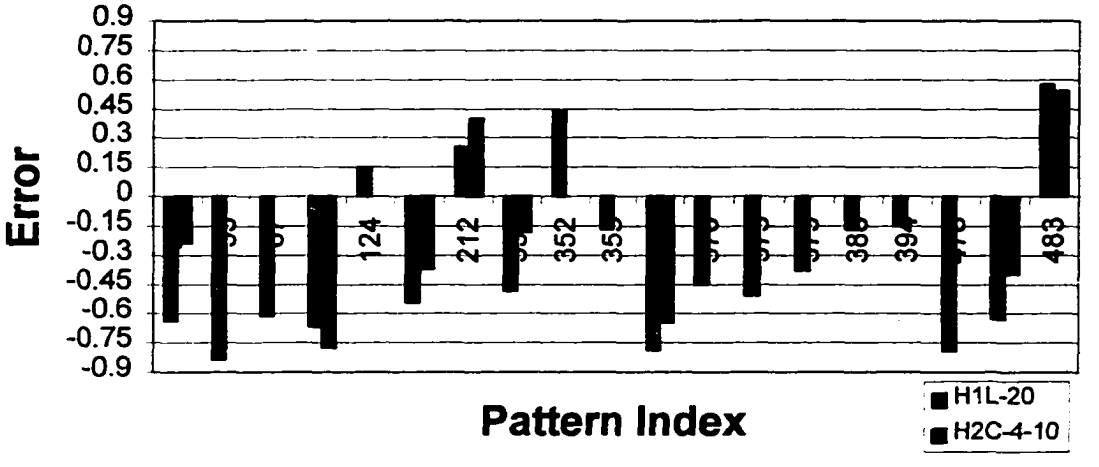
Table 5-10 H2C-4-10 and H1L-20 Types of LOSANN
(The Max. training Epoch = 3000)

Model Type	Hidden Layer	Hidden Neurons	Learn Mode	Learn Type	Learn Rate	Training		Testing	
						Bad	RMSE	Bad	RMSE
H2C-4-10	2	4-10	Pattern	Constant	0.5	16	3.23%	8	7.91%
H1L-20	1	20	Pattern	Linear	2	0	1.32%	19	15.27%

these two types of LOSANN. Type H2C-4-10 model is the "smarter" one. It only generated 8 bad outputs (10.88% of the testing patterns) and its RMSE is 7.91% during testing. Type H1L-20 is a good "student" since its Training RMSE is only 1.34%.

Figure 5-10 explores the testing results of these two types of LOSANN in detail. The horizontal axis of the figure is the pattern index that indicates the patterns, which generated bad outputs. The vertical axis displays the error values of these patterns. Figure 5-10 shows that the patterns which generated bad outputs when using Type H2C-4-10 of LOSANN also generated bad output when using Type H1L-20. This indicates that it is necessary to analyze the patterns that always generated bad outputs.

Figure5-10: H2C-4-10 Type and H1L-20 Type LOSANN



After analyzing the patterns that always generated bad output during testing H2C-4-10 and H1L-20 types of LOSANN, we found that the distribution of types of the training patterns was the critical issue. Although the number of training patterns is 416 for this analysis, there are only 66 patterns (15%) that had more than 2 phases, 11 patterns that had 1 phase (2.64%), and 8 patterns that had more than 3 lanes (1.92%) in the training pattern set (see Table 5-11). The patterns with only 1 phase are used as elements to analyze 5-leg intersections or other abnormal intersections in the NYCDOT's

studies. It is obvious that using this kind of pattern distribution to train LOSANN will affect its general knowledge development.

The distribution of these three patterns is shown in Table 5-11. The percentages of these patterns in the testing pattern set are higher that that in the training pattern set. Table 5-11 also lists the percentages of these three "problem" patterns among the bad pattern set, which were identified while testing H2C-4-10 and H1L-20 LOSANN models. The total ratio of the "problem" pattern is 5 out of 8 (62.50%) for the H2C-4-10 model and 7 out of 19 (36.84%) for H1L-20 model. It was found that the ratio of the "problem" patterns increased as the number of bad outputs decreased. These findings indicate that it was these patterns (the number of lanes is more than 3 in any approach of a intersection, traffic signal phase sequences more than 2 or equal to 1) that had a high probability of generating bad outputs.

Table 5-11: Testing Result Analysis

Patterns Type	No. of Pattern	Phases				Lanes		Total
		>2		=1		>3		
		No.	%	No.	%	No.	%	
Training Pattern	416	66	15.87%	11	2.64%	8	1.92%	20.43%
Testing Pattern	74	15	20.27%	3	4.05%	2	2.70%	27.03%
Bad pattern generated when testing (H2C-4-10)	8	2	25.00%	2	25.00%	1	12.50%	62.50%
Bad pattern generated when testing (H1L-20)	19	4	21.05%	2	10.53%	1	5.26%	36.84%

Judging from this analysis, the leaning behavior of LOSANN is expected to improve significantly if more patterns are used for training and types of training patterns are more evenly distributed.

5.5 Comment on the LOSANN Model

The following summarizes the discussions in Section 5.3 and 5.4:

1. LOSANN is a general model which can be used to analyze a variety of signalized intersections. This means that LOSANN has the ability to understand the complex relationships between average stopped delay and the specific factors included in the analysis, such as traffic volume, vehicle types, bus stops, parking conditions, pedestrian impacts, and so forth.
2. The learning behavior depends on the type of LOSANN. A LOSANN with one or two hidden layers learns more effectively and has higher testing quality than a single layer model (see Figure 5-9).
3. Using a variable learning rate type, such as a linear type or weighted exponential type as defined by Formula (4- 18) and (4-19), can increase the training speed (see Tables 5-6 and 5-7).
4. Batch learning mode does not improve the learning behavior of LOSANN in this study (see Table 5-8).
5. The results of testing LOSANN depend on the type of LOSANN and the distribution of the training pattern types used.

Based on this analysis, H1C-4-10 and H2L-20 LOSANN types were selected as the models to be used in the Optimal Traffic Signal Control System. The main characteristics of these two types of LOSANN are listed in Table 5-12.

Table 5-12 Learning Behavior of Different Types of LOSANN

Type	Hidden Layer	Hidden Neurons	Learn Mode	Learn Type	Learn Rate	Training		Testing		Max. Epochs
						Bad	RMSE	Bad	RMSE	
SL	0	-	Pattern	Constant	0.5	67	9.93%	26	26.53%	3000
H1L-20	1	20	Pattern	Linear	2	0	1.34%	19	15.27%	2549
H2C-4-10	2	4-10	Pattern	Constant	0.5	16	3.23%	8	7.91%	3000

CHAPTER 6: OPTIMAL TRAFFIC SIGNAL CONTROL MODEL

6.1 Problem Presentation

The purpose of developing a dynamic traffic signal optimization model is to optimize traffic signal timing on-time at a specific intersection. In order to realize this goal, we must first understand the relationship existing between the traffic environment and the MOE, and then design an effective timing optimization algorithm based on this relationship. In the last chapter, LOS Analysis Neural Network (LOSANN) models have been discussed to gain insight into this relationship. The relationship can be represented as

$$Delay \Leftarrow [LOSANN] \Leftarrow \begin{Bmatrix} Traffic\_Information \\ Geometric\_Condition \\ Signalization\_Condition \end{Bmatrix} \qquad (6\text{-}1)$$

where "Delay" is the MOE that refers to the average stopped delay time per vehicle at an intersection. [LOSANN] is a computer-simulated artificial neural network that consists of linear combination functions and activation functions (see Equations (4-3) and (4-4)). The output of LOSANN is on the left and the inputs, shown as "variables" in {}, are on the right. During the training process, the output and the inputs are given, but the weights in LOSANN are not fixed, that is, they need to be adjusted. Once the training process is successful, LOSANN has the ability to understand the relationships between the output(s) and the inputs.

In the optimizing model, the problem can be represented as:

$$\text{Minimize} \quad Delay \Leftarrow [LOSANN^*] \Leftarrow \begin{Bmatrix} Traffic\_Information^* \\ Geometric\_Condition^* \\ Signalization\_Condition \end{Bmatrix} \qquad (6\text{-}2)$$

$$C_{min} \le C \le C_{max} \qquad (6\text{-}2a)$$

$$\sum_{i=1}^{\le 4} G_{Ai} \ge G_{Ap} \qquad (6\text{-}2b)$$

$$\sum_{i=1}^{\le 4} G_{Bi} \ge G_{Bp} \qquad (6\text{-}2c)$$

$$\sum_{i=1}^{\le 4} (G_{Ai} + G_{Bi} + Y_{Ai} + Y_{Bi}) = C \qquad (6\text{-}2d)$$

$$C_{min} > 0; \quad G_{Ap} > 0; \quad G_{Bp} > 0; \quad Y_i > 0; \quad L_i > 0 \qquad (6\text{-}2e)$$

In the problem represented by Equation (6-2), the weights and the architecture of LOSANN are fixed. The geometric conditions in the input of LOSANN are given. The traffic information is transferred from traffic sensors or detectors and they reflect the traffic demand at a specific intersection which cannot be modified by our will. Thus, traffic information and geometric condition have the symbol "*". In contrast, the signalization condition needs to be determined. It is our purpose that setting an optimal traffic signal timing to improve the level of service at an intersection.

Equations (6-2a ~2e) are the constraint conditions, where

C_{min} = the minimum cycle length (sec)

C_{max} = the maximum cycle length (sec)

G_{Ap} = the minimum pedestrian green requirement for crossing the north-south street (sec)

G_{Bp} = the minimum pedestrian green requirement for crossing the east-west street (sec)

G_{Ai} = the green time of phase i for the east-west traffic movement (sec)

G_{Bi} = the green time of phase i for the north-south traffic movement (sec)

Y_i = the amber time of phase i (sec)

i = the index of signal phase. The subscript i is less than or equal to 4 since there are no more than 4 phases in either direction.

Equation (6-2a) indicates that a possible signal cycle length, C, must be bounded by the minimum cycle length, C_{min} and the maximum cycle length, C_{max}. Usually cycle lengths between 45 and 180 s are used in the field. However, computational experience showed that the size of cycle length significantly affects optimization speed (see Section 6.5). Thus, to reduce computational time, the size of cycle length can first be estimated by the standard "Webster" formula: A reference cycle length can be found by the following:

$$C_o = \frac{1.5L + 5}{1 - R} \qquad (6\text{-}3)$$

where

C_o = optimal cycle length (sec)

L = total lost time during a cycle, which consists of the startup delay minus the portion of amber utilized by drivers (sec)

R = sum of the flow ratios of critical movements

The minimum cycle length C_{min} and the maximum cycle length C_{max} can be determined by empirical results (FHWA, 1987). Empirical results showed that cycle lengths within a ± 30% from the optimal cycle-length estimated by Webster's formula are still performing nearly optimally (Berry, 1978); thus we can set

$$C_{min} = 0.7C_0 \quad \text{and} \quad C_{max} = 1.3\,C_0 \qquad (6\text{-}4)$$

In addition to letting vehicles pass through an intersection, a traffic signal must also provide enough time for pedestrians to cross the street. This requirement becomes the lower bound of green time design. Equations (6-2b) and (6-2c) present the formulas to estimate the minimum green time

$$G_{Ap} = W + \frac{D_{ns}}{V} \qquad (6\text{-}5a)$$

$$G_{Bp} = W + \frac{D_{ew}}{V} \qquad (6\text{-}5b)$$

where

G_{Ap} = the minimum pedestrian green time requirement for east-west traffic movement (sec)

G_{Bp} = the minimum pedestrian green requirement for north-south traffic movement (sec)

W = pedestrian start-up time (4~7 sec)

D_{ew} = crossing distance in east-west bound (ft)

D_{ns} = crossing distance in north-south bound (ft)

V = walking speed (4 ft/sec)

Equation (6-2d) is the definition of cycle length. It states that cycle length is a complete sequence of signal indications; it is the duration of time in which the whole set of phases at a signalized intersection takes place once.

In addition to constraints (6-2a) through (6-2e), there are other constraints that must be considered. These conditions that determine the volume domain of inputs depend on the knowledge space (the pattern set) used to train LOSANN. For example, describing the traffic flow at 4-legs intersection needs 12 factors to describe it since there are four approaches and each approach has three kinds of movement (left turn, through, and right turn). If there are 300 patterns consisting of a knowledge space, the maximum and the minimum volumes of these factors in the training set specify their volume domains, respectively. Other factors, whether they belong to the traffic or to the geometric or to the signalization conditions, also have their own domains. Although the volume domains of the factors are not listed, they must be carefully considered when applying LOSANN. If an input data is out of its domain, the output is meaningless.

6.2 Search Algorithms

In this study, state space search and heuristic search methods are used to optimize traffic signal timing. The problem defined by (6-2) can be described using "artificial intelligence language". The output of the equation (6-2) is the state of the optimal problem. The inputs of the equation are the facts of the problem. Facts determine states. Among the facts relevant to this problem, only those related to the signalization condition are undetermined. These facts include phases sequence, cycle length, green time, amber time and lost time (usually amber time and lost time are given). All of the possible states comprise the state space. The goal of the search is to find the state with the minimum value in the whole state space. The state with the minimum value is the signal timing that results in the minimum average stopped delay per vehicle at an intersection.

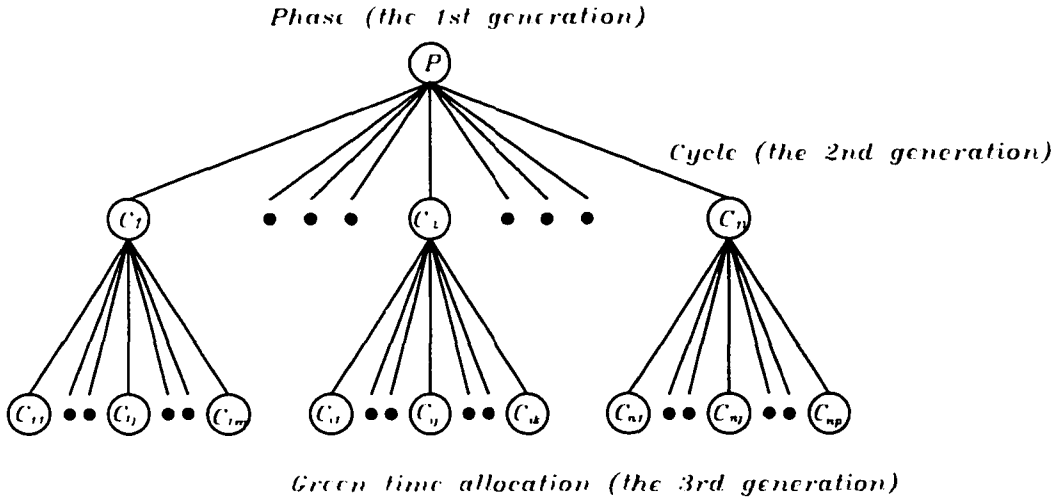


Figure 6-1 State space for searching

In order to navigate through the search space, it is necessary to define the structure of the state space (Figure 6-1). In this study, the first generation child is generated by the traffic phase sequence. The second-generation child is generated by the cycle length. The third generation is generated by the green time allocation. In this research, only signals with two-phase sequences are studied. Therefore, there is only one child, a two phase sequence, in the first generation level. For the convenience of describing the search process, the child is indexed by its subscript, which is displayed in the circles of the figure. For example, the ith child in the second generation is defined as C_i and the jth child of C_i is C_{ij}. The number of second-generation children can be calculated by

$$n = \frac{C_{max} - C_{min}}{C_s} \qquad (6\text{-}6)$$

where

n = the number of second-generation children

C_{max} = the maximum cycle length defined by Equation (6-4)

C_{min} = the minimum cycle length defined by Equation (6-4)

C_s = the search step size in second. In a practical application, C_s is usually set to 5, since cycle lengths typically end in 0 or 5.

Each child in the second generation C_i represents the cycle length. Its value can be calculated by

$$C_i = C_{min} + C_s * (i-1) \qquad 1 \le i \le n \qquad (6\text{-}7)$$

The third generation child C_{ij} has two subscripts. The first indicates its parent and the second its order in the siblings. The number of children who have the same parent C_i can be calculated by

$$m_i = Round\left[\frac{C_i - (Y_A + Y_B + G_{Ap} + G_{Bp})}{g_s}\right] \qquad 1 \le i \le n \qquad (6\text{-}8)$$

where

C_i = the cycle length represented by child C_i (see Equation 6-7)

Y_A = the amber time for east-west traffic movement (sec)

Y_B = the amber time for north-south traffic movement (sec)

G_{Ap} = the minimum green time requirement for pedestrians crossing street in the north-south bound direction

G_{Bp} = the minimum green time requirement for pedestrians crossing street in the east-west bound direction

g_s = the green time increment in seconds. The smaller its volume, the more detailed the search becomes. In this study, g_s is set to 0.2 second.

The total number of third generation children can be calculated by

$$m = \sum_{i=1}^{n} m_i$$

The third generation child C_{ij} represents the green time for east-west traffic movement G_{Aij}. It can be calculated by

$$C_{ij} = G_{Aij} = G_{Ap} + g_s*(j-1) \qquad 1 \le j \le m_i \qquad (6\text{-}9)$$

Substituting C_{ij} into Equation (6-2c), the north-south green time G_{Bij} can be calculated easily since

$$G_{Bij} = C_i - (C_{ij} + Y_A + Y_B) \qquad (6\text{-}9a)$$

The search state is the average stopped delay per vehicle which is written as D_{ij}. The relationship between the state D_{ij} and its factors can be described by

$$D_{ij} = Delay \Leftarrow [LOSANN*] \Leftarrow \begin{Bmatrix} Traffic\_Information* \\ Geometric\_Condition* \\ G_{Aij}(C_i, C_{ij}), G_{Bij}(C_i, C_{ij}), Y_A, Y_B \end{Bmatrix} \qquad (6\text{-}10)$$

Since amber time is preset, Equation (6-10) can simply be stated as by

$$D_{ij} = D_{ij}(C_i, C_{ij}) = [LOSANN]\{C_i, C_{ij}\} \qquad 1 \le i \le n;\ 1 \le j \le m_i \qquad (6\text{-}10a)$$

Equation (6-10a) indicates that the average stopped delay D_{ij} occurs if the cycle length is equal to C_i and the green time for east-west movement equal to C_{ij}, based on the experience.

6.3 State Space Search Model with Depth-First Search Algorithm

6.3.1 General Description

A state space may be searched in two directions: from the given data of a problem instance toward a goal or from a given goal back to the data. To find an optimal traffic

signal scheme, using a data-driven search is an appropriate procedure since the goal of the search is unknown. The search algorithm order is depth-first. In depth-first search, when a state is examined, all of its children and their descendants are examined before any of its siblings. Depth-first search goes deeper into the search space whenever this is possible. Only when no further descendants of a state can be found are its siblings considered.

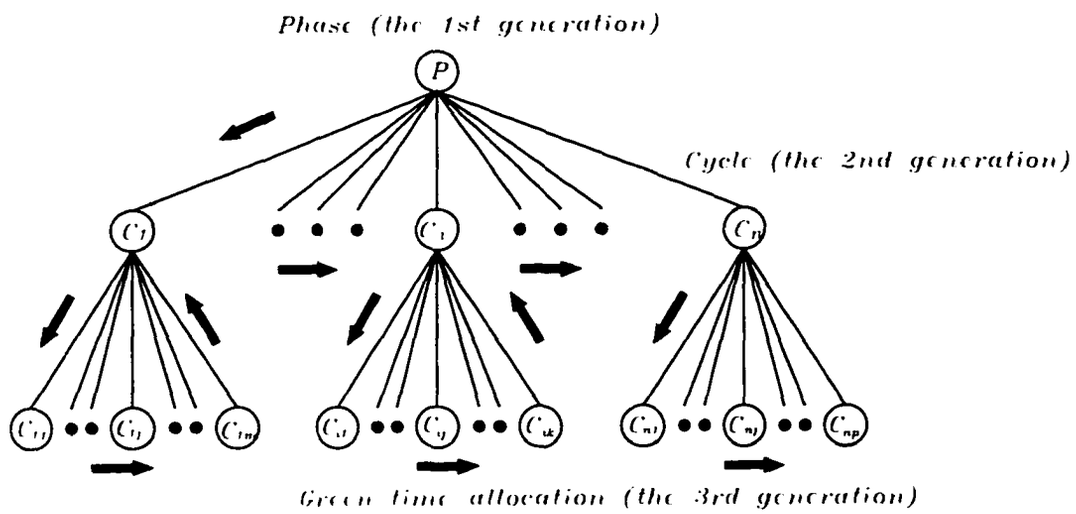


Figure 6-2 Depth-first search

The advantage of using depth-first search is that it is able to find the global optimal solution. The disadvantage is that the search time may be lengthy.

6.3.2 Design a Depth-First Search Algorithm for Optimal Traffic Signal Timing

This dissertation focuses on the feasibility of a dynamic optimization procedure; hence, a simple two-phase signalization is considered the test case. This decision is also based on the input patterns that were available for training LOSANN.

The traffic signal optimal algorithm with depth-first search model is shown in Figure 6-2. The symbols used in the figure are the same as that in Figure 6-1. As

described in the last section, there is only one child in the first generation and n children in the second generation. Each second generation child C_i has m_i children. The arrows display the search path.

The search begins at the phase sequence level and arrives at the deepest level via the path P→ C_1 → C_{11}. C_1, defined as the first child in the second generation, is equal to the minimum cycle length C_{min}. C_{11} is the first child of C_1. As defined by Equation (6-9), C_{11} represent the minimum green time for east-west traffic movement through an intersection.

The search algorithm examines the states of C_i's family, which is defined by Equation (6-10), in the order $C_{11} \rightarrow C_{1m_1}$ and keeps a record:

$$D_1^* = min\ \{D_{1j}(C_1, C_{1j}),\ 1 \le j \le m_1\} \qquad (6\text{-}11a)$$

where D_i^* is the local solution indexed by cycle length C_1. C_{1j} can be calculated by Equation (6-9). The input that produced the local solution D_1^* is also saved at the same time. At this point, we can state the general form used to calculate the local solution as

$$D_i^* = min\ \{D_{ij}(C_i, C_{ij}),\ 1 \le i \le m_1\} \qquad (6\text{-}11b)$$

where D_i^* is the local solution indexed by cycle length C_i. The third generation child C_{ij} and state $D_{ij}(C_i, C_{ij})$ were defined by Equations (6-9) and (6-10a).

After getting the first local solution, the search goes back to one of the current children's "uncles". The path is shown in Figure 6-2: $C_{1m_1} \rightarrow C_2$. Since C_2 has its children C_{2j}, the search must go into the search space as deeply as possible, that is, go

into the third generation via the path: $C_2 \rightarrow C_{21}$. After examing all the children in C_2's family, the search can find the second local solution D_2^* by using Equation (6-11b). The procedure is repeated until the last child C_{nm_n} has been examined. During the iteration, all the local solutions D_i^* are found. In the end, the algorithm select the solution that produces the minimum delay as expressed by

$$D^* = min \{D_i^*,\ 1 \leq i \leq n\} \qquad (6\text{-}12)$$

where D* is the global solution. After finding the state D*, the optimal signal scheme can be found by backtracking along the search path.

6.4 Heuristic Search Model with Best-First Search Concept

6.4.1 General Description

As described in the previous section, a depth-first search algorithm is able to find the global optimal solution but it may take too much time. We need an algorithm that can get a solution, even if it is not the best one, within a very short time, for example, 10 seconds. For this reason, the heuristic search is applied to find a good suboptimal solution.

One of the heuristic search procedures called "hill-climbing" (Pearl 1984) is used in this traffic signal optimal model. The general concept of hill climbing is that the search always selects the best child of the current parent for further expansion, until it reaches a state that is better than any of its children. Therefore, the search order of hill-climbing procedures is a best-first search. The advantage of the best-first search is that the search time can be controlled. The disadvantage is that a best-first search method is

unable to distinguish between local and global maxima. This means that a best-first search strategy may choose a local optimal solution if it reaches a state that is better than any of its children. Another problem is that an erroneous heuristic can lead along infinite paths that fail.

In this research, a search strategy is designed based on the concept of best-first search. This search, each time, always proceed along a specific direction to find the "best child", that is, a local optimal solution. According to the features of traffic signal timing with two -phases, two directions must be considered since there are two independent variables (cycle length and ratio of green time split) which determine a traffic signal timing scheme. Each run of the search can only consider either cycle length changing by fixing the ratio of green time split, or verse-visa. Then the search is switched to another direction. The process will be performed in cycles until it finds an optimal solution, which might be a suboptimal solution. This search strategy is defined as "direction search" in this dissertation. Direction search is a heuristic search.

6.4.2 Design a Direction Search Algorithm for Optimal Traffic Signal Timing

In this section, a direction search strategy designed to optimize traffic signal timing with two phases sequences will be introduced. This search is based on the traffic engineer's experience and intuition, which are helpful in finding a good suboptimal solution, and may turn out to be a global solution.

The green time split ratio must satisfy the traffic demand in different approaches as much as possible. This is a necessary condition under any cycle length. Therefore, the direction search is designed to force the search in two directions, cycle length C_i and green time split R_k. The subscript of these two directions is the index of the search epoch. The values of the subscript of these two directions are always different since each time the search can occur in only one of these two directions. The direction navigated by the split radio R_k is very important. The direction search algorithm concentrates the search in the area where lies a high probability of finding the optimal solution so that the search can save much time.

There are only two independent variables or facts, cycle length C_i and green time split ratio R_k, which are capable of controlling the traffic operation at an intersection. Amber time and the minimum green time are usually predetermined. . Green time split ratio R_k is introduced not only for navigating the search direction, but also for calculating a green allocation for the k^{th} epoch searching:

$$C_{ik} = R_k(C_i - Y_A - Y_B) \qquad (6\text{-}13)$$

where C_{ik} is one child of C_i's family and it represents the green time for east-west traffic movement. Y_A and Y_B are the amber time. If the increment of green time is set, the C_{ik} must be round to the next down (or the next up) value of green time. Figure 6-3 shows the optimal traffic signal timing algorithm with a direction search strategy. One coordinate system is built by the cycle length C_i, the ratio of green time split R_k, and the average stopped delay per vehicle D_{ik}. The D_{ik}-*axis* is the output of LOSANN. The input

facts, C_i and $C_{ik}(C_i, R_k)$, determine the output state D_{ik} through LOSANN (Equation 6-11).

There are four lines in the C_i-R_k plane: C_{min}, C_{max}, R_{min}, and R_{max}. These four lines determine a feasible state search space. C_{min} and C_{max} are the minimum and the maximum cycle length. R_{min} and R_{max} are the minimum and the maximum green time split ratio for east-west traffic movement. It is easy to understand that

$$R_{min} = G_{Ap}/(Ci - Y_A - Y_B)$$

$$R_{max} = (Ci - G_{Bp} - Y_A - Y_B)/(Ci - Y_A - Y_B) \qquad (6\text{-}14)$$

where

G_{Ap} = the minimum green requirement for pedestrians crossing in the north-south direction

G_{Bp} = the minimum green requirement for pedestrians crossing in the east-west direction

Note that the line R_{max} is not parallel to the R_k-*axis* because R_{max} is related to C_i.

Direction search begins with an initial green split rate R_1 and examines the states along the line A-B (see Figure 6-3). The R_1 is set to 0.5, indicating that green time is equally allocated to the two directions. When all states that are determined by the facts C_i and $C_{i1}(C_i, R_1)$ are examined, the first epoch search is complete. A first local optimal solution is found by Equation (6-11a):

$$D_1 = D_{r1} = min\ \{D_{i1}(C_i, C_{i1}(C_i, R_1)),\ 1 \le i \le n\}$$

where $1 \leq r \leq n$. D_{r1} indicates that the cycle length C_r and green time allocated by R_1 produces the lowest delay after the first epoch searching. It must be noted that green time split ratio R_1 is fixed in this search epoch.

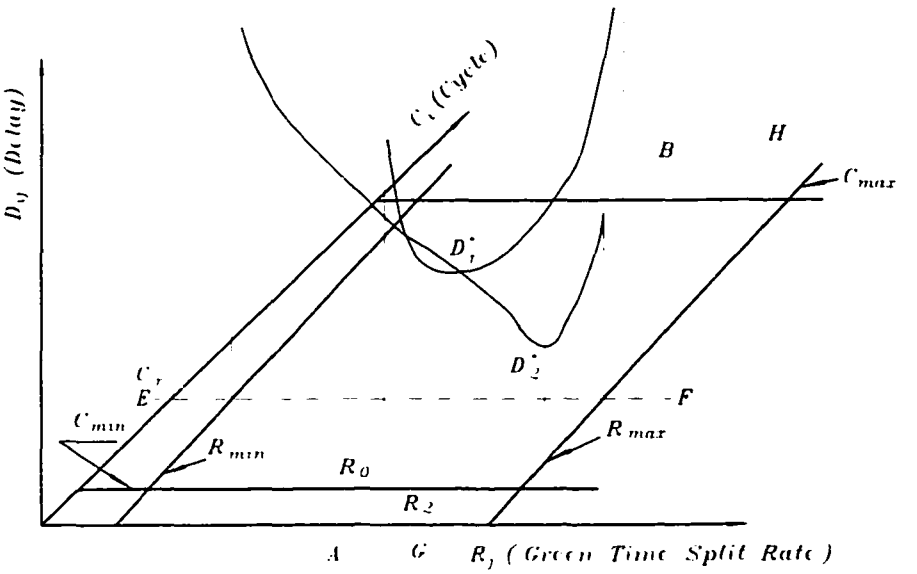


Figure 6-3 Direction search

The first epoch search not only finds a local optimal solution but also provides the best child C_r as the guide for the second epoch search. With the cycle length fixed as C_r, the second epoch search examines all the states along the line E-F (see Figure 6-3). Each time, C_{rj} is determined by

$$C_{rj} = G_{min} + g_s*(j-1) \qquad 1 \leq j \leq m_r \quad (6\text{-}15)$$

where g_s is the increment of green time search and m_r is the number of children in C_r's family that is defined by Equation (6-8). The value of m_r represents the total options of green time allocation within a specific cycle length C_r.

When every state is examined, the second local solution D_2 is found by Equation (6-11b):

$$D_2 = min\ \{D_{kj}(C_r, C_{jk})\ 1 \le j \le m_r\}$$

where j is the index of search step. Based on the relationship between R_k and C_{ij}, the new green time split ratio R_2 is

$$R_2 = \frac{C_{rq}}{C_r - Y_A - Y_B} \qquad (6\text{-}16)$$

If $D_2 < D_1$, the search continues. In that case, the C_{rq} is the best child in the second epoch search and R_2 determined by C_{rq} becomes the guide to navigate the next epoch search, that is, $R_3 = R_2$. Equations (6-11a) and (6-11b) are very similar; the difference is that the search described by Equation (6-11a) is implemented in the second-generation, while Equation (6-11b) is used to examine the children in the third-generation.

The direction search algorithm can be rewritten in a more general form:

$$D_{2k-1} = D_{rj} = min\ \{D_{i(2k-1)}(C_i, C_{i(2k-1)}(C_i, R_{2k-1}))\};\quad 1 \le i \le n \qquad (6\text{-}17a)$$

$$D_{2k} = D_{rq} = min\ \{D_{r(2k)}(C_r, C_{rj})\};\quad 1 \le j \le m_r \qquad (6\text{-}17b)$$

$$R_1 = 0.5$$

$$R_{2k} = C_{rq}/(C_r - Y_A - Y_B)$$

$$R_{2k-1} = R_{2(k-1)};\quad k \ne 1 \qquad (6\text{-}17c)$$

where k is a positive integer; i is the index for cycle length increment step; j is the index for green time increment step of east-west traffic movement; and the subscript included k is the index of the search epoch. If D_{k+1} is less than or equal to D_k, then the search continues.

The direction search algorithm continues until one of the following two events happen:

1. The new optimal solution is larger than the previous one, that is, $D_{k+1} > D_k$.
2. The search epoch k reaches the maximum value set externally to force the algorithm to stop.

6.5 Analyzing Algorithms

In this analysis and comparison of searches, the cost of search is measured by computational time.

6.5.1 Analyzing the Depth-First Search Algorithm

A pseudocode used in this study is very much like "Basic". Depth-first search implemented to optimize the traffic signal scheme is listed in the second column of Table 6-1. We use the following conventions in our pseudocode.

1. Indentation indicates block structure. For example, the body of **For i** loop begins on line 1 and ends on line 6. The indentation style applies to **If-Then-EndIf and Do-While-Loop** statements as well.
2. A string composed of upper-case letters represents a function. For example, LOSANN is LOS Analysis Neural Network.
3. Variables associated with a function are listed in the bracket followed by the function.

The labels of computation costs measured by time are listed in the third column and their corresponding runs in the fourth column. The domain of variable or the equation(s)

included in a function are listed in the fifth column. The definitions of the variables used in pseudocode can be found in the "Note" column.

Table 6-1 Depth-First Search Pseudocode

	Pseudocode	Cost	Runs	Note
1	For i = C_1 To C_n Step C_s	t_1	n	$C_{min} \le C_i \le C_{max}$
2	For j = C_{i1} To C_{im_i} Step g_s	t_2	m	$G_{Ap} \le C_{ij} \le C_{im_i}$
3	D_{ij} = LOSANN(C_i, C_{jj})	t_3	m	Eq. (6-11b)
4	If $D_{ij} < D^*$ Then	t_4	m	
	$D^* = D_{jj}$			
	$C^* = C_i$			
	$G^* = C_{ij}$			
	End If			
5	Next j	t_5	m	
6	Next i	t_5	n	

Note: the initial D^* must be set larger enough.

Time labeled "n" in the table refers to the number of runs necessary to examine the second-generation child C_i. The value of n is equal to the population of the second generation, that is

$$n = \frac{C_{max} - C_{min}}{C_s}$$

Time m is the total number of runs necessary to examine all children in the third generation. The value of m must be equal to the population of the third generation:

$$m = \sum_{i=1}^{n} m_i$$

Substituting Equation (6-8) into the above equation, we obtain

$$\begin{aligned} m &= \sum_{i=1}^{n} m_i = \sum_{i=1}^{n} \frac{C_i - Y_A - Y_B - G_{Ap} - G_{Bp}}{g_s} \\ &= -\frac{n(Y_A + Y_B + G_{Ap} + G_{Bp})}{g_s} + \sum_{i=1}^{n} \frac{C_i}{g_s} \\ &= \frac{(C_1 + C_n)n}{2g_s} - \frac{n(Y_A + Y_B + G_{Ap} + G_{Bp})}{g_s} \end{aligned} \qquad (6\text{-}18a)$$

Based on Equations (6-6) through (6-7), it is easy to prove that

$$C_n = C_{max} \; ; \qquad C_1 = C_{min}$$

Substituting these results into Equation (6-17a), and considering Equation (6-8), the total number of runs necessary to examine the third generation is

$$\begin{aligned} m &= \frac{(C_{\max} + C_{\min})n}{2g_s} - \frac{(Y_A + Y_B + G_{Ap} + G_{Bp})n}{g_s} \\ &= \frac{(C_{\max} - C_{\min} + 2C_{\min})n}{2g_s} - \frac{n(Y_A + Y_B + G_{Ap} + G_{Bp})}{g_s} \\ &= \frac{C_{\cdot} n^2}{2g_s} + \frac{1}{g_s}(C_{\min} - Y_A - Y_B - G_{Ap} - G_{Bp})n \end{aligned} \qquad (6\text{-}18)$$

We sum the products of cost and times columns in the table, obtaining the total search cost of the depth-first search model, in terms of time, as

$$T_d = (t_1 + t_5)n + (t_3 + t_4 + t_5)m \qquad (6\text{-}19)$$

It is obvious that the study cost T_d is a quadratic function of n, since Equation (6-18) is a quadratic function of n. From Equation (6-19), we get the following conclusions:

1. The traffic signal optimal algorithm with depth-first search has a order of growth of $\Theta(n^2)$ for a two-phase signal. This growth order usually is acceptable for sort or search problems, but it may not satisfy the requirement of dynamic traffic signal control since it may take much more time.
2. The optimum solution found by this model is a global optimum solution since the entire state space is searched exhaustively.
3. The search time critically depends on the number of cycle length options, which is determined by the maximum and the minimum cycle lengths whose values are fixed. Therefore, specifying the domain of cycle length based on traffic engineering knowledge and experience will increase the speed of the bsearch.

4. The increment of the green time search g_s is also a factor which determines the state space.

6.5.2 Analyzing the Direction Search Algorithm

The pseudocode used to implement the direction search algorithm is shown in Table 6-2. Similar to Table 6-1, LOSANN defined by Equation (6-11b) is the LOS Analysis Neural Network. M is the maximum searching epoch which is preset to control the computational time. The value range of the maximum searching epoch M will be discussed later in this section. The searching epoch refers to examining all of the second-generation children or all of the third-generation children who belong to the same family.

Table 6- 2 Pseudocode of the Direction Search Model

	Pseudocode	Cost	Runs	Note
1	k= 0; D_0 = 100; R_q = 0.5	t_0	1	
2	Do While k ≤ M	t_6	n_1	
3	k =k+1	t_7	n_1	
4	For i = C_1 To C_n Step C_s	t_1	n_2	$C_{min} \le C_i \le C_{max}$
5	$C_{ik} = R_k*(C_i-Y_A-Y_B)$	t_8	n_2	
6	D_{ik} = LOSANN(C_i, C_{ik})	t_3	n_2	Eq. 6-11

7	If $D_{ik} < D_k$ Then
	$D_k = D_{ik}$
	$C_r = C_i$
	End If
		t_9	n_2	

8	Next i	t_5	n_2	
9	If $D_k > D_{k-1}$ Then Exit Do	t_{10}	0 or 1	
10	k = k+1	t_7	n_1	
11	For j = C_{r1} To C_{im} Step g_s	t_2	n_3	$G_{Ap} \le C_{rj} \le C_{rm}$
12	D_{rj}= LOSANN(C_r, C_{rj})	t_3	n_3	Eq. 6-11

13	If $D_{rj} < D_k$ Then
	$D_k = D_{rq}$
	$R_k = C_{rq}/(C_r-Y_A-Y_B)$
	End If
		t_{11}	n_3	

14	Next j	t_5	n_3	
15	If $D_k > D_{k-1}$ Then Exit Do	t_{10}	0 or 1	
16	Loop	t_{12}	n_1	

In Table 6-2, n_1 is the actual number of search epochs when the direction search algorithm terminates. In other words, it refers to how many times the algorithm has changed its searching direction during the whole process. Thus, the actual number of search epochs is always less then or equal to the maximum number of search epochs allowed to implement the algorithm, that is,

$$n_1 \leq M \tag{6-19}$$

The value of n_2 in Table 6-2 is the total number of runs necessary to examine the second-generation children's states when the green time split ratio is fixed. The number of second-generation children is equal to n (see Equation (6-7)). The green time split ratio changes one time every two implemented searching epochs. Therefore, the total number of times necessary to examine the second-generation children's state is

$$n_2 = nn_1 \leq nM \tag{6-20}$$

where n is the number of cycle segment and M is the preset maximum number of search epochs. The number of the third generation children who have been visited is defined as n_3. Since the algorithm may stop before the search effort is completed, that is, M epochs, as it meets a local optimum state, the value of n_3 can be estimated by

$$\begin{aligned}
n_3 &\leq \sum_{j=n-M+1}^{n} \frac{C_j - Y_A - Y_B - G_{Ap} - G_{Bp}}{g_s} \\
&= \frac{M(C_{\max} + C_{(n-M+1)} - 2(Y_A + Y_B + G_{Ap} + G_{Bp}))}{2g_s} \\
&= \frac{M(nC_s + C_{(n-M+1)} + C_{\min} - 2(Y_A + Y_B + G_{Ap} + G_{Bp}))}{2g_s} \\
&= \frac{1}{2g_s} MC_s n + \frac{1}{2g_s} M(C_{n-M+1} + C_{\min} - 2(Y_A + Y_B + G_{Ap} + G_{Bp})) \\
&= \frac{1}{2g_s} MC_s n + BM
\end{aligned} \tag{6-21}$$

where $$B = \frac{1}{2g_s}(C_{n-M+1} + C_{\min} - 2(Y_A + Y_B + G_{Ap} + G_{Bp})) \quad (6\text{-}22)$$

Based on the data listed in Table 6-2, the computational cost of the direction search algorithm can be estimated by

$$T_b = t_0 + t_{10} + (t_6 + 2t_7 + t_{12})n_1 + (t_1 + t_3 + t_5 + t_8 + t_9)n_2 + (t_2 + t_3 + t_5 + t_{11})n_3 \quad (6\text{-}23)$$

where T_b is the total computational cost. Substituting Inequality (6-19) through (6-22) into Equation (6-23), we get the order of growth of direction search model:

$$\begin{aligned} T_b &\le t_0 + t_{10} + (t_6 + 2t_7 + t_{12})M + (t_1 + t_3 + t_5 + t_8 + t_9)nM \\ &+ \frac{1}{2g_s}(t_2 + t_3 + t_5 + t_{11})Mn + (t_2 + t_3 + t_5 + t_{11})BM \quad (6\text{-}24) \\ &= T(n) \end{aligned}$$

Equation (6-24) gives a formula to estimate the computational time used to complete M search epochs. In the best case, the solution may be found at the first run of search. Thus, the direction search has a low bound runtime $\Omega(1)$.

For a specific algorithm, the analysis of its worst case is more important than that of the best case. As mentioned previously, the maximum search epoch M is used to control the search running time and a new green time split ratio R_k is generated once the green time for east-west traffic movement is reset (see Equation (6-16)). Therefore, M must not be larger than the total number of the third-generation children in the biggest family. The number of the third-generation children in the biggest family can be estimated by

$$M_{\max} = \frac{C_{\max} - Y_A - Y_B - G_{Ap} - G_{Bp}}{g_s} = \frac{C_{\max} - C_{\min} + C_{\min} - Y_A - Y_B - G_{Ap} - G_{Bp}}{g_s}$$
$$= \frac{C_{\max} - C_{\min}}{g_s} + \frac{C_{\min} - Y_A - Y_B - G_{Ap} - G_{Bp}}{g_s} \qquad (6\text{-}25)$$
$$< 2\frac{C_{\max} - C_{\min}}{g_s} = 2n\frac{C_s}{g_s}$$

where M_{max} = the number of the third-generation children in the largest family

C_{max} = the maximum cycle length (sec)

C_{min} = the minimum cycle length (sec)

Y_A = the amber time for east-west traffic movement (sec)

Y_B = the amber time for north-south traffic movement (sec)

G_{Ap} = the minimum green requirement for pedestrians crossing in the north-south direction

G_{Bp} = the minimum green requirement for pedestrians crossing in the east-west direction

g_s = the size of the step to examine green time allocation (sec)

C_s = the size of the step to examine cycle length (sec)

During the derivation of Formula (6-25), the relationship defined by Equation (6-7) was considered. Substituting M_{max} for M in Equation (6-24), the worst case order of growth of the direction search is $O(n^2)$.

Based on the computational time estimated by formulas (6-24) and (6-25), we can list the characteristics of this model as follows:

1. The traffic signal optimal algorithm with a direction search has runtime $T(n)$ and a low bound runtime $\Omega(1)$ for a two-phase signal. This search is faster than the depth-first search algorithm ($\Theta(n^2)$) introduced in section 6.5.1.
2. As mentioned before, the computational time T_b can be controlled by M.
3. This kind direction search does not have an infinite loop problem since it has complexity $O(n^2)$.

6.6 Numeral Examples of Using OTSCS

As discussed in Sections 6.3 and 6.4, the artificial intelligence-based traffic signal control optimal algorithm which includes the LOS Analysis Neural Network (LOSANN) is named Optimal Traffic Signal Control System (OTSCS) in this dissertation. The OTSCS model was demonstrated by several numerical examples.

In order to confirm the conclusions made in the last section, depth-first search and direction search were used for comparison. The search step for cycle length, C_s, was set at 5 seconds and the search step for green allocation g_s was set at 0.2 second.

As a major premise, LOSANN must have enough knowledge to correctly analyze LOS at a given intersection. We assumed that the H2C-4-10 model of LOSANN was qualified for this analysis, based on their testing results discussed in section 5.4.2. This LOSANN model has two hidden layers. It has 124 input neurons, 4 and 10 neurons in the first and the second hidden layer respectively, and 1 output neuron. All the neurons are feed-forward connected. Its training RMSE was 3.27% and the testing RMSE was 7.91%.

Optimal results from OTSCS are listed in Table 6-3. The first column in the table lists the type of search algorithm. The last column refers to the figure that shows the specific environments of an analyzed signalized intersection. The data for these examples are obtained from the Bronx Center Transportation Study (New York City DOT, 1995). The time cost column shows the computational time needed to complete the search. It must be mentioned that the time cost depends on the computer system. The computational time listed in Table 6-3 reflects the CPU time required by a Compaq 166MHz Pentium with 16 MB of RAM.

Table 6-3 The Optimal Signal Timing Estimated by OTSCS

Search algorithm	Search Size	Time cost mm:ss	Runing times	Opt. Delay (second)	Opt. Cycle (second)	Green Split (EW)	Green Split (NS)	Traffic Environment information
DF[a]	45\120	1:15	-	15.36	105	85	10	Example 1
DS[b]	45\120	0:07	5	15.36	105	85	10	see
WDF[c]	31\59	0:08	-	30.51	56	35.8	10.2	Table 6-4
WSF[d]	31\59	0:02	3	30.51	56	35.8	10.2	
DF[a]	45\120	1:12	-	42.51	110	71.5	28.5	Example 2
DS[b]	45\120	0:04	3	42.6	120	71.15	38.85	see
WDF[c]	31\59	0:11	-	49.19	56	36.35	9.65	Table 6-5
WSF[d]	31\59	0:02	5	49.19	56	36.5	9.5	
DF[a]	45\120	1:02	-	8.43	120	14.5	94.5	Example 3
DS[b]	45\120	0:01	3	8.63	45	14.5	19.5	see
WDF[c]	40\59	0:04	-	8.60	55	14.5	29.5	Table 6-6
WSF[d]	40\59	0:01	3	8.64	40	14.5	14.5	

[a] depth-first search algorithm
[b] direction search algorithm
[c] depth-first search algorithm using Webster's formula (Equation (5-3))
[d] direction search algorithm using Webster's formula (Equation (5-3))

The following findings are based on the results shown in Table 6-3:

1. The computational time of using the heuristic direction search is significantly less than that of using the depth-first search algorithm. Generally speaking, the search

speed of the direction is about 10 to 100 times faster than that of depth-first algorithm for these numerical examples.

2. The solution obtained by using the direction search is not necessary equal to the global optimal solution. The difference in the solutions obtained by the direction search and depth-first search is minor.
3. Search speed can be raised if Webster's formula is used to narrow the search space. But in the case of using the direction search, this improvement is minimal in practical terms.
4. Sometimes there is a large difference between the solutions obtained by the search space bounded by Webster's formula and the global optimal solution (see Example 1). This last finding indicates that the subspace bounded by Webster's formula may not always contain a global optimum solution.

Table 6-4: Intersection at 163rd St. and Elton St., Bronx, New York

```
================================================================================
Streets: (E-W) E. 163rd ST.              (N-S) WASH. / ELTON
Area Type: Other                         12-7-95 PM PEAK
Comment: EXISTING CONDITION 1995
================================================================================
            |   Eastbound   |  Westbound    |  Northbound   |  Southbound
            | L    T    R  | L    T    R   | L    T    R   | L    T    R
            |---- ---- ----|---- ---- ----|---- ---- ----|---- ---- ----
No. Lanes   |      2       |    > 2       |           2   |    > 2  <
Volumes     |      241     | 434  143     |           399 |   1   89    1
PHF or PK15 |     0.93     |0.93 0.93     |          0.90 |0.90 0.90 0.90
Lane Width  |     12.0     |     11.0     |          12.0 |     12.0
Grade       |        0     |        0     |        0      |        0
% Heavy Veh |        2     |   6    6     |               4|   3    3    3
Parking     |(Y/N) N       |(Y/N) N       |(Y/N) Y      6 |(Y/N) Y      6
Bus Stops   |            0 |              0|             0 |             0
Con. Peds   |          106 |             68|             51|            68
Ped Button  |(Y/N) N       |(Y/N) N       |(Y/N) N        |(Y/N) N
Arr Type    |        4     |   3    3     |               4|   3    3    3
RTOR Vols   |            0 |              0|             0 |             0
Lost Time   |     2.00     |2.00 2.00     |          2.00 |2.00 2.00 2.00
Prop. Share|  -1        -1|  -1        -1|  -1         -1|  -1        -1
Prop. Prot.|            -2|              -2|             -2|            -2
--------------------------------------------------------------------------------
                          Signal Operations
Phase Combination 1     2     3     4 |          5     6     7     8
EB  Left          *                   |NB  Left    *
    Thru          *                   |    Thru    *
    Right         *                   |    Right   *
    Peds          *                   |    Peds    *
WB  Left          *                   |SB  Left    *
    Thru          *                   |    Thru    *
    Right         *                   |    Right   *
    Peds          *                   |    Peds    *
Green          55.0P                  |Green    55.0P
Yellow/AR       5.0                   |Yellow/AR 5.0
Cycle Length: 120 secs  Phase combination order: #1 #5
--------------------------------------------------------------------------------
```

Table 6-5: Intersection at 149th St. and Park Avenue, Bronx, New York

```
Streets: (E-W) E. 149TH ST.              (N-S) PARK AVE.
Analyst: J. FAN                          File Name: 149PAPEX.HC9
Area Type: CBD                           10-27-95 PM
```

	Eastbound			Westbound			Northbound			Southbound		
	L	T	R	L	T	R	L	T	R	L	T	R
No. Lanes		> 2 <			> 2 <			> 2 <				
Volumes	187	484	87	17	792	60	197	124	22			
PHF or PK15	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95			
Lane Width		11.0			11.0			10.0				
Grade		0			0			0				
% Heavy Veh	2	2	2	2	2	2	2	2	2			
Parking	(Y/N)	N		(Y/N)	N		(Y/N)	N				
Bus Stops			0			0			0			
Con. Peds			0			0			0			
Ped Button	(Y/N)	Y	7.8 s	(Y/N)	Y	9.0 s	(Y/N)	Y	13.6 s			
Arr Type	3	3	3	3	3	3	3	3	3			
RTOR Vols			0			0			0			
Lost Time	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00			
Prop. Share	-1		-1	-1		-1	-1		-1			
Prop. Prot.			-2			-2			-2			

Signal Operations

Phase Combination	1	2	3	4		5	6	7	8
EB Left	*				NB Left	*			
Thru	*				Thru	*			
Right	*				Right	*			
Peds	*				Peds	*			
WB Left	*				SB Left				
Thru	*				Thru				
Right	*				Right				
Peds	*				Peds	*			
NB Right					EB Right				
SB Right					WB Right				
Green	55.0P				Green	55.0P			
Yellow/AR	5.0				Yellow/AR	5.0			

Cycle Length: 120 secs Phase combination order: #1 #5

Table 6-6: Intersection at 153rd St. and Grand Concourse, Bronx, New York

```
================================================================================
Streets: (E-W) 153rd Street              (N-S) Grand Concourse
Analyst: J. Fan                          File Name: 153GCPEX.HC9
Area Type: Other                         8-4-95 PM Peak
Comment: 1995 Existing Volumes-Northbound Lefts are U-turns
================================================================================
            |   Eastbound   |  Westbound    |  Northbound   |  Southbound
            | L    T    R  | L    T    R   | L    T    R   | L    T    R
            |---- ---- ----|---- ---- ----|---- ---- ----|---- ---- ----
No. Lanes   |              | 1         1  |    > 3    1   | 1    3
Volumes     |              |   8         3| 14 1576  114|  59 1005
PHF or PK15|              |0.61      0.61|0.95 0.95 0.95|0.93 0.93
Lane Width  |              |15.0      15.0|     10.0 15.0|10.0 10.0
Grade       |              |        0     |        0     |        0
% Heavy Veh|              |   0         0|   0    2    1|   2    3
Parking     |              |(Y/N) Y    20 |(Y/N) Y    20 |(Y/N) N
Bus Stops   |              |              0|              0|             0
Con. Peds   |             0|              0|              0|             0
Ped Button  |              |(Y/N) N       |(Y/N) N       |(Y/N) N
Arr Type    |              |   3         3|   4    4    4|   4    4
RTOR Vols   |              |              0|              0|             0
Lost Time   |              |3.00      3.00|3.00 3.00 3.00|3.00 3.00
Prop. Share|              |  -1        -1|  -1        -1|  -1       -1
Prop. Prot.|              |            -2|            -2|           -2
--------------------------------------------------------------------------------
                          Signal Operations
Phase Combination 1     2     3     4 |          5     6     7     8
EB  Left                              |NB  Left    *
    Thru                              |    Thru    *
    Right                             |    Right   *
    Peds                              |    Peds
WB  Left          *                   |SB  Left    *
    Thru                              |    Thru    *
    Right         *                   |    Right
    Peds                              |    Peds
NB  Right                             |EB  Right
SB  Right                             |WB  Right
Green          36.0A                  |Green    73.0P
Yellow/AR       6.0                   |Yellow/AR 5.0
Cycle Length: 120 secs  Phase combination order: #1 #5
--------------------------------------------------------
```

CHAPTER 7: SUMMARY AND CONCLUSIONS

7.1 Research Summary

The Optimal Traffic Signal Control System (OTSCS) was developed in this study as an artificial intelligence based signal timing system. The system was intended to dynamically control a traffic signal by finding an optimal signal timing to minimize delay at signalized intersections. Software based on this system was developed using Microsoft Visual Basic 5.0 not only as a signal timing tool, but also as an education / training tool to help the user understand the learning behavior of artificial neural networks and properties of heuristic search methods.

OTSCS consists of a computer simulation-neural network and an artificial intelligence-based search model. This computer simulation-neural network is called Level of Service Analysis Neural Network (LOSANN) and its major function is to conduct a comprehensive analysis of the level of service at a signalized intersection. LOSANN is a knowledge-based model. It has the capability of understanding complex relationships between traffic delay and traffic environment for 3- and 4-leg signalized intersections. This study used data from New York City, but the system can evaluate LOS of signalized intersections in any area. Using traditional methodology such as HCM 94, it is difficult to correctly estimate LOS in a short time. LOSANN can immediately evaluate LOS given necessary on-line data. This system is appropriate for use in an Advanced Traffic Management System (AMTS) since it considers almost all factors which may affect the traffic performance at a signalized intersection (Fan and Saito, 1998).

In the signal timing optimization procedure, the input end of LOSANN is linked with the traffic sensors, and the output end of LOSANN is connected to a specially designed heuristic search machine named as Optimal Traffic Signal Timing Model (OTSTM). OTSTM can perform a depth-first search or a direction search to find an optimal signal timing based on the analysis of LOSANN and the signal cycle length range determined by the user. Search strategy selection depends on how fast the optimization must be completed. OTSTM can provides an optimal traffic signal timing, which includes signal cycle length and green time allocation, within a few seconds using a typical Pentium-based PC.

As an application, OTSCS includes a group of supporting modules. These include a data conversion program, a pattern set preparation program and an analysis report program. The data conversion program transfers data from HCS's output files to the format designed for LOSANN, if the user wants to make a simulated training. The pattern set preparation program provides the user an interface to view, select and organize data to train and test LOSANN. The analysis report program automatically records the process and the results of training as well as testing LOSANN.

The major efforts for OTSCS software development included:

- Model logic design
- Graphic user interface (GUI) design
- Traffic operation data collection

- Creating Level of Service Analysis Neural Network (LOSANN) by using "Basic" programming
- Creating changeable learning rate types for error back-propagation algorithm.
- Training and testing LOSANN
- Designing and programming an optimal machine with depth-first search and best-first search strategies
- Analyzing computational cost of the optimal search algorithms
- Integrating LOSANN and the artificial intelligent search algorithm
- Numerically demonstrating OTSCS model

7.2 Conclusions of the Study

The following conclusions are drawn from this study:

1. Optimal Traffic Signal Control System (OTSCS) is capable of quickly providing an optimal traffic signal timing for a signalized intersection given the traffic situation. It takes only a few seconds to complete all the computations when a Pentium-based PC is used.
2. Level of Service Analysis Neural Network (LOSANN) is capable of understanding the complex relationships between the average stopped delay for the entire intersection and the traffic, geometric and signalization environment through self-learning. Since LOSANN is a knowledge-based model rather than a mathematical or statistical model, it can adapt itself to a variety of situations of signalized intersections.
3. Multilayer LOSANNs perform significantly better than single-layer LOSANNs.

4. Using a variable learning rate can increase the convergence speed for training LOSANN. After 3000 epochs, the RMSE of LOSANN becomes lower than 1.5% when a changing learning rate is used with the tolerance set at 15%.
5. The batch learning mode was less effective in improving the learning behavior of LOSANN, as compared to the pattern learning mode according to the data set used in this study, but it takes much more time for training LOSANN model.
6. Pattern set preparation is a critical step to develop LOSANN. The pattern set not only needs to have enough patterns, but also to include a variety of situations.
7. The direction search strategy is very effective. The search speed of the direction search is at least 10 times faster than that of the depth-first search strategy based on the results of numerical examples. The results from these two strategies are very similar. OTSCS should, therefore, employ a search machine with a direction search strategy.

7.3 Recommendations for Further Research

The following topics are recommended for further research:

1. Optimal Traffic Signal Control System (OTSCS) software can be immediately used in level of service evaluation and signal timing optimization (two-phases only). In order to make the model more accurate in testing, more patterns are needed to train and test LOSANN since the pattern set used for this study was limited to those obtained from Bronx Center Study and Flushing Area Study in New York City.

2. Application of OTSCS to traffic network signal system is a challenge. In order to use OTSCS to improve traffic network performance, key factors affect network traffic performance should be identified.
3. The learning rate is a critical issue of applying error back-propagation algorithm. Although variable learning rates created in this study greatly enhanced the learning behavior of artificial neural network, they are not the optimal learning rate. Finding the optimal learning rate through mathematical method is very important for application of artificial neural networks.
4. The heuristic search methods specifically designed for OTSCS functioned successfully. It is possible to limit the optimization time to 10 seconds. This type of traffic signal optimization can be extended to other popular traffic operation analysis software such as HCS, TRANSIT-7F. This linkage should be studied to make this software more universal.
5. The heuristic direction search needs to be extended to 4-dimension space which includes optimizing phase sequence.

APPENDIX A: DERIVATION OF GENERAL FORM FOR WEIGHT ADJUSTMENT

It is necessary to derive a general formula to calculate weight adjustments, since multilayer feed-forward networks usually have more than one hidden layer. Figure A-1 shows a multilayer network. The final output layer is indexed as a symbol *"0"*. For convenience, we define the hidden layer next to the output layer as layer 1. The numerical designations of the layers increase as we move backward along the network untill the hidden layer before the input layer is reached.

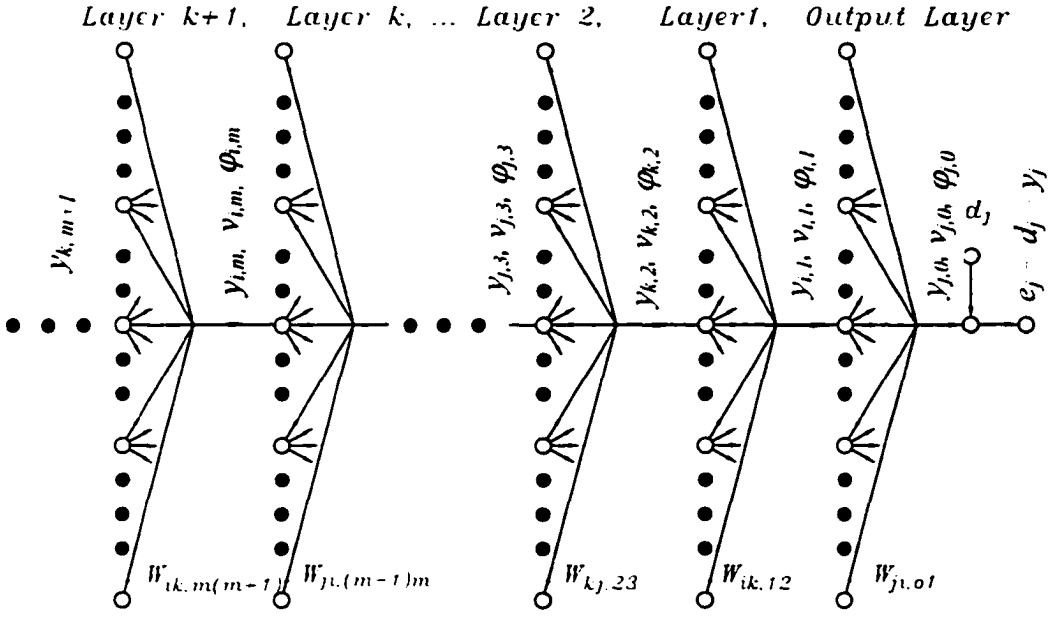


Figure A-1 Multilayer perceptron

Notations defined in this appendix have a composite subscript, except the symbol *e*, an error term discussed shortly. The subscript is separated into two parts by a comma. The portion which precedes the comma indicates the neuron position in a layer; the portion which follows the comma indicates the ordinal number of a layer counting back

from the output layer. The subscript for weights has elements in each part before and after the comma. The elements in these subscript parts show the relationship between the two connected neurons in different layers. For example, the subscript $ji,(k-1)k$ means that the neuron j in the layer $k-1$ is connected with the neuron i in the layer k.

An error signal is defined as

$$e_j(n) = d_{j,0}(n) - y_{j,0}(n) \qquad j \in C \qquad (4\text{-}5a)$$

where $e_j(n)$ = an error signal,

$d_{j,0}(n)$ = a desired signal,

$y_{j,0}(n)$ = an activated signal,

C = the amount of neurons in the output layer,

n = the epoch number in a pattern set.

In order to train the network by the least mean-squares (LMS) algorithm, the *instantaneous sum of squared errors* of the network is defined as

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \qquad (4\text{-}6)$$

It should be noted that the *instantaneous sum of squared errors* is a function of the weights in the network. Once a pattern n is entered into the network, the error signal $e_j(n)$ is generated (see Equation (4-5a)).

The activated signal $y_{j,0}(n)$ is determined by the activation function $\varphi(*)$ as follows

$$y_{j,0}(n) = \varphi(v_{j,0}(n)), \qquad (A\text{-}1)$$

where

$$v_{j,0}(n) = \sum_{i \in C_0} w_{ji,01}(n) y_{i,1}(n) \qquad \text{(A-2)}$$

In general, Equations (A-1) and (A-2) can be written as

$$y_{j,k}(n) = \varphi(v_{j,k}(n)) \qquad n \in N \qquad \text{(4 - 2)}$$

and

$$v_{j,k}(n) = \sum_{i \in C_{k+1}} w_{ji,k(k+1)}(n) y_{i,(k+1)}(n) \qquad \text{(4 - 4)}$$

where C_{k+1} is the neuron set in layer *(k+1)*. Introducing the steepest descent concept, the weights in the network can be modified by

$$w_{ji,k(k+1)}(n+1) = w_{ji,k(k+1)}(n) + \Delta w_{ji,k(k+1)}(E(n)) \qquad \text{(4 - 8)}$$

where the weight adjustment $\Delta w_{ji,k(k+1)}(n)$ is determined by the *delta rule*:

$$\Delta w_{ji,k(k+1)}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji,k(k+1)}(n)} \qquad \text{(4 -10)}$$

where η is the learning rate. The use of the minus sign in Equation (4-10) accounts for gradient descent in the weight space. Since the epoch number n is a constant during adjusting weights, it is omitted henceforth.

Using the "chain rule" of calculus, the gradient in the output layer can be written as

$$\frac{\partial E}{\partial w_{ji,01}} = \frac{\partial E}{\partial e_{j,0}} \frac{\partial e_{j,0}}{\partial y_{j,0}} \frac{\partial y_{j,0}}{\partial v_{j,0}} \frac{\partial v_{j,0}}{\partial w_{ji,01}} = -e_j \varphi_{j,0}'(v_{j,0}) y_{i,1} \qquad \text{(A-3)}$$

If we define the local gradient in this layer as

$$\delta_{j,0} = -\frac{\partial E}{\partial y_{j,0}}\frac{\partial y_{j,0}}{\partial v_{j,0}} = -\frac{\partial E}{\partial e_{j,0}}\frac{\partial e_{j,0}}{\partial y_{j,0}}\frac{\partial y_{j,0}}{\partial v_{j,0}} = e_j \varphi'_{j,0}(v_{j,0}) \qquad \text{(A-4a)}$$

the weight adjustment in the output layer can be determined by

$$\Delta w_{ji,01} = \eta \delta_{j,0} y_{i,1} \qquad \text{(A-4)}$$

Now we use the *Inductive Method* to prove that Equation (4-13) is the general form of the weight adjustments. Equation (4-13) is recalled below:

$$\Delta w_{ji,k(k+1)}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji,k(k+1)}(n)} = \eta \delta_{j,k}(n) y_{i,(k+1)}(n) \qquad \text{(4-13)}$$

where the local gradient is

$$\delta_{j,k} = -\frac{\partial E}{\partial y_{j,k}}\frac{\partial y_{j,k}}{\partial v_{j,k}} = \frac{\partial \varphi(v_{j,k}(n)}{\partial v_{j,k}(n)} \sum_{m \in C_{k-1}} \delta_{m,(k-1)} w_{mj,(k-1)k} \qquad \text{(4 -14)}$$

It is noted that the derivation of the weight adjustments in a hidden layer is complicated since there is no specified desired signal for the neuron j. The error signal for a hidden neuron would have to be determined recursively in terms of the error signals of all the neurons to which that hidden neuron is directly connected (see Figure A-1).

Step 1: For hidden layer l, the gradient is

$$\frac{\partial E}{\partial w_{ji,12}} = \frac{\partial E}{\partial y_{j,1}}\frac{\partial y_{j,1}}{\partial v_{j,1}}\frac{\partial v_{j,1}}{\partial w_{ji,12}} = \varphi_{j,1}'(v_{j,1}) y_{i,2} \sum_{m \in C_o} \frac{\partial E}{\partial y_{m,0}}\frac{\partial y_{m,0}}{\partial v_{m,0}}\frac{\partial v_{m,0}}{\partial y_{j,1}} \qquad \text{(A-5)}$$

Substituting Equation (A-4a) into Equation (A-5), the gradient in the hidden layer 1 can be rewritten as

$$\frac{\partial E}{\partial w_{ji,12}} = \varphi_{j,1}'(v_{j,1}) y_{i,2} \sum_{m \in C_0} (-\delta_{m,0} w_{mj,01}) \qquad \text{(A-5')}$$

If the local gradient on the hidden layer *1* is defined as

$$\delta_{j,1} = -\frac{\partial E}{\partial y_{j,1}} \frac{\partial y_{j,1}}{\partial v_{j,1}} = \varphi_{j,1}'(v_{j,1}) \sum_{m \in C} \delta_{m,0} w_{mj,01} \qquad \text{(A-6)}$$

then, the weight adjustments in the layer 1 is

$$\Delta w_{ji,12} = -\eta \frac{\partial E}{\partial w_{ji,12}} = \eta \delta_{j,1} y_{i,2} \qquad \text{(A-7)}$$

Step 2: In the hidden layer *2*, the gradient can be calculated by

$$\frac{\partial E}{\partial w_{ji,23}} = \frac{\partial E}{\partial y_{j,2}} \frac{\partial y_{j,2}}{\partial v_{j,2}} \frac{\partial v_{j,2}}{\partial w_{ji,23}} = \varphi_{j,2}'(v_{j,2}) y_{i,3} \sum_{m \in C_1} \frac{\partial E}{\partial y_{m,1}} \frac{\partial y_{m,1}}{\partial v_{m,1}} \frac{\partial v_{m,1}}{\partial y_{j,2}} \qquad \text{(A-9)}$$

where C_1 is the neuron set in layer 1. Substituting Equation (A-6) with Equation (A-9), the weight adjustments in the layer 2 are rewritten by

$$\frac{\partial E}{\partial w_{ji,23}} = \varphi_{j,2}'(v_{j,2}) y_{i,3} \sum_{m \in C_1} (-\delta_{m,1} w_{mj,12}) \qquad \text{(A-9')}$$

As in Step 1, the local gradient in this layer is defined as

$$\delta_{j,2} = -\frac{\partial E}{\partial y_{j,2}} \frac{\partial y_{j,2}}{\partial v_{j,2}} = \varphi_{j,2}'(v_{j,2}) \sum_{m \in C_1} \delta_{m,1} w_{mj,12} \qquad \text{(A-10)}$$

The weight adjustments in the layer 2 are

$$\Delta w_{ji,23} = -\eta \frac{\partial E}{\partial w_{ji,23}} = \eta \delta_{j,2} y_{i,3} \qquad \text{(A-11)}$$

Step 3: Assume that the weight adjustment in the layer *k-1* is

$$\Delta w_{ji,(k-1)k} = -\eta \frac{\partial E}{\partial w_{ji,(k-1)k}} = \eta \delta_{j,(k-1)} y_{i,k} \qquad \text{(A-12)}$$

where the local gradient is defined as

$$\delta_{j,(k-1)} = -\frac{\partial E}{\partial y_{j,(k-1)}}\frac{\partial y_{j,(k-1)}}{\partial v_{j,(k-1)}} = \varphi_{j,(k-1)}{}'(v_{j,(k-1)}) \sum_{m \in C_{k-2}} \delta_{m,(k-2)} w_{mj,(k-2)(k-1)} \qquad \text{(A-13)}$$

The gradient in the hidden layer k can be derived as

$$\frac{\partial E}{\partial w_{ji,k(k+1)}} = \frac{\partial E}{\partial y_{j,k}}\frac{\partial y_{j,k}}{\partial v_{j,k}}\frac{\partial v_{j,k}}{\partial w_{ji,k(k+1)}} = \varphi'_{j,k}(v_{j,k}) y_{i,(k+1)} \sum_{m \in C_{k-1}} \frac{\partial E}{\partial y_{m,(k-1)}}\frac{\partial y_{m,(k-1)}}{\partial v_{m,(k-1)}}\frac{\partial v_{m,(k-1)}}{\partial y_{j,k}}$$

(A-14)

Substituting Equation (A-13) into the above equation, Equation (A-14) is rewritten as

$$\frac{\partial E}{\partial w_{ji,k(k+1)}} = \frac{\partial E}{\partial y_{j,k}}\frac{\partial y_{j,k}}{\partial v_{j,k}}\frac{\partial v_{j,k}}{\partial w_{ji,k(k+1)}} = \varphi'_{j,k}(v_{j,k}) y_{i,(k+1)} \sum_{m \in C_{k-1}} (-\delta_{m,(k-1)} w_{mj,(k-1)k})$$

(A-14')

If the local gradient in the hidden layer n is defined as

$$\delta_{j,k} = -\frac{\partial E}{\partial y_{j,k}}\frac{\partial y_{j,k}}{\partial v_{j,k}} = \varphi_{j,k}{}'(v_{j,k}) \sum_{m \in C_{k-1}} \delta_{m,(k-1)} w_{mj,(k-1)k} \qquad \text{(A-15)}$$

then the weight adjustment for an arbitrary hidden layer is derived as

$$\Delta w_{ji,k(k+1)} = -\eta \frac{\partial E}{\partial w_{ji,k(k+1)}} = \eta \delta_{j,k} y_{i,(k+1)} \qquad \text{(A-16)}$$

This is what we try to demonstrate if we reverse the ordinal indexes of layers in Equation (A-16)

APPENDIX B: OTSCS SOFTWARE

The Optimum Traffic Signal Control System (OTSCS) is an application developed to determine optimal signal timings for isolated signalized intersections using artificial neural network and optimization techniques. The Level of Service Analysis Neural Network (LOSANN) in OTSCS, once trained, can perform level of service analysis faster than traditional methods. The software can not only be adapted to practical LOS analysis and signal timing, but is also helpful in understanding the behaviors of artificial neural networks and learning the techniques for training and testing them. When the input module of OTSCS is adjusted to interface with dynamic traffic volume counters and its output module is connected to a traffic signal, OTSCS becomes a computerized dynamic optimum traffic signal control system.

B1 Components and User Interfaces of OTSCS

OTSCS consists of four main modules:

1. Data Conversion Module: Convert data from HCS detailed report files to the standard format designed for the Level of Service Analysis Neural Network (LOSANN). The transferred data comprise the knowledge space for training and testing LOSANN.
2. LOS Analysis Neural Network: Analyze Level of Service in an isolated signalized intersection based on the knowledge space.
3. Report Generation Module: Generate LOSANN's training report or testing report for user review.
4. Optimal Traffic Signal Timing Module: Provide an optimal traffic signal timing to the user using the dynamic traffic situations.

All of these main modules are designed according to the Graphic User Interface (GUI) concept and are controlled by System Menu. Therefore, OTSCS is convenient for the user.

The major interfaces in OTSCS include:

1. System Menu
2. Data Conversion (for data conversion module)
3. LOS Analysis Neural Network (for LOSANN)
4. Training and Testing Analysis (for report generation module)
5. Traffic Signal Optimum Algorithm (for OTSTM)

The following five sections will discuss these five major interfaces.

B2 System Menu

The system menu of OTSCS (Figure B-1) is the main menu used to control the whole system. There are four options in this menu. When an option is selected by clicking one of the options, the interface related to the selected module will automatically appear. To exit the program, the user simply clicks the control icon on the top right corner or double-click any part of the form.

Option selections can be made by clicking *Options* in the menu bar of the main menu's menu bar.

From *Utility*, the user can select one of two options: Previewing the data or creating a pattern set (see the sections of Data Preview and Create Pattern Set).

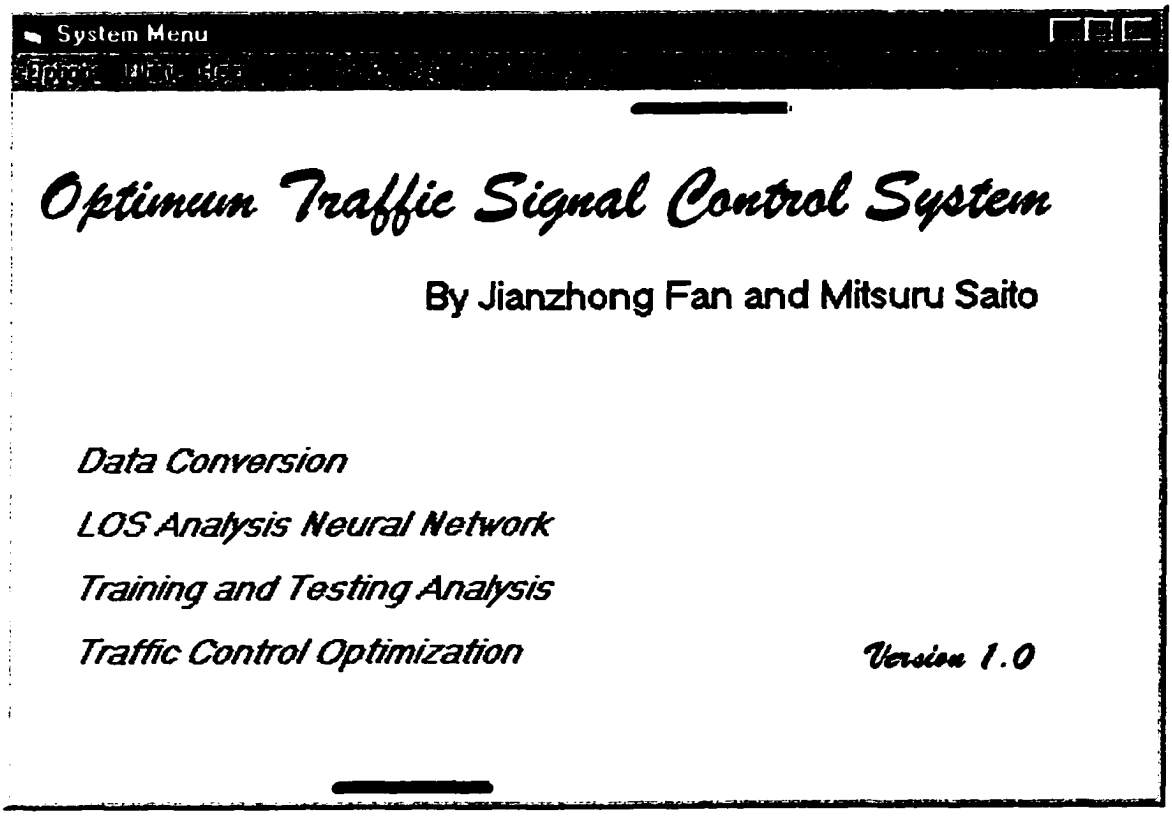


Figure B-1 System Menu of OTSCS

B3 Data Conversion

Data conversion is the first step to using OTSCS. If it is the first time that the user uses OTSCS, or an adaptable LOSANN is not available, the user must prepare patterns to train and test LOSANN. Data conversion has three phases:

a) Convert the source data

b) Preview the converted data

c) Create a pattern set.

B3.1 Converting the Source Data

The data conversion interface (Figure B-2) was designed to activate the data conversion module. At present, the data conversion module is only able to convert the data from HCS output files into the standard format that can be read by LOSANN. In the future, the data conversion module will be adapted to any type of data.

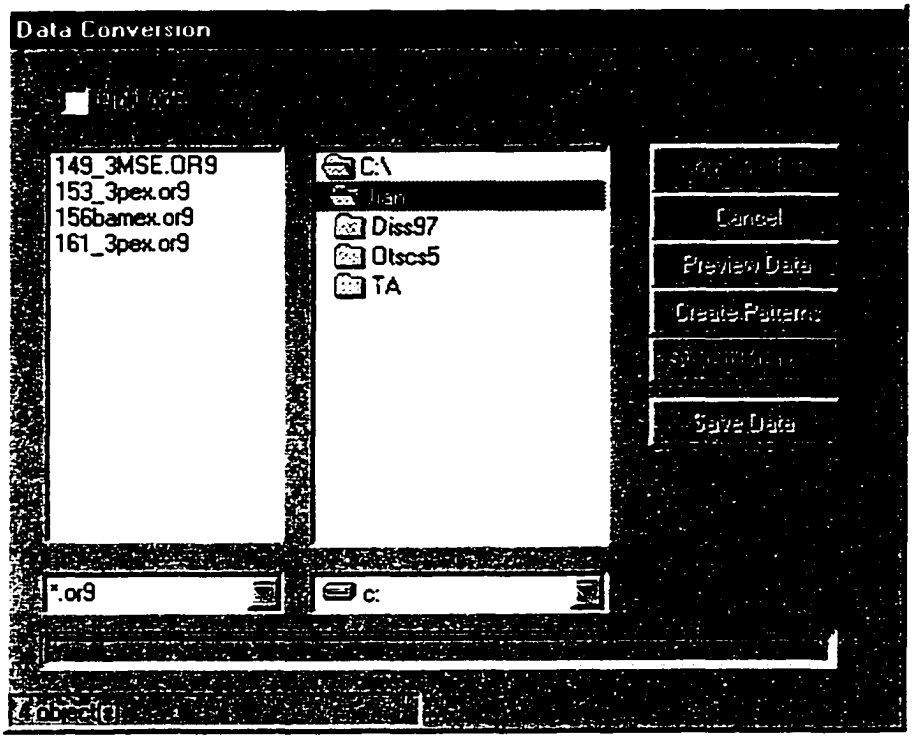


Figure B-2 Data Conversion Form

Before converting data, the user must place all of the HCS output files into one directory. The user can select the disk driver from the driver "combination box" and select the directory in which the source files are loaded from the directory list box. Whenever the user highlights the selected directory, all of the files with the specific extension name will appear in the file list box. For this dissertation, the default file extension name is "or9" as shown in Figure B-2, indicating "operations analysis of

Chapter 9 of HCS". The user can check the files stored in the file list box. The status bar located at the bottom left of the form indicates how many files are listed in the specific directory.

There are six control buttons on the right side of this form:

1. Conversion control button – used to trigger the conversion program. When the user clicks this button, all the files listed in the file list will be converted in the order of appearance into the standard format designed for LOSANN. If Only One is checked, only the file highlighted in the list box will be converted.
2. Cancel button – used to cancel the conversion process. When the cancel button is clicked, the conversion is terminated and the converted files are deleted.
3. Save button -- used to save the converted data file. The data file extension must be ".tem". Once data are converted, the data is automatically saved into a temporary file, "~convert.tem", in the current path. However, unless explicitly saving as ".tem", the temporary file will be deleted when any one of the following cases occurs:
    a. The user comes back to the data conversion or to the system form from the data preview form
    b. The user comes back to the data conversion or to the system form from the pattern creation form
    c. A pattern file is created.
4. Preview data button – used to preview the data converted from HCS files before the user creates a pattern set (see the section Previewing Data).

5. Create pattern button – used to create a pattern set (see the section Creating Pattern Set).
6. System button – used to return to the system menu.

The status bar on the bottom of the form provides the user with messages such as the number of HCS files in the current list, the number of files converted, and the location of the data file just saved.

B3.2 Previewing Data

The data preview form (Figure B-3) provides the user an opportunity to examine the profile of the data before saving them as a pattern set for LOSANN. This form consists of five areas in addition to the data file listing box:

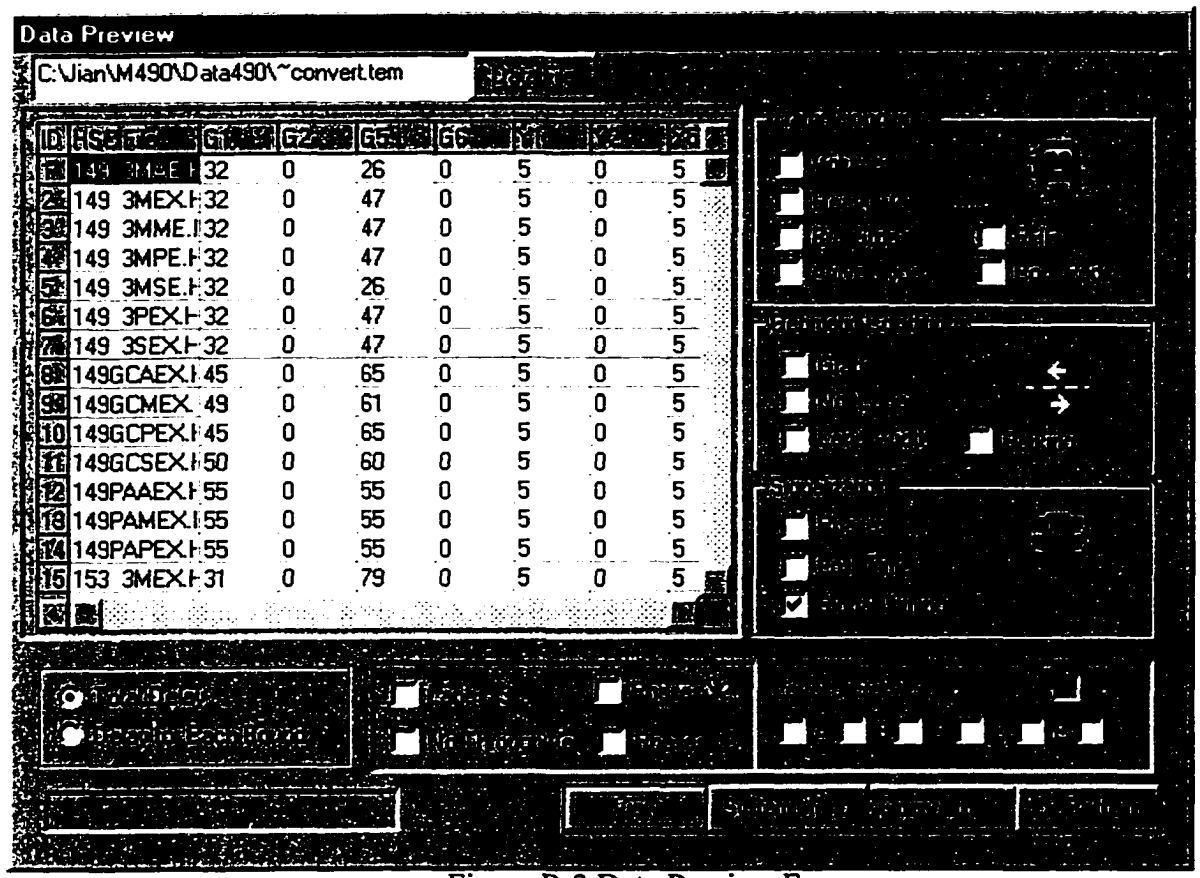


Figure B-3 Data Preview Form

1. Data exploration area in which data values are shown
2. Column selection area, which consists of four frames to select traffic, geometric, signalization conditions, and an average stop delay per vehicle.
3. Row selection area, which consists of two frames to select the level of service and / or to make queries about phasing and the non-existence of through vehicles.
4. Control button area at the bottom right corner of the form
5. Status bar

Before previewing the converted data, the user must type the data file name in the text box on the top-left corner of the form. If the form is called from the data conversion form, the data saved in the temporary file will be automatically displayed in the data preview form. The control button with the caption " Data File" helps the user find the data file desired.

Data values are shown in tabular format in the data grid. The columns of the data grid represent the factors that affect the traffic operation at an intersection and the rows of the grid count the number of the converted files. Each converted file becomes a record in the data grid. The data grid needs 136 columns to list all the information. However, the user has the total control over how many should be selected through the column selection frames. Similarly, the user can restrict the selection of records based on the constraints presented in the rows selection area. Once the user determines factor and record type selections, he can click the view button to see requested information in the data grid. The status bar will show the user the number of records shown in the data grid.

B3.3 Creating a Pattern Set

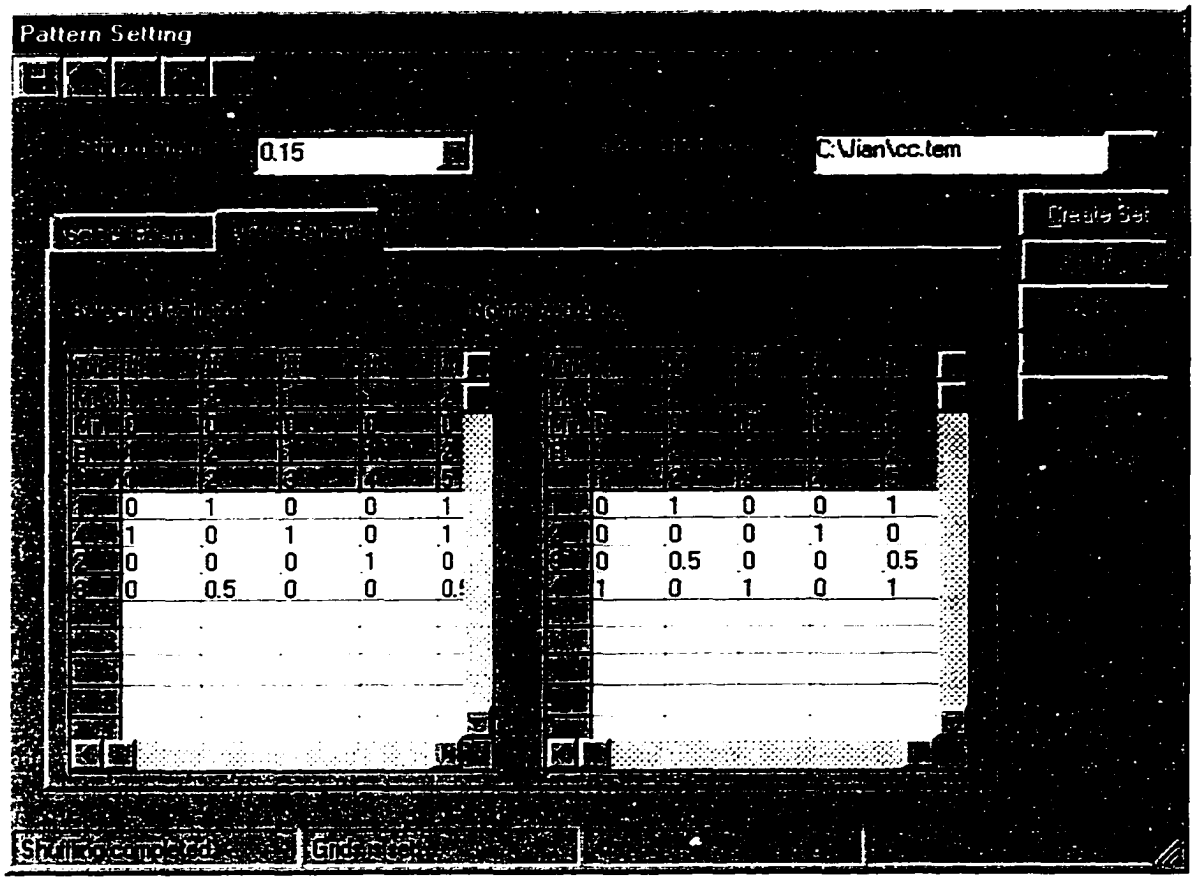


Figure B-4 Pattern Setting Form

After the user previews the converted data, he should click the control button named ">>Pattern", to get the pattern creation form. The user can directly switch to the pattern creation form, if so desired, from the *Utility* menu on the system menu form. The interface of the pattern creation form (Figure B-4) is similar to the data preview form, but its function is completely different from the latter. The pattern creation form provides the user an interface to create a pattern set based on the user's requirements, rather than just reviewing data. The procedure for creating a pattern set is described below:

1. Type the converted file name with the complete path in the text box in the top right corner of the form. If the user cannot remember the path and / or the file

name, he can click the control button labeled "..." on the right of the text box, which will help the user find the required converted data file.

2. Select the percentage of the patterns to be saved as test pattern from the comb box in the top-left corner of the form. The default value of the percentage is 15%.
3. Check the factors that you want to input LOSANN from the tab named "Select Factors". You may find that there are some factors that are always checked since they are basic factors for traffic level of service analysis.
4. Click the control button label as "Create Set", on the right side of the form. After waiting for a few minutes if the data set is large, the user can view the patterns from the tab labeled as "View Patterns". There are two grids on that tab. The left grid displays the original values of the patterns and the right grid displays their normalized patterns.
5. Click the control button, "Shuffle", on the right side of the form to shuffle the patterns in the set in random orders. Shuffled patterns will improve the training of LOSANN. The user can view the shuffling results from the left grid on the "View Patterns" tab.
6. Click the control button, "Save", on the right side of the form to save the pattern set. OTSCS automatically saves the training pattern set by the extension ".spt" and the test pattern set by "_tes.spt".
7. Read the information provided in the status bar on the bottom of the form to check where and how many patterns have been separately saved for training and testing.

8. Click the control button, ">>LOSANN", to load the LOSANN interface.

B4 LOS Analysis Neural Network (LOSANN)

The form of the LOS Analysis Neural Network (LOSANN) is shown in Figure B-5. It is the interface to train, test and modify artificial neural network models (ANNs). Since each of these tasks needs to follow a set of procedures, six subsections are used to guide the user. The first three subsections guide the user in using this interface to train, test, and modify ANNs, and make one ANN model to become the best LOSANN model. The fourth subsection will describe control buttons and their functions. The fifth subsection introduces the tabs which are used to represent the pattern set information and to set the parameter for LOSANN model. The last subsection is the status bar that provides dynamic run-time information.

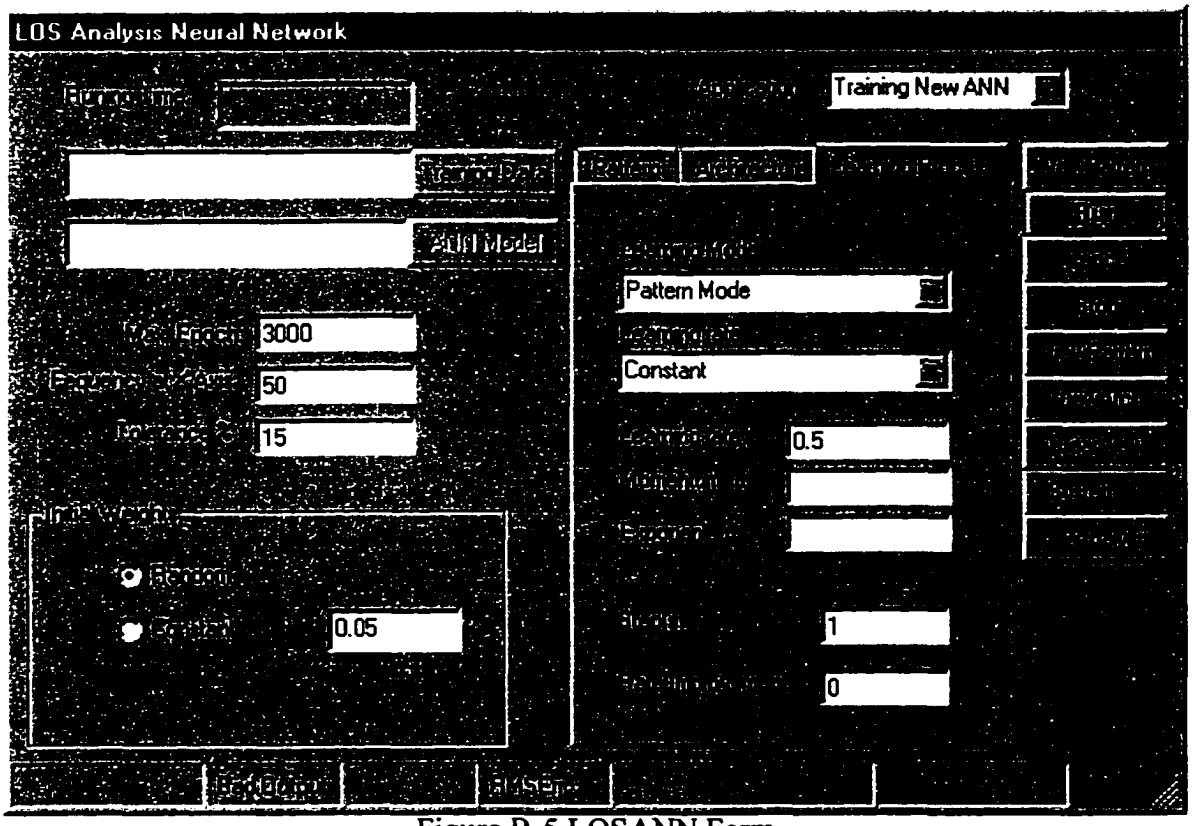


Figure B-5 LOSANN Form

B4.1 Training Artificial Neural Network Models

If the user does not have an acceptable neural network to perform the level of service analysis of signalized intersections, one can be developed. The following steps describe the training procedure:

1. Select the "Training New ANN" option from the comb box labeled "Application" on the top-right of the form
2. Type the pattern set name with its full path in the text box that is found to the left of the "Training Data..." button. If the user cannot remember the path and the name of the desired pattern set, he can click the "Training Data ..." button to see a list of available pattern sets. Note that the training pattern set is always saved with the ".spt" extension.
3. Check the tab labeled "Patterns" to see whether the opened pattern set is the desired one. This tab tells the user the number of input factors (signals) and the number of desired signal(s) on the output layer, as well as the total number of training patterns.
4. Select the architecture for the LOSANN from the tab labeled "Architecture". There are three architecture options available: single layer, one hidden layers and two hidden layer ANN model.
5. Select the learning mode and learning rate from the tab labeled "Learning Process". There are two possible learning modes (pattern mode and batch mode) and three possible learning rate types (constant, linear and weighted exponential learning rate), as discussed in Chapter 5.

6. Enter the maximum epoch allowed for training in the text box labeled "Maximum Epoch" on the left side of the form. The default value of the maximum epoch is 3,000.
7. Enter the frequency interval desired for the analysis from the text box labeled "Frequency Interval". The frequency interval determines how often the results are recorded. The default value is set to 50.
8. Select the tolerance level that can be accepted by the user. The default tolerance value is 15%, that is, if the activation signal in the output layer is within ±15% of the desired value, LOSANN is considered to have generated a "good" output.
9. Select the method to determine the initial weights on the ANN by selecting from the Initial Weight option frame in the form.
10. Press "Run" button to start the training process.
11. Save the ANN model by clicking the OK button, if desired, in the Save Dialog Window which appears automatically after the training process is completed. The default extension of the ANN model is ".tan".
12. Save the training result by clicking the OK button. The training results file is saved in a file with the default extension ".pre" for later analysis.

B4.2 Testing an Artificial Neural Network Model

If an artificial neural network model is trained successfully, LOS Analysis Neural Network form can be used to test the trained ANN model. The test procedure is described below:

1. Select "Testing" from the comb box labeled as "Application" on the top right of the form.
2. Type the artificial neural network name with its full path in the text box labeled as "ANN's Mode". The user can click the "ANN's Model" button to find the ANN model desired, if they are previously developed.
3. Review all the information about the selected ANN model from the tab "Architecture" and the tab "Learning Process". The information in these tabs is read only.
4. Type the pattern set name with its full path in the text box that is left to the "Testing Data" button. The "Testing Data" button is the same button as the"Training Data" button. Once "Application" is selected as "Testing", the caption of the button automatically changes from "Training Data" to "Testing Data". If the full path of the desired testing pattern set cannot be remembered, click the "Testing Data"" button to see a list of available test pattern sets. The extension of the testing pattern set must always be "_tes.spt". This is not a problem if the testing pattern set was named correctly in the pattern creation form.
5. Check the "Patterns" tab to see whether the opened testing pattern set is desired. This tab displays the number of input factors and the number of desired signals on the output layer, as well as the total number of the testing patterns in the selected data file.
6. Press the "Run" button to start the testing process.

7. Save the test results by clicking the OK button in the Save Dialog Window which appears automatically after the test process is completed. The default extension of the test result file is ".tst".

B4.3 Modifying Artificial Neural Network Models

LOS analysis neural network, like any other neural network, needs to be modified or improved since traffic characteristics will change in time and in space. An existing artificial neural network model can be modified by continually training it. In order to modify or retrain an already available LOSANN, the user must proceed step by step:

1. Select the "Training Exit ANN" from the combination box labeled as "Application" on the top right of the form.
2. Type the name of the selected artificial neural network with its full path in the text box "ANN's Model". The user may click the "ANN's Model" button to find the desired model from a list.
3. Type the pattern set name with its full path in the text box next to the "Training Data" button. If the path and / or the name of the desired pattern set cannot be remembered, the user clicks the "Training Data" button to list available training data file names. The training pattern set must always end with the "_.spt" extension.
4. Reset the learning mode or learning rate if necessary. Note that the architecture of the LOSANN cannot be changed.
5. Reset the value of the maximum epoch. The value must be larger than the value shown in the first panel of the status bar, which represents the epoch number of the previous training.

6. Click the "Run" button to start training process
7. Save the improved model with the "_.tan" by clicking the OK button, if desired for later use, in the Save Dialog Window which appears automatically after the training process is completed.
8. Similarly, save the training results with "*.pre" by clicking the OK button in the Save Dialog Window if desired for later analysis.

B4.4 About the Control Buttons

There are eleven control buttons in the form of LOS Analysis Neural Network. Table B-1 lists their names and functions in the order of their appearances.

Table B-1 Control Buttons on the Form LOS Analysis Neural Network Form and Their Functions

Name of Button	Function
Run	Run the programs for training, testing, and modifying ANN models based on the user's options.
Cancel	Cancel the currently running program
Stop	Stop the programming temporarily
View Pattern	Display detail information about the opened pattern set by restoring an additional message box in the pattern tab.
View Arch.	Display detail information about the architecture of the current ANN model by presenting an additional message box in the architecture tab.
Clear Views	Clear the additional message boxes
New Pattern <<	Go to the Pattern Creation form
System <<	Go to the System Menu form
>> Analysis	Go to the Results Analysis form
>> OTSCS	Go to the Optimum Traffic Signal Algorithm

B4.5 About the Tabs

The tabs in this LOS Analysis Neural Network form are used to present information about the pattern set, the architecture of the ANN model, and the mode, the rate and the parameters of the learning process.

The Pattern tab lists the number of the input and output signals, the total number of patterns, and the name of the input signals when the "View Pattern" button is clicked.

The Architecture tab explores the structure of the ANN model. The information in the tab includes the number of neurons on the input or on the output layers, and on the hidden layer(s) if there is a hidden layer(s).

The Learning Process tab shows the learning process mode and the value of learning rate. It also provides the user an opportunity to select the training schedule. The single training schedule means when the number of bad outputs is equal to 0 or the maximum epoch has been reached, training is terminated. The multiple training schedules mean more than one training process will be sequentially implemented. The value in the text box labeled "Training Times" in the "Learning Process" tab determines the schedule type. Its default value is 1, that is, "using the single running schedule". The value in the text box labeled "Rate Improvement" sets the step size for the learning rate improvement. For example, the constant learning rate is set at 0.5, the value of the training schedule is set to 4 and the rate improvement is set at 0.5, then the program will

sequentially train the ANN models with the learning rate 0.5, 1.0, 1.5 and 2.0, respectively. The default value for the rate improvement is 0.

If the user chooses the multiple training schedule, OTSCS will provide the dialog box for entering the file names for saving trained ANN models and their learning results before training begins. OTSCS will automatically add the learning rate on the entered file names. For example, the user enters his LOSANN model' name "ANN1". The constant learning rate is equal to 0.5. The training schedule is set at 4 and the rate improvement is set to 0.5. The LOSANN models will be saved as ANN1_5.tan, ANN1_10.tan, ANN1_15 and ANN1_20.

B4.6 About the Status Bar

The status bar on the bottom of the form dynamically shows the order number of the current learning epoch, the bad output from the last epoch, and the value of the roots mean square error from the last epoch.

B5 Training and Testing Analysis

The Training and Testing Analysis form provides an interface to view the results of training or testing. Since this form is linked to Microsoft Excel 7.0, the user can use any of Excel's functions to perform the analysis and prepare the report. The Training and Testing Analysis form (Figure B-6) includes two areas and four control buttons. The file selection area provides an easy way to select one or more files for analysis. All of the selected files should be moved into the "List for Analysis" area which is located on the

top right of the form. Whenever a file in the file list box in the top left corner of the form is highlighted, its description will be displayed in the information area in the bottom left corner of the form.

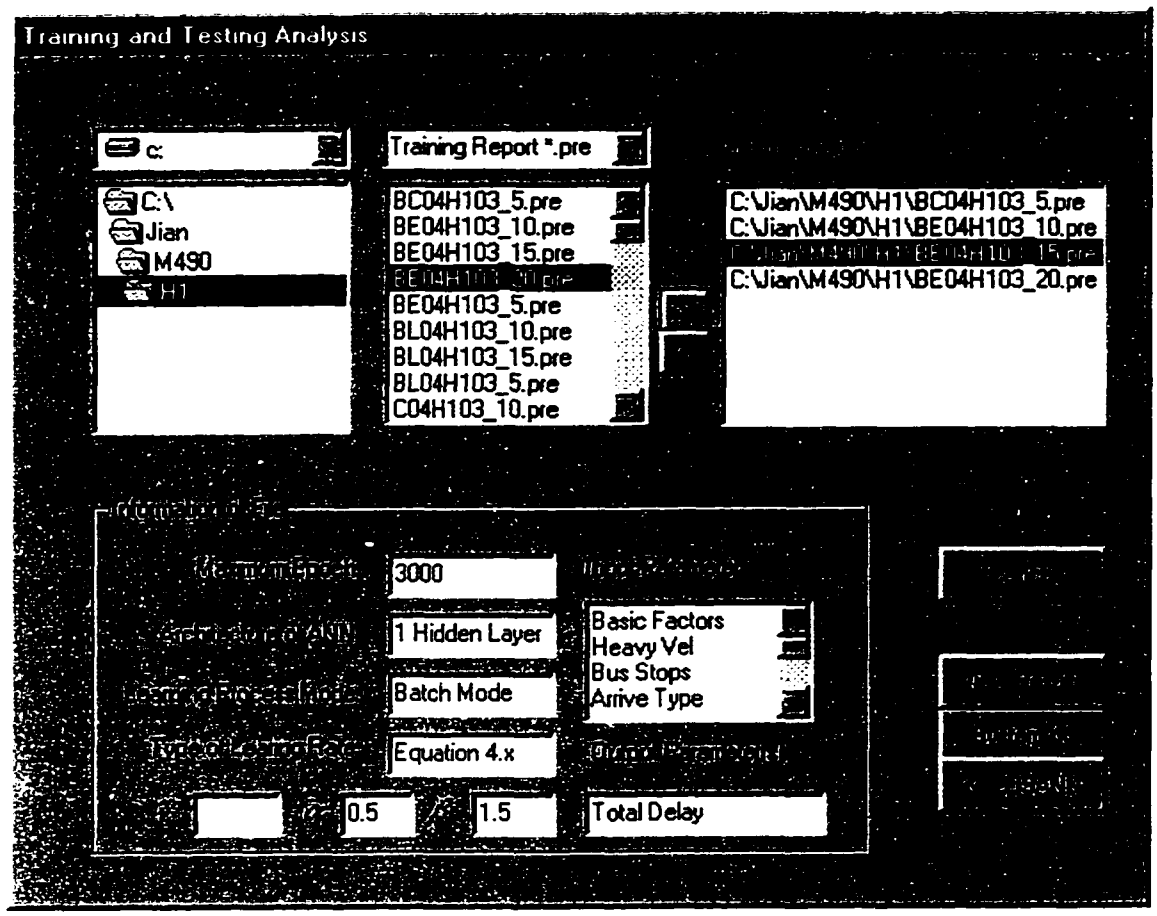


Figure B-6 Training and Testing Analysis Form

Once the result file(s) is moved into the analysis area, the "View Results" control button is enabled. Click this button to see all the interesting results from the spreadsheet.

The functions of the "System" button, "Cancel" button, and "LOSANN" button are similar to the buttons on the LOS Analysis Neural Network form.

B6 Optimum Traffic Signal Control System

The Optimum Traffic Signal Control System window provides an interface to perform optimal traffic signal timing for an isolated signalized intersection. This form consists of two information windows, three groups of tabs, one frame, a status bar with four panels, and five control buttons (see Figure B-7). The two information windows on the top of the form show the user the LOSANN model and the intersection which is picked up from the pattern set for testing the algorithm, respectively. The latter is not always necessary since the user can enter the traffic condition, geometric condition, and signalization requirement into the specified tabs manually.

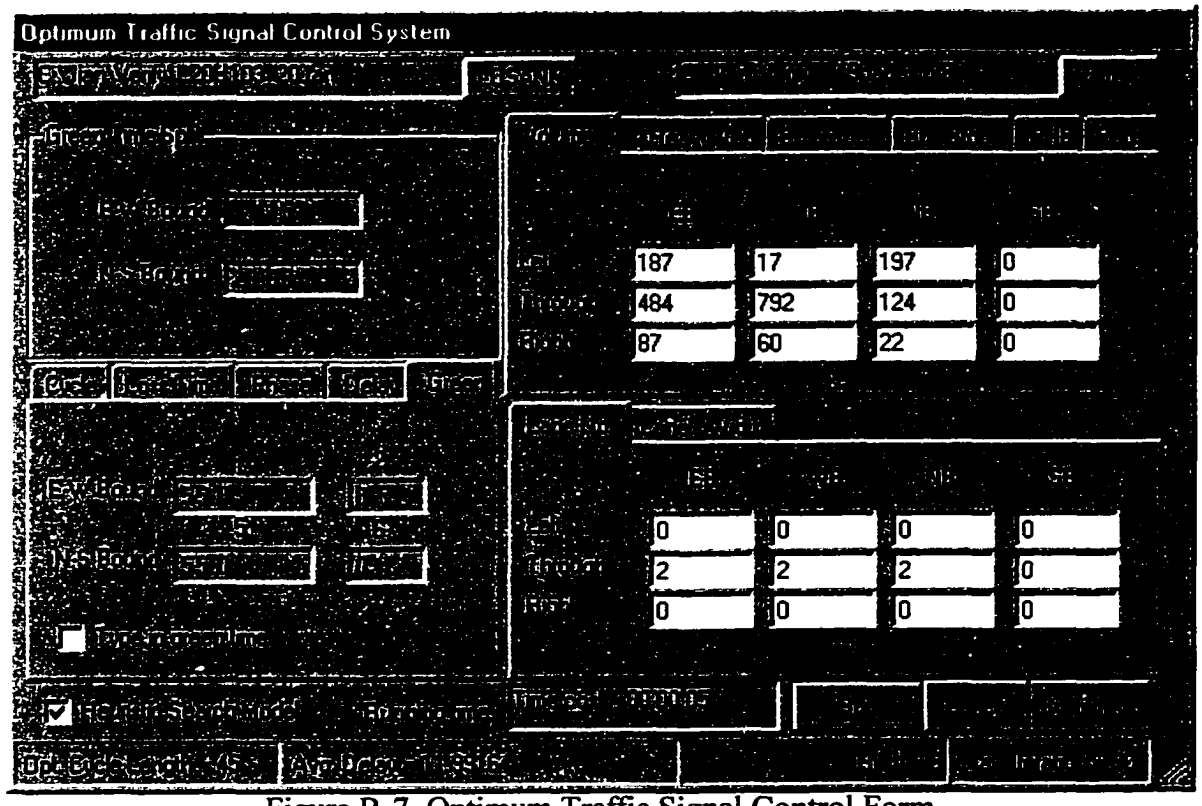


Figure B-7 Optimum Traffic Signal Control Form

The three groups of tabs are designed to input the factors to describe the traffic condition, geometric and signalization requirements. The frame with the caption "Green Time Split" and the status bar display the process and the results of the optimization.

The functions of the control buttons are self-explanatory from their captions.

The following are the steps in the optimization procedure:

1. Select a LOSANN by clicking the LOSANN control button. If more information about LOSANN is required, double click the window showing the name of the ANN model after selecting the model.
2. Input all the information required into the tabs. The information entered must be detailed enough to describe the traffic environment. For running the optimization routine, a pattern is selected by clicking the control with the caption "Pattern". Once the selected pattern appears in the window on the top right corner of the form, required factors will be filled in the text boxes on the tabs. The information also can be entered manually.
3. Set the maximum and minimum traffic signal cycle length
4. Set the search step size (in second(s))
5. Click the RUN control button

When the optimization process ends, the optimum signal timing results will be displayed in the green time split frame and the status bar. The user can obtain the time spent on running the optimization process from the running time text box.

Note that only two-phase signals are dealt with by the current version of OTSCS.

REFERENCES

Aleksander, I., and H.Morton. *An Introduction to Neural Computing,"* London: Chapman & Hall. 1990.

Berry, D. S., *Notes on Traffic Engineering,* unpublished, Northwestern University, Evanston, III., 1978.

Bronx Center Transportation Study, New York City Department of Transportation, Technical Memorandum No. 1, September 1995.

California Scientific software. *BrainMaker Professional Neural Network Simulation Software User's Guide and Reference Manual,"* 4th Edition, Nevada City, July 1993.

Downtown Flushing Transportation Improvement Study, New York City Department of Transportation, PL900T(021), PT4446.896, July 1996.

Faghri, A. and J. Hua. Evaluation of Artificial Neural Net work Applications in Transportation Engineering. *Transportation Research Record,* 1358, 1992, pp.71-80.

Federal Highway Administration, *TRANSYT 7-F: User's Manual,* FHWA, Washington, D.C. 1987.

Fan, Jianzhong and Mitsuru Saito, Application of an Artificial Neural Network to Level of Service Analysis of Signalized Intersection, Presented at the Transportation Research Board 77th Annual Meeting, January, 1998.

Feuer A. and E. Weinstein. Convergence Analysis of LMS filters with Uncorrected Gaussian Data. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol. ASSP - 33, No.1, 1994, pp. 222 -230.

Fwa, T.F., and W. T. Chan. Priority Rating of Maintenance Needs by Neural Networks. *Journal of Transportation Engineering*, Vol. 120, 1994.

Gallant, Stephen I. *Neural Network Learning and Expert Systems.* The MIT Press, Cambridge, Massachusetts, 1994.

George, F. L. & S. A. William, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* 2nd Edition, The Benjamin / Cummings Publishing Company, Inc. 1993, Chapter 3 and 4.

Guyon, I. Applications of Neural Networks to Character Recognition. *International Journal of Pattern Recognition and Artificial Intelligence* 5, 1991, pp. 353 – 382.

Hagiwara, M. *Theoretical Derivation of a Momentum Term in Back - Propagation.* International Joint Conference on Neural Networks, Vol. 1, 1992, pp. 682 – 686.

Haykin, S. *Neural networks—a comprehensive foundation.* Maxwell Macmillan Canada, Toronto, 1994.

Hebb, D. O. *The Organization of Behavior.* Wiley, New York, 1949

Highway Capacity Manual. Special Report 209, Transportation Research Board, National Research Council, 1994.

Hua, J. and A. Faghri. *Development of Neural Signal Control System.* Transportation Research Board 74th Annual Meeting, Washington D.C. January 22-26, 1995.

Huang, Xiao Hui *A Simulation-Neural Network Model for Traffic Control at High-Speed Signalized Intersections,* Dissertation, The Department of Civil and Environmental Engineering of the University of Cincinnati.

Lee, Y., S. Oh, and M. Kim. *The Effect of Initial Weights on Premature Saturation in Back - Propagation Learning.*" International Joint Conference on Neural Networks, Vol. 1, 1991, pp. 765 - 770.

Lin, Hsiao C., C. and M. Cassidy. Application of Fuzzy Logic and Neural Network to Automatically Detect Freeway Traffic Incidents. *Journal of Transportation Engineering,* Vol. 120 No. Sep./Oct. 1994, pp.753-771.

Mendel, J. M., and R. W. McLaren. "*Reinforcement-learning control and pattern recognition systems.*" In *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications* (Mendel, J.M. and K,S. Fu, eds.) New York: Academic Press, 1970, pp. 287-318.

Nakatsuji, T. and T. Kaku. Development of a Self-organizing Traffic Control System Using Neural Network Models *Transportation Research Record* 1324, 1991a, pp.137-145.

Nakatsuji, T. and T. Kaku. Development of a Self-organizing Traffic Control System Using Neural Network Models (Part 2). 1991b.

Papcostas, C.S. & P.D. Prevedouros. *Transportation Engineering and Planning.* 2nd Edition, Prentice-Hall, Inc., 1993.

Pant, Prahlad D. and P. Balakrishnan. Neural Network for Gap acceptance at Stop-Controlled Intersections. *Journal of Transportation Engineering,* Vol. 120 No.3, 1994 pp.43.

Ritchiel, M. Kaseko, and B. Bavarian. Development of an Intelligent System for Automated Pavement Evaluation. *Transportation Research Record* 1324, 1991.

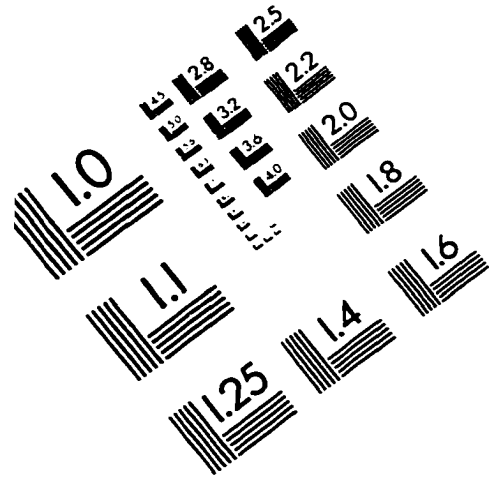
Roy, S., and J.J. Shynk, Analysis of the Momentum LMS Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing* ASSP - 38, 1990, pp. 2088 – 2098.

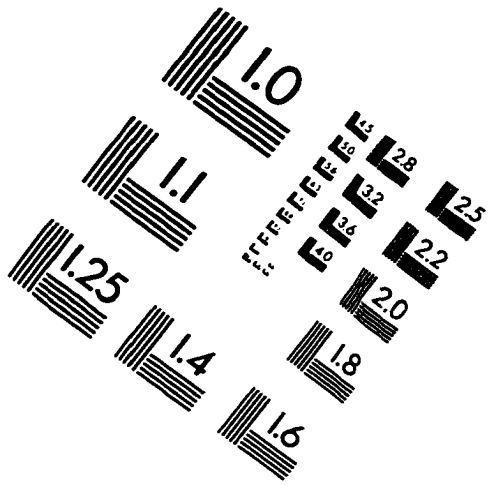
Rumelhart, D.E. G.E. Hinton, and R.J. Williams. Learning representations by Back - Propagation Errors. *Nature* (London), 323, 1986, pp. 533 – 536.

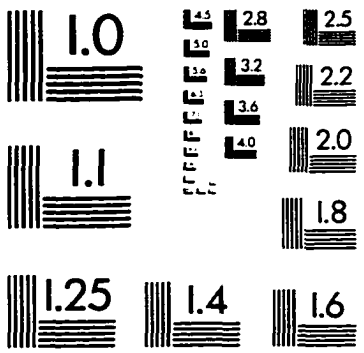
Russo, A. P. *Neural Networks for Sonar Signal Processing.* Tutorial No. 8. IEEE Conference on Neural Networks for Ocean Engineering, Washington, DC., 1991.

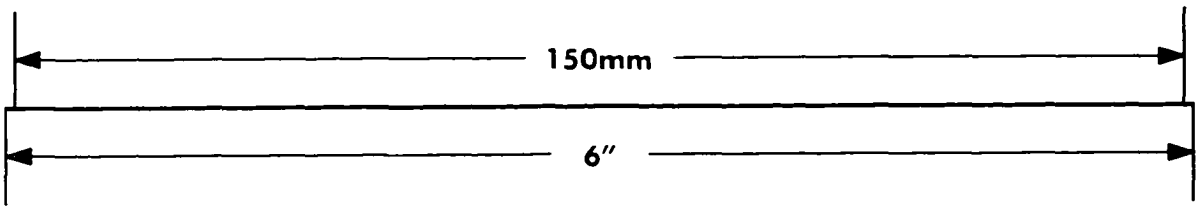
United States General Accounting Office, *"SMART HIGHWAYS, An Assessment of Their Potential to Improve Travel,"* GAC/PEMD-91-18, May 1991.

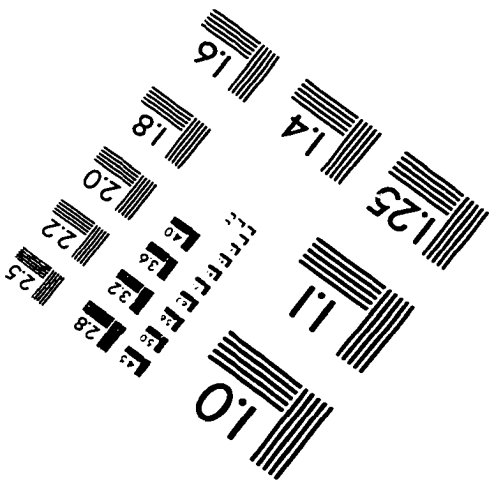
IMAGE EVALUATION
TEST TARGET (QA-3)

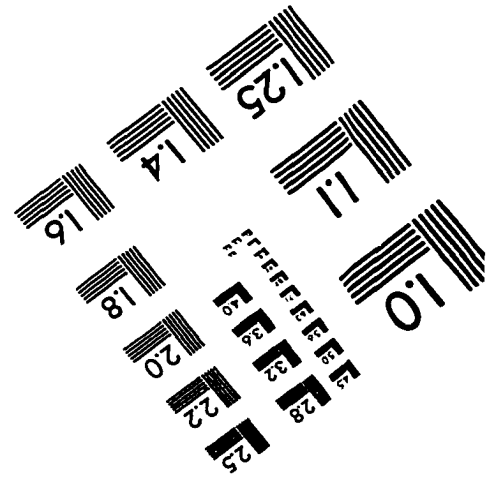












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989