CHAPTER 3 SYSTEM OVERVIEW

3.1 Introduction

This chapter serves as an overview of the entire dissertation. The next three sections follow the topical research protocol by introducing the goals of the research (Section 3.2), describing the methodologies applicable to the research (Section 3.3) and presenting the system architectures of the proposed system (Section 3.4). In Section 3.4, interfaces, subsystems, and program flow are emphasized. Finally, Section 3.5 presents the system function requirements.

3.2 Objectives

3.2.1 Motivation

The motivation behind this research is to systematically approach the grade crossing problems found in the literature review summary (Section 2.7). To achieve state-of-the-art significance in this field, research needs to be done on both railroad and highway sides.

On the railroad sides, the current railroad grade crossing safety studies concentrate on grade crossing protection devices, such as two- and four-quadrant gates. Those safety measures aim at preventing highway vehicles from entering the right-of-way within a grade crossing and conflicting with trains. The example of such a dramatic measure is the impenetrable barrier applied in the HST situation.

Instead of protecting grade crossing passively and showing the hostility to the highway drivers, we could develop an advanced traffic management system smart enough to control the traffic near grade crossings. In such a system, we could incorporate grade crossing information into traffic control and prevent the queue from backing onto the railroad tracks. Preemption of the traffic signal at/near a grade crossing is such an

alternative to target safety improvement. However, the current preemption practice leaves room for improvements suggested by the accident investigations and the researchers (from Section 2.3.3 to Section 2.3.6). The proposed system in this dissertation should be able to promote the safety of grade crossing and improve traditional preemption limitations.

On the highway side, the congestion and the traffic delays are of the primary concerns. Most of us could understand that, if the highway traffic signals were preempted to incorporate the railroad operations, it would be in our best interest to sacrifice our personal time for extra delays. We expect such preemption promotes safety at a grade crossing. However, could we promote grade crossing safety and reduce traffic delay at the same time?

This research explores the possibility of incorporating grade crossing closure information into the highway traffic optimization, which at least reduces the impact of the preemption to the highway vehicles' delays. Fortunately enough, the experiments in case study show that the benefits of the proposed system are quite beyond our expectations: the *overall* network delay is mitigated significantly. We don't have to sacrifice our time in preemption-related delays in exchange of a safer grade crossing, in this case.

In summary, to promote grade crossing safety and to mitigate the traffic delays establishes the research objectives.

3.2.2 Objectives

Based on the above discussions, there are two objectives of this research project.

- Promote grade crossing safety
- Reduce highway traffic delays

More specifically, the safety of grade crossing can be promoted by de-queuing those vehicles detected on the railroad tracks before the arrival of trains. The proposed implementation differs from the preemption call of the traditional traffic controller (e.g. 170/NEMA) in that the embedded inference engine takes the on-line surveillance detector input, dynamically chooses the phase sequence and calculates the de-queuing time. In the traditional preemption call, however, a pre-determined phase is actuated and a fixed length of preemption time is applied.

The proposed implementation is safer than the traditional approach because the preemption does not end unless there is no vehicle detected on the tracks. On the other hand, because vehicle speed varies, it is difficult to design a pre-calculated preemption time applicable to all vehicle types on various occasions; there still might be one or more vehicles left on the tracks.

Similarly, responding to the real time detector data, the proposed implementation causes less interruption on highway traffic flow. For instance, if there is no vehicle detected on the preemption link, a preemption phase may not be necessary.

The second objective is very intuitive and is implemented in many traffic optimization algorithms. The implementation here distinguishes itself from other algorithms in that grade crossing closure information is the constraint to the optimization variables--the phase length. The constraints contain critical grade crossing safety factors

and dynamically respond to the detector surveillance data. The implementation also inherits the dynamic phase sequence selection explained in the first objective.

3.2.3 MOEs and Delay Objective Function

In the dissertation so far, delay reduction has been selected as one of the objectives without any elaboration. Actually, the traffic signal control is designated to improve the Measurements of Effectiveness (MOEs) at the controlled intersection. This section explains how delay is chosen as an objective function. The common MOEs include:

- Delays
- Total travel time
- Total travel
- Number and percentage of stops
- Total minute-kilometers of congestion
- Average speed
- Accidents
- Throughput
- Fuel consumption
- Emission of pollutants

Among them, the delays are particularly interesting since the Highway Capacity Manual (HCM) (TRB 1997) adapted Intersection Control Delay (ICD) as the only MOE to determine intersection Level of Service (LOS). In the HCM, the ICD was defined as follows: "Control delay includes initial deceleration delay, queue move-up time, stopped delay and final acceleration delay." The procedures to analytically determine the LOS and the ICD are carefully explained in the HCM. For this reason, the summation of the approximation of the ICD on all surveillance links in the next two minutes is chosen as the objective function throughout this research project.

Measuring control delays on-site is difficult. There are several approaches to measure the ICDs at the intersections:

- Test-car observing
- Path tracking
- Arrival and departure queue observing

The HCM (Appendix III of Chapter 9) details how to measure the ICD using the arrival departure queue observation. In addition to the HCM approach and the on-site measuring approaches, simulation provides engineers an option to collect ICDs. For example, in micro simulation software like CORSIM, it is possible to track every individual vehicle profile and to make accurate counts of ICDs (Zhang and Zhang 2000).

Despite of the varieties and the difficulties of measuring and modeling the ICD, an approximation to the ICD is proposed in CHAPTER 4 . The SOURCAO delay model is based on the proposed delay surveillance in Section 4.2. The proposed delay model takes the simulated surveillance detector data as input; therefore, it is possible to implement SOURCAO in the real traffic network. The details of the delay model are discussed in Section 4.4 and the validation is presented in Section 6.2.1.

Furthermore, the traffic signal optimization requires not only approximating the delays until current time, but also forecasting the delays in a projected period (2 minutes in this research). The delay forecast is addressed in Section 4.5.

The scope of "delay" in SOURCAO is approximated to the ICD defined by the HCM, except that the "acceleration delay" is not considered. However, the acceleration delay contributes only to a small proportion of the ICDs. The term "delay" from this point refers to the approximation delay unless specified.

3.2.4 Approaches

The implementation is divided into two steps, corresponding to the two objectives. The first step is to choose a proper preemption phase sequence; the second step is to find the optimized phase length.

The appropriate next phase favors the safety of grade crossing. An inference engine can achieve the first objective. The inference engine takes the on-line detector input, dynamically chooses the phase sequence and calculates the de-queuing time constraints according to the detected queue length.

The optimization process uses the traffic delay at the intersections within grade crossing vicinities as an objective function. A delay model is proposed and developed with the system. The optimization is implemented into two steps. In the first step, the delay function is approximated and represented by a multilayer perceptron neural network (off-line training). After the function is trained and obtained, an algorithm named Successive Quadratic Programming (SQP) searches the optimized length of the phases so that the total delays in the network can be minimized (on-line).

Moreover, simulation software is chosen to evaluate and validate SOURCAO. Due to cost, availability and performance of the simulation software, the summary of simulation software in Section 2.7 indicates that TSIS/CORSIM is the favorite. In other words, the interface between SOURCAO and TSIS/CORSIM should be carefully considered.

3.3 Methodologies

Today's technologies can be characterized as "explosive." The proposed system benefits the advanced technologies from integrating the different methodologies to achieve the designated objectives. The technologies are explored briefly in the section.

3.3.1 Artificial Intelligent

3.3.1.1 Intelligent Agent

The core mechanism to implement the rule of an intelligent agent in the proposed system is an inference engine. The inference engine in this research is a simple rule-based deduction procedure. The goal of the agent is to choose the next phase, according to the current available surveillance detector and grade crossing information. The complete details are presented in Section 5.2.

3.3.1.2 Neural Network

In this research, a multilayer perceptron neural network is applied to forecast the network traffic delay. The weights in a three-layer perception neural network are trained to store the traffic patterns so that the delay in the next 2 minutes can be predicted. The network traffic delay forecast can be expressed as:

$$D = \sum_{m \in M} d_m = \sum_{m \in M} f(X; W, U) \tag{3-1}$$

Where:

m is index to a link;

M is the collection of surveillance links;

d_m is the delay on link m;

W is the weight trained from the neural network (off-line training);

U is the detector and grade crossing information obtained from on-line;

X is control variables or phase length of all traffic signals in this case. In the proposed systems, three phases are considered to join the optimization. However, only the first phase is implemented and the second phase length is optimized again when the first phase is about to end. The total number of variables in SOURCAO is three times the traffic signals.

3.3.2 Optimization

The optimization is widely used in traffic signal control. Almost every traffic control algorithm employs at least an optimization algorithm.

After the weights in the delay function described in Section 3.3.1.2 become stabilized, the optimization can be started. The objective function in Section 3.3.1.2 is subject to the constraints below:

$$x_{\min} \le x_{ij} \le x_{\max} \tag{3-2}$$

$$x_1 + x_2 + x_3 \le 120 \tag{3-3}$$

Where:

i is the number of traffic signals;

j is the optimized signal phase number, *j=1, 2, 3;*

120 is the time to project the future delays (two minutes in seconds in the proposed system);

x_{min} and x_{max} are the minimum and maximum green time. The maximum green time is determined dynamically.

3.3.3 Software Engineering

Objective-Oriented Analysis is one of the modern software engineering approaches and follows three essential principles: encapsulation, class and inherence. There are dozens of OOA methods and the definition of OOA changes from one approach to another (Pressman 1997). Taking the Booch Method as an example, the OOA development can be outlined as follows:

- Identify classes and objects;
- Identify the semantics of classes and objects;
- Identify relationships between classes and objects;
- Conduct a series of refinements;
- Implement classes and objects.

Generally, the software design in this research follows those rules and OOA helps programming, debugging and testing.

3.4 System Architecture

Since TSIS/CORSIM is used to evaluate SOURCAO, the interface between SOURCAO and TSIS/CORSIM should be carefully considered. The details of this interface are presented in Section 3.4.1. Section 3.4.2 reveals the SOURCAO software program flow. Section 3.4.3, 3.4.4 and 3.4.5 briefly describe all subsystems. Their interactions are emphasized throughout the whole section.

3.4.1 Interface between TSIS and CORSIM

TSIS provides a fundamental and new tool to interact the traffic software, such as CORSIM and its Run Time Extension (RTE). Under the TSIS umbrella, various programs can communicate with CORSIM through a standard interface. Currently, TSIS facilitates Dynamic Link Library (DLL[1]) as the interface under WINDOWS 95/98/NT. Data among the different programs (DLLs) are exchanged through Microsoft conventions. For each simulation time step (1 second), TSIS calls a series of functions within CORSIM and RTE to drive the simulation event-loop.

There are two kinds of data exchanges between CORSIM and SOURCAO: network feature data (off-line) and on-line data. The network feature data include CORSIM network nodes, links, lanes, detector geometry and the CORSIM internal IDs. The online data are detector counts and signal control codes. In terms of data flow, SOURCAO can be imagined as a black box. In every second, it reads all detector counts (CORSIM FORTRAN name: DETON) from CORSIM and returns the traffic signal control codes (CORSIM FORTRAN name: SDCODE and AMBSPC) back to CORSIM.

---

[1] *A DLL is an executable file that acts as a shared library of function. Dynamic linking provides a way for a process to call a function that is not part of its executable code. The executable code for the function is located in a DLL, which contains one or more functions that are compiled, linked, and stored separately from the processes that use them. DLLs also facilitate the sharing of data and resources.* (Microsoft 1996)

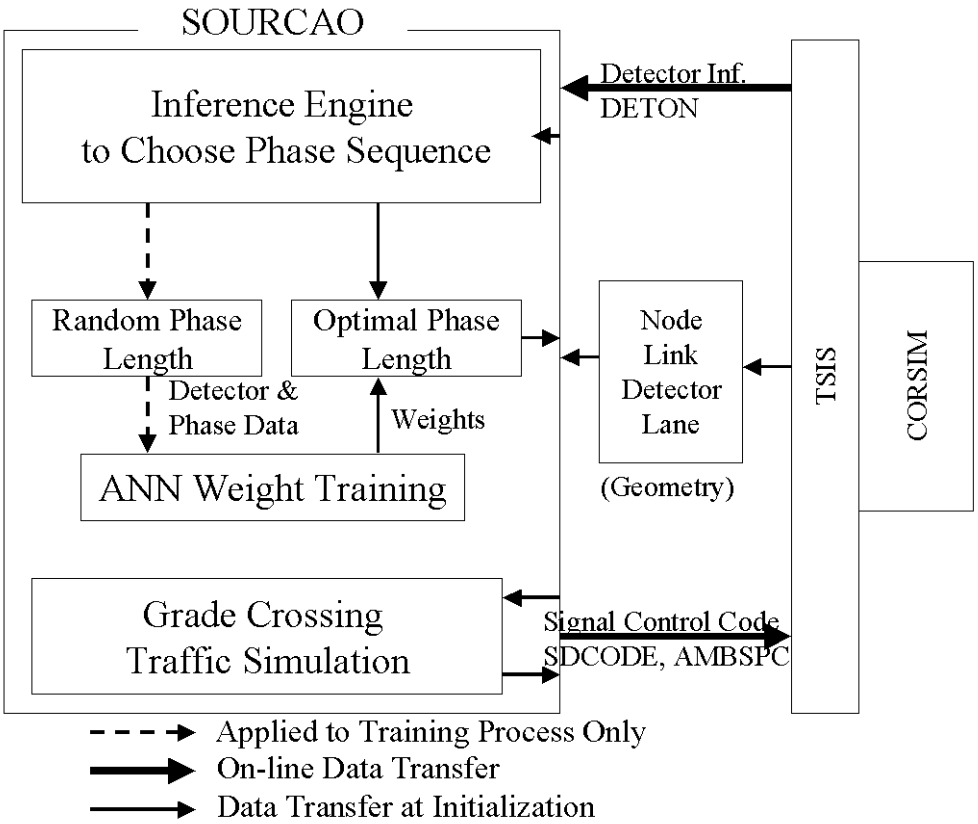

Figure 3-1 Data Flow at Interface between CORSIM and SOURCAO

3.4.2 Program Flows

In objective-oriented programming, *class* is introduced to support the idea of the system designing and modeling. The concepts of the user-defined data type and the class hierarchies are supported in the class. As described in Section 3.3.3, OOA guides the designing, debugging and testing of SOURCAO programming.

In this section, the class name starts with a capital letter "C." For example, the class "*CPriorityNetwork*" means the "*PriorityNetwork*" class. In addition, because function calls are associated with classes, *class name* is shown before the *function* calls, separated by two columns "*::*". For example, in Figure 3-2, *CPriorityNetwork::Update* means call the *Update* function in the *PriorityNetwork* class. (The priority control network is named since each of the phases is assigned a priority.)

In order to explain the program flow, some necessary classes are listed in Table 3-1. In addition to the table, three interface functions in TSIS/CORSIM are JMAIN, JINIT and JEXIT.

- JMAIN: An interface FORTRAN function, called by TSIS in each simulation step (1 second). JMAIN in turn calls a C++/C function (upctrl) inside SOURCAO.
- JINIT: An interface FORTRAN function, called by TSIS at the start of CORSIM simulation. JINIT in turn calls C++/C initialization functions (upinit) inside SOURCAO.
- JEXIT: An interface FORTRAN function, called by TSIS at the end of simulation. JEXIT in turn calls C++/C functions.

Table 3-1 Selected Classes and Member Functions

CPriorityNetwork	It represents a traffic network with grade crossing and highway traffic signals near HRGC. It inherits all the features of the CORSIM network. In addition, it genarates variables and function members necessary to implement intelligent agent and phase length optimization, as well as provide data to filter neural network training.	
	CreateRail	Defines grade crossing data and functions; genarates or inputs gate close/open time.
	CreateCORSIMNetwork	Copies the CORSIM network to priority network, like link/node/detector attributes
	CreatePriorityNetwork	Reads PriorityControl phase coding, phase priority and surveillance layout settings.
	CreateControl	Maps PriorityNetwork data in SOURCAO to CORSIM data
	UpdateSurveillance	Updates detectors and surveillance (queue, delay etc.)
	UpdateControl	Invokes the grade crossing control and intersection signal control updates
CGate	It represents a physical HRGC traffic controll mechanism.	
	Update	
	CPriorityControl Class	Updates grade crossing gate open/close control status.
CPriorityControl	It represents a physical traffic controller near highway intersections.	
	UpdatePhase	Calls all of the following functions and invokes phase updating.
	UpdateState	Updates grade crossing state defined in Table 3-2.
	Inference	Deducts and chooses the next phase
	PhaseLength; OptimizedPhaseLength; RandomPhaseLength	Determines the next phase length. One of the three functions can be chosen at a time.

TSIS controls the execution of CORSIM and a user-defined DLL run time extension. Before initializing CORSIM, TSIS invokes the interface function INIT. INIT calls a function (upinit) inside SOURCAO. Then in every second, before the execution of CORSIM, the JMAIN function is executed and triggers a function (upctrl) inside SOURCAO. On exiting the CORSIM, JEXIT is called.

In CORSIM, the traffic signals are represented in SDCODE and AMBSPC variables by the link and the intersection (Zhang and Hobeika 1999). The two arrays finally control the movement logic of vehicles in CORSIM. Before an intersection, every

vehicle in the CORSIM network checks those two arrays to determine if a vehicle needs to stop or go. With the proper values of those two arrays, the traffic signals function as desired. The rest of the section explains the program flow of SOURCAO in Figure 3-2 and Figure 3-3 in plain language and Objective-Oriented (OO) function point of view.

The number in a cycle in the following paragraphs of this section corresponds to the closest box in Figure 3-2 and Figure 3-3. The number in a blank cycle (e.g. ①) shows optimization program flow and the sequence (e.g. from ① to ②, ..., to ⑨) while the number in a dark cycle ❶ stands for a branch. The sequence ①, ②, ③, ④, ❶ to ⑨ indicates the branch route to simulate the current conditions without the proposed system (do-nothing). The sequence ①, ②, ③, ④, ⑤, ❷, ❸ and ⑦ directs the branch route to train neural network (❷ sends phase length to ⑨).

During the TSIS initialization, *INIT* calls *upinit* in SOURCAO. It in turn invokes other initialization functions inside *CPriorityNetwork*① such as copying CORSIM link/node/detector attributes, reading phases and priorities, reading surveillance layout settings, mapping and sorting data in PriorityNetwork. After the initialization in SOURCAO is completed, TSIS takes over the program control and calls CORSIM and *JMAIN* in every second until the end of simulation. During the execution of the *JMAIN* function, surveillance data② are updated first. Then *CGate*::*Update*③ decides whether grade crossing needs to be closed or not. According to the open- and closed-state status, traffic control code for each grade crossing intersection (SDCODE and AMBSPC in FORTRAN) is assigned the desirable value. After that, the complicated highway traffic signal control ④ is updated. SOURCAO checks if a phase is starting its amber interval and is about to complete. If not, the program directly flows from ④ to ⑨ and the signal control code in the previous simulation step is sent back to CORSIM. However, when a phase enters its amber interval, the actions are taken according to different user options.

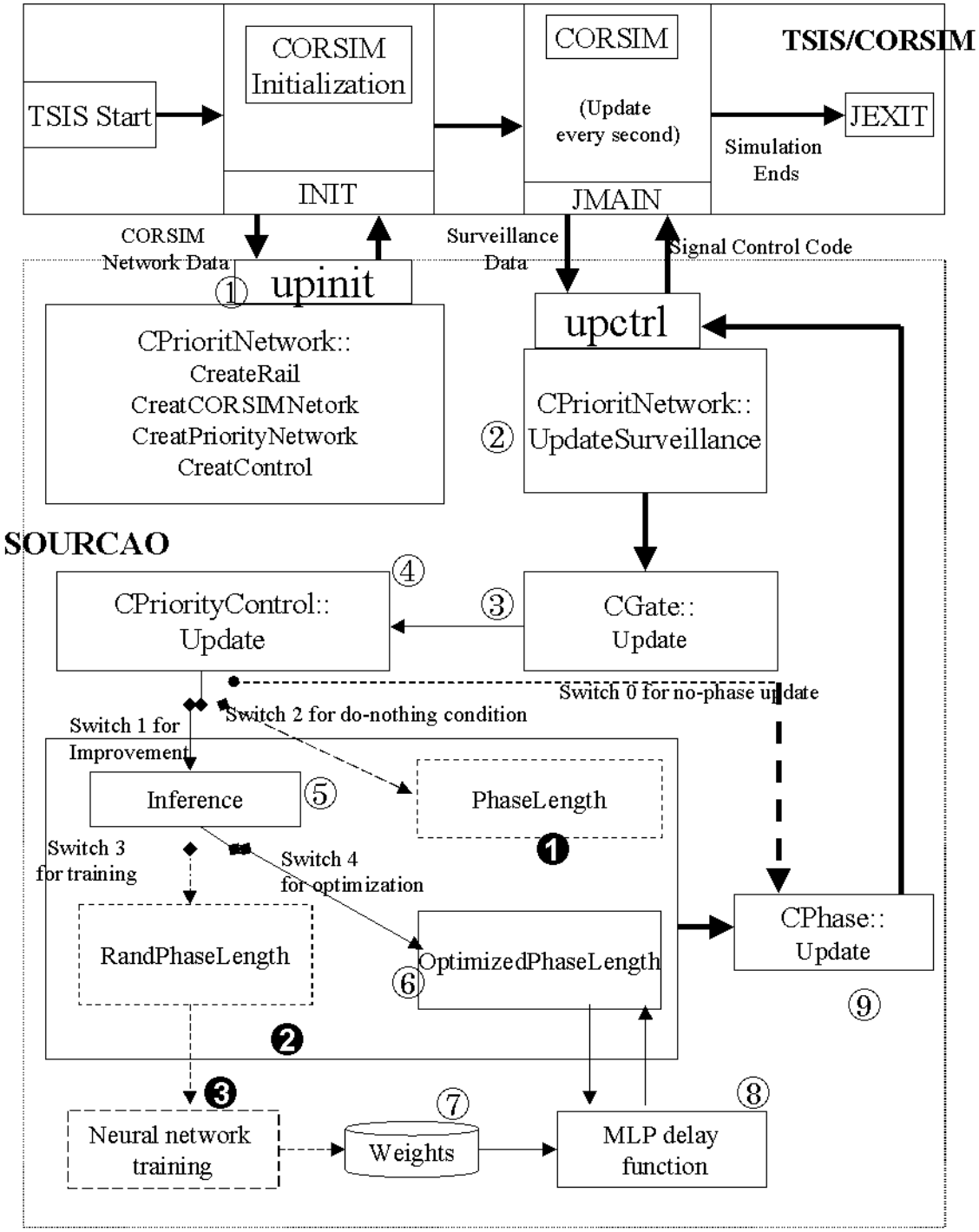

Figure 3-2 SOURCAO Program Flow by the Class and the Function

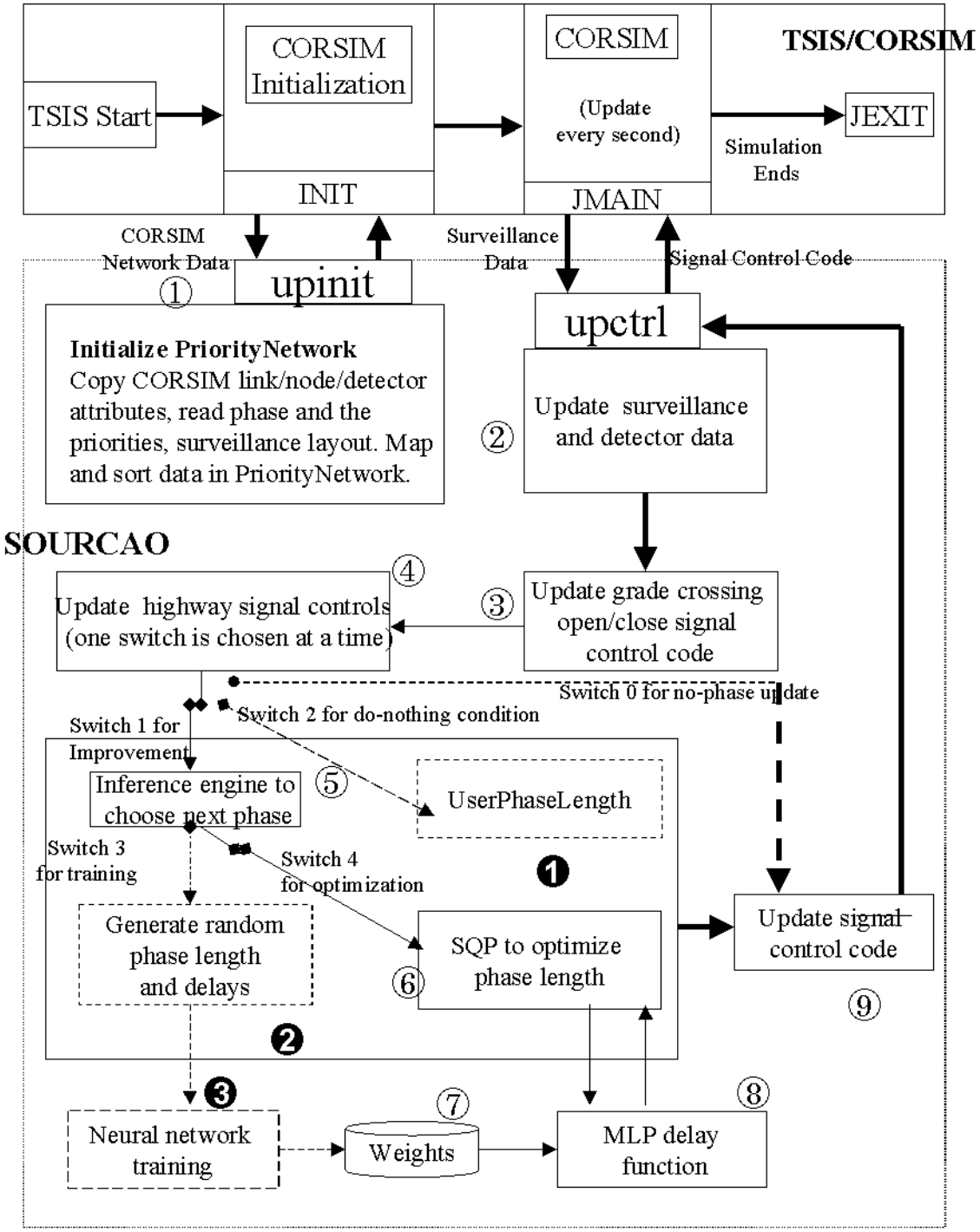

Figure 3-3 SOURCAO Program Flow

If a user chooses Switch 2, the pre-timed control strategy is employed and the user defines and specifies all phase sequences and phase lengths ❶. In order to make the comparison with the optimized phase length, the user-specified phases ❶ are used to simulate the current conditions without the proposed system (do-nothing).

Otherwise, Switch 1 brings the inference engine ⑤ to deduce the next phase. Because the optimization of phase length is achieved in two steps, Switch 3 in the first step is required before Switch 4 can be turned on. Switch 3 leads to the neural network training and is not required in every simulation run (In the case study, switch is turned on only once.). After Switch 3 is turned on, wide ranges of phase length are randomly generated ❷, and the relationship between phase length and delay functions is mapped from the multilayer perceptron neural network training ❸. The random phase length is also sent back to ⑨ to control signal. After the convergence of the neural network training is attained ❸, the neural network weights ⑦ are stored for phase length optimization ⑥ function usage. When Switch 4 is chosen, the optimization ⑥ takes delays forecast by the neural network as an objective function ⑧. After optimization process ends, the optimized phase length is sent to ⑨.

Whichever switch is chosen, the priority network phase in SOURCAO is finally converted to the CORSIM representation ⑨, which is sent back to CORSIM to update the signal status.

3.4.3 Traffic Control Intelligent Agent

Within SOURCAO, four grade crossing states are defined, as shown in Table 3-2. During the simulation, the state is updated in every simulation interval (one second). The traffic control intelligence agent takes the surveillance detector and grade crossing state information as input to decide the appropriate signal sequence (from all possible phase collections) to promote the safety of grade crossing.

Table 3-2 Grade Crossing States

State	Descriptions
0	Grade crossing is opened to highway traffic (default)
-1	Grade Crossing is about to close to highway traffic; pre-signal preemption may be needed.
1	Grade Crossing just opens to highway traffic; restore-signal preemption may be needed.
9	Grade Crossing just closes to highway traffic.

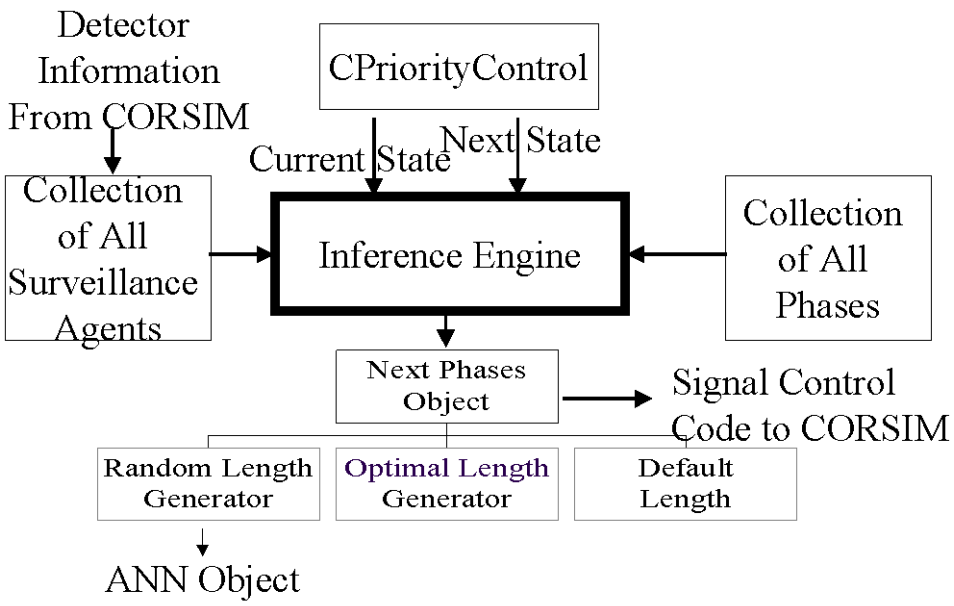

Figure 3-4 Inference Engine Data Flow

3.4.4 Neural Network Delay Prediction

The neural network model predicts the traffic delays over all surveillance links in the next two minutes. A multilayer perceptron neural network is propagated to attain weights so that the delays calculated from the weights are close to the delays from delay model. After the convergence is attained, the weights are stored to the disk file for the optimization step use. The data flow is shown in Figure 3-5, and the variables are described in Section 3.3.1.2. In the optimization step, the weights are loaded to evaluate the objective function. During the optimization process, the weights are assumed constant.

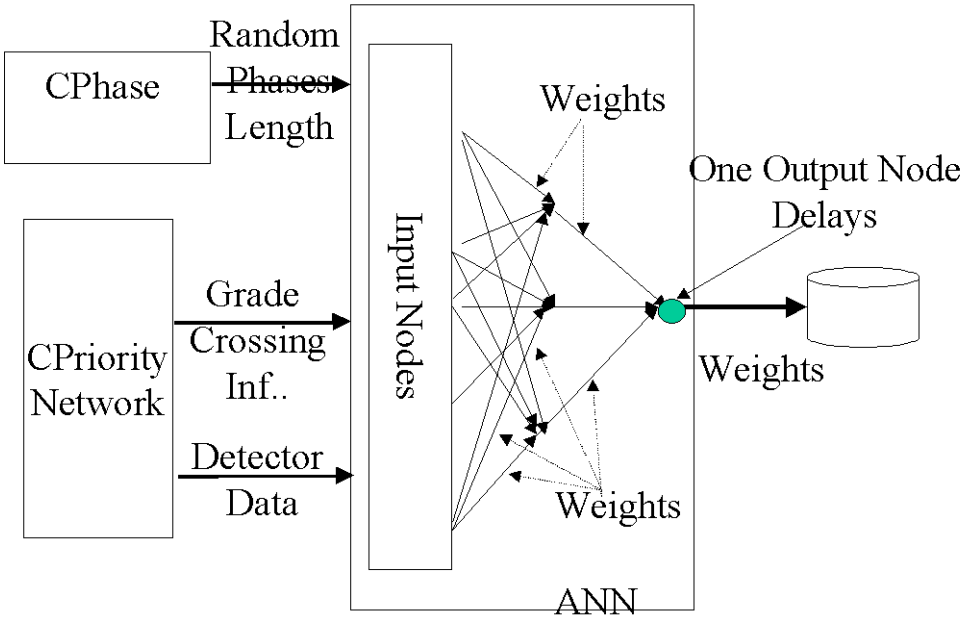

Figure 3-5 Delay Forecast Neural Network Training

3.4.5 Traffic Signal Optimization

The optimizer is quite independent of the rest of the programs. The optimization algorithm can take any form of derivable functions and linear constraints, and then generate an optimized vector X^* such that:

$$f(X^*) \leq f(X) \qquad \forall x \in \Phi \tag{3-4}$$

Where F is the collection of vectors subject to maximum and minimum constraints.

In SOURCAO, the objective function is the delay function $f(X;W,U)$, which is the function of weights (W) generated in Section 3.4.4, current and historical sensor data (U_1) provided by surveillance agents and the historical delays (U_2) generated by the delay model in the *CPriorityNetwork* class. Therefore, the interface could be defined as follows in Figure 3-6. In the figure, the optimizer takes two types of information from SOURCAO: the objective function generated by neural network training and the deviation of the objective function. The optimizer produces the phase lengths after iterations when the objective function converges.

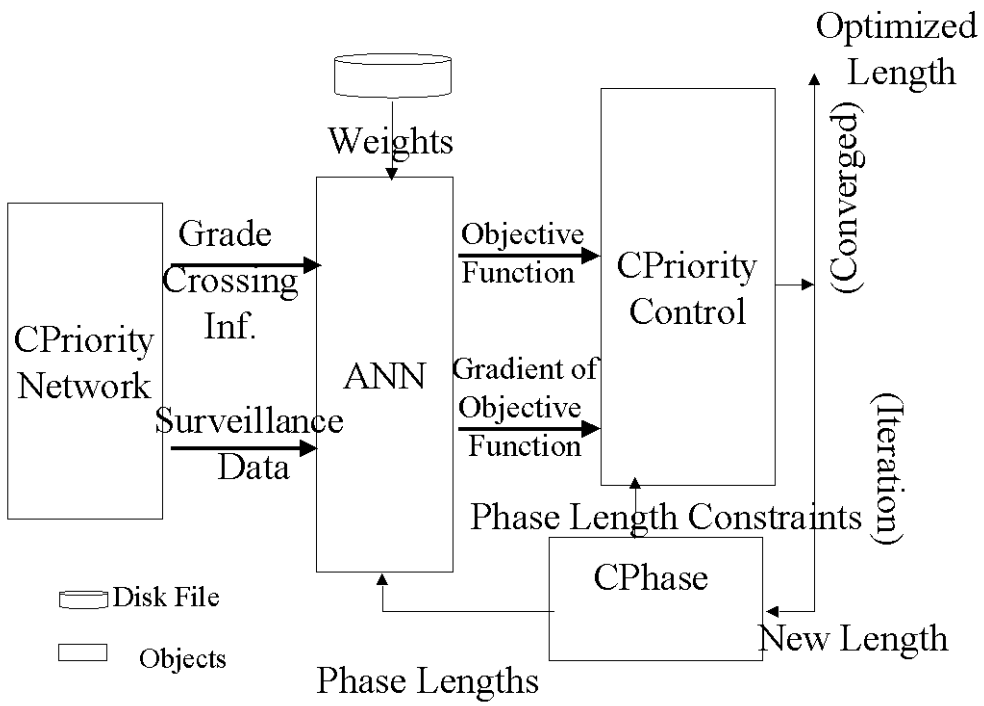

Figure 3-6 The Optimization Interface

3.5 System Function Requirements

Before any software programming starts, the system requirements should be drawn up. The following briefly lists the requirements.

- Independent delay model;
  - Surveillance detector data as only input data;
  - Expandability (to other surveillance measure like video camera etc.);
  - Close to HCM 1997 control delay.
- Neural Network Delay Forecast;
  - Expandable to other ANN architecture (instead of MLP);
  - Weights updateable to the traffic pattern change.

- Embedded Inference Engine;
  - Dynamic phase selection to prevent queue from backing onto the tracks to promote grade crossing safety;
  - Responsive to grade crossing status (e.g. if grade crossing is closed and the queue before gate is full of the link storage).
- Optimized Phase Length ;
  - Minimized network delay function;
  - Incorporate grade crossing information into constraints.
- Software Architecture.
  - Standard ANSI C++[2];
  - Portability, independent of platform [2](Windows/UNIX/VMS);
  - Compatible with the FHWA next generation of simulation;
  - Currently interface with TSIS/CORSIM and evaluated by TSIS/CORSIM.

[2] The only exception is SQP is called from Microsoft FORTRAN 4.0 Library (maths.lib). However, Visual Numerics licensed that library to Microsoft. The library is available in C++ on Windows/UNIX platform from www.vni.com