CHAPTER 5 METHODOLOGIES

5.1 Introduction

In CHAPTER 3 , the objectives are de-composed into promoting grade crossing safety and minimizing highway network traffic delays. How the intelligent agent perceives this traffic network with a grade crossing is described in Section 4.2 of the last chapter. The details of the intelligent design and implementation are discussed in Section 5.2.

In the previous chapter, efforts are made to forecast the traffic network delays through an MLP neural network since the traffic signal optimization takes traffic network delays as the objective functions (Section 4.3 to 4.7). Once the objective function is ready, Section 5.3 presents the details of the optimization process.

5.2 The Intelligent Agent and Grade Crossing Safety

The first objective of SOURCAO is defined as promoting grade crossing safety. An intelligent agent is designed for this purpose. The intelligent agent perceives the traffic environments and grade crossing threats, directs the traffic on the intersections near a grade crossing and appropriately avoids the queue from backing up on the upstream grade crossing. We can view the intelligent agent as a traffic policeman who could oversee several intersections and grade crossings at the same time.

5.2.1 Intelligent Agent Design

In recent years, intelligent agents have become a hot topic in artificial intelligent research. Russel (1994) defined the agent as anything that can be viewed as **perceiving** its environment through **sensors** and as **acting** upon that the environment through **efforts**. Alternatively, an agent can be simply viewed as:

Architecture + Program.

The above definition perfectly fits the solution set for the first object defined in Section 3.2.2. “A traffic control agent **perceives** its environment through **sensors (detectors)** and acts upon that the environment through **efforts (choosing sequences and phase lengths)**.”

In the proposed system, a reflex agent with the internal state presented by Russel (1994) is implemented. Replacing the corresponding components in Figure 5-1 with a specific simulated traffic environment, the traffic control agent in Figure 5-2 can be obtained. Furthermore, Russell (1994) suggested a function of the reflex agent with the state as follows:

function REFLEX-AGENT-WITH-STATE (precept) **returns** action
    static: *state*, a description of the current state
        rules, a set of condition-action rules
    state-UPDATE-STATE (state, precept)
    rule-RULE-MATCH (state, rule)
    action-RULE-ACTION (rule)
    state-UPDATE-STATE (state, action)
    return action

Following the above agent template, how the agent behaves is introduced in Section 5.2.2. The intelligent agent is implemented through Section 5.2.3. How the agent perceives his traffic world through the surveillance detectors is presented in Section 4.2.

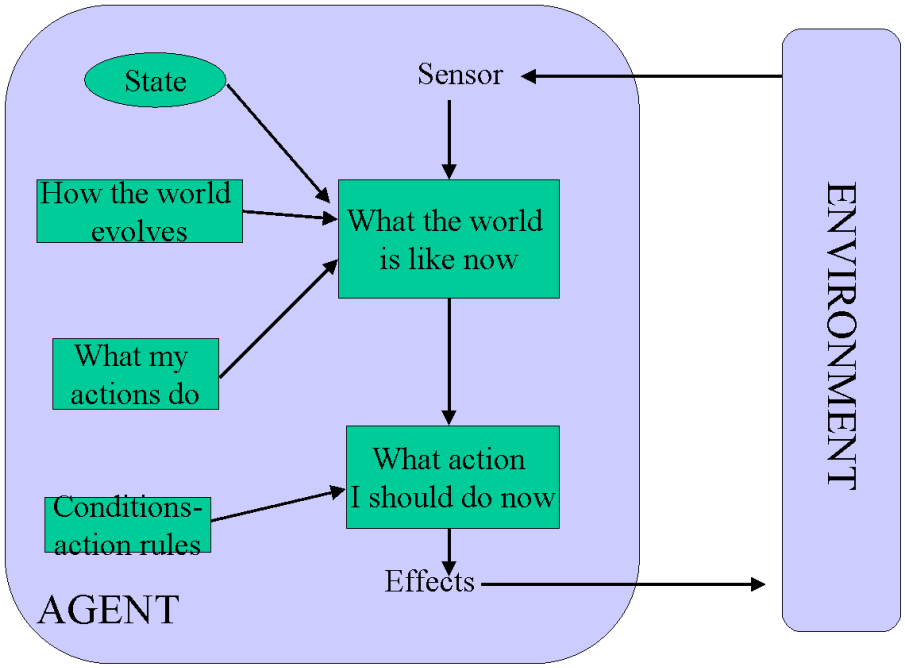


Figure 5-1 A Reflex Agent with Internal State

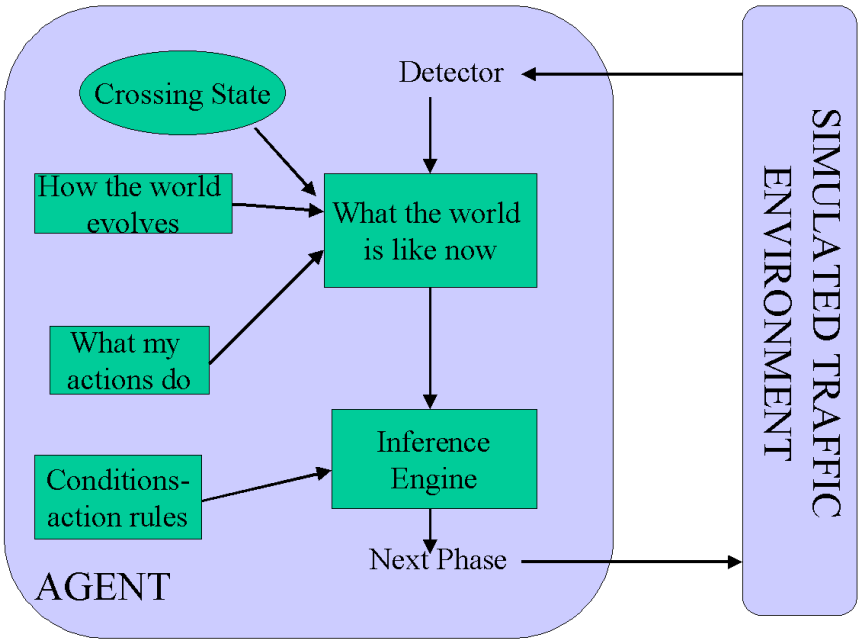


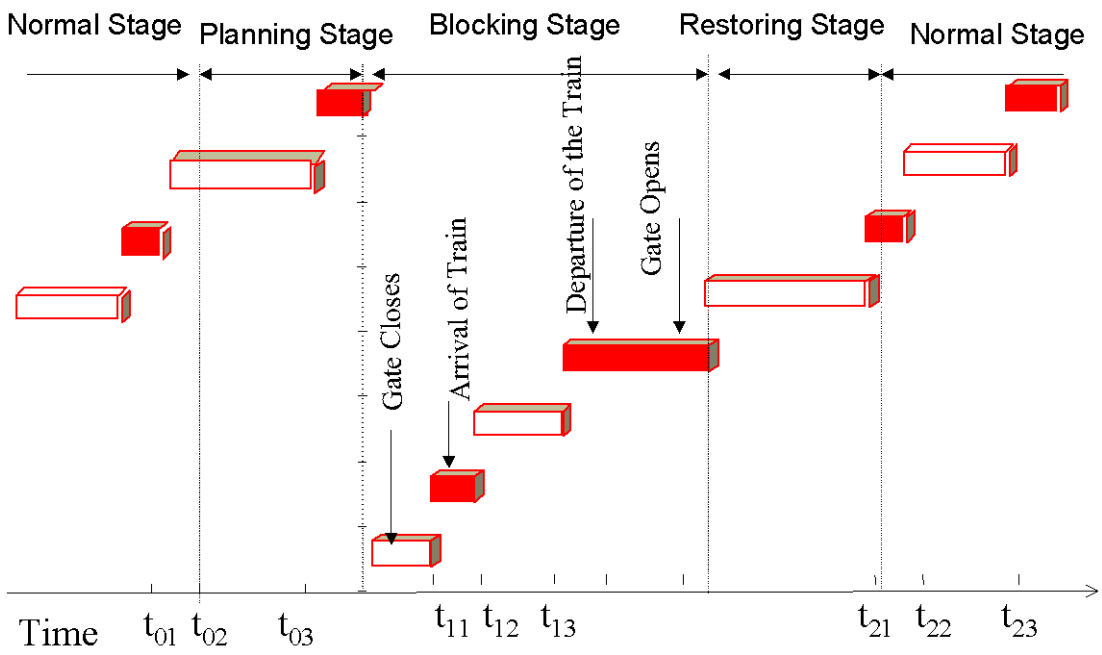
Figure 5-2 A Traffic Control Reflex Agent

5.2.2 Deduction Rules

Throughout this research, it is assumed that highway vehicles always yield to the railroad operations, which are correct in most of the cases. That rule requires that the highway traffic signal controller within a certain distance of grade crossing vicinities cooperate with the railroad operations.

The traffic signal is actuated by the status of the surveillance detectors on the traffic network. The rule of the traffic signal control is to reduce the traffic delay on all approaches. On the other hand, the right-of-way at the highway intersection near a grade crossing yields to the vehicles passing the grade crossing (preemption). Moreover, the safety rule of a grade crossing requires the traffic signals to de-queue the highway vehicles on the rail tracks. The preemption interrupts the smooth operation of the highway traffic.

The safety of grade crossing is given the highest priority. If any vehicle occupies a grade crossing and the train is about to come, an inference engine should assign the highest priority to a phase so that queuing vehicles on the tracks can be dispersed. If this safety rule is not violated, three scenarios are considered beyond the normal stages:



.

Figure 5-3 Traffic Signal Stages and Grade Crossing Closure

- Planning Stage

Before the arrival of the train (current state=0, next state = -1), the traffic signal should be preempted to the traffic, which is blocked by the closure of grade crossing. Furthermore, the time to start this preemption should be determined and optimized by the anticipated traffic delay in the projected future (the second objective).

- Blocking Stage

During grade crossing closure time (current state = -1, next state = 1), if the link toward grade crossing is full, the traffic signal should be changed to prevent the vehicles from entering the links towards a grade crossing. In that way, the right-of-way could yield to the other traffic in order to maximize the capacity and reduce the traffic delay on the approaches not towards grade crossing.

- Restoring Stage

After a grade crossing is re-opened (current state = -1, next state = 0), the traffic control plan should give priority to the approaches blocked by the closure of the grade crossing. Again, the length of this period needs to be optimized according to the status of the traffic network.

5.2.3 Data Structure and Inference

Because SOURCAO is evaluated in CORSIM/TSIS, the CORSIM traffic signal coding convention is adapted. In CORSIM, the traffic control for all lane groups in a link is represented by a signal code defined in Figure 5-4. The combination of all signal control codes on an intersection is defined as a *phase* (amber time and all red time are considered included in a phase in the research). All possible phases are defined as a signal plan (a cycle if in a pre-timed signal control). The process to choose the next phase from a signal plan is called *phase transition*. In the inference engine of SOURCAO, all phases are stored orderly. By checking the status of the associated detectors, the rules in Section 5.2.2 can be implemented to deduce whether the next phase can be ***skipped***. Following the module component rule of the first-order logic (Russell 1994, 269), the data structures of the detectors for a phase are arranged as:

surveillance detectors 11, surveillance detector 12, surveillance detector 13...
surveillance detector 21, surveillance detector 22, surveillance detector 23...
.....
surveillance detector m1, surveillance detector m2, surveillance detector m3...

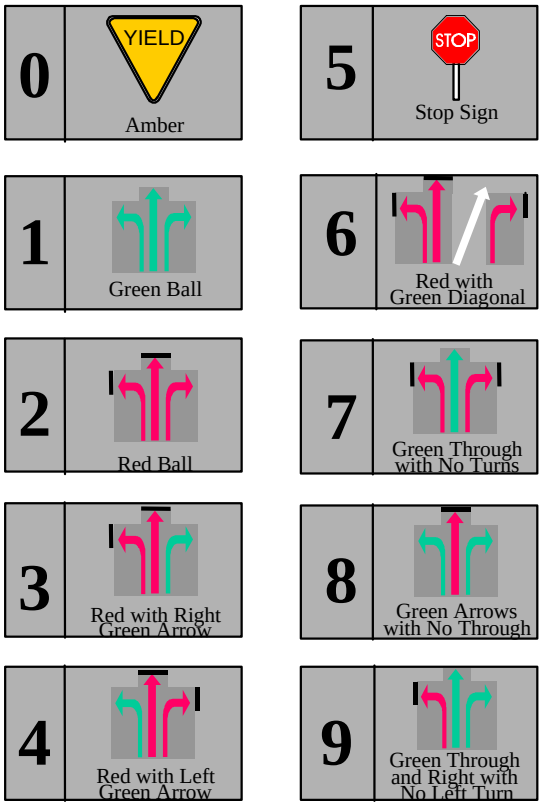


Figure 5-4 A CORSIM Signal Control Code (FHWA, 1999)

Each row of detectors is defined as a *CSurveillanceAgent* class in which the detectors are stored in a linked list; the class provides a method to deduct a true or false conclusion. For each row, the deduction algorithm checks the status of detector 11, status of detector 12 and status of detector 13. If all are true, the program returns true; otherwise, it returns false. Assuming there are m *CSurveillanceAgent* objects associated with a phase plan, if any of the *CSurveillanceAgent* objects returns true, the phase is skipped; otherwise, the phase is stored in a temporary linked list as a potential phase candidate. By checking the history of the phases, one of which is used furthest from now and in the temporary linked list, is chosen for the next phase.

5.3 Phase Length Optimization

5.3.1 Characteristics of Traffic Signal Operations near HRGC

The traffic control operations near an HRGC is similar to the emergency or transit vehicle operations in that the vehicles are granted a certain degree of "priority." However, usually, the difference has to be emphasized on the HRGC.

First of all, usually, a train has absolute "priority" over highway vehicles, and the discussion on this dissertation is limited to this scenario. Next, the characteristics of railroad operations blocks a HRGC much longer than the highway priority vehicle or emergency vehicle does. The block time varies depending on the length of the trains and the passing speed. For example, if the train is traveling at a speed of 50-160 km per hour and the length varies from 100 to 2000m, the train occupies the intersection from 1 to 5 minutes.

The above two sets of characteristics distinguish the traffic operations from the priority signal operations and priority signal control strategies could not directly be applicable here.

According to the characteristics of the railroad traffic, in addition to normal operation state, the traffic signal control is divided into three stages as discussed in Section 5.2.2:

- The planning stage
- The block stage
- The recovering stage

Phase length optimization in SOURCAO is trying to determine the phase length in order to incorporate those stages and minimize the delays in the next two minutes. The signal phase length optimization works somewhat like the concept of “rolling horizon” in OPAC of RT-TRACS (Pooran 1998).

In SOURCAO, however, the concept of traditional cycle length is not applied. Instead, three phase lengths in the future are optimized for a signal at a time although only the first optimized phase is implemented. Another two are optimized again when any of them becomes the next phase.

The second characteristics in SOURCAO is that the optimization only takes place when the optimized phase ends (actually, the green interval ends since a phase does not end until the amber and all red intervals end). The author assumes the traffic condition does not change dramatically during the period in a phase.

Finally, in SOURCAO, only one traffic signal is optimized at a time, although the network delay is adapted as an objective function. The reason for this is to save the optimization execution time in a real time traffic world. In addition, the author believes that the global optimization may be approximated locally since one of the characteristics of a neural network is that it can store some information that can not be modeled in a mathematical formula, like the hidden influence of other traffic signal controllers. However, this claim can only be supported by a theoretical proof or a test.

Another argument is that, however smartly the optimization algorithm could search the global optima solution, the objective function approximates current and future traffic conditions. The unpredictable nature of traffic flow could diminish some optimization efforts. This explains the importance of the independent evaluation of the traffic optimization algorithm as well.

5.3.2 Optimization Algorithm

The objective function of the traffic signal optimization (5-1) is defined as the network delay in Section 4.4 (same as d). The function is subject to the boundary equality constraint condition (5-2) and the inequality condition (5-3).

$$f = f(X;U,W) \tag{5-1}$$

$$x_{\min} \le x_{ij} \le x_{\max} \tag{5-2}$$

$$x_1 + x_2 + x_3 \le 120 \tag{5-3}$$

Where:

i is the number of the traffic signals;

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

x_i is the number j future phase length;

120 is the projected period in seconds for the delay forecast.

After the weights W in the objective function are trained and the state variables U are sensed through the surveillance detectors, the objective function (5-1) only depends on the control variable X then.

The above optimization problems can be classified as a non-linear programming problem with linear constraints. Usually, there are two classes of approaches. One of the approaches is to take the constraints as the penalty or the barrier functions. Those functions are added to the original objective function to form an augmented objective function. Thus, the constrained non-linear programming becomes non-constrained.

On the other hand, the second class of algorithms is named “the method of the feasible directions.” In the second algorithm, the solution set is initiated from a feasible point to an improved feasible point so that the constraints could be maintained while the objective function keeps decreasing.

Successive Quadratic Programming or the Projected Lagrangian approach is one popular projection method. Section 5.3.3 discusses the mathematical foundation for that algorithm. Section 5.3.4 presents the implementation of the algorithm and interfaces with the proposed system.

5.3.3 Successive Quadratic Programming

The discussion here is limited to the non-linear objective function f (x) subject to the linear inequality function g(x) and the linear equality functions h(x), as expressed below:

Minimize f (x)

Subject to:

$$g(x) \leq 0$$
$$h(x) = 0 \tag{5-4}$$

By applying the Karush-Kuhn-Tucker (KKT) optimality condition (Bazaraa and Sherali 1993, 163) in the vector format, we can have:

$$\nabla f(x) + \nabla g(x)^T \bullet u + \nabla h(x)^T \bullet v = 0$$
$$u^T g(x) = 0$$
$$u \geq 0 \tag{5-5}$$

Instead of finding the KKT point directly from (5-5), we could start from an initial point (x_k, u_k, v_k) to iteratively minimize the problem in (5-6). If the question below is solved, the KKT point is found.

$$\text{Minimize: } f(x) + u^T g(x) + v^T h(x) \tag{5-6}$$

The motivation behind Successive Quadratic Programming is to approximate (5-6) by successive quadratic functions. Quadratic Programming is well studied and has a solid solution. For example, Bazaraa and Sherali (1993, 503-509) showed that a quadratic programming problem subject to the linear constraints could be reduced to a linear complementary problem, which converges within the finite number of the iterations.

The conversion of Quadratic Programming problem (5-6) is shown below. By applying Newton's method (Bazaraa and Sherali 1993, pp. 308), we have:

$$\begin{aligned}(x,u,v)^T &= (x_k,u_k,v_k)^T - \left(\nabla^2 L(x_k,u_k,v_k)\right)^{-1} \bullet \\ &\left(\nabla f(x_k) + u^T \bullet \nabla g(x_k) + v^T \bullet \nabla h(x_k), g(x_k), h(x_k)\right)^T\end{aligned} \tag{5-7}$$

Or,

$$\begin{aligned}&(\nabla^2 L(x_k,u_k,v_k)) \bullet \left((x,u,v) - (x_k,u_k,v_k)\right)^T \\ &= -(\nabla f(x_k) + u^T \bullet \nabla g(x_k) + v^T \bullet \nabla h(x_k), g(x_k), h(x_k))^T\end{aligned} \tag{5-8}$$

Where

$$d = x - x_k \tag{5-9}$$

$$\nabla^2 L(x_k, u_k, v_k) = \begin{bmatrix} \nabla^2 f(x_k) & \nabla g^T(x_k) & \nabla h^T(x_k) \\ \nabla g^T(x_k) & 0 & 0 \\ \nabla h^T(x_k) & 0 & 0 \end{bmatrix}$$

(5-10)

$$\begin{bmatrix} \nabla^2 f(x_k) & \nabla g^T(x_k) & \nabla h^T(x_k) \\ \nabla g^T(x_k) & 0 & 0 \\ \nabla h^T(x_k) & 0 & 0 \end{bmatrix} \bullet (d, u - u_k, v - v_k)^T$$
$$= \begin{pmatrix} \nabla^2 f(x_k) \bullet d + \nabla g_I(x_k)^T \bullet (u - u_k) + \nabla h(x_k)^T \bullet (v - v_k) \\ \nabla g_I(x_k)^T \bullet (u - u_k) \\ \nabla h(x_k)^T \bullet (v - v_k) \end{pmatrix}$$

(5-11)

Applying (5-9), (5-10) and (5-11) into (5-8), we can have (5-12), (5-13) and (5-14):

$$\frac{1}{2}\nabla^2 f(x_k) \bullet d + \nabla g_I(x_k)^T \bullet u_I + \nabla h(x_k)^T \bullet v = -\nabla f(x_k)$$

(5-12)

$$\nabla g(x_k) \bullet d = -g(x_k)$$

(5-13)

$$\nabla h(x_k) \bullet d = -h(x_k)$$

(5-14)

Moreover, the above Hessian $\tilde{N}^2 f(x)$ can be approached by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) updating scheme (Bazaraa and Sherali 1993, 325) without actually calculating the $\tilde{N}^2 f(x)$.

5.3.4 Implementation

The following five steps could be implemented to solve (5-4).

Step 1. Check the given initial $\boldsymbol{x}$. If all constraint conditions are satisfied, go to step 2; if not, adjust $\boldsymbol{x}$ to satisfy the equality constraints. Go to step 2.

Step 2. Solve the quadratic programming defined in (5-13). After this step, the projection direction d is obtained.

Step 3. Linearly search $x_{k+1}=x_k+\boldsymbol{a}\cdot d$, go to Step 4 afterwards.
Where

$$f(x^k+\boldsymbol{a}^k d^k)\le f(x^k)+0.1\boldsymbol{a}^k(d^k)^T\nabla f(x^k) \tag{5-15}$$

$$(d^k)^T\nabla f(x^k+\boldsymbol{a}^k d^k)\ge 0.7\boldsymbol{a}^k(d^k)^T\nabla f(x^k) \tag{5-16}$$

Step 4. The Hessian matrix is updated by BFGC scheme if (5-17) holds. Go to Step 5 then.

$$(d^k)^T\nabla f(x^k+\boldsymbol{a}^k d^k)-\nabla f(x^k)>0 \tag{5-17}$$

Step 5. Check convergence. If (5-18) is satisfied ends the iteration; otherwise, go to Step 2.

$$\left\|\nabla f(x^k)-A^k l^k\right\|\le t \tag{5-18}$$

Where

A=($g_I(x_k)$, $h(\boldsymbol{x_k})$), λ is its Lagrangian multiplier and τ is the user defined accuracy.

Fortunately, there is a built-in function in Microsoft Developer Studio FORTRAN 4.0 or DIGITAL FORTRAN 5.0 (or later) to implement the SQP algorithm described in this section.

The advantage of calling a mathematical library is time saving in programming and reliability gaining in running. However, since the algorithm is not specially formulated to solve optimization in SOURCAO, the efficiency is compromised. Another disadvantage of using this library function is that sometimes the convergence could not be well controlled within a given time. The optimization iteration will not end until (5-18) is satisfied. There is no way to dig into Microsoft library and jump out when the optimization program running time exceeds in a predefined time (e.g. 3 seconds in this case). For a real time signal controller, we may sacrifice accuracy to satisfy the time constraints. Obviously, we could not manipulate the Microsoft library to do so. Section 6.5.3 examines how serious the time constraints are.