APPLICATION OF STOCHASTIC LEARNING AUTOMATA FOR MODELING DEPARTURE TIME AND ROUTE CHOICE BEHAVIOR

BY

Kaan Ozbay, Assistant Professor
Department of Civil & Environmental Engineering
Rutgers, The State University of New Jersey
Piscataway, NJ 08854-8014
Phone: (732) 445-2792
Fax: (732) 445-0577
Email: kaan@rci.rutgers.edu

Aleek Datta, Research Assistant
Department of Civil & Environmental Engineering
Rutgers, The State University of New Jersey
Piscataway, NJ 08854-8014
Phone: (732) 445-2792
Fax: (732) 445-0577
Email: adatta@eden.rutgers.edu

Pushkin Kachroo, Assistant Professor
Bradley Department of Electrical & Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Phone: (540) 231-2976
Fax: (540) 231-2999
Email: pushkin@vt.edu

Submitted for Possible Publication in the
Journal of Transportation Research Record
ABSTRACT

This paper uses Stochastic Learning Automata (SLA) theory to model the learning behavior of commuters within the context of the combined departure time route choice (CDTRC) problem. The SLA model uses a reinforcement scheme to model the learning behavior of drivers. In this paper, a multi-action Linear Reward-Penalty reinforcement scheme is introduced to model the learning behavior of travelers based on past departure time choice and route choice. In order to test the model, a traffic simulation is developed. The results of the simulation are intended to show that the drivers learn the best CDTRC option, and that the network achieves user equilibrium in the long run. Results evidence that the developed SLA model accurately portrays the learning behavior of drivers while the network satisfies user equilibrium conditions.

Key Words: Departure Time Decision, Route Choice Behavior, Traffic Simulation
1 INTRODUCTION & MOTIVATION

Advanced Traveler Information Systems (ATIS) are becoming one of the most common traffic management tools that are used to improve the performance of both the transportation system itself, and individual travelers using this system. ATIS systems provide real-time information regarding incidents, route travel times, and traffic congestion to assist travelers in selecting their best departure time and route choices. Direct benefits of ATIS include reduced travel time and cost, improved safety, whereas indirect benefits are improved driving conditions, which help in reducing day-to-day stress and anxiety and more importantly increased travel time reliability. The exploration and development of dynamic trip choice models, which incorporate both route and departure time choices, has been spurred by the development of such ATIS systems. The success of these ATIS systems mainly depends on the availability of reliable models that reflect traveler response to route guidance or other traveler information. Moreover, these models of combined route and departure time choice should be able to capture the “day-to-day learning behavior” of drivers.

The importance of the day-to-day learning behavior of drivers becomes more accentuated when information regarding traffic conditions is given to users through ATIS because travelers tend to make their decision based on their previous experiences. Thus, in response to these developments and needs in the area of ATIS, Ozbay et al. (1) proposed to use stochastic learning automata approach (SLA) to model the day-to-day learning behavior of drivers in the context of “route choice behavior”. Stochastic learning automata mimics the day-to-day learning of drivers by updating the route choice probabilities based on the basis of information received and the experience of drivers. The appropriate selection of the proper learning algorithm as well as the parameters it contains is crucial for its success in modeling route choice behavior. In simple terms, the stochastic learning automata approach is an inductive inference mechanism that updates the probabilities of its actions occurring in a stochastic environment in order to improve a certain performance index, i.e. travel time of users. Simulation results using a single-origin-single-destination and multiple route transportation network show that the SLA model accurately depicted driver learning, as well as helping achieve travel time equilibrium on the network.

In the context of ATIS, the issue of travel time reliability has recently become a major issue. Several researchers have recently shown that the “reliability” of travel time is in fact of major importance to drivers (2). It is now well known that the majority of travelers would like to be able to plan ahead their daily trips with an acceptable level of certainty. The frequent occurrence of accidents or other events causing congestion reduces this level of trip time predictability and creates an additional level of negative perception about the level of service on a specific route. Especially in a society where a large number of people are driven by tight schedules, this kind of unpredictable non-recurrent congestion becomes
unacceptable. A good example is the late arrival policy most day care centers employ. The majority of day care centers in New Jersey charge $15 per every 15-minute delay after, say, 6:30 pm. Thus, the working parents would like to be able to plan quite accurately the time they leave their work to pick up their children. In such a situation, the reliability of trip time plays an important role with respect to the way a person plans his/her trip. More important than the trip travel time is the reliability of trip time in terms of “the deviation of trip time” from the expected trip time on a daily basis. In fact, drivers are observed to be more concerned about the reliability of their trip travel times than the overall length of their trip. Thus, it has been observed that travelers are willing to adjust their routes as well as their departure times to improve the reliability of their trip times. Wunderlich et al. (2) show that travelers tend to learn best route and departure time based on their day-to-day experiences. Thus, their report clearly shows that a successful ATIS system should be able to model the learning behavior of travelers in terms of their route and departure time choice. In this paper, we propose to extend the SLA route choice model previously introduced by Ozbay et al. (1) to include departure time choice by introducing the “multi-action stochastic learning automaton”.

2 LITERATURE REVIEW
Numerous researchers have addressed the departure time choice problem. De Palma et al. (3) and Ben-Akiva et al. (4) modeled departure time choice over a network with one bottleneck using the general continuous logit model. Mahmassani and Herman (5) used a traffic flow model to derive the equilibrium joint departure time and route choice over a parallel route network. Mahmassani and Chang (6) further developed the concept of equilibrium departure time choice and presented the boundedly-rational user equilibrium concept. Noland (7) developed a simulation methodology using a model of travel time uncertainty to determine optimal home departure times. Finally, Ran et al. (8) developed a model to solve the for the dynamic user optimal departure time and route choice using a link-based variational inequality formulation.

Several dynamic tradeoffs are of interest when analyzing departure time decisions. To illustrate, suppose that individuals have a particular work start time. As discussed in Hendrickson et al. (9), several influences may cause a traveler to plan to arrive earlier or later than this start time:

1. **Congestion avoidance**: by avoiding peak hour traffic, travel time might be considerably reduced.
2. **Schedule delay**: early or late arrivals at work may be dictated by the schedule of shared ride vehicles or carpools.
3. **Peak/Off-peak tolls** and **parking availability**: Charges for parking or roadway fares, due to congestion pricing, may vary by time of day, thereby inducing changes in planned arrival or
departure times. Furthermore, parking availability may be restricted for late arrivals as parking lots fill up.

In each of these cases, time dependent variations in travel characteristics may affect departure time choice and route choice.

Recently, researchers have also conducted several computer-based experiments to produce route choice models. Iida, Akiyama, and Uchida (10) developed a route choice model based on actual and predicted travel times of drivers. Another route choice model developed by Nakayama and Kitamura (11) assumes that drivers “reason and learn inductively based on cognitive psychology”. The model system is a compilation of “if-then” statements in which the rules governing the statements are systematically updated using algorithms. In essence, the model system represents route choice by a set of rules similar to a production system. Mahmassani and Ghang (12) developed a framework to describe the processes governing commuter’s daily departure time decisions in response to experienced travel times and congestion. It was determined that commuter behavior can be viewed as a “boundedly-rational” search for an accepted travel time. Results indicated that a time frame of tolerable schedule delay existed, termed an “indifference band”. Jha et al (13) developed a Bayesian updating model to simulate how travelers update their perceived day-to-day travel time based on information provided by ATIS systems and their previous experience.

3 DESCRIPTION OF LEARNING AUTOMATA

The first learning automata models were developed in mathematical psychology by Bush and Mosteller (14), and Atkinson et al. (15), Tsetlin (16), and Fu (17). Recent applications of learning automata to real life problems include control of absorption columns (17) and bioreactors (18). The learning paradigm governing the automaton discussed in works such as these is straightforward; the automaton can perform a finite number of actions in a random environment, and then adjusts its actions based upon the response of the environment. The aim is to design an automaton that can determine the best action guided by past actions and corresponding responses. The relation between the automaton and the environment can be seen in Figure 1a.

**Insert Figure 1a**

When a specific action is performed at time, $\alpha(n)$, the environment responds by producing an environment output $\beta(n)$, which is stochastically related to the action. In our application, the
environment’s response is an element of the set \( \mathbb{R} = [0,1] \). The output value of 1 corresponds to an “unfavorable” (failure, penalty) response, while output of 0 means the action is “favorable.”

The automaton’s environment is defined by a set \( \{\alpha, c, \mathbb{P}\} \) where \( \alpha \) is the action set, \( \mathbb{P} \) represents a (binary) output set, and \( c \) is a set of penalty probabilities (or probabilities of receiving a penalty from the environment for an action) where each element \( c_i \) corresponds to one action \( \alpha_i \) of the action set \( \alpha \). If the probability of receiving a penalty for a given action is constant, the environment is called a stationary environment; otherwise, it is non-stationary. For this implementation, the environment is considered non-stationary since, in reality, the physical environment changes as a result of actions taken. Finally, a learning algorithm is necessary in order to compensate for environmental changes over time. The aim of the algorithm is to choose actions that minimize the expected penalty.

The main concept behind the learning automaton model is the concept of a probability vector defined for a P-model environment as

\[
p(n) = \{ p_i(n) \in [0,1] | Pr[\alpha(n) = \alpha_i] \}
\]  

where \( \alpha_i \) is one of the possible actions. Models in which the output can be only one of two variables, 0 and 1, for example, are referred to as P-models. In our case, a response of 0 can be considered favorable and 1 for unfavorable. We consider a stochastic system in which the action probabilities are updated at every stage \( n \) using a reinforcement scheme. The updating of the probability vector with this reinforcement scheme provides the learning behavior of the automata. If the automaton is “learning” in the process, its performance must be superior to an automaton for which the action probabilities are equal.

**Advantages of SLA vs. Classical Route / Departure Time Choice Models Based on Utility Theory**

Classical approaches for modeling the day-to-day dynamics of the route / departure time choice methods problem in the literature, mostly in the context of dynamic or static traffic assignment problems such as in Depalma *et al.* (3) and Ben-Akiva *et al.* (4), requires a fair amount of knowledge of the system to be controlled. Moreover, the majority of them do not explicitly deal with the learning problem. The mathematical models used in these approaches are often assumed to be exact, and the inputs are deterministic functions of time. On the other hand, the modern control theory approach proposed in this paper, “Stochastic Learning Automata”, explicitly consider the uncertainties present in the system, but the stochastic control methods assume that the characteristics of the uncertainties are known. However, all assumptions concerning uncertainties and/or input functions may not be valid or accurate. It is therefore necessary to obtain further knowledge of the system by observing it during operation, since *a priori* assumptions may not be sufficient.
It is possible to view the problem of route choice as a problem in learning. Learning is defined as a change in behavior as a result of past experience. A learning system should therefore have the ability to improve its behavior with time. In a purely mathematical context, the goal of a learning system is the optimization of a “functional not known explicitly” (1).

The stochastic automaton attempts a solution of the problem without any \textit{a priori} information on the optimal action. One action is selected at random, the response from the environment is observed, action probabilities are updated based on that response, and the procedure is repeated. A stochastic automata acting as described to improve its performance is called a learning automaton (LA). \textbf{This approach does not require the explicit development of a utility function since the behavior of drivers is implicitly embedded in the parameters of the learning algorithm itself.}

4 SLA COMBINED DEPARTURE TIME ROUTE CHOICE MODEL

Learning and forecasting processes for route and departure time choice have been modeled through the use of statistical models such as those proposed by Cascetta and Canteralla (20) or Davis and Nihan (21). These applications model the learning and forecasting process using one general approaches briefly described below (22):

- Deterministic or stochastic threshold models based on the difference between the forecasted and actual cost of the alternative chosen the previous day for switching choice probabilities (21).
- Extra utility models for conditional path choice models where the path chosen the previous day is given an extra utility in order to reflect the transition cost to a different alternative (20).
- Stochastic models that update the probability of choosing a route based on previous experiences according to a specific rule, such as Bayes’ Rule.

Stochastic learning is also the learning mechanism adopted in this paper. However, the SLA learning rule is a general one and different from Bayes’ rule as follows. The SLA model consists of two components, the choice set and the learning mechanism. In the case of a combined departure time and route choice (CDTRC) problem, travelers will choose their route and departure time simultaneously. For the CDTRC problem, let’s assume that there exists an action set \( \alpha \) that is comprised of possible actions that each user can perform. In other words, \( \alpha \) is the set of combined departure time/route choice (CDTRC) decision combinations Thus \( \alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \) where \( \alpha_n \) is the specific action performed by an individual driver. We define \( \alpha_n \) as the combination of CDTRC options that is possible
given the number of routes and departure time periods. Thus, if there are “r” routes and “k” departure
time periods, there will be \( r \times k \) possible CDTRC combinations. For example, if there are 2 routes,
denoted as route 1 and route 2, and 4 departure time periods denoted as 1,2,3, and 4, the set of
route/departure time choice (CDTRC) decision combinations will be:

\[
a = \{a_1 = a_{11}, a_2 = a_{12}, a_3 = a_{13}, a_4 = a_{14}, a_5 = a_{21}, a_6 = a_{22}, a_7 = a_{23}, a_8 = a_{24}\}
\]

where \( a_{rk} \) represents the action for taking route “r” during departure time period “k”. The environment,
the traffic system in this case, responds by producing a response \( \beta \) that is stochastically related to the
action. Thus, the set of responses can be defined as \( \beta = \{\beta_1, \beta_2, \ldots, \beta_m\} \). In its simplest case, the
response may be favorable or unfavorable (0 for reward and 1 for penalty). The action probability,
\( p(t+1) \) for each user at time “t+1” is then updated on the basis of its value at time “t” and the response
at time ‘t’, which can be represented as \( \beta(t) \). The action probability is updated using one of the
reinforcement schemes discussed in the next section of this paper and in Narendra (23).

This system, described above can be made into a feedback system where the effect of user choice
on the traffic and vice versa is modeled, using a “variable structure stochastic automata”, which is
defined by a triplet \( \{\alpha, c, \beta\} \) where \( \alpha \) is the action set, \( \beta \) represents a (binary) output set, and \( c \) is a set of
penalty probabilities (or probabilities of receiving a penalty from the environment for an action) where
each element \( c_i \) corresponds to one action \( \alpha_i \) of the action set \( \alpha \).

****INSERT FIGURE 1b****

The response of the environment is considered to be a random variable. If the probability of
receiving a penalty for a given action is constant, the environment is called a stationary environment;
otherwise, it is non-stationary. The need for learning and adaptation in systems is mainly due to the fact
that the environment changes with time. Performance improvement can only be a result of a learning
scheme that has sufficient flexibility to track the better actions. The aim in these cases is not to evolve to
a single action that is optimal, but to choose actions that minimize the expected penalty. For our
application, the (automata) environment is non-stationary since the physical environment changes as a
result of actions taken.

The main goal of this paper is to show the applicability of SLA to the combined route / departure
choice problem and the study of its properties in terms of network equilibrium, learning rates, and
convergence when used in the context of a purely stochastic traffic simulation environment. The simple
case study presented here allows us to better understand all of these properties without being limited by the shear size of a general network.

However, the application of the theory presented here to a “multiple-origin-multiple-destination” network is quite straightforward from an algorithmic point of view. Basically, the algorithm that will be presented can be directly applied to route-departure time choice sets generated for each origin destination pairs that exist in the study network. The only addition will be to determine the set of feasible route-departure time choices for each OD pair and this would require a path generation algorithm. A mathematical treatment of this extension to a multiple-origin-multiple destination network and simulation results are presented in Ozbay et al. (2001).

4.1 Linear Reward–Penalty ($L_{R-P}$) Scheme

This learning scheme, proposed by Ozbay et al. (1) for modeling learning based route choice processes, was first used in mathematical psychology. The idea behind a reinforcement scheme, such as linear reward-penalty ($L_{R-P}$) scheme, is a simple one. If the automaton picks an action $\alpha_i$ at instant $n$, and a favorable input ($\alpha(n) = 0$) results, the action probability $p_i(n)$ is increased and all other components of $p(n)$ are decreased. For an unfavorable input $\beta(n) = 1$, $p_i(n)$ is decreased and all other components of $p(n)$ are increased.

In order to apply this idea to our situation, first assume that there are only $r \times k$ distinct CDTRC options to choose between an origin-destination pair for a time period. A general scheme for updating action probabilities can be represented as follows:

If

$$\alpha(n) = \alpha_i \quad (i = 1, \ldots, r)$$

$$p_j(n + 1) = p_j(n) - g_j[p(n)] \quad \text{when} \quad \beta(n) = 0$$

$$p_j(n + 1) = p_j(n) - h_j[p(n)] \quad \text{when} \quad \beta(n) = 1$$

where

- $j \neq i$
- $j = (1, \ldots, r)$
- $r$ = total number of options
- $g_j[p(n)]$ and $h_j[p(n)]$ are continuous, non-negative functions used to update $p(n)$

To preserve the probability measure we have

$$\sum_{j=1}^{r} p_j(n) = 1$$

so that
\begin{align*}
p_i(n+1) &= p_i(n) + \sum_{j=1}^{r} g_j(p(n)) \quad \text{when } \beta(n) = 0 \\
p_i(n+1) &= p_i(n) - \sum_{j=1}^{r} h_j(p(n)) \quad \text{when } \beta(n) = 1
\end{align*}

The updating scheme is given at every instant separately for that action which is attempted at stage \( n \) in equation (3) and separately for actions that are not attempted in equation (2). Reasons behind this specific updating scheme are explained in Narendra and Thathnachar (23). In the above equations, the action probability at stage \( (n+1) \) is updated on the basis of its previous value, the action \( \alpha(n) \) at the instant \( n \) and the input \( \beta(n) \). In this scheme, \( p(n+1) \) is a linear function of \( p(n) \), and thus, the reinforcement (learning) scheme is said to be linear. For example, if we assume a network with one origin destination pair, two routes, and a single departure time period between this O-D pair, we can consider a learning automaton with two actions in the following form:

\begin{align*}
g_j(p(n)) &= ap_j(n) \quad \text{(4)} \\
\text{and} \\
h_j(p(n)) &= b(1 - p_j(n)) \\
\text{for } j = (1, ..., r)
\end{align*}

In equation (3) \( a \) and \( b \) are reward and penalty parameters and \( 0 < a < 1, \ 0 < b < 1 \). If we substitute (4) in equations (2) and (3), the updating (learning) algorithm for this simple two route and single departure time system can be re-written as follows:

\begin{align*}
p_1(n+1) &= p_1(n) + a(1 - p_1(n)) \\
p_2(n+1) &= (1 - a)p_2(n) \quad \alpha(n) = \alpha_1, \beta(n) = 0 \\
p_1(n+1) &= (1 - b)p_1(n) \\
p_2(n+1) &= p_2(n) + b(1 - p_2(n)) \quad \alpha(n) = \alpha_1, \beta(n) = 1
\end{align*}

Equation (5) is generally referred to as the general \( L_{R-EP} \) updating algorithm. From these equations, it follows that if action \( \alpha_i \) is attempted at stage \( n \), the probability \( p_j(n) \) is decreased at stage \( n+1 \) by an amount proportional to its value at stage \( n \) for a favorable response and increased by an amount proportional to \( 1 - p_j(n) \) for an unfavorable response.

If we think in terms of only route choice decisions in a network with one O-D pair and two links connecting them and a single departure time period, then, if at day \( n+1 \), the travel time on route 1 is less than the travel time on route 2, we consider this as a favorable response and the algorithm increases the
probability of choosing route 1 and decreases the probability of choosing route 2. If the travel time on route 1 at day \( n+1 \) is higher than the travel time on route 2 the same day, then the algorithm decreases the probability of choosing route 1 and increases the probability of choosing route 2. **However, in order to consider both route choice and departure time choice decisions, we must modify the updating algorithm to incorporate “multiple actions”**. This extension to multiple actions represents a more general model that is capable of modeling the combined route choice/departure time choice decision. The scheme for multi-action learning automata can be obtained by updating equation (4) to the following:

\[
g_j(p(n)) = ap_j(n) \tag{6}
\]
and
\[
h_j(p(n)) = \frac{b}{r-1} - bp_j(n)
\]

The primary difference between (4) and (6) is the means by which the penalty is applied. Instead of applying the entire penalty to the incorrect choice, only a proportion of the penalty, based on the number of options \( r \times k \), is applied. This modification ensures that the correct choice is still fully rewarded, but equally penalizes the incorrect choice. Furthermore, this modification is necessary to ensure that \( \sum p_i(n) = 1 \). By substituting (6) into (5), the algorithm for the multi-action learning automaton is obtained:

\[
\begin{align*}
p_i(n+1) &= p_i(n) + a(1 - p_j(n)) \quad \alpha(n) = \alpha_i, \beta(n) = 0 \\
p_j(n+1) &= (1 - a)p_j(n), j \neq i \quad \alpha(n) = \alpha_j, \beta(n) = 1 \\
p_i(n+1) &= (1 - b)p_i(n) \\
p_j(n+1) &= \frac{b}{r-1} + (1 - b)p_j(n), j \neq i
\end{align*}
\tag{7}
\]

In order to apply the SLA-CDTRC model, the reward and penalty parameters must be calibrated. A travel simulator was developed using Java to calibrate this model. The simulator is accessible via the Internet, and is the basis for data collection. Through the travel simulator, experiments were conducted to determine the learning rate for users based upon a sample scenario, two route choices, and departure time choices. Based on the results of the experiments and the data collected, the SLA model was calibrated. A detailed explanation of the calibration methodology can be found in Ozbay et al. (1).

### 4.2 Convergence Properties of SLA-CDTRC Model

A natural question is whether the updating is performed in a way, which is compatible with intuitive concepts of learning or in other words if it is converging to a final solution as a result of the modeled
learning process. The following discussion based on Narendra (24) regarding the expediency and optimality of the learning provides some insights regarding these convergence issues. The basic operation performed by the learning automaton described in equation (3) is the updating of the choice probabilities on the basis of the responses of the environment. One quantity useful in understanding the behavior of the learning automaton is the average penalty received by the automaton. At a certain stage \( n \), if action \( \alpha_i \), i.e. the combined departure time route choice \( i \), is selected with probability \( p_i(n) \), the average penalty (reward) conditioned on \( p(n) \), the action probability, is given as:

\[
M(n) = E[\beta(n) | p(n)] = Pr[\beta(n) = \| | p(n)] \\
= \sum_{i=1}^{r} Pr[\beta(n) = \| | \alpha(n) = \alpha_i] Pr[\alpha(n) = \alpha_i] \\
= \sum_{i=1}^{r} p_i(n)c_i
\]

If we define a set as the set of penalty probabilities of the environment, we can calculate an average penalty. Here, if no \textit{a priori} information is available, and the actions are chosen with equal probability for a random set, the value of the average penalty, \( M_0 \), is calculated by:

\[
M_0 = \frac{c_1 + c_2 + \ldots + c_r}{r}
\]

\( M_0 \) is also called pure chance automaton. The term of the learning automaton is justified if the average penalty is made less than \( M_0 \) at least asymptotically as \( n \to \infty \). This asymptotic behavior, which is called expediency, is defined in Definition 1.

\textbf{Definition 1} (24): A learning automaton is called expedient if

\[
\lim_{n \to \infty} E[M(n)] < M_0
\]

Expediency represents a desirable behavior of the automaton. However, it is more important to determine if an updating procedure results in \( E[M(n)] \) attaining its minimum value. Thus, if the average penalty is minimized by the proper selection of actions, the learning automaton is called optimal and the optimality condition is given in Definition 2.

\textbf{Definition 2} (24): A learning automaton is called \textit{optimal} if
\[ \lim_{n \to \infty} E[M(n)] = c_i \]  

where 
\[ c_i = \min_j \{c_j\} \]  

By analyzing the data obtained on user combined departure time route choice behavior, we can ascertain if the convergence properties of the behavior in terms of the learning automata. Since, the choice converges to the correct one in the experiments we conducted (1), it is clear that the 
\[ \lim_{n \to \infty} E[M(n)] = \min\{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\} \]
Hence, the learning schemes that are implied by the data for the human subjects are optimal. Since, they are optimal, they also satisfy the condition that 
\[ \lim_{n \to \infty} E[M(n)] < M_0 \]
Hence, the behavior is also expedient.

5 SLA MODEL TESTING USING TRAFFIC SIMULATION
The stochastic learning automata model developed in the previous section was applied to a transportation network to determine if network equilibrium is reached. The logic of the traffic simulation is shown in the flowchart in Figure 2. The simulation consists of one O-D pair connected by two routes with different traffic characteristics. All travelers are assumed to travel in one direction, from the origin to the destination. Travelers are grouped in “packets” of 5 vehicles. The simulation period is 1 hour, assumed to be in the morning peak, and each simulation consists of 100 days of travel. Travelers are given four departure time choices, representing fifteen-minute intervals within the simulation time. Therefore, each traveler, or packet, is responsible for deciding a route of travel and the period in which to depart. After the traveler makes these two decisions, the arrival time of each traveler within the given departure time period is determined. The arrival time is generated randomly from a uniform probability distribution.

Each iteration of the simulation represents one day of travel and consists of the following steps, except for Day 1; (i) based on the CDTRC probabilities \( p_1 \) through \( p_8 \), select a CDTRC for the customer, (ii) for each departure time period, assign an arrival time for each customer and then sort the arrival times in ascending order. Next, for each CDTRC, from earliest departure time to latest departure time, have a (iii) customer arrive at time \( t \), (iv) check for departures and calculate volume on link, (vi) assign travel time, (vii) calculate and store departure time, travel time, and arrival time. Finally, at the end of the iteration, update \( p_i \) for each CDTRC possibility based upon the experienced travel time. As mentioned earlier, the first day is unique because CDTRC selection is based upon a random number. This method is
used to ensure that the network is fully loaded before “learning” begins. The most important component
of our methodology shown in Figure 2 is the algorithm used to simulate daily CDTRC decisions. As the
flowchart depicts, the CDTRC choice is determined using a random number (Rand_Num) that is used to
find a value from an array containing all $p_i(n)$ values. The one-dimensional 100-element CDTRC array
(CDTRC_Array[ ]) is created by converting $p_i(n)$ into an integer value (i.e. (int) 0.23 = 23) and then
randomly filling $n$ elements of the array with value, where $n$ is equal to the integer. For example, if $p_i(n)$
= 0.34, then the algorithm would fill 34 elements of CDTRC_Array[ ] with the value 34. Next, the
CDTRC is determined by selecting the value contained in CDTRC_Array[Rand_Num].

****INSERT FIGURE 2****

This algorithm ensures that the CDTRC is always random, and prevents any bias or corruption in the
CDTRC process. Finally, the action probabilities, $p_i(n)$, are updated using the reinforcement schemes
derived earlier in this paper. The simulation network used here is the same network used in Ozbay et al.
for the analysis of route choice behavior (2).

“Route 1”

<table>
<thead>
<tr>
<th>Length</th>
<th>21.45 miles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>$q_c = 4000$ veh/hr</td>
</tr>
</tbody>
</table>

Travel Time Function

\[
v = v_0 (1 - \frac{k}{k_j})
\]

\[
t = \frac{l}{v}
\]

where

$k_j = 250$ veh/mi

$v_0 = 65$ mph

“Route 2”

<table>
<thead>
<tr>
<th>Length</th>
<th>15 km</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>$q_c = 2800$ veh/hr</td>
</tr>
</tbody>
</table>

Travel Time Function

\[
v = v_0 (1 - \frac{k}{k_j})
\]

\[
t = \frac{l}{v}
\]

where
Analysis of the simulation results focused on the learning curves of the packets and the travel times experienced by the overall network. First, the network is examined, and then the learning curves of various participants are analyzed. Before the results are presented, the two types of equilibrium by which our results are analyzed must be defined.

**Definition 3 (Instantaneous User Equilibrium) (IUE):** The network is considered to be in instantaneous user equilibrium if the travel times on all routes for all time departure periods during any time period \([t, t+\Delta t]\), which is equal to the simulation time step (one minute), are almost equal. Instantaneous user equilibrium exists if

\[
\left| t_{route_1}^{(t,t+\Delta t)}(V_1) - t_{route_2}^{(t,t+\Delta t)}(V_2) \right| \leq \varepsilon
\]

where

\[t_{route_1}^{(t,t+\Delta t)}(V_1) \text{ and } t_{route_2}^{(t,t+\Delta t)}(V_2)\] are the average travel times during \([t, t + \Delta t]\) respectively

\(\varepsilon\) is a user specified small difference in minutes. This small difference can be considered to be the indifference band, or the variance a traveler allows in the route travel time. Thus, the driver will not switch to the other route if the travel time difference between the two routes is within \([-2.5, +2.5]\) minutes.

**Definition 4 (Global User Equilibrium) (GUE):** Global user equilibrium is defined as

\[
t_{1j}^{\text{total}}(V_1) = t_{2j}^{\text{total}}(V_2) \quad \forall j = 1, \ldots, k
\]

where

\(V_1^{\text{total}}\) and \(V_2^{\text{total}}\) are the total number of travelers that used route 1 and 2 respectively

\(j\) is the departure time period chosen by the traveler.

GUE states that the travel time on each route calculated using the total number of cars that used routes “i” during the overall simulation period should be equal. The main difference between GUE and IUE is that GUE is calculated using the cumulative amount of traffic volume on each route at the end of each simulation run whereas IUE is evaluated at each time step. It should be noted that IUE can be different than GUE since IUE is evaluated at each time step.
The results of the simulation can be found in Table 1. Average travel times on each CDTRC choice are shown for 10 simulations. As can be seen, the daily travel time difference between the 8 CDTRC choices is minimal, which indicates there is stability in the transportation network in terms of travel time. Comparing the CDTRC choices over the 10 runs evidence the similarity in travel times for each simulation run, which indicates that the learning algorithm is functioning correctly. Furthermore, the difference in average travel times for route 1 and route 2 are also small, the largest difference being 0.40 minutes. This less than a minute difference also confirms that the SLA is performing properly, and as a result, network equilibrium is obtained. Analysis of Figure 3a also graphically shows the similarity in travel time between the CDTRC options.

****INSERT TABLE 1****

****INSERT FIGURE 3A****

IUE conditions, as defined in Definition 3, occur when the travel time the travel times on routes 1 and 2 are within the given indifference band. Figure 3b shows the percentage of travelers within the pre-defined indifference band of ±2.5 minutes. This condition exhibits the similarity in travel times on the link, on a per minute basis, and proves that the network is stable in terms of travel time. As seen in Figure 3b, IUE conditions improve drastically after day 20. For this simulation, the final percentage of IUE conditions is 94.5%. For the other simulations, IUE conditions ranged from 85-92%. Such large percentages indicate that throughout all of the simulations, the travel times on the network remain similar, and that network equilibrium has been reached.

****INSERT FIGURE 3B****

Next, the learning curves for arbitrary packets are analyzed as seen in Figures 4a and 4b. Both of the packets exhibit a high degree of learning, as evidenced by their steep slope of their learning curve. The flat portion of the curves indicates that learning is not yet taking place; rather, instability exists within the network at that time. However, once network stability increases, the learning rate gradually increases. Furthermore, in both cases, each packet learned the shortest CDTRC by day 20. In the case of Packet #655 (Figure 4a), this day also corresponds to the time at which IUE conditions begin to increase. After this point, the learning curve for one CDTRC option increased, while the probability of choosing the other options decreased significantly. To further verify the learning process, the percentage of correct decisions was analyzed. Packet #655 made the correct CDTRC choice 84% of the time, while packet #390 made the correct decision 78% of the time. The variability in the plots is due to the
randomness introduced by the RTDC algorithm implemented in the traffic simulation; specifically, due to the random number generated by the random number generator. On any day of the simulation, the random number generated can cause travelers to choose the wrong CDTRC option, even though the traveler’s CDTRC probability, \( p_i(n) \), favors a certain route. Nonetheless, these percentages again prove that the learning algorithm is accurately modeling the learning rate of each packet, and is also permitting each packet to choose the shortest RTDC option.

****INSERT FIGURE 4A****

****INSERT FIGURE 4B****

Finally, in order to better prove the SLA learning algorithm, the simulation time period was run for a longer time period to see if the probability of choosing one route would converge to 1. Due to hardware constraints and the memory required by the current simulation program, the longest simulation run possible was 180 days. However, the results of this simulation, seen in Figure 5, strongly suggest that the learning algorithm allows a traveler’s CDTRC probability to converge towards 1. Currently, we are working on modifying the data structures within the simulation in order to run the simulation for a longer number of days. In this simulation, the packet’s CDTRC probability for choice 5 reached a value of 80%. The graph also shows that the packet’s learning rate is increasing rapidly, and furthermore, only one CDTRC choice is being rewarded. From these results, we can conclude that the probability of choosing CDTRC 5 would converge to 1 if the simulation ran for a longer period of time.

****INSERT FIGURE 5****

6 CONCLUSIONS

This study is an extension of the work conducted by Ozbay et. al. (1) using Stochastic Learning Automata. The SLA theory was adapted to model the day-to-day learning behavior of drivers in the context of the combined departure time-route choice decisions. Modifications to the SLA theory primarily centered upon updating the reinforcement scheme used to model driver learning. The introduction of multiple choices required the use of a multi-action learning scheme to accurately depict the combined departure time route choice problem. A Linear Reward-Penalty scheme, in the context of a multi-action learning automaton, was proposed to represent the day-to-day learning process for CDTRC choices. Next, the SLA-CDTRC model was tested using a traffic simulation that involved one origin-destination pair and two routes. Within the simulation, travelers were required to choose one
of eight CDTRC possibilities. The purpose of the simulation was to determine if the SLA-CDTRC model effectively and accurately modeled the day-to-day learning behavior of drivers, and also to determine if network equilibrium is reached based on this learning behavior. Simulation results show that network equilibrium, given in Definition 3 and 4, was achieved. Furthermore, results also prove that simulated driver learning did occur, and that the probability of choosing the best CDTRC choice converges towards 1 as shown in Figure 5. Future work includes extending this methodology to include multiple origin-destination networks.

REFERENCES
LIST OF FIGURES

FIGURE 1a  The automaton and the environment.

FIGURE 1b  Feedback mechanism for the SLA-CDTRC Model.

FIGURE 2  Flowchart For Simulation.

FIGURE 3a  Comparison of Travel Times on CDTRC Choices for Simulation #5

FIGURE 3b Instantaneous Equilibrium Conditions for Simulation #6.

FIGURE 4a  Evolution of CDTRC Probability for Packet #655 from Simulation #6.

FIGURE 4b  Evolution of CDTRC Probability for Packet #390 from Simulation #9.

FIGURE 5  Evolution of CDTRC Probability for Random Packet in Extended Simulation.

LIST OF TABLES

TABLE 1  Simulation Results – CDTRC Travel Times and Global User Equilibrium
FIGURE 1a The automaton and the environment.
FIGURE 1b Feedback mechanism for the SLA-CDTRC Model.
Start

Day = 1
Assign $p_t(day + 1)$ Values To each Packet

Day = Day + 1

Packet = Packet + 1

Calculate Volume based on Arrivals and Departures

Choose CDTRC such that $CDTRC = DTRC_{Array}[Rand\_Num]$

Calculate Travel Time, Link Departure Time

Packet > Total Packets

Update All $p_t$ Based On Linear Reward-Penalty Reinforcement Scheme

Day > Day N

Stop

FIGURE 2 Flowchart For Simulation.
FIGURE 3a. Comparison of Travel Times on CDTRC Choices for Simulation #5

FIGURE 3b. Instantaneous Equilibrium Conditions for Simulation #6.
Figure 4a  Evolution of CDTRC Probability for Packet #655 from Simulation #6.

Figure 4b  Evolution of CDTRC Probability for Packet #390 from Simulation #9.
FIGURE 5 Evolution of CDTRC Probability for Random Packet in Extended Simulation.
<table>
<thead>
<tr>
<th>Simulation Number</th>
<th>Route 1 Travel Time (min)</th>
<th>Route 2 Travel Time (min)</th>
<th>Simulation Statistics (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CDTRC 1</td>
<td>CDTRC 2</td>
<td>CDTRC 3</td>
</tr>
<tr>
<td>1</td>
<td>21.51</td>
<td>23.46</td>
<td>22.94</td>
</tr>
<tr>
<td>2</td>
<td>21.53</td>
<td>23.63</td>
<td>22.38</td>
</tr>
<tr>
<td>3</td>
<td>21.67</td>
<td>23.53</td>
<td>22.41</td>
</tr>
<tr>
<td>4</td>
<td>21.37</td>
<td>23.32</td>
<td>22.58</td>
</tr>
<tr>
<td>5</td>
<td>21.33</td>
<td>23.07</td>
<td>22.67</td>
</tr>
<tr>
<td>6</td>
<td>21.36</td>
<td>23.32</td>
<td>22.58</td>
</tr>
<tr>
<td>7</td>
<td>21.59</td>
<td>23.44</td>
<td>22.92</td>
</tr>
<tr>
<td>8</td>
<td>21.63</td>
<td>23.48</td>
<td>22.11</td>
</tr>
<tr>
<td>9</td>
<td>21.77</td>
<td>24.04</td>
<td>22.4</td>
</tr>
<tr>
<td>10</td>
<td>21.49</td>
<td>23.37</td>
<td>22.93</td>
</tr>
</tbody>
</table>