11-1995

Intelligent Control of Vehicles: Preliminary Results on the Application of Learning Automata Techniques to Automated Highway System

Cem Unsal
Virginia Polytechnic Institute and State University

John S. Bay
Virginia Polytechnic Institute and State University, bay@vt.edu

Pushkin Kachroo
University of Nevada, Las Vegas, pushkin@unlv.edu

Intelligent Control Of Vehicles: Preliminary Results on the Application of Learning Automata Techniques to Automated Highway System

Cem Ünsal[†], John S. Bay[†], and Pushkin Kachroo[†††]
[†, †††]*Center for Transportation Research* & [†, ††]*Machine Intelligence Laboratory*
Virginia Tech, Blacksburg, VA 24061
[†]*unsal@vt.edu*, [††]*bay@vt.edu*, [†††]*pushkin@ctr.vt.edu*

Abstract

In this paper, we suggest an intelligent controller for an automated vehicle to plan its own trajectory based on sensor and communication data received. Our intelligent controller is based on an artificial intelligence technique called learning stochastic automata. *The automaton can learn the best possible action to avoid collisions using the data received from on-board sensors. The system has the advantage of being able to work in unmodeled stochastic environments. Simulations for the lateral control of a vehicle using this AI method provides encouraging results.*

1. Introduction

Growing traffic congestion and the number of traffic casualties are two of the most significant problems today. It is also becoming increasingly difficult, for both monetary and environmental reasons, to continue to build additional highways. One of the solutions proposed for this problem is Automated Highway System. It is proposed that AHS will *evolve* from today's roads, and provide a fully automated "hands-off" operation at better levels of performance than today's roadways in terms of safety, efficiency, and operator comfort.

The 1997 AHS demonstration requested by the Congress will include lateral and longitudinal control, maintenance of position in the roadway traffic flow, and lane changing, all in a collision-free automated driving environment [5]. After 1997, AHS research will continue to gain importance. Vehicle control is one of the most vital parts of the AHS research. Considering the complexity of an Intelligent Vehicle Highway System (IVHS), it is obvious that the current control methods are not sufficient to provide a fully automated, collision-free environment. Intelligent control is one of the important tools that is useful in AHS research, although it is obvious that we may not solve the "whole problem" using a single method. The task of creating intelligent systems that we can rely on consequently brings the idea of "artificial intelligence" to mind.

Automatic vehicle control (AVC), as defined in AHS, will remove the driver as the source of control in the vehicle. Technologically, this step will be the natural consequence of the previous progress. In the early stages of "evolution," all vehicles may not be equipped with this technology right away. "Intelligent" and "non-intelligent" vehicles will have to coexist for some time. In this paper, we suggest an intelligent controller for an automated vehicle to plan its own trajectory based on sensor and communication data received. We visualize our controller as a part of the structure[1] described by Varaiya [15]. Our intelligent controller is part of the planning layer, and it is based on an artificial intelligence technique called *learning stochastic automata*. The aim is to design a system which can learn the best possible action based on the data received from on-board sensors and/or roadside-to-vehicle communications.

We visualize the intelligent controller of a vehicle as a stochastic automaton in a nonstationary environment. The system will control the path of the vehicle on the automated highway in the case of communication loss with the higher layer in the hierarchy and/or during the transition from automatic to manual control. A learning automaton system for vehicle control has the advantage of being able to work in unmodeled dynamic environments, unlike adaptive control methods or expert systems. It is also possible to model driver characteristics as a part of the system.

In the next section, we will introduce the learning automata and related definitions. Section 3 describes our application of learning automata to intelligent vehicle control. Simulation results and discussion of improvements and further research are in sections 4 and 5 respectively.

[1] Varaiya describes a five-layer hierarchical control architecture to achieve the vision of IVHS network. The layers of the architecture, starting at the top, are *network, link, planning & coordination, regulation* and *physical* layers. The network layer assigns a route to each vehicle as it enters the system. The link layer assigns each vehicle a path which balances traffic for all lanes and the target speed and desired platoon size for each section of highway. The planning layer creates a plan which approximates the desired path. The regulation layer controls the vehicle trajectory so that it confirms to this plan. Below the regulation layer, the physical layer provides sensor data and responds to actuator signals.

1082-3409/95 $04.00 © 1995 IEEE

2. Learning automata

Classical control theory requires a fair amount of knowledge of the system to be controlled. The mathematical model is assumed to be known, and the inputs are deterministic functions of time. Modern control theory, on the other hand, explicitly considers the uncertainties present in the system. Stochastic control methods assume that the characteristics of the uncertainties are known. However, all those assumptions on uncertainties and/or input functions may be insufficient to successfully control the system. It is therefore necessary to obtain further knowledge of the system by observing it in operation, since *a priori* assumptions may not be sufficient. We may view the problem as a problem in learning. A learning system has 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" [12].

The stochastic automaton attempts a solution of the problem without any information on the optimal action (initially, equal probabilities are attached to all the actions). A stochastic automaton acting as described to improve its performance is called a *learning automaton* (LA). The automaton can perform a finite number of actions in a random environment. One of the actions is selected at random. When a specific action is performed, the environment provides a response stochastically connected to the chosen action. This response may be favorable or unfavorable (or may define the degree of "acceptability" for the action). Action probabilities are then updated based on the response. An important point is that the knowledge of the nature of the environment is minimal. The environment may be time varying; the automaton may be a part of a hierarchical decision structure, but unaware of its role, or the stochastic characteristics of the output of the environment may be caused by the actions of other agents unknown to the automaton.

The first learning automata models were developed in mathematical psychology. Early research in this area is surveyed by Bush and Mosteller [3] and Atkinson *et al.* [1]. Tsetlin [14] introduced deterministic automata operating in random environments as a model of learning. Fu and colleagues were the first researchers to introduce stochastic automata into the control literature [6,7]. Applications to parameter estimation, pattern recognition and game theory were initially considered by this school. Properties of linear updating schemes and the concept of a 'growing' automaton are defined by McLaren [10]. Chandrasekaran and Shen [4] studied nonlinear updating schemes, nonstationary environments and games of automata. Narendra and Thathachar have studied the theory and applications of LA and carried out simulation studies in the area [13].

2.1 Types of environment and reinforcement schemes

The environment responds to the (input) action of the automaton by producing an output (Figure 1). There are several *environment models* defined by the output set of the environment. Models in which the output can take only one of two values (0 or 1) are referred to as *P-models*. The output value of 1 corresponds to an "unfavorable" (failure, penalty) response, while output of 0 means the action is "favorable." When the output of the environment is a continuous random variable with possible values in an interval [a, b], the model is named *S-model*.

Narendra and Thathachar [13] state that an S-model environment is relevant in control systems with a continuous valued performance index. The outputs lie in the interval $[0,1]$[2] in the S-model, and therefore, are neither favorable nor totally unfavorable. The main problem is to determine how the probabilities of all actions are to be updated. It has been shown that all the principal results derived for the P-model carry over to the more realistic S-model.

On the other hand, if the probability of receiving a penalty for a given action is constant, the environment is called a *stationary environment*; otherwise, it is *nonstationary*. 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 the actions to minimize the expected penalty. As in adaptive control theory, the practical justification for the study of stationary environments is based on the assumption that if the convergence of a scheme is sufficiently fast, then acceptable performance can be achieved in slowly changing environments.

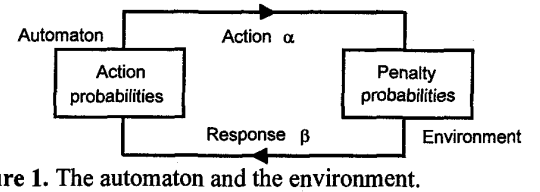


Figure 1. The automaton and the environment.

The main concept behind the learning automaton model is a probability vector defined as $p(n) = \{ p_i(n) \in \{0,1\} \mid p_i(n) = Pr[\alpha(n)=\alpha_i] \}$ where α_i is one of the possible actions. The action probabilities are updated at every stage n using a *reinforcement scheme*. The updating of the probability vector provides the learning behavior of the automata. A learning automaton generates a sequence of actions on the basis of its interaction with the environment. If the

[2] In the case where the environment outputs β_k are not in the interval [0,1], but in $[a,b] \subset \mathbf{R}$, it is always possible to map the output into the unit interval.

automaton is "learning" in the process, its performance must be superior to a "pure chance" automaton, for which the action probabilities are equal. The quantitative basis for assessing the learning behavior is quite complex, even in the simplest P-model and stationary random environments [13]. Based on the average penalty to the automaton, several definitions of behavior, such as *expediency*, *optimality*, and *absolute expediency*[3], are given in the literature. Reinforcement schemes are categorized based on the behavior type they provide, and the linearity of the algorithm used. In general terms, a reinforcement scheme can be represented as:

$$p(n+1) = T[p(n), \alpha(n), \beta(n)]$$

where T is a mapping, α is the action, and β is the input from the environment. If $p(n+1)$ is a linear function of $p(n)$, the reinforcement scheme is said to be linear; otherwise it is termed nonlinear. Early studies of reinforcement schemes were revolved mostly around linear schemes for reasons of analytical simplicity. However, a few attempts were made to study nonlinear schemes [2, 4, 16]. Generalization of such schemes to the multi-action case was not straightforward [13]. Researchers started looking for the conditions on the updating functions that ensure a desired behavior. This approach led to the concept of absolute expediency. Absolutely expedient learning schemes are presently the only class of schemes for which necessary and sufficient conditions of design are available [2, 13].

2.2 Multiple environments/teachers

The learning automaton may send its action to multiple environments. In that case, the actions of an automaton elicit a vector of outputs. Then, the automata have to "find" an optimal action which "satisfies" all the environments (better yet, all the "teachers"). In a multi-teacher environment, the automaton is connected to N separate teachers. The action set of the automaton is of course the same for all teacher/environments. Baba discussed the problem of a variable-structure automaton operating in many-teacher (stationary and nonstationary) environments [2]. Conditions for absolute expediency are given in his work. (Our initial algorithm is an adapted version of the algorithms described in [2].)

Some difficulties arise while formulating a mathematical model of the learning automaton in a multi-teacher environment. Since we get multiple outputs from the environment, the question of how to "interpret" the output vector[4] $\beta(n)$ is important. The elements of the output vector must be combined in some fashion to form the input to the automaton. A straight-forward method is to define the input to the automaton as a weighted sum of all outputs where k_i's denote the weights attached to each teacher output and must be chosen in order to guarantee $\beta(n) \in [0,1]$:

$$\beta(n) = k_1 \beta^1(n) + k_2 \beta^2(n) + \ldots + k_N \beta^N(n)$$

3. Application of learning automata to AHS: Intelligent vehicle control

Our approach to the problem of vehicle control makes use of Learning Automata techniques described in previous section. We visualize the controller of an intelligent car as an automaton (or automata group) in a nonstationary environment. The aim here is to design an automata system which can learn the best possible action based on the data received from on-board sensors, vehicle-to-vehicle and/or roadside-to-vehicle communications. The significance of this system is that the learning automata system we defined will be useful as a backup system (or *the* system in a homogeneous traffic in the near future) in controlling the path of a vehicle in the case of communication loss with the higher layer in the hierarchy as well as during the transition from fully automatic to manual control.

3.1 The model

The basic model of a simple planning/coordination layer for lane changing and speed control of a vehicle is shown in Figure 2. The automaton that constitutes the decision structure has five actions: *stay in lane* (the "idle" action/state), *shift to right* (lane), *shift to left*, *accelerate* and *decelerate*. We assume that there are four different types of sensors (*front sensor*, *right sensor*, *left sensor* and *speed sensor*) and a roadside-to-vehicle communication (*link layer*) that provide data. Each sensor block and the link layer are "teachers" in a nonstationary environment (or multi-environment system). The response of the environment is then a combination of the outputs of all five teachers, as discussed in section 2.2. The mapping F from sensor block outputs to the input β of the automaton can be a binary function (for a P-model environment) or a linear combination of five teacher outputs. The final automaton model will use a linear combination of teacher outputs with adjustable weight factors.

It is important to differentiate between an "automaton environment" and the "physical environment." The action α of the automaton is a signal to the regulation layer which defines the current choice of action. It is the regulations layer's responsibility to interpret this signal. When an action is carried out, it affects the physical environment. The teachers/sensors in turn sense the changes in the environment, and the feedback loop is closed with the signal β.

[3] A learning automaton is absolutely expedient if the expected value of the average penalty at one iteration step is less than the previous step *for all steps*.

[4] Are the output from different teachers to be summed after normalization? Can we introduce weight factors associated with specific teachers? If so, how?

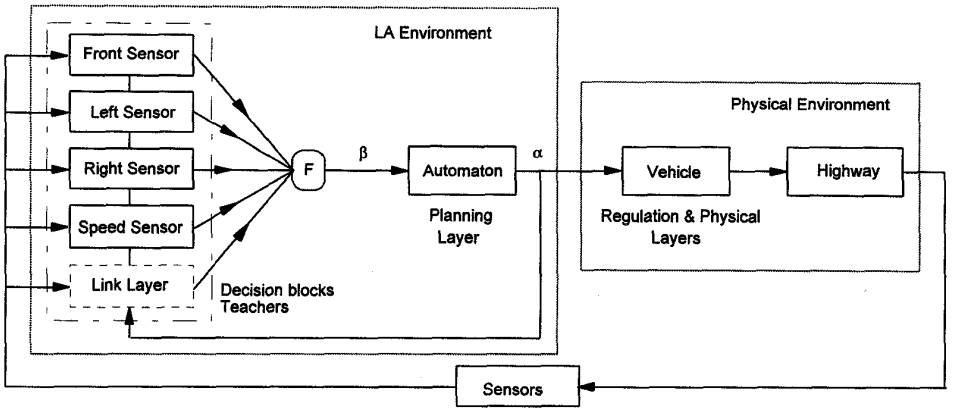


Figure 2. Automata in a multi-teacher environment connected to the physical layer

The discussion of nonstationary environments in the previous section is based on the changing penalty probabilities of actions. In this application, the action probabilities in the learning automaton environment are functions of the status of the physical environment (*e.g.*, a vehicle in front will result in a penalty response from front sensor/teacher if the chosen action is *stay in lane* or *accelerate*). The realization of a mathematical model of this physical environment may be extremely difficult.

3.2 Sensors

The four sensors/teachers listed above are actually simple decision blocks which "calculate" the penalty response associated with the corresponding sensor, based on the chosen action. Tables 1-4 below describe the output of these decision blocks for different actions.

The output of the two side sensor blocks may also have values in the interval [0,1] depending on the distance of the vehicle from the sensor source. However, that type of design may result in a more expensive implementation, as it may require distance measurement and/or additional sensors.

We do assume that the front sensor is capable of providing the headway distance. Functions f_i are shown as linear functions of the headway distance; but, they can be any arbitrary function valued between 0 and L_i. The difference between the action *Decelerate* and other actions comes from the fact that decreasing the speed is a possible way of avoiding a collision.

The value *dev* is defined as the difference between current speed and the desired speed. Again the functions f_i do not have to be linear. The penalty for actions *Sil, SL* and *SR* in the case of a deviation is defined differently than the penalty for actions *Acc* and *Dec*, since actions *Sil, SL* and *SR* are not related to longitudinal control. The values of the limits d_i defines the capabilities of the sensors (in the case of side and front sensor blocks) as well as the "behavior" of the vehicle, *i.e.*, the sensitivity to the headway distance and to the speed fluctuations.

Table 1. Output of the *Left Sensor* block.

	Sensor Status	
Current Action[5]	Vehicle in sensor range	No vehicle in sensor range
SiL	0	0
SL	1	0
SR	0	0
Acc	0	0
Dec	0	0

Table 2. Output of the *Right Sensor* block.

	Sensor Status	
Current Actions	Vehicle in sensor range	No vehicle in sensor range
SiL	0	0
SL	0	0
SR	1	0
Acc	0	0
Dec	0	0

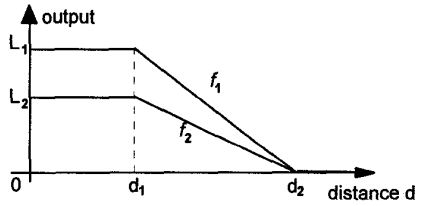


Figure 3. The definition of the limits and functions for front sensor block.

[5] Actions are: Stay in Lane, Shift Left, Shift Right, Accelerate, and Decelerate.

Table 3. Output of the *Front Sensor* block.

Cur. Action	Sensor Status: Vehicle in region A ($d < d_1$)	Vehicle in region B ($d_1 < d < d_2$)	No vehicle in range
SiL	L_1	$f_1(d)$	0
SL	0	0	0
SR	0	0	0
Acc	L_1	$f_1(d)$	0
Dec	L_2	$f_2(d)$	0

Table 4. Output of the *Speed Sensor* block (See figure 4 for the definitions of the functions f_i).

Cur. Action	Sensor Status: region A	region B	region C	region D	region E
SiL	M_2	$f_2(dev)$	0	$f_2(dev)$	M_2
SL	M_2	$f_2(dev)$	0	$f_2(dev)$	M_2
SR	M_2	$f_2(dev)$	0	$f_2(dev)$	M_2
Acc	0	0	0	M_1	$f_1(dev)$
Dec	M_1	$f_1(dev)$	0	0	0

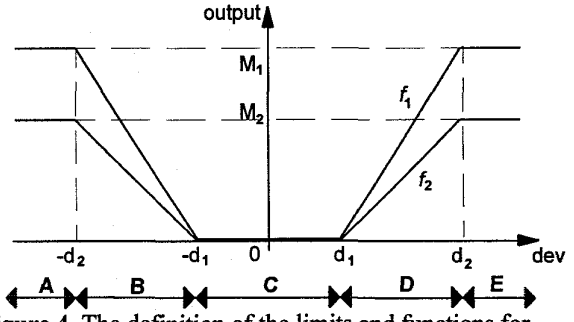


Figure 4. The definition of the limits and functions for speed sensor block.

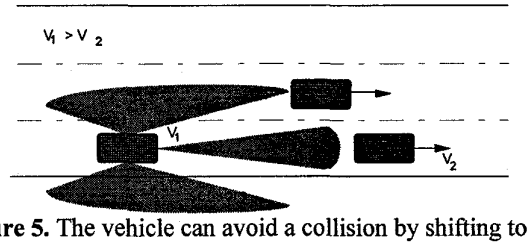


Figure 5. The vehicle can avoid a collision by shifting to the leftmost lane.

The fifth block which represents the evaluation of the roadside-to-vehicle data, is not defined yet. The importance of this block is that the global characteristic of the data received from the roadside should be used to solve some of the problems in the lane changing maneuvers. For example, in the case shown in Figure 5, the action SL will receive a penalty response, although it is the best action to avoid a collision (considering only the lateral control). The data received from the link layer can be used to suppress the penalty response of the left sensor block[6]. We expect our automata and multi-teacher environment to guide the vehicle without collision using the learning algorithm described below.

3.3 The algorithm

The initial algorithm we used is adapted from [2] which describes a reinforcement scheme for a nonstationary multi-teacher S-model environment as:

$$\text{if } \alpha(n) = \alpha_i,$$

$$p_i(n+1) = p_i(n) + \left[\frac{s_i^1 + \cdots + s_i^N}{N}\right] \cdot \sum_{\substack{j=1 \\ j \neq i}}^{r} \phi_j[\underline{P}(n)] - \left[1 - \frac{s_i^1 + \cdots + s_i^N}{N}\right] \cdot \sum_{\substack{j=1 \\ j \neq i}}^{r} \psi_j[\underline{P}(n)]$$

$$p_j(n+1) = p_j(n) - \left[\frac{s_i^1 + \cdots + s_i^N}{N}\right] \cdot \phi_j[\underline{P}(n)] - \left[1 - \frac{s_i^1 + \cdots + s_i^N}{N}\right] \cdot \psi_j[\underline{P}(n)] \quad \text{for all } j \neq i \tag{3.1}$$

where the functions ϕ_i, ψ_i satisfy the following conditions:

$$\frac{\phi_1}{p_1(n)} = \cdots\cdots = \frac{\phi_r}{p_r(n)} = \lambda(\underline{P}(n))$$

$$\frac{\psi_1}{p_1(n)} = \cdots\cdots = \frac{\psi_r}{p_r(n)} = \mu(\underline{P}(n)) \tag{3.2}$$

$$p_j(n) + \psi_j(\underline{P}(n)) > 0$$

$$p_i(n) + \sum_{\substack{j=1 \\ j \neq i}}^{r} \phi_j(\underline{P}(n)) > 0$$

$$p_j(n) - \phi_j(\underline{P}(n)) < 1$$

for all $i, j = 1, \ldots, r$, where λ and μ satisfy:

$$0 < \lambda(\underline{p}) < 1$$

$$0 < \mu(\underline{p}) < \min_{j=1,\ldots,r}\left\{\frac{p_j}{1-p_j}\right\} \tag{3.3}$$

An automaton using the above algorithm is proven to be ϵ-optimal in a nonstationary multi-teacher environment [2]. Initial simulations we devised do not use such an S-model environment. The adjustment of function parameters in functions ϕ_i, ψ_i is too difficult a task. We initially used the P-model environment with a simplified environment model to test the feasibility of the use of learning automata in vehicle control. Initial simulations are based on the following model/assumptions:

[6] One way to do this is to assign a larger weight to the *link layer* teacher output in an S-model environment.

- There are three actions defined for the automaton: Sil, SL, and SR.
- Three sensor modules are assumed: front sensor, left sensor and right sensor.
- The environment is of P-model, *i.e.*, all inputs are from the set {0,1} (*i.e.*, $d_1 = d_2$ for front sensor block). The mapping F which computes the combined input to the automaton is chosen as an OR-gate. In a sense, the environment is a single-teacher environment in which a single module computes the response based on the data obtained from all three sensors.
- In the simulation, we assumed that the controlled vehicle's speed is higher than other vehicles, *i.e.*, we tried to control a vehicle which is passing other vehicles in the highway.
- The technological requirements for the model used in the simulation (as well as the model described above) are no different than those defined in the present research. Communication requirements, on the other hand, may be less.
- The regulation layer carries out an action if it is chosen m times consecutively by the automaton, where m is a predefined parameter. (This may be changed to "k times in the last m choices," or to a more sophisticated decision rule.) When an action is carried out (*i.e.*, shifting lane), the action probabilities are initialized to $1/r$.
- A minimum processing speed of 25 iterations per second is assumed. This is only related to computation; the sensor data feed can be slower than 25 Hz. This value is much slower than the limit of 200 Hz which is considered in current AHS research [11].

Since we chose a P-model environment, and an OR-gate to compute combined response of the environment, the updating algorithm is relatively simple than defined above. The update algorithm with the functions ϕ_i, ψ_i is given below:

$$\begin{aligned} &\text{if } \alpha(n) = \alpha_i \\ &p_i(n+1) = p_i(n) + \max_k\{s_k\}(-k\theta)\cdot H\cdot[1-p_i(n)] \\ &\qquad +[1-\max_k\{s_k\}]\cdot(-\theta)\cdot[1-p_i(n)] \\ &p_j(n+1) = p_j(n) - \max_k\{s_k\}(-k\theta)\cdot H\cdot p_j(n) \\ &\qquad +[1-\max_k\{s_k\}]\cdot(-\theta)\cdot p_j(n) \quad \text{for all } j \neq i \end{aligned} \tag{3.4}$$

i.e.:

$$\psi_k(\underline{P}(n)) \equiv -\theta\cdot p_k(n)$$
$$\phi_k(\underline{P}(n)) \equiv -k\cdot\theta\cdot H\cdot p_k(n)$$

where parameters k, θ and the function H are defined as:

$$0<\theta<1,\quad 0<k\theta<1 \qquad H \equiv \min\left\{1;\ \max\left[\frac{p_i}{k\theta(1-p_i)}-\varepsilon;\ 0\right]\right\}$$

where ε is an arbitrarily small positive real number. Note that the function H includes p_i which is the action probability corresponding to the current action. There are two reasons for this definition: first, H cannot be negative, in order to keep the penalty-reward characteristics of the functions ϕ_i and ψ_i. Furthermore, the fourth condition in Equation 3.2 implies:

$$p_i(n)+\sum_{\substack{j=1\\ j\neq i}}^{r}(-k\theta)Hp_j(n)>0 \Rightarrow p_i(n)-k\theta H\cdot(1-p_i(n))>0$$

$$\Rightarrow H<\frac{p_i(n)}{k\theta(1-p_i(n))}$$

This function is found to work better than the one given in [2], considering the convergence to the optimal action. The convergence rate is faster for an action probability associated with the optimal action, when the current action's probability is close to 1. (Note that H is a function of the probability of the current action p_i and is not related to the index j in Equation 3.3.) The situation where the probability of the optimal action is close to 0 occurs frequently in our application. In order have a fast update on the probability vector, we had to change the definition of the function H to the largest possible value satisfying the conditions for ϕ_i in Equation. 3.2.

4. Simulation results

The simulations were written and run in Matlab. Here, we will give only the snapshots of two very short segments of simulation (6 seconds each) with the action probability vectors plotted for corresponding time intervals.

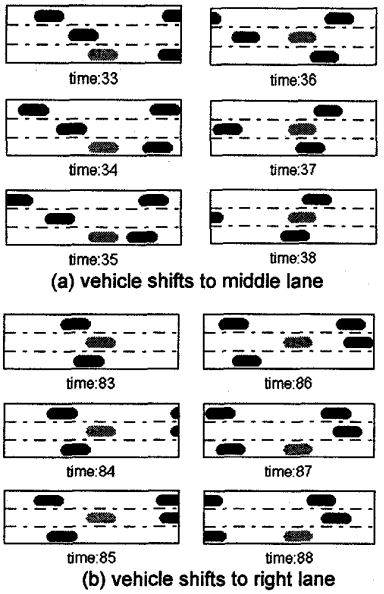
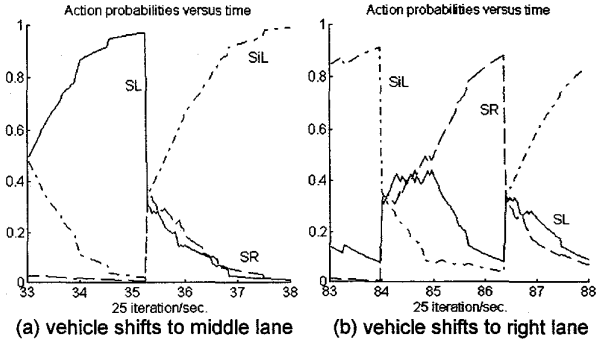


Figure 6. The trajectory of a vehicle (gray) going faster than other vehicles; traveling left-to-right.



(a) vehicle shifts to middle lane (b) vehicle shifts to right lane

Figure 7. Action probabilities for the same time intervals.

As seen in Figures 6 and 7, the action probability vector is updated based on the sensor data received. Sensor ranges are defined as 5m for side sensors, and 15 m for front sensor. In case (a), the probability of the action SL approaches 1 since other actions (SiL and SR) receive penalty responses from the front and right sensor blocks. Around t = 35 sec., the regulation layer decides to "fire" this action; the probabilities are then initialized to $1/r$. In case (b), the probability of the action SR approaches 1 around t = 86 sec., due to the fact that the front and left sensors detect the two approaching vehicles. In order to avoid a collision, the action *Shift Right* is fired, and the probability vector is initialized. In both cases, the third action SiL is fired due to the presence of vehicles in the adjacent lanes. For example, in case (b), the probability of the action SiL approaches 1 and SiL is sent to the regulation layer around t = 84 sec. since it is the best possible action for the last m iteration steps.

The update speed for both the sensor data and the iteration algorithm is 25 iterations per second. All other vehicles' velocities are between 81 and 89 kmh. while the controlled vehicle's speed is 90 kmh.

5. Discussion on simulation results and further research

With its limited sensor capabilities, the vehicle cannot obtain a global view of its environment. This is even impossible for a highway portion of length, say, 100m. The need of a higher layer of hierarchy (such as the *link layer* in [15]) is inevitable. In the situation shown in Figure 5, one of the two possible solutions for avoiding collision is shifting to the leftmost lane; however, the vehicle's decision block based on local sensor data cannot avoid an imminent collision[7]. For this reason a connection to a higher "layer" of information is necessary. A higher layer which has more complete data of the changing environment must assist/supervise the vehicle in its actions/decisions.

Another "hole" in the automata/teacher model is the problem of preference for a vehicle in the middle lane (or in a lane with adjacent lanes in both sides) while the front sensor block is the only one sending a detection signal. Our simple model cannot find the best action in that case; the action probabilities of SR and SL both approach to 0.5. Again, a higher-level command from the link layer or a "behavioral" adjustment for the decision block is needed.

The behavior of the controlled vehicle currently depends on several factors. For example, the vehicle sometimes changes its lane after a short period of time, sometimes in the last possible moment[8]. There are two important parameters affecting the behavior of the controlled vehicle. One is the frequency of the update for action probabilities. The higher the frequency, the faster the reaction to sensor block inputs. Although we did not present the results here, slower update rates resulted in collision for the same situations shown in Figure 7.

The range of the sensors and the definition of the limits (for sensor block output functions) are also important. It is possible to decrease the update rate for larger sensor ranges, but the relation is not deterministic since the behavior of the car depends on many other factors. Again, in a situation similar to one presented in Figure 7a, a shorter (front sensor) range resulted in a collision.

The extension to the S-model may help to decrease the update frequency, because it will give the automaton more time to "adjust" (*e.g.*, $d_1 < d_2$ for front sensor). Adjusting the parameters for optimum results is a very difficult task, even for a simple automaton/environment pair in a very crude simulation.

The need for a more complicated sensor definition and for "hierarchical interference" is obvious. Our research will continue toward the development of a more complex decision system. Initially, learning automata algorithms are found to be a promising tool for intelligent control of vehicle in AHS. Since the highway traffic may have a heterogeneous character (*i.e.*, automatically and manually controlled vehicles in the same highway), it is important for an automatically controlled vehicle to differentiate between "intelligent" and 'dumb" vehicles. This is important because the roadside structure will also not have the "complete" information about the vehicles without communication capabilities. Therefore, a controlled vehicle must rely on its own sensor data for a complete data on its immediate neighborhood, in a heterogeneous traffic.

Our model of vehicle control is consistent with the current assumptions on sensors and communications capabilities; desired sensor and communication

[7] Again,we consider only lateral motion while longitidunal speed is constant. Implementition of two additional actions (*Acc* and *Dec*) may solve the problem.

[8] It sometimes fails to change the lane in time, due to parameter definitions.

characteristics for the controller described here are generally the same, if not lower, than the ones required by other control aspects of AHS[9].

Our initial controller design is far from perfect. Some of the issues that need investigation are:

- The P-model environment must be extended to S-model in order to incorporate a priority level with the teacher outputs. The multi-teacher characteristic of the environment will then show its potential application. The weighting factors associated with each teacher/sensor output defines the "behavior" of the vehicle. The adjustment of these can be viewed as "another level" of learning. Several other AI methods can be used for this purpose.
- The first additions to the current decision structure will be the two actions *Acc* and *Dec*. We are also considering the possibility of a second automaton for these actions, separating the longitudinal and lateral control. This will move our research toward a multi-automata system, which could subsequently bring us to an application of game theory to interconnected automata [13].
- It is possible to treat a nonstationary environment as a sequence of stationary environments. It is possible to "discretize" the physical status of a vehicle to several "stationary statuses," and to use learning algorithms for stationary environments. The literature on the stationary case is much more detailed than the nonstationary case.
- Since the algorithm increases the possibility of the other actions while penalizing an undesired action, the use of the *H*-function results in fast convergence, but not in an absolutely expedient scheme (when the environment is treated as a sequence of stationary environments, as we mentioned above). The absolute expediency conditions are not met due to the choice of function *H*. The reason for this can also be seen by examining the definition of absolute expediency. With this algorithm, all action probabilities except that of the current action are increased whenever the chosen action receives a penalty from the environment. Therefore, the sum of penalties may increase at some time steps. As we update our algorithm to the S-model, similar convergence tests will be carried out.
- Our intelligent controller model must be incorporated with a realistic vehicle dynamics model described in [9] to simulate the physical constraints of the actions to be carried out by the regulation layer.
- The algorithms and methods we described here will be tested using scaled model vehicles currently designed in the FLASH (Flexible Low Cost Automated Scaled Highway) laboratory in the Virginia Tech's Center for Transportation Research [8].

[9] For example, current methods for longitudinal control of vehicles requires large bandwidth communications and accurate sensors.

Acknowledgment

This material is based upon work supported in part by the Naval Research Laboratory under Grant no. N00014-93-1-G022, and in part by the Center for Transportation Research/VDOT under Smart Road Project. The content of this paper does not necessarily reflect the position or policy of the United States Government.

References

[1] Atkinson, R.C., G.H. Bower, and E.J. Crothers, *An Introduction to Mathematical Learning Theory*, New York: Wiley, 1965.

[2] Baba, N., *Lecture Notes in Control and Information Sciences: New Topics in Learning Automata Theory and Applications.* Springer-Verlag, 1984.

[3] Bush, R.R., and F. Mosteller, *Stochastic Models for Learning*, New York: Wiley, 1958.

[4] Chandrasekharan, B., and D.W.C. Shen, "On Expediency and Convergence in Variable Structure Stochastic Automata," *IEEE Transactions on System Science and Cybernetics*, SSC-5, 1968, pp. 145-149.

[5] DOT's IVHS Strategic Plan Report to Congress, in *November 10th, 1993 Hearing before the Subcommittee on Investigations and Oversight of the Committee on Science, Space and Technology, US. House of Representatives, 103 Congress, First Session*, pp. 35-36.

[6] Fu, K.S., "Stochastic Automata as Models of Learning Systems," in *Computer and Information Sciences II*, J.T. Lou, Editor, New York: Academic, 1967.

[7] Fu, K.S., and G.J. McMurtry, "A Study of Stochastic Automata as Models of Adaptive and Learning Controllers," Technical Report TR-EE 65-8, Purdue University, Lafayette, Ind., 1965.

[8] Kachroo P., K. Özbay, R. G. Leonard, and C. Ünsal, "Flexible Low-cost Automated Scaled Highway (FLASH) Laboratory for Studies on Automated Highway Systems," *IEEE Conference on Systems, Man and Cybernetics, Vancouver, B.C.*, 1995.

[9] Kachroo, P., and M. Tomizuka, "Vehicle Traction Control and its Application," Technical Report UI-PRR-94-08, Program on Advanced Technology for the Highway, Institute of Transportation Studies, University of California at Berkeley, 1994.

[10] McLaren, R.W., "A Stochastic Automaton Model for Synthesis of Learning Systems," *IEEE Transactions on System Science and Cybernetics*, SSC-2, 1966, pp. 109-114.

[11] Lasky, T.L., and B. Ravani, "A review of Research Related to Automated Highway System (AHS)," Interim Report for Federal Highway Administration, Contract No. DTFH61-93-C-00189, University of California (Davis), October 25, 1993.

[12] Narendra, K.S., and M.A.L. Thathachar, "Learning Automata—A Survey," *IEEE Transactions in Systems, Man and Cybernetics*, Vol. SMC-4, No. 4, July 1974.

[13] Narendra, K.S. and M.A.L. Thathachar, *Learning Automata.* Prentice Hall, New Jersey, 1989.

[14] Tsypkin, Ya. Z., *Adaptation and Learning in Automatic Systems*, New York: Academic, 1971.

[15] Varaiya, P., "Smart Cars on Smart Roads: Problems of Control," *IEEE Transactions on Automatic Control*, Vol. 38., No. 2, Feb. 1993, pp. 195-207.