1-1999

Multiple Stochastic Learning Automata for Vehicle Path Control in an Automated Highway System

Cem Unsal
Carnegie Mellon University

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

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

should be adjusted for home advantage [3]. It is interesting that the home advantage increases from regular season to playoff competition in the NBA and for European club soccer, remains the same for the NHL and declines for MLB. Adjustive ratings have been proposed using maximum likelihood, least squares and exponential smoothing. The probabilities of the various outcomes have been calculated using Gaussian and Poisson assumptions. For sports with few draws, the correct winner can be predicted correctly about 70% of the time. See [3] for comparisons of various methods and for a bibliography.

VI. Conclusions

Systems analysis was applied to a comprehensive list of 83 sports. Each of those sports can be placed into one of three categories: combat sports where each competitor tries to control the opponent, object sports where each competitor tries to control an object or independent sports where each competitor in unimpeded by the opponent and therefore must control him(her)self. Each ensuing SRS consists of three phases: evaluation, weighting and rating. The input to the evaluation phase is a performance in a sport. Each performance is then evaluated by either judgement, measurement or objective scoring. In the weighting phase, a matrix of weights is applied to the evaluated performances. In the rating phase, operations are classified as being accumulative or adjustive. Most accumulative SRS's are ad-hoc using a sum of some or all of the weighted evaluated performances. For accumulative SRS's, each rating changes monotonically except for data ageing. Most adjustive SRS's are based on some theoretical concept in which predictor-corrector action causes ratings to either increase, decrease or remain the same as required by the corrector. Each SRS is analogous to a one-level neural network (weighting phase) followed by an inference engine (rating phase) generating the ratings.

Accumulative SRS's, in contrast to adjustive SRS's, appear to encourage more participation by high-rated players as noted by administrators of men's and women's professional tennis. Conversely, adjustive SRS are about equally sensitive to motion up or down and are more commonly used to predict future performance. For adjustive SRS's applied to sports with few draws and operating on margin of victory adjusted for home advantage, it is possible to predict the correct winning team about 70% of the time.

It is hoped that this survey and taxonomy of SRS's may be an aid to those who are interested in evaluating human performance (not necessarily restricted to sports).

References

[1] *Olympic Rev.*, no. XXV-4, Aug./Sept. 1995, p. 72.
[2] R. Stefani, "Survey of the major world sports rating systems," *J. Appl. Statist.*, vol. 24, no. 6, pp. 635–646, 1997.
[3] , "Predicting outcomes," *Statistics in Sport*. London, U.K.: Arnold, 1998, ch. 12.
[4] *The 1994 Information Please Sports Almanac*. Boston, MA: Houghton Mifflin, 1994.
[5] *The World Almanac and Book of Facts 1995*. New York: Funk and Wagnalls, 1995.
[6] *The Official World Encyclopedia of Sports and Games*. New York: Paddington, 1979.
[7] "Nordic combined," *Olympic Rev.*, no. 195, p. 195, Apr. 1992.
[8] C. Lee *et al.*, "An empirical study of boxing match prediction using a logistic regression analysis," in *Proc. Section Statistics Sports, Amer. Statistical Assoc., Joint Statistical Meeting*, Anaheim, CA, Aug. 10–14, 1997.

Multiple Stochastic Learning Automata for Vehicle Path Control in an Automated Highway System

Cem Ünsal, Pushkin Kachroo, and John S. Bay

***Abstract*—This paper suggests an intelligent controller for an automated vehicle planning its own trajectory based on sensor and communication data. The intelligent controller is designed using learning stochastic automata theory. Using the data received from on-board sensors, two automata (one for lateral actions, one for longitudinal actions) can learn the best possible action to avoid collisions. The system has the advantage of being able to work in unmodeled stochastic environments, unlike adaptive control methods or expert systems. Simulations for simultaneous lateral and longitudinal control of a vehicle provide encouraging results.**

***Index Terms*— Automated highway system (AHS), intelligent vehicle control, reinforcement learning, stochastic learning automata.**

I. Introduction

Growing traffic congestion and the number of traffic casualties are two of the most significant transportation problems today. At the same time, building additional highways is becoming increasingly difficult for both monetary and environmental reasons. One of the solutions to this problem is the Automated Highway System (AHS). 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. Vehicle control is one of the most vital parts of the AHS research. Considering the complexity of an intelligent vehicle/highway system, it is becoming apparent that conventional methods are not sufficient to provide a fully automated, collision-free environment. The task of creating intelligent systems that we can rely on consequently brings the idea of "artificial intelligence" (AI) to mind. Several AI paradigms are emerging as candidate solutions.

Automatic vehicle control, as defined within AHS, proposes to remove the driver as the source of control in the vehicle. In the early stages of this evolution, not all vehicles may be equipped with this technology; "intelligent" and "nonintelligent" vehicles will have to coexist for some time. In this paper, we suggest an intelligent controller for an automated vehicle that plans its own trajectory based on sensor—and, in the future, communication—data received. We visualize our controller as a part of the five-layer hierarchical control architecture described by Varaiya [20]. The layers of this architecture, starting at the top, are *network, link, planning and coordination, regulation* and *physical* layers. The planning layer creates a plan that approximates the desired path. The regulation layer controls the vehicle trajectory so that it conforms to this plan. Below the regulation layer, the physical layer provides sensor data and responds to actuator signals.

Manuscript received May 17, 1996; revised August 7, 1997, July 21, 1998, and August 17, 1998. This material was based upon work supported in part by the Naval Research Laboratory under Grant N00014-93-1-G022, and in part by the Center for Transportation Research/VDOT under Smart Road Project.

C. Ünsal is with the Institute for Complex Engineered Systems, Carnegie Mellon University, Pittsburgh, PA, 15213-3890 USA (e-mail: unsal@ri.cmu.edu).

P. Kachroo and J. S. Bay are with the Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, 24061-0111 USA.

Publisher Item Identifier S 1083-4427(99)00248-9.

Our intelligent controller can be visualized as part of this planning layer. It uses an AI technique called *learning stochastic automata*. The aim is to design a system that can learn the best possible action(s) based on the data from on-board sensors and/or roadside-to-vehicle communications. We visualize the intelligent controller of a vehicle as two stochastic automata (one for lateral, the other for longitudinal control) in a nonstationary environment. The system will control the path of the vehicle in 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. It may also be used as the sole system for path planning, though it is not necessarily fail-safe. 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 that need detailed information about the system. 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 III describes our application in intelligent vehicle control. Simulation results and discussion of improvements and further research are in Sections IV and V, respectively.

II. Learning Automata

Classical and modern control techniques require a fair amount of knowledge of the system to be controlled, in the form of a mathematical model of the system or statistical values such as mean and variances of the uncertainties. However, not all those assumptions on the system or uncertainties may be valid or accurate. It is therefore necessary to obtain further knowledge of the system by observing it during operation. One approach is to view this as a problem in learning [6], [16], [18]. Rule-based systems, although performing well on many control problems, cannot handle unanticipated situations. The idea behind designing a learning system is to guarantee robust behavior without the complete knowledge, if any, of the system and/or environment. A crucial advantage of reinforcement learning compared to other learning approaches is that it requires no information about the environment except for the reinforcement signal [9], [14]. "In a purely mathematical context, the goal of a learning system is the optimization of a functional not known explicitly" [13].

The stochastic automaton attempts a solution of the problem without any *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 automaton acting as described to improve its performance is called a *learning automaton* [3], [14]. Stochastic automata were first used in control applications by Fu *et al*. [5]. Several researchers studied different learning schemes (e.g., [4] and [7]) and applications to different control problems (e.g., [12] and [14]). Recent applications to real life problems include control of absorption columns [11], pattern recognition [17], active vehicle suspension [9], path planning for manipulators [15], and mobile robots [1]. Here, we will describe the use of learning automata as an intelligent controller for vehicle path planning in unmodeled traffic.

When a specific action α is performed at time n, the environment responds by producing an environment β that is stochastically related to the action (Fig. 1). In the simplest case, this response may be favorable or unfavorable (0 for reward and 1 for penalty). The environment may be time varying, i.e., the stochastic characteristics of the output of the environment may be caused by the actions of other agents unknown to the automaton. If the probability of receiving a penalty for a given action is constant, the environment is called a *nonstationary environment*. 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.

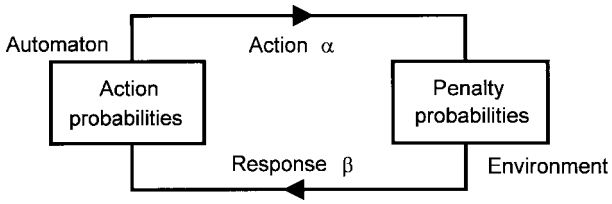

Fig. 1. The automaton and the environment.

The main concept behind the learning automaton model is the concept of a probability vector defined as $p(n) = \{p_i(n) \in \{0,1\} | p_i(n) = \Pr[\alpha(n) = \alpha_i]\}$ where α_i is one of the possible actions. The updating of the probability vector at every time step with a *reinforcement scheme* provides the learning behavior of the automaton. A reinforcement scheme, is a mapping from current action α, current environment response β, and current action probability vector $p(n)$ to $p(n+1)$, the action probability vector at the next time step.

If the automaton is "learning" in the process, its performance must be superior to an automaton for which the action probabilities are equal. Based on the average value of the penalties received by the automaton, several definitions of behavior, such as *expediency*, *optimality*, and *absolute expediency*, are given in the literature [14]. An automaton is absolutely expedient if the expected value of the average penalty at one iteration step is less than it was at the previous step *for all steps*. Absolutely expedient learning schemes are presently the only class of schemes for which necessary and sufficient conditions of design are available [2], [14]. The algorithm we will present in this paper is derived from a nonlinear absolutely expedient reinforcement scheme presented by Baba [2].

A learning automaton may send its action to multiple environments at the same time. In that case, the action of the automaton results in a vector of responses from environments (or "teachers"). Then, the automaton has to find an optimal action that satisfies all the teachers. This problem was previously discussed in [2]. Some difficulties arise while formulating a mathematical model of the learning automaton in a multiteacher environment. The question of how to interpret the output vector $\beta(n)$ is important: Are the outputs from different teachers to be summed after normalization? Can we introduce weight factors associated with specific teachers? If so, how?

The elements of the output vector must be combined in some fashion to form the input to the automaton. In this application, since the environment response is binary, teacher responses are combined using an OR gate, which forces the system to satisfy all the teachers simultaneously. However, due to safety reasons, the finalized function for the combined response includes conditions in which one teacher response inhibits the others, as we discuss later in Section III-B.

III. Intelligent Vehicle Control

We model the path controller of an intelligent vehicle as a pair of automata. The aim is to design an automata system that can learn the best possible action(s) based on the information received from on-board sensors. Vehicle-to-vehicle and roadside-to-vehicle communications could be future extensions. The system we define will be useful as a backup system (or the only system in a homogeneous traffic network) 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 between fully automatic and manual control.

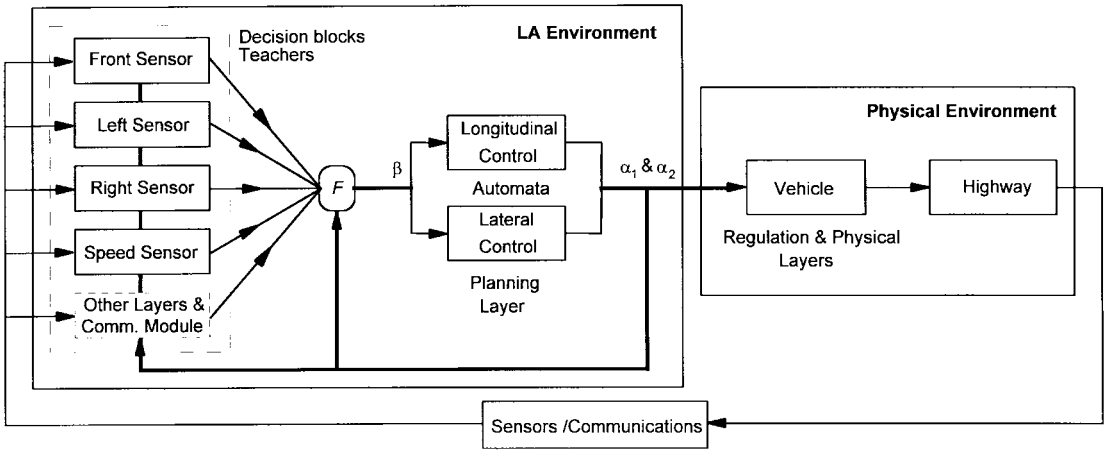

Fig. 2. Automata in a multiteacher environment connected to the physical environment.

Our model for lane and speed control of a vehicle is shown in Fig. 2. The automata that constitute the decision structure have three actions each: *stay in lane, shift to right* (lane), *shift to left* for lateral control (SiL, SR, and SL); *accelerate, decelerate* and *same speed* for longitudinal control (ACC, DEC, and SS). We assume that there are four different types of sensors (*front, right, left* and *speed*) that provide data. Each sensor block is a "teacher" in a nonstationary environment. (Inter-vehicle and roadside-to-vehicle communications, as well as a feedback from the regulation layer, could be added as additional "teachers."). The response of the environment is a combination of the outputs of all four teachers. The mapping F from sensor block outputs to the input β to the automata can be a binary function, a linear combination of five teacher outputs, or a more complex function—as is the case in this application. The function F is described in Section III-B.

It is important to differentiate between the "automaton environment" and the "physical environment." The automaton environment is the stochastic system model that provides responses to the automata actions. The action α of the automaton is a signal to the regulation layer that defines the current choice of action. It is the regulation 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. The discussion of nonstationary environments 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 SiL or ACC). The realization of a deterministic model of this physical environment would be extremely difficult.

A. Sensors

The four sensor/teacher modules listed above are simple decision blocks that calculate the penalty response associated with the corresponding sensor, based on the chosen action. Tables I and II below describe the output of the decision blocks for front and side sensors. As seen in Table I, a penalty response from left sensor module is received only when the lateral action is SL and there is a vehicle in the left sensor's field of view. Similarly, the action SR is penalized if there is a vehicle on the right lane. All other situations result in a reward response for the lateral actions. Longitudinal actions are not affected by the side sensor responses.

TABLE I
OUTPUT OF THE LEFT AND RIGHT SENSOR BLOCKS

	Sensor Status (L/R)	
Current Action	Vehicle in sensor range	No vehicle in sensor range
SiL	0 / 0	0 / 0
SL	1 / 0	0 / 0
SR	0 / 1	0 / 0

TABLE II
OUTPUT OF THE FRONT SENSOR BLOCK (REGIONS DEFINED IN FIG. 3)

	Sensor Status			
	Vehicle in region A	Vehicle in region B (d_1 < distance < d_2)		No vehicle in range
Action	(distance < d_1)	$\dot{d} \geq 0$	$\dot{d} < 0$	(distance > d_2)
SiL	1	1	0	0
SL	0	0	0	0
SR	0	0	0	0
ACC	1	1	0	0
DEC	0	0	0	0
SS	1	1	0	0

We assume that the front sensor is capable of providing the headway distance *and* we can measure its rate of change by comparing two consecutive sampling values. If the sensor "sees" a vehicle at a very close distance (d_1), a penalty is sent in response to automata actions *SiL, ACC,* and *SS*. All other actions may serve to avoid a collision, and therefore, would be "encouraged." If the vehicle in front is not too close (i.e., the distance to the vehicle is greater than d_1, but less than d_2) *and* is not approaching, the response from front sensor is favorable (Table II).

Consider the situation shown in Fig. 3, where the controlled vehicle's left and front sensors detect vehicles, and the vehicle in front is slower that the controlled vehicle. If the current automata actions are SL and ACC, the lateral automaton will receive a penalty from the left sensor. At the same time, the longitudinal automaton will receive a penalty from the front sensor because of the approaching vehicle.

Table III gives the penalty definitions for the speed sensor. The deviation (*dev*) is defined as the difference between current speed and the desired speed. A penalty response is received only by longitudinal actions. Lateral actions are not affected by the speed sensor block.

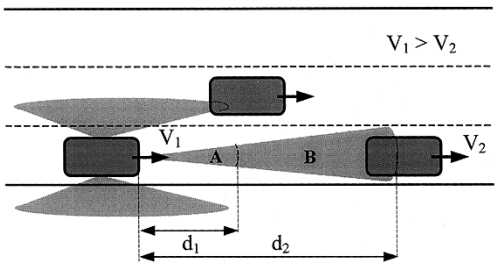

Fig. 3. The vehicle, sensing the approaching vehicle in front, can avoid a collision by shifting to the leftmost lane.

TABLE III
OUTPUT OF THE SPEED SENSOR BLOCK

	Sensor Status		
Current Action	*dev* < -*ds*	|*dev*| < *ds*	*dev* > *ds*
ACC	0	0	1
DEC	1	0	0
SS	1	0	1

The values of the limits d_1, d_2, and ds define the capability of the sensors as well as the "behavior" of the vehicle, i.e., the sensitivity to the headway distance and to the speed fluctuations. The fifth block in Fig. 2 represents the evaluation of the information received via vehicle-to-vehicle and/or roadside-to-vehicle communications [19]. The addition of communication capabilities increases an autonomous vehicle's capability to handle complex traffic situations. Here, we will emphasize the learning algorithm and its application to automated vehicle control, and therefore will not include this block in our discussion for reasons of clarity.

In the case shown in Fig. 3, the action SL will receive a penalty response, although it is one of the two actions needed to avoid a collision (the other one is DEC). Using the present algorithm, the vehicle decelerates in a situation like this (the mapping F forces this), and although it is able to avoid a collision, the resulting path is far from optimal. We can show that additional information about the environment would enable the vehicle to "escape" the pocket created by the slow-moving vehicles [19]. We expect our automata and multiteacher automata environment to guide the vehicle without collision using the learning algorithm described below.

B. The Algorithm

In a stationary N-teacher P-model environment, if an automaton has the action α_i and the environment responses are $\beta_i^j = 1, \cdots, N$ at time step n, then the probabilities for r actions $\underline{P}(n+1) = \{p_1\ p_2 \cdots p_r\}$ are updated as follows:

if $\alpha(n) = \alpha + i$,

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

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

where the functions ϕ_i and ψ_i satisfy the following conditions:

$$\frac{\phi_1(\underline{P}(n))}{p_1(n)} = \cdots = \frac{\phi_r(\underline{P}(n))}{p_r(n)} = \lambda(\underline{P}(n))$$
$$\frac{\psi_1(\underline{P}(n))}{p_1(n)} = \cdots = \frac{\psi_r(\underline{P}(n))}{p_r(n)} = \mu(\underline{P}(n))$$

$$\begin{aligned} p_j(n) + \psi_j(\underline{P}(n)) &> 0 \\ p_i(n) + \sum_{\substack{j=1 \\ j\neq 1}}^{r} \phi_j(\underline{P}(n)) &> 0 \\ p_j(n) - \phi(\underline{P}(n)) &< 1 \end{aligned} \quad (2)$$

for all $i, j = 1, \cdots, r$, i.e., the update functions ϕ_i and ψ_i have a constant value for all components of $\underline{P}(n)$, and updates of probability vector do not force the values outside of the range (0, 1) for all $p_i \in (0, 1)$.

Theorem: If the functions $\lambda(\underline{P})$ and $\mu(\underline{P})$ satisfy the following conditions:

$$\begin{aligned} \lambda(\underline{p}) &\leq 0 \\ \mu(\underline{p}) &\leq 0 \\ \lambda(\underline{p}) + \mu(\underline{p}) &< 0 \end{aligned} \quad (3)$$

then the automaton with the reinforcement scheme in (1) is absolutely expedient in a stationary environment.

The proof of this theorem can be found in [2]. In this application, we defined a single environment response that is a function of four teacher outputs, however, the theorem is also valid for our description of a single-teacher model. Our simulations are based on the following model and assumptions.

- There are three actions defined for each of two automata: *stay in lane, shift left, shift right,* and *accelerate, decelerate, same speed*.
- Four sensor modules are assumed: front, left and right range sensors and a speed sensor.
- The environment is of P-model, i.e., all inputs are binary. The basic function for the mapping F that computes the combined input to the automaton is an OR-gate. However, the front sensor block inhibits the penalty response from the speed sensor block in order to avoid a collision. This enables the action DEC to be sent to the regulation layer even if the actual speed of the car is much less than the desired speed. Otherwise, the action DEC would receive a penalty response from the speed sensor block that would restrain its action probability from approaching 1. Prioritizing the sensor information in this way is important.
- To smooth the system output, the regulation layer carries out an action if it is recommended m times consecutively by the automaton. (In other words, the length of the *memory* vector is m. Whenever this vector/buffer is filled with the same action, the action is fired.) This may of course be changed to "k times in the last m choices," or to a more sophisticated decision rule. When an action is carried out, action probabilities are re-initialized to $1/r$. The value m is chosen to be equal to the processing speed in iterations per second. Therefore, the regulation layer executes and action if it is sent consecutively over a period of 1 s.
- The technological requirements for the models used in the simulation (as well as the model described above) are no different from those defined in the current AHS research [8]. Processing speed range is [25, 200]. This is related only to computation; the sensor data feeds have a different rate.
- Mapping F can be represented as shown in Fig. 4.

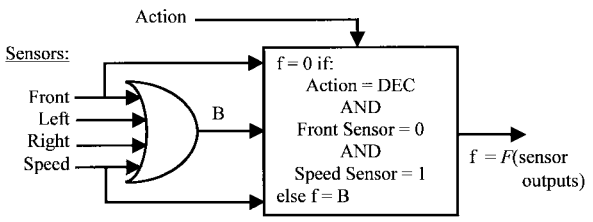

Fig. 4. Definition of the mapping F.

From (1), we can write our update algorithm as

$$\begin{aligned} &\text{if } \alpha(n) = \alpha_i \\ &p_i(n+1) = p_i(n) + f \cdot (-k \cdot \theta \cdot H(n)) \cdot [1 - p_i(n)] \\ &\qquad -[1-f] \cdot (-\theta) \cdot [1 - p_i(n)] \\ &p_j(n+1) = p_j(n) - f \cdot (-k \cdot \theta \cdot H(n)) \\ &\qquad \cdot p_j(n) + [1-f] \cdot (-\theta) \cdot p_j(n) \quad \text{for all } j \neq i \end{aligned} \tag{4}$$

i.e.:

$$\begin{aligned} \psi_k(\underline{P}(n)) &\equiv -\theta \cdot p_k(n) \\ \phi_k(\underline{P}(n)) &\equiv -k \cdot \theta \cdot H(n) \cdot p_k(n) \end{aligned} \tag{5}$$

where learning parameters k and θ are real values and satisfy:

$$0 < \theta < 1 \qquad 0 < k\theta < 1 \tag{6}$$

The feedback f is given as in Fig. 4, and the function H is defined as:

$$H(n) \equiv \min\left\{1; \quad \max\left[\frac{p_i(n)}{k\theta(1-p_i(n))} - \varepsilon; 0\right]\right\} \tag{7}$$

Parameter ε is an arbitrarily small positive real number. Also note that the function H includes p_i which is the action probability corresponding to the *current* action.

We now show that the defined algorithm [(4)–(7)] satisfies all the conditions in (2) and (3).

- From (5), we have

$$\begin{aligned} \frac{\phi_k(\underline{P}(n))}{p_k(n)} &= \frac{-k \cdot \theta \cdot H(n) \cdot p_k(n)}{p_k(n)} \\ &= -k \cdot \theta \cdot H(n) \equiv \lambda(\underline{P}(n)) \end{aligned} \tag{8}$$

and

$$\frac{\psi_k(\underline{P}(n))}{p_k(n)} = \frac{-\theta \cdot p_k(n)}{p_k(n)} = -\theta \equiv \mu(\underline{P}(n)). \tag{9}$$

That is, our definition is consistent with the first two conditions of (2).

- Using (4) and (5), the rest of the conditions on ϕ_i and ψ_i translates to the following:

$$\begin{aligned} &\text{a)} \quad p_i(n) + \theta \cdot (1 - p_i(n)) < 1 \\ &\text{b)} \quad p_j(n) - \theta \cdot p_j(n) > 0 \\ &\text{c)} \quad p_i(n) - k \cdot \theta \cdot H(n) \cdot (1 - p_i(n)) > 0. \end{aligned} \tag{10}$$

Conditions (a) and (b) are associated with the reward updates while the last one is associated with the penalty updates. These conditions guarantee that the probabilities stay in the range (0, 1) at all times (with the assumption that none of the probabilities is initially 0 or 1). Conditions (a) and (b) can be shown to be satisfied as follows, using the fact that the sum all probabilities is 1:

$$\begin{aligned} \text{a)} \quad & p_i + \theta \cdot (1 - p_i) < p_i + \theta \cdot \sum_{\substack{j=1 \\ j \neq i}}^{r} p_j < p_i \\ & + \sum_{\substack{j=1 \\ j \neq i}}^{r} p_j = 1 \text{ since } 0 < \theta < 1 \end{aligned} \tag{11}$$

$$\begin{aligned} \text{b)} \quad & p_j(n) - \theta \cdot p_j(n) = p_j \cdot (1 - \theta) > 0 \\ & \text{since } 0 < p_j < 1 \text{ and } 0 < \theta < 1 \end{aligned} \tag{12}$$

For the third condition, we have

$$\begin{aligned} \text{c)} \quad & p_i(n) - k \cdot \theta \cdot H(n) \cdot (1 - p_i(n)) > 0 \\ & \Leftrightarrow k \cdot \theta \cdot H(n) \cdot (1 - p_i(n)) < p_i(n) \\ & \Leftrightarrow H(n) < \frac{p_i(n)}{k \cdot \theta \cdot (1 - p_i(n))}. \end{aligned} \tag{13}$$

This condition is already satisfied by the previous definition of the function $H(n)$. For the limiting values 0 and 1, we have:

- $H = 0 \Rightarrow p_i(n) > 0$ which is true for all probabilities at all times.
- For $H = 1$, we must have $p_i(n) - k \cdot \theta \cdot (1 - p_i(n)) > 0$ to satisfy condition (c). From the definition of the function, we conclude that $H = 1$ implies

$$\frac{p_i(n)}{k \cdot \theta \cdot (1 - p_i(n))} - \varepsilon > 1. \tag{14}$$

This inequality can be rewritten as

$$p_i(n) > (1 + \varepsilon) \cdot k \cdot \theta \cdot (1 - p_i(n)) \tag{15}$$

or, omitting the time step variable and rearranging

$$p_i - k \cdot \theta \cdot (1 - p_i) - \varepsilon \cdot k \cdot \theta \cdot (1 - p_i) > 0. \tag{16}$$

Since, all the factors of the third term are positive real, we may omit this term without affecting the inequality to obtain

$$p_i - k \cdot \theta \cdot (1 - p_i) > 0 \tag{17}$$

which is exactly what we must have for $H = 1$.

Thus, all the conditions (2) in the definition of the reinforcement scheme are satisfied. Furthermore, the functions λ and μ satisfy the following:

$$\begin{aligned} \lambda(\underline{P}(n)) &= -\theta < 0 \\ \mu(\underline{P}(n)) &= -k \cdot \theta \cdot H \leq 0 \\ \lambda(\underline{P}(n)) + \mu(\underline{P}(n)) &= -\theta - k \cdot \theta \cdot H < 0 \end{aligned} \tag{18}$$

because $0 < \theta < 1, 0 < l \cdot \theta < 1$, and $0 < H < 1$.

Comparing (18) and (2), we see that the theorem is satisfied, i.e., the algorithm given in (4)–(7) is absolutely expedient in a stationary environment.

The physical environment (the automated highway) is of course changing with time. However, the automata environment can be considered to be stationary, since the update rate for the action probabilities is relatively high. That is, for a physical environment that does not change quickly, the automata would be capable of finding

TABLE IV
CONVERGENCE RATES FOR A SINGLE OPTIMAL ACTION OF A 3-ACTION AUTOMATON IN A STATIONARY ENVIRONMENT

Parameters		Average number of steps to reach $p_{opt} = 0.995$			
		(a) 3 actions with p(0)=[1/3 1/3 1/3]		(b) 3 actions with p_{opt} 0.005, $p_{1\neq opt}$ −0.995/2	
K	θ	ε-optimal algorithm	New algorithm with H	ε-optimal algorithm	New algorithm with H
1	0.010	544.99	306.48	952.94*	534.42
	0.035	155.67	155.59	358.46**	159.11
	0.050	108.79	66.22	263.84**	113.98
	0.100	54.11	35.13	184.85	60.13
	0.200	26.85	19.21	141.64	31.16
	0.050	102.80	48.36	259.77	77.60
2	0.100	51.42	27.57	185.83	41.70
	0.200	25.58	15.73	117.84	23.53

* 80 runs did not converge in 1200 steps.
** 1 run did not converge in 1200 steps.
H is defined in Equation 7.
500 runs for each parameter set.

the necessary action by a learning process that does better at each time step. Here, the reinforcement scheme updates the probability values so that the expected value of the total penalty received from the environment decreases at each iteration.

The choice of the function H is also due to another factor besides the conditions for absolute expediency (2). Our algorithm converges to a solution faster than the one given in [2]. We compared two reinforcement schemes using three actions and two different initial conditions. Our definitions result in a faster update in general as shown in Table IV. The new definitions are based on the fact that the update rate for a penalty response from the environment is higher when the probability of the current action is close to 1. [Note that H is a function of the probability of the current action p_i; it is not related to the index j in (1).] This provides a much faster convergence rate when the actions receiving a penalty from the environment have high associated probability values.

The data shown in Table IV are the results of two different initial conditions where 1) all probabilities are initially the same with only one action receiving reward (i.e., optimal action) and 2) the optimal action initially has a small probability value. The difference in convergence rate is more distinct in the situation where the probability of the optimal action is initially very close to 0. (This situation occurs frequently in our application. For example, while the probability of the lateral action *shift left* is converging to 1, a vehicle may enter the left sensor range. In this case, we need a "strong" penalty update to decrease the probability of this action, while "encouraging" the action *stay in lane*.) In order to have a fast update on the probability vector, the function H is set to the highest possible value [see (6) and (7)] satisfying the conditions given in (2). As seen in Table IV, when $p_{\rm opt} = 0.005$, the number of iteration steps to reach $p_{\rm opt} = 0.995$ is reduced drastically for relatively large values of the learning parameter θ.

IV. SIMULATION RESULTS

In this section, we will give the snapshots of a very short segment of the simulation (14 s) with the action probability vectors of two automata. As seen in Fig. 5, the controlled vehicle (darker color) travels in a traffic moving from left-to-right with an speed of 88 km/h at time $t = 10$ s. The action probabilities for lateral and longitudinal automata (Fig. 6) are updated based on the sensor data received. Sensor ranges during the given simulation are $d = 10$ m for side sensors, and $d_1 = 6$ and $d_2 = 12$ meters for the front sensor.

Controlled vehicle that is initially in the middle lane decelerates from 90 km/h as it detects the car in front. While decelerating, the lateral action SR (shift right) is fired at around $t = 15$ s, due to the penalty responses received from left and front sensors. After shifting to the leftmost lane, vehicle speed increases (ACC is fired around $t = 18$ s) because the front sensor no longer detects any vehicle.

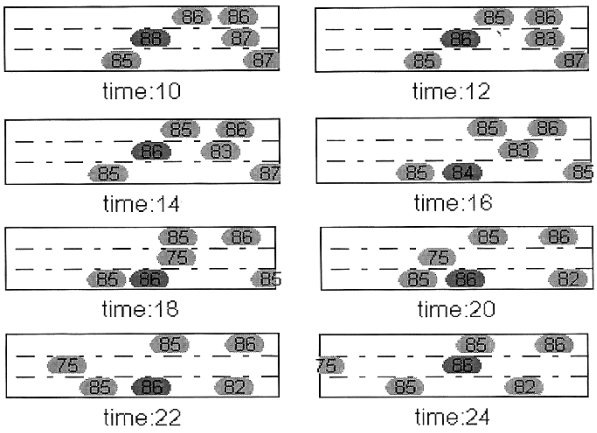

Fig. 5. The trajectory of a vehicle (darker color) traveling left-to-right.

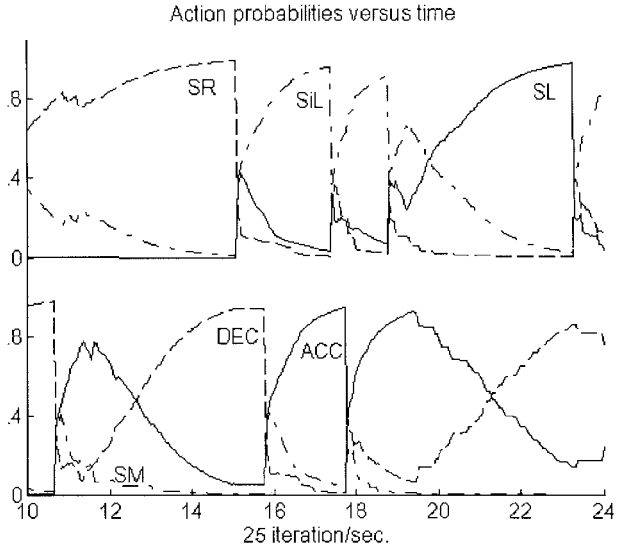

Fig. 6. Action probabilities for the same time interval.

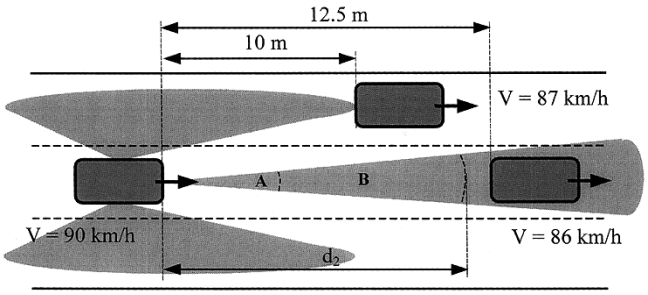

Fig. 7. A situation where Shift Right is the optimal lateral action.

This increase is the result of our definition of the desired speed range which is 90 ± 3 km/h (i.e., $ds = 3$). As soon as the car in the rightmost lane enters the front sensor range, the controlled vehicle shifts to the middle lane. It can also be seen from Fig. 6 that the probability of the action DEC increases until the moment that the lateral action SL (shift left) is fired ($t \approx 23$ s). The velocity keeps increasing after $t = 24$ s, until it is in the desired range. Between $t = 14$ s and $t = 16$ s, the probability of the action DEC is decreased although the velocity is already less than the desired value. This is the direct result of the definition of mapping F as described in Section III-

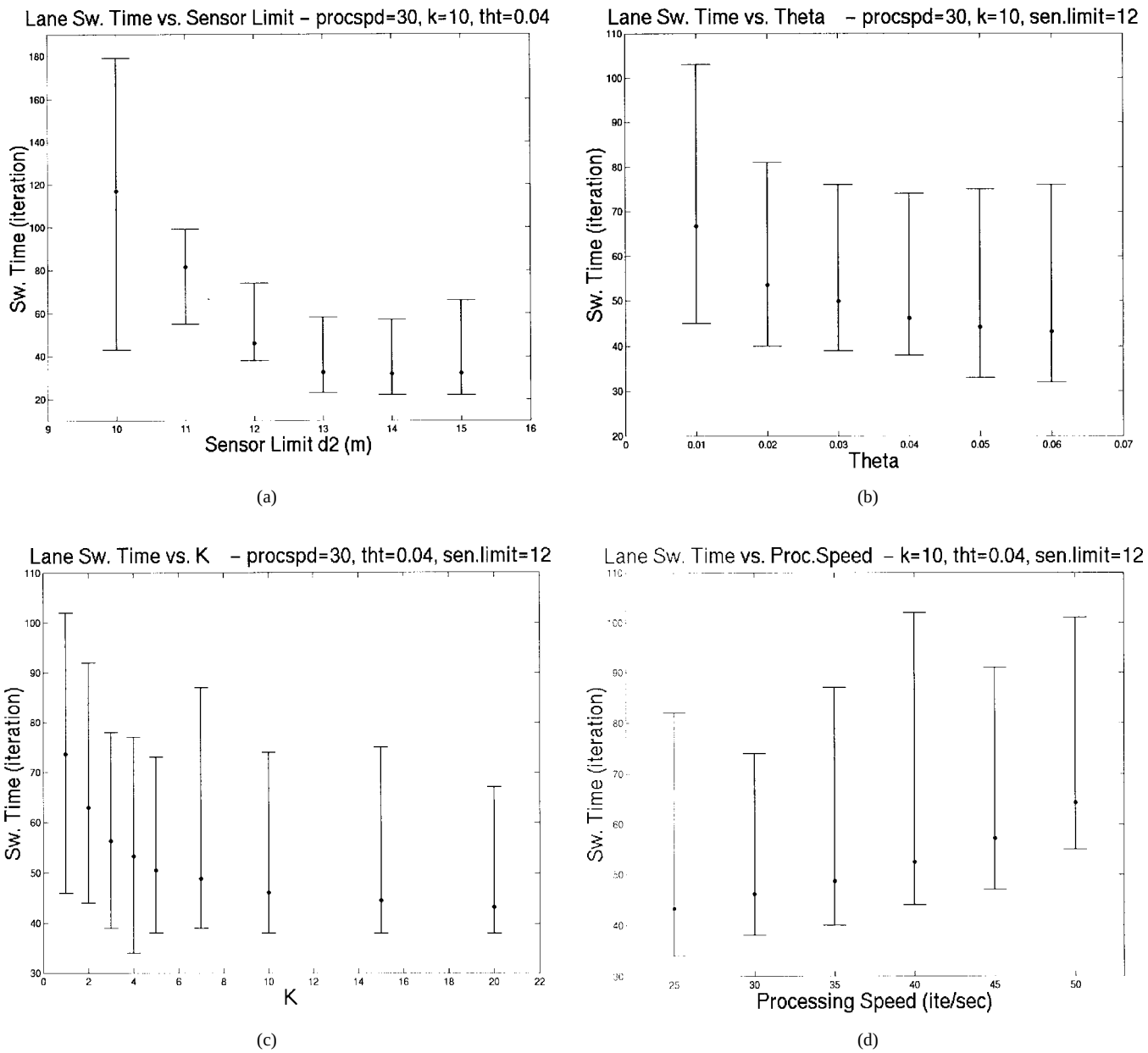

Fig. 8. Effect of parameters on action decision (the dot shows the average value, and vertical lines show the range of data over 500 runs).

B, and is necessary. Otherwise, the vehicle could not shift lanes in time because of the presence of vehicles on both sides. The slowest update rate used for both the sensor data and the decision algorithm is 25 iterations per second which is very low considering the values assumed in current AHS research [8].

V. Discussion on Results and Further Research

With its limited sensor capabilities, the vehicle cannot obtain a global view of its environment. The need for a higher layer of hierarchy (such as the *link layer* in [20]) and vehicle-to-vehicle communications is inevitable. In the situation shown in Fig. 3, one of the two possible solutions for avoiding a collision is shifting to the leftmost lane. If the vehicle is not capable of quickly changing its speed, the vehicle's decision block based on local sensor data cannot avoid an imminent collision, because action SL is prohibited by the left sensor. For similar reasons, a connection to a higher layer (of information) is necessary. Such a layer that has more complete data of the changing environment, must assist/supervise the vehicle in its actions/decisions.

In the simulations, we assumed an execution time of 0.5 s for actions in order to guarantee a comfortable acceleration for lane change maneuvers. However, the regulation layer that controls the vehicle trajectory may not always be ready to execute the recommended action. Therefore, the regulation layer must inform the planning layer about the feasibility of the actions.

The behavior of the controlled vehicle currently depends on several factors, such as frequency of update, sensor definitions, and learning parameters. For example, the vehicle sometimes changes its lane after a short period of time, and sometimes in the last possible moment.

Consider the situation shown in Fig. 7. Controlled vehicle ($V = 90$ km/h) approaches two other vehicles, one in the middle lane ($V = 86$ km/h), the other in left lane ($V = 87$ km/h). Suppose that at the instant that the other vehicles are at 12.5 m (in front) and 10m (on

the left) from the controlled vehicle, the probabilities of the three actions SiL, SR, and SL are equal (1/3). It is obvious that the vehicle will shift to the rightmost lane as soon as the action SR is chosen m times, in order to avoid collision (if it does not change its speed).

Fig. 8 shows the effect of sensor limit d_2, learning parameters k and θ, and processing speed on the convergence to the optimal solution. As seen in Fig. 8(a), the decision to change lane is obtained faster if the sensor limit d_2 is higher. Since the initial headway distance is defined as 12.5 m, the response is the same for $d_2 \geq 13$ m. A 2 m increase (from 10 to 12 m) in the sensor limit provides a change of $179 - 74 = 105$ iterations (i.e., $105/30 = 3.5$ s) on the maximum, and $116.81 - 46.15 \approx 70$ iterations (2.3 s) on the average in the decision process. Again, for an action to be carried out, it must be chosen 30 times consecutively.

A second important factor is the learning parameters. As θ increases, the time to shift lane decreases, since the probability updates are faster [Fig. 8(b)]. The data obtained suggest that values greater than or equal to 0.04 must be used for this application. When the learning parameter k is increased, the time to shift lane decreases, this time due to faster updates when a penalty is received [Fig. 8(c)]. For $k \geq 10$, the effect on the decision is the same. Small values of k can be used to adjust the "reaction" time. Furthermore, the effects of the learning parameters are not as significant as the sensor limit definition.

The last parameter we tested is the processing speed. If we keep the length of the memory vector for a regulation layer decision the same, increasing the frequency of the updates will of course decrease the time to reach the optimal decision. However, if the definition of this memory vector is kept the same as the processing speed (i.e., one-second memory), larger values of the processing speed increase the time to reaction [Fig. 8(d)]. Again, the data show that doubling the processing speed (and the length of the memory vector) results in a loss of (82/25)-(101/50) = 1.26 s on the maximum, and (43.23/25)-(64.31/50) = 0.44 s on the average.

Extension to the S-model (where the teacher outputs are continuous) may help to decrease the update frequency, because it will give the automaton more time to adjust. Adjusting the parameters simultaneously for optimum results is a very difficult task, even for a simple automaton/environment pair.

The need for more complicated sensor definitions and for a "hierarchical interference" (such as commands from a higher layer [20]) 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 vehicles in AHS. Since 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 may not have complete information about those vehicles without communication capabilities. Therefore, a controlled vehicle must rely on its own sensors input for complete data on its immediate neighborhood in heterogeneous traffic. Also, some form of feedback from the lower/regulation layer, describing the plausibility of the action requested by the automata (planning layer), must be included in order to guarantee safe operation of the vehicle.

As a practical matter, we should note that our model of vehicle control is consistent with the current assumptions on sensors and communications capabilities. Desired sensor and communication characteristics for the controller described here are generally the same, if not lower, than the ones required by other control aspects of AHS. Some applications of longitudinal vehicle control require a communication rate of 200 Hz and headway sensors with a few centimeters resolution.

There are many ways to improve our initial controller design. Some of the issues that need investigation are as follows.

- The P-model environment may be extended to S-model in order to incorporate a priority level with the teacher outputs. The multiteacher characteristic of the environment will then show its potential application. The weighting factors and/or parameters associated with each teacher/sensor output define the "behavior" of the vehicle. The adjustment of these can be viewed as another level of learning.
- The use of two or more automata for vehicle control will move the control problem toward a multi-automata system, which could subsequently bring us to an application of game theory to interconnected automata.
- Our intelligent controller model must be incorporated with a realistic vehicle model to simulate the physical constraints of the actions to be carried out by the regulation layer. Emerging hybrid system modeling applications provide the necessary simulation platform for such applications.

The parameters of the learning automata and the definition of sensor ranges, as well as the processing speed and the size of the memory vector for the regulation layer, define the behavior of a single vehicle on a highway. Simulation of vehicle flow in a single lane highway using learning automata may prove to be useful, since previous attempts using more "discrete" methods (e.g., cellular automata) and simpler control rules were successful [10]. Our approach should bring a "less discrete" model to traffic simulations, and multilane simulations would be possible.

VI. Conclusion

The intelligent controller for vehicle path planning described here consists of two stochastic learning automata. Using the data received from on-board sensors, each automaton can learn the best possible lateral/longitudinal action to be sent to lower layer in the control hierarchy in order to avoid collisions. This nonmodel-based adaptive method would be especially useful in situations where complete information about the flowing traffic is not provided by the higher levels of the control hierarchy. The reinforcement scheme for the learning process is designed to guarantee a fast convergence to an action that is optimal. Simulations for simultaneous lateral and longitudinal control provided encouraging results. This method is also capable of capturing the overall dynamics of the system that includes the vehicle, the driver and the roadway.

References

[1] T. Aoki, T. Suzuki, and S. Okuma, "Acquisition of optimal action selection in autonomous mobile robot using learning automata (experimental evaluation)," in *Proc. IEEE Conf. Fuzzy Logic Neural Networks/Evol. Computation*, 1995, pp. 56–63.

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

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

[4] B. Chandrasekharan and D. W. C. Shen, "On expediency and convergence in variable structure stochastic automata," *IEEE Trans. Syst., Sci., Cybern.*, vol. SSC-5, pp. 145–149, 1968.

[5] K. S. Fu, "Stochastic automata as models of learning systems," in *Computer and Information Sciences II*, J. T. Lou, Ed. New York: Academic, 1967.

[6] U. Herkenrath, D. Kalin, and W. Wogel, Eds., *Mathematical Learning Models: Theory and Applications*. New York: Springer-Verlag, 1983.

[7] S. Lakshmivarahan, *Learning Algorithms: Theory and Applications*. New York: Springer-Verlag, 1981.

[8] T. L. Lasky and B. Ravani, "A review of research related to automated highway system (AHS)," Interim Rep. for FHWA, DTFH61-93-C-00189, Univ. California, Davis, Oct. 1993.

[9] C. Marsh, T. J. Gordon, and Q. H. Wu, "Stochastic optimal control of active vehicle suspensions using learning automata," in *Proc. Mech. Eng. Part I, J. Syst. Contr. Eng.*, vol. 207, pp. 143–152, 1993.
[10] K. Nagel and S. Rasmussen, "Traffic at the edge of chaos," in *Artificial Life IV*. Cambridge, MA: MIT Press, 1994, pp. 222–235.
[11] K. Najim, "Modeling and self-adjusting control of an absorption column," *Int. J. Adap. Contr. Signal Process.*, vol. 5, pp. 335–345, 1991.
[12] K. Najim and A. S. Poznyak, Eds., *Learning Automata: Theory and Applications*. Oxford, U.K.: Elsevier, 1994.
[13] K. S. Narendra and M. A. L. Thathachar, "Learning automata—A survey," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-4, July 1974.
[14] K. S. Narendra and M. A. L. Thathachar, *Learning Automata*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[15] K. Naruse and Y. Kakazu, "Strategy acquisition of path planning of redundant manipulator using learning automata," *IEEE Int. Workshop Neuro-Fuzzy Ctrl.*, 1993, pp. 154–159.
[16] M. F. Norman, *Markov Processes and Learning Models*. New York: Academic, 1972.
[17] B. J. Oommen and E. V. de St. Croix, "String taxonomy using learning automata," Tech. Rep. TR-234, School of Computer Science, Carleton Univ., Ottawa, Ont., Canada, Mar. 1994.
[18] M. L. Tsetlin, *Automaton Theory and Modeling of Biological Systems, Vol. 102 in Mathematics in Science and Engineering*. New York: Academic, 1973.
[19] C. Ünsal, "Intelligent Navigation of Autonomous Vehicles in an Automated Highway System: Learning Methods and Interacting Vehicles Approach," Ph.D. dissertation, Virginia Tech., Blacksburg, Jan. 1997, (http://www.theses.org/).
[20] P. Varaiya, "Smart cars on smart roads: Problems of control," *IEEE Trans. Automat. Contr.*, vol. 38., pp. 195–207, Feb. 1993.

Semantic Constraints for Membership Function Optimization

J. Valente de Oliveira

***Abstract*—The optimization of fuzzy systems using bio-inspired strategies, such as neural network learning rules or evolutionary optimization techniques, is becoming more and more popular. In general, fuzzy systems optimized in such a way cannot provide a linguistic interpretation, preventing us from using one of their most interesting and useful features. This work addresses this difficulty and points out a set of constraints that when used within an optimization scheme obviate the subjective task of interpreting membership functions. To achieve this a comprehensive set of semantic properties that membership functions should have is postulated and discussed. These properties are translated in terms of nonlinear constraints that are coded within a given optimization scheme, such as backpropogation. Implementation issues and one example illustrating the importance of the proposed constraints are included.**

***Index Terms*—Adaptive systems, fuzzy systems, learning, membership functions, neural networks, optimal interfaces, optimization, semantic constraints.**

I. Introduction

When compared with other nonlinear modeling techniques such as artificial neural networks, Hammerstein, Wiener, or more generally NARMAX models (cf. [3]), fuzzy systems have the important advantage of providing an insight on the linguistic relationship between the variables of the system.

In the context of approximation tasks, the optimization of fuzzy systems (such as controllers, models, or classifiers) using biological inspired strategies (such as neural networks learning rules, or evolutionary optimization techniques like genetic algorithms) is becoming more and more popular. What is common in these bio-inspired strategies is that the only indicator guiding the search process is concerned exclusively with the system quality in terms of the approximation task (e.g., with the errors between system output and training data). For example, this is case with the so-called performance index of the neural networks learning rules [7] and with the fitness function of the genetic algorithms [5].

In the presence of such unconstrained optimization processes the parameters of membership functions (e.g., centers and widths) are treated just like any other parameter of the system. Furthermore, usually a fuzzy system has very many independent parameters (degrees of freedom). As a consequence, several local minima of the performance function are likely to exist. In face of this we simply cannot ensure that the resulting membership functions have any semantic value in what concerns their ability to be interpretable as linguistic terms. To ignore this aspect during fuzzy system design is first of all to renounce the richness of concepts and techniques provided by fuzzy sets and systems. It is comparable to let the fuzzy parameters outgoing their range (usually [0, 1]) violating the set or logic nature of fuzzy calculus. It is equivalent to consider fuzzy systems as a tool for initializing artificial neural networks (in the sense that without semantics a fuzzy system can be viewed as a neural net).

While this is undebatable, one may legitimately ask 1) whether constraining the optimization to preserve semantics would not degrade the fuzzy system performance, and 2) what would be the reasons to translate the functioning of, e.g., a ready-to-implement fuzzy control algorithm into a natural-language equivalent description.

For addressing the first issue, it is recalled that, in general, unconstrained optimization methods are more susceptible to get stuck in local minima than conveniently constrained ones. One the other hand, minima are necessarily local in that region of parameters space where there is no possible interpretation. A typical case being the absence of membership functions in a subset of the universe of discourse. This being so, there would be input values for which the system would have no response. Constraints may help to avoid local minima. The systematic avoidance of local minima leads to the global minimum, and thus to optimal performance. This contradicts the naive belief that constrained optimization cannot perform better than unconstrained optimization. Take adaptive control as an example. In certain model-reference schemes unconstrained optimization of the controller gains leads to unstable closed-loop responses. However when equipped with convenient constraints, these control schemes leads to guaranteed stability, cf. [1].

Later on, this semantics/performance synergy is further illustrated by means of an included example where a fuzzy system optimized with semantic constraints performs better than other with the same structure and initial conditions, but optimized without taking care of semantics.

Notice also that the first applied fuzzy (control) systems were designed using heuristic methods based on the knowledge and experience of experts, and thus they were necessarily semantically valid. However this has not degraded their performances; by the opposite in

Manuscript received April 15, 1996; revised February 10, 1997 and April 10, 1998.

The author is with the Department of Mathematics and Computer Science, Universidade Da Beira Interior, 6200 Covilhã, Portugal.

Publisher Item Identifier S 1083-4427(99)00895-4.